

AngularJS

...

Carlos Vicient Monllaó

Contenidos

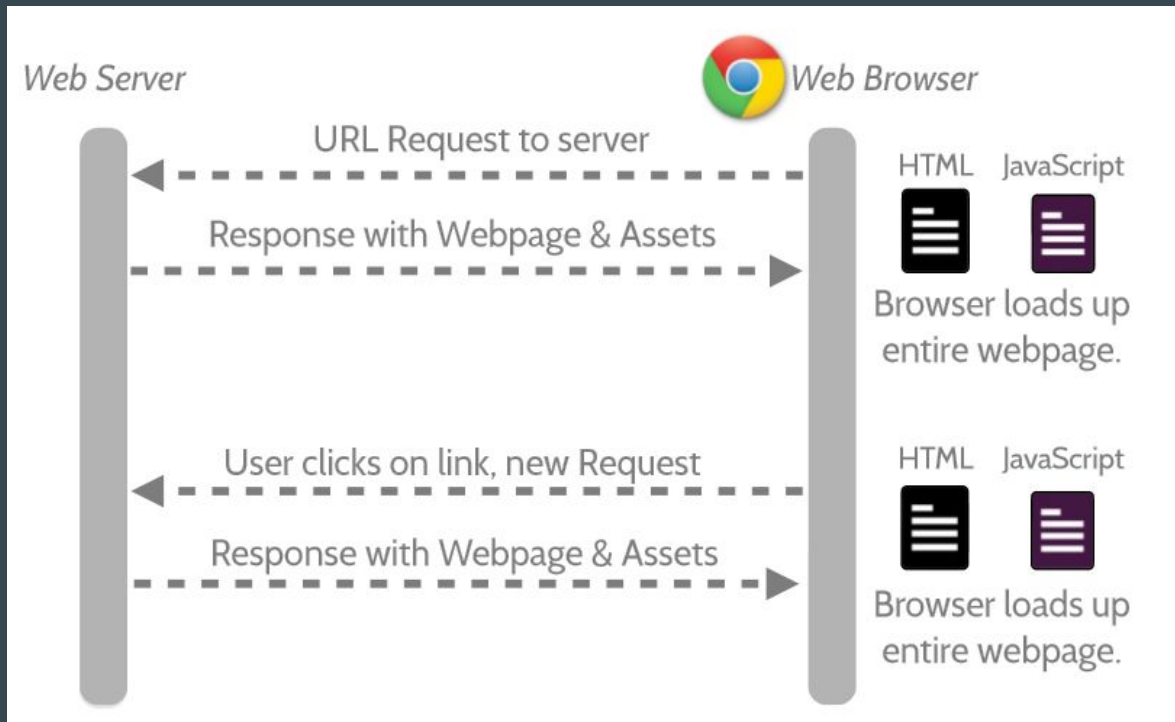
- Introducción
- Angular
- Estructura recomendada y herramientas
- Conclusiones
- Enlaces de interés

Introducción

- Web clásica
- Web moderna
- API-Driven
- Tendencias actuales

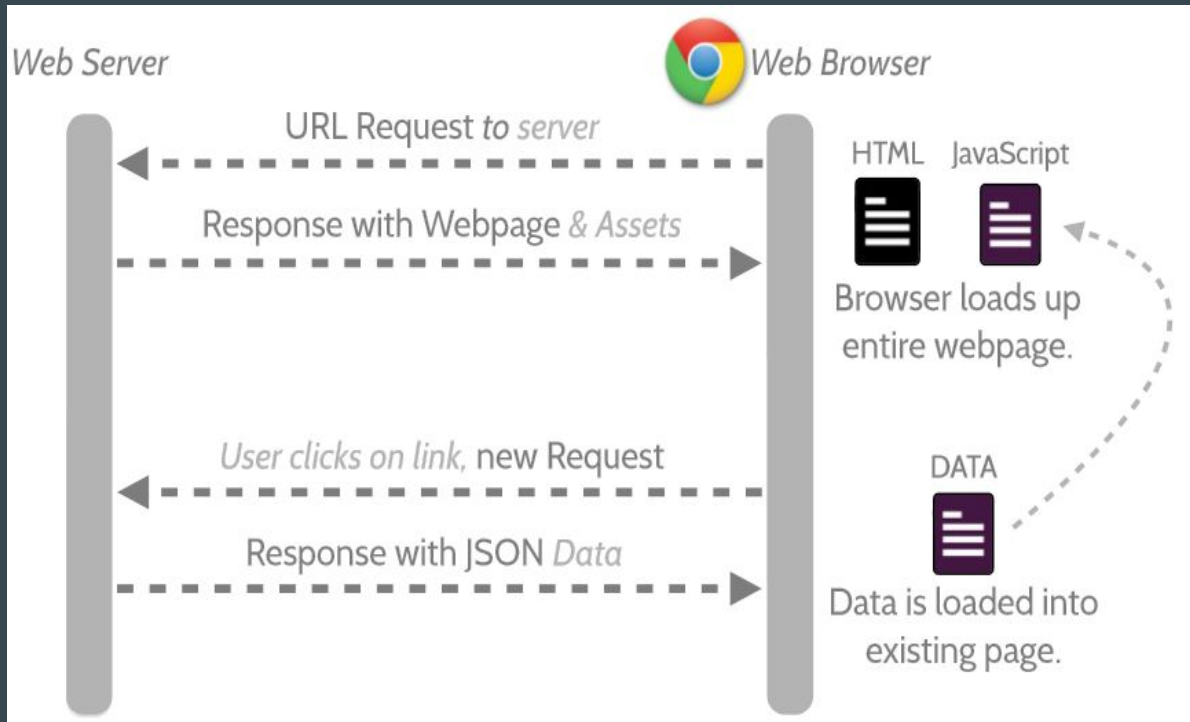


Introducción. Web clásica



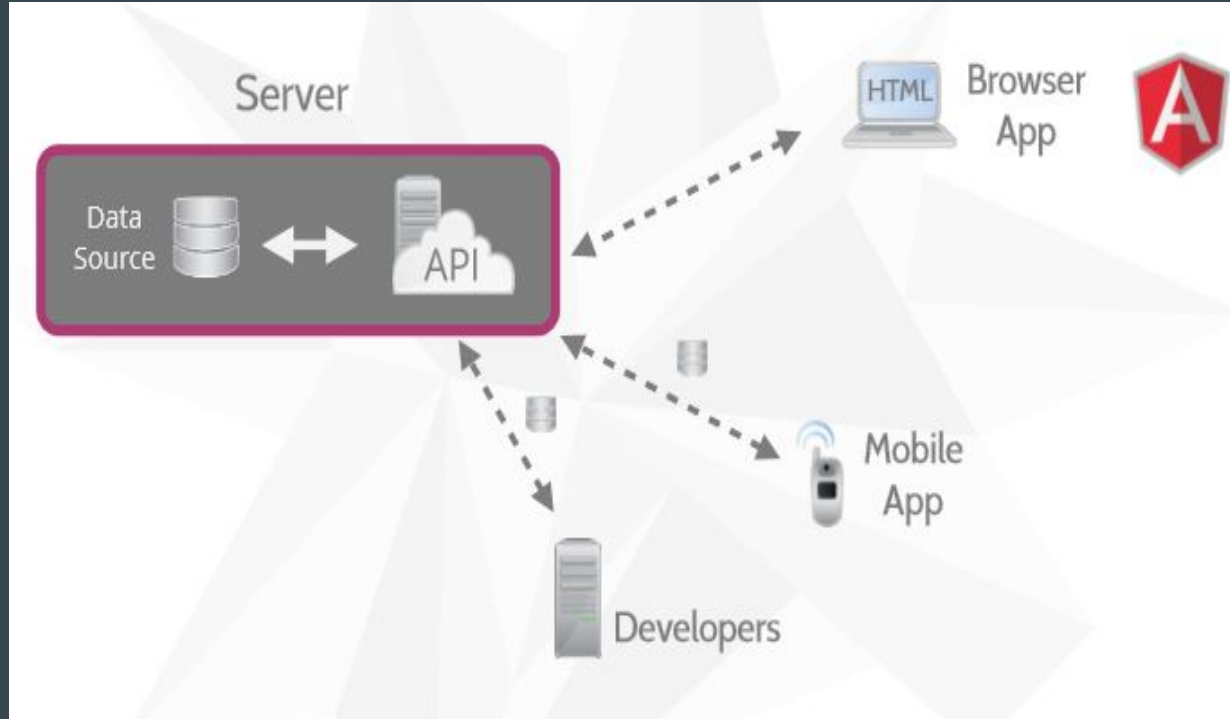
- El servidor y el cliente intercambian muchos datos en cada petición
- El servidor tiene mucha carga de trabajo
- Poco eficiente

Introducción. Web moderna (I)



- El cliente descarga la página (HTML) y código Javascript en la primera petición
- El cliente y el servidor intercambian datos
- El cliente renderiza los datos
- La carga de trabajo se traslada al cliente
- Eficiente
- Barato

Introducción. Web moderna (II)

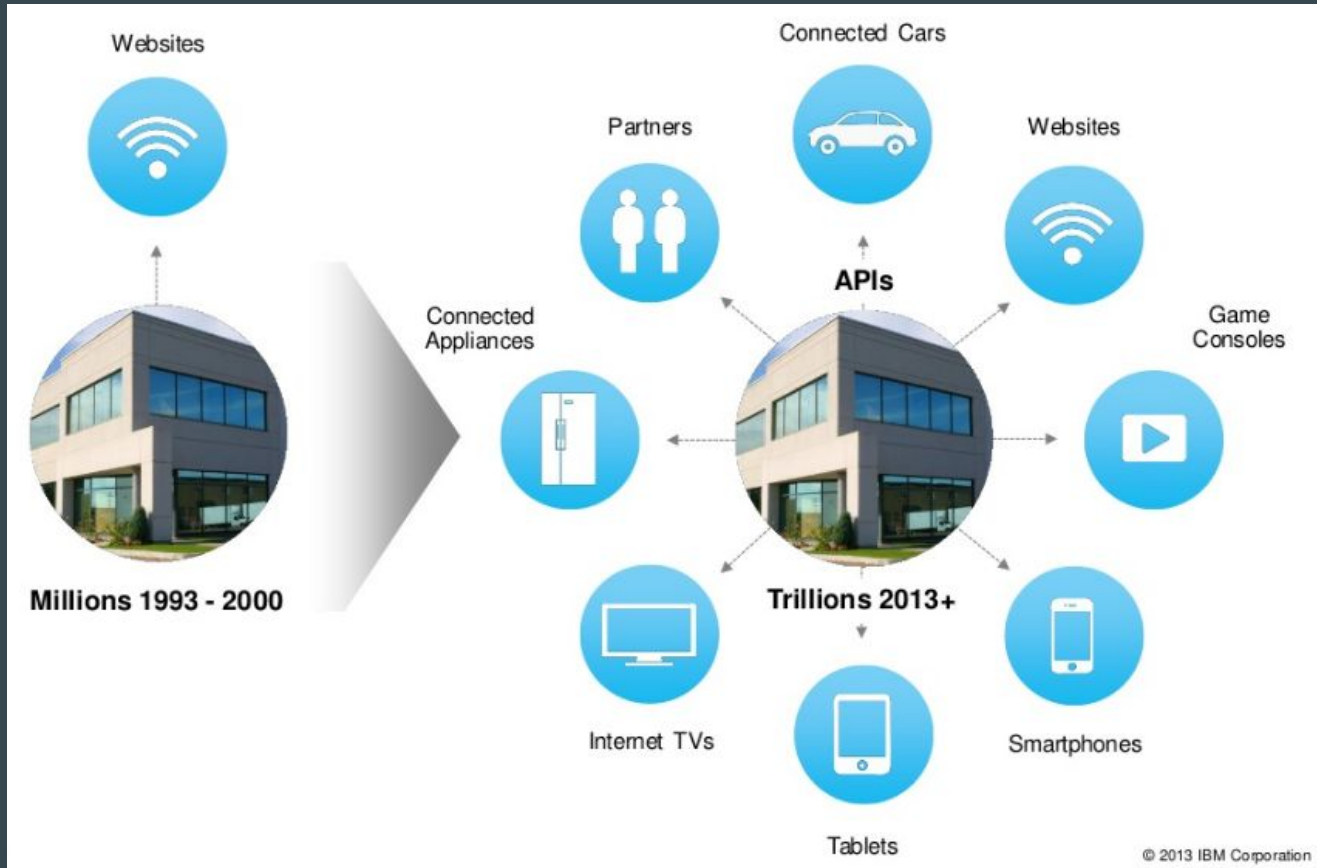


- *API-driven development*
- El cliente descarga la página (HTML) y código Javascript en la primera petición
- El cliente y el servidor intercambian datos
- El cliente renderiza los datos
- La carga de trabajo se traslada al cliente
- Eficiente
- Barato

Introducción. API-Driven

API-Driven development

- *Application Programming Interface*
- Metodología de programación: Estructurar datos -> implementar código para interactuar con los datos
- Provee una forma bien estructurada y consistente para acceder a un recurso
- Herramienta esencial para diseñar arquitecturas escalables y de calidad
- SPAs (Single Page Applications) como AngularJS, EmberJS, BackboneJS, etc. pensados para complementar aplicaciones basadas en RESTful APIs (Backend)



1 billion

*Smartphone
users by 2016*



7 billion

*Dollars in eBay transactions
processed through APIs*



69.1%

*Increase in mobile
transaction volume
since 2011*



10x traffic

*Reaches Twitter
through APIs than the
Web*



42%

*Of enterprises say
poor integration hurts
customer-facing apps*



126 million

Tablet users by 2016



ebay



del.icio.us

flickr

facebook



amazon
web services™

twitter



Expedia Affiliate
Network

NETFLIX



PayPal

Google

Google
Maps



Google Drive

Introducción. Tendencias actuales

LAMP: Linux, Apache, MySQL, PHP

- Finales del 2000
- Tecnología consolidada y robusta
- Depende del S.O*
- Web Server “blocking”
- Base de datos relacional (overrated)
- Varios lenguajes
- Difícil de escalar
- Rendimiento:
 - Cada conexión (usuario) independiente al resto => lentitud solo para el usuario que realiza tarea compleja
 - Muchos usuarios recurrentes => Agotar recursos

MEAN: MongoDB, ExpressJS, AngularJS, NodeJS

- ~2009
- Tecnología muy nueva
- No depende del sistema operativo
- **Non-blocking (event-based)**
- Base de datos no relacional (JSON)
- 100% javascript
- Escalable y muy rápido
- Rendimiento:
 - Una única instancia(Node) => optimización de los recursos
 - Tarea compleja un usuario afecta a todos.

Node.js

Software Devel...

Symfony

Software

Laravel

Web application..

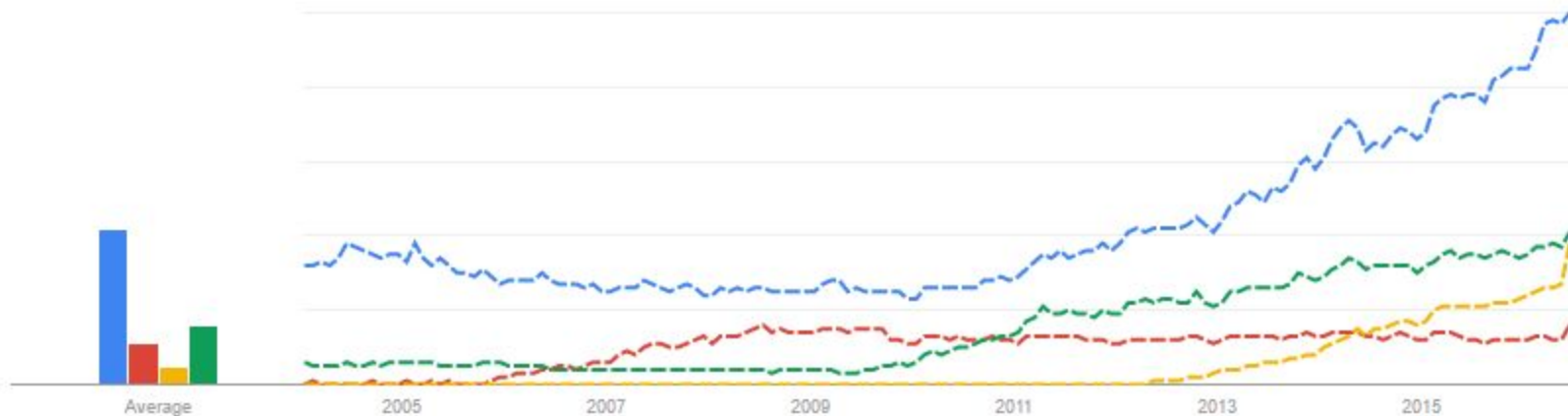
MongoDB

Database Softw...

[+ Afegeix un terme](#)

Versió beta: el mesurament de l'interès de cerca de diferents *temes* és una funció beta que proporciona ràpidament mesuraments precisos sobre l'interès de cerca general. Per mesurar l'interès de cerca d'una *consulta* específica, seleccioneu l'opció "terme de cerca". [?](#)

Interest over time [?](#)

☐ News headlines [?](#)☐ Forecast [?](#)

</>

MongoDB

Database Software

MySQL

System software

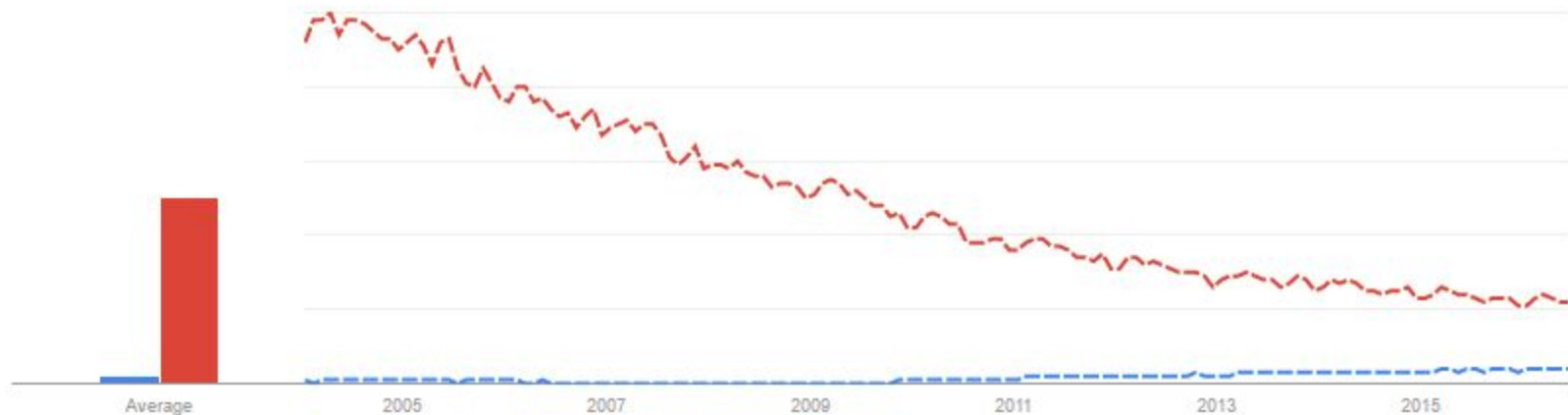
+ Afegeix un terme

Versió beta: el mesurament de l'interès de cerca de diferents *temes* és una funció beta que proporciona ràpidament mesuraments precisos sobre l'interès de cerca general. Per mesurar l'interès de cerca d'una *consulta* específica, seleccioneu l'opció "terme de cerca". ?

Interest over time ?

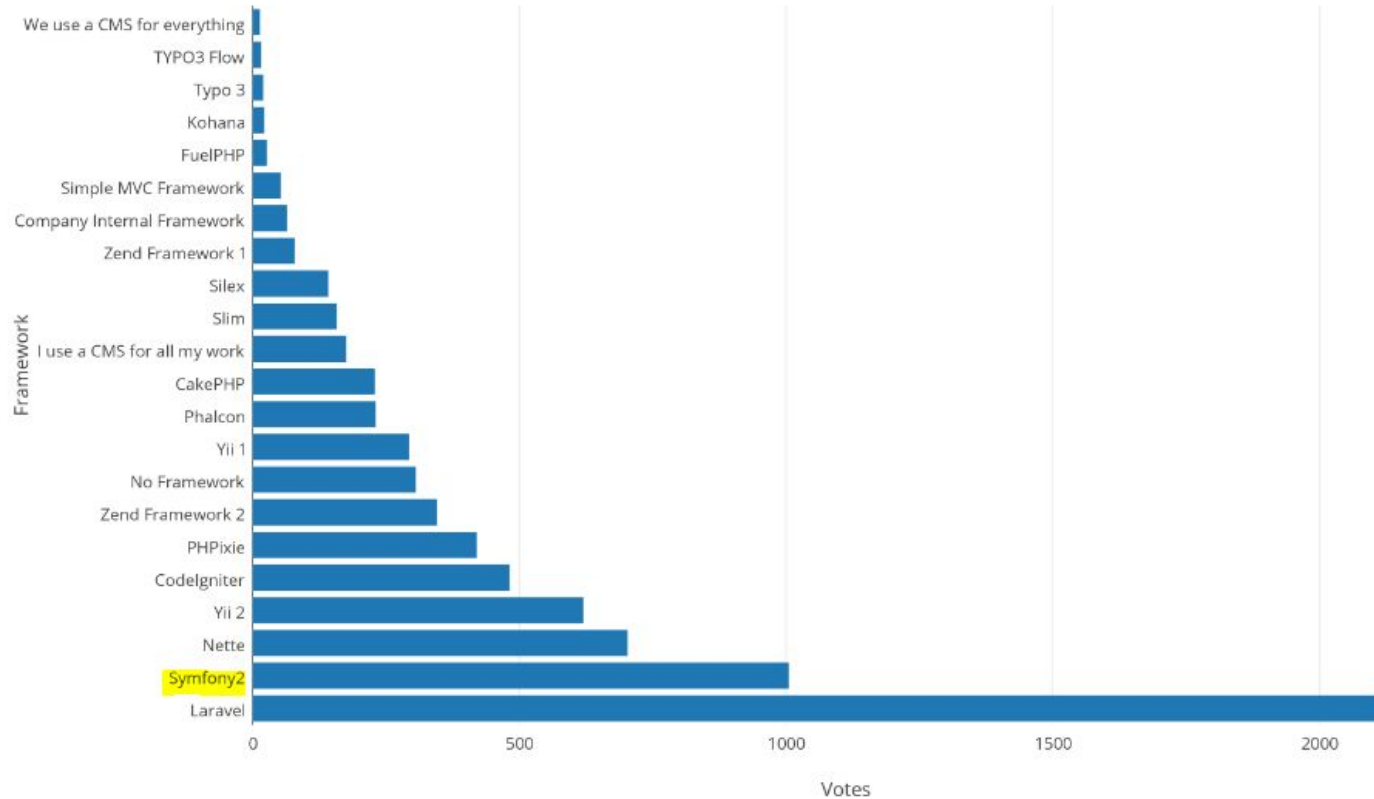
☐ News headlines ?

☐ Forecast ?



</>

PHP Framework Popularity in Personal Projects - SitePoint, 2015



Angular

<https://github.com/carlosvicient/angular101>

- ¿Qué es AngularJS?
 - ¿Qué no es AngularJS?
 - Arquitectura
 - Data binding
 - Instalación y ng-app
 - Expressions
 - Built-in directives
 - Filters
 - Modules
 - Controllers
 - Un poco de orden
-

Angular. ¿Qué es Angular?

- **Framework** basado en Javascript (client-side)
- Open-source (Google*)
- Enriquece HTML => Legibilidad y mantenimiento
- MVC (o MVVM: Model View ViewModel)
- Especialmente pensado para páginas dinámicas basadas en servicios REST-API
- Facilita desarrollo de SPAs (Single Page Applications)

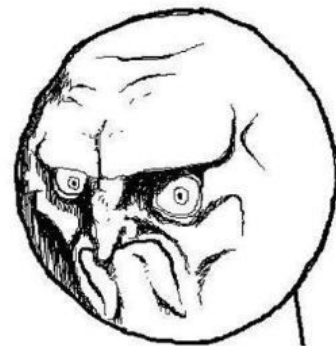
“A complete client-side solution”

“Angular is what HTML would have been, had it been designed for applications.”

What is Angular? <https://docs.angularjs.org/guide/introduction>

Angular. ¿Qué no es Angular?

- No es una librería JavaScript
- No es jQuery
- No es una plataforma (.Net, Java, ...)
- No es un lenguaje de programación
- No es un plugin una extensión del navegador
- No es *one-way data binding*



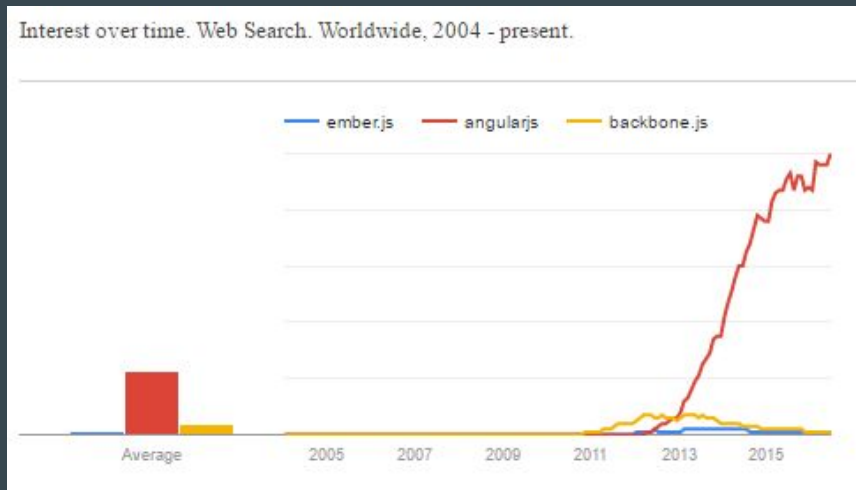
NO.

Angular. Backbone, Ember, Angular (I)

	Backbone	Ember	Angular
Github	https://github.com/jashkenas/backbone	https://github.com/emberjs/ember.js/	https://github.com/angular/angular.js
Dependencias	Underscore, jQuery	handlebars, jQuery	No tiene
Data binding	No (plugins)	Sí (getters and setters)	Sí (json)
Routers	Muy simple	Complejo de usar	Parecido a backbone (angular-route vs angular-ui-router)
Vistas	Fácil de usar (jQuery y DOM) Templating third-party	Fácil. Componentes reusables. Templating handlebars	Muy fácil. Basado en HTML
Testing	Third party (Jasmine, Sinon)	Al principio no. Ahora Qunit	Excelente (Karma)
Data	jQuery ajax	jQuery ajax	\$http, \$resource

Angular. Backbone, Ember, Angular (II)

	Backbone	Ember	Angular
Comunidad documentación	KO	ok	OK
Integración (third party)	OK	ok	ko*
Herramientas desarrollo	KO	ok	OK



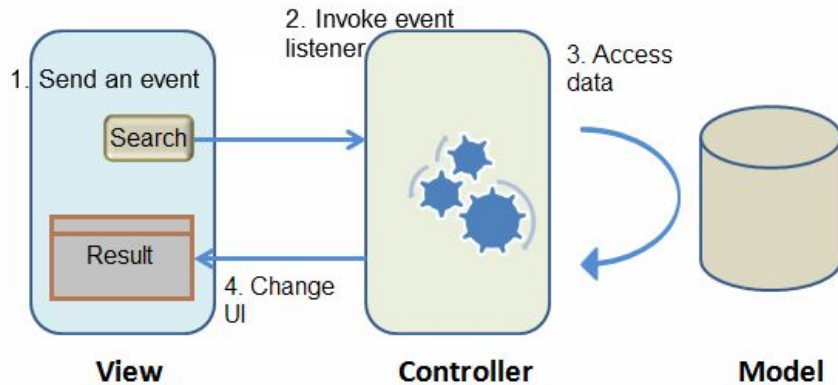
¿Qué pasa con ReactJS? Difícil de comparar. React es una librería para renderizar vistas.
AngularJS + ReactJS = Performance

* <http://www.slideshare.net/deepusnath/javascript-frameworks-comparison-angular-knockout-ember-and-backbone>
<http://www.devadictos.com/javascript-posts/angularjs-emberjs-backbonejs-fight/>

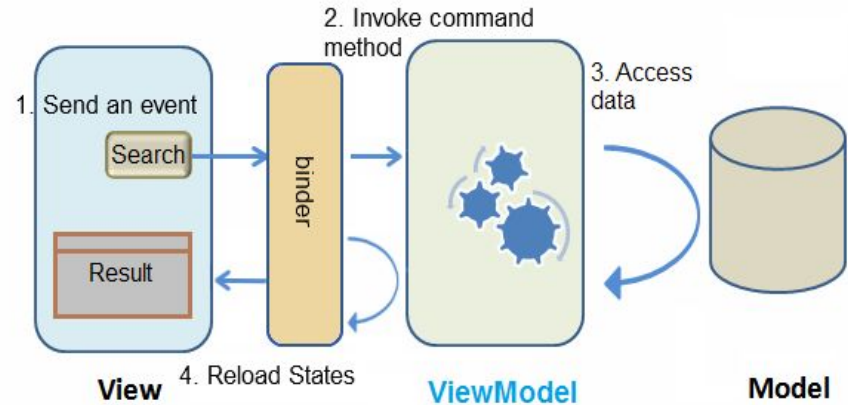
Angular. Patrones de arquitectura (Architectural pattern)

MVC? MVVM?

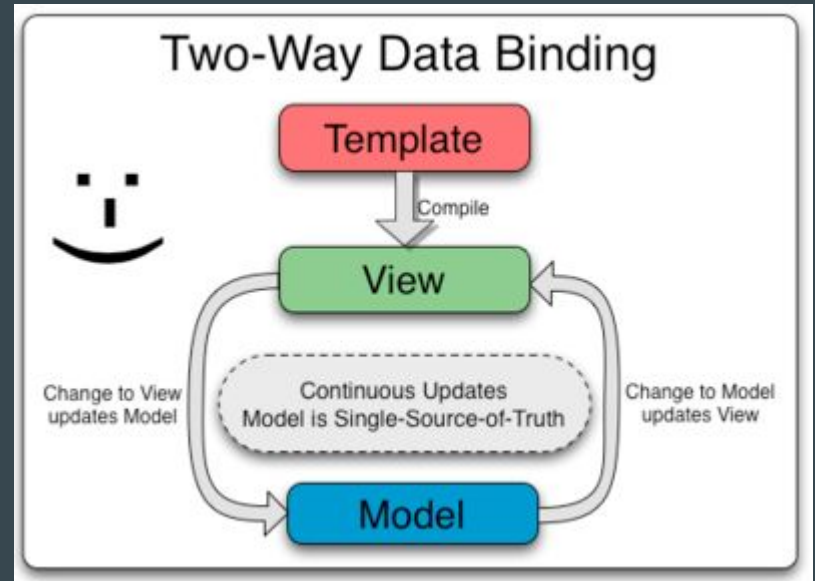
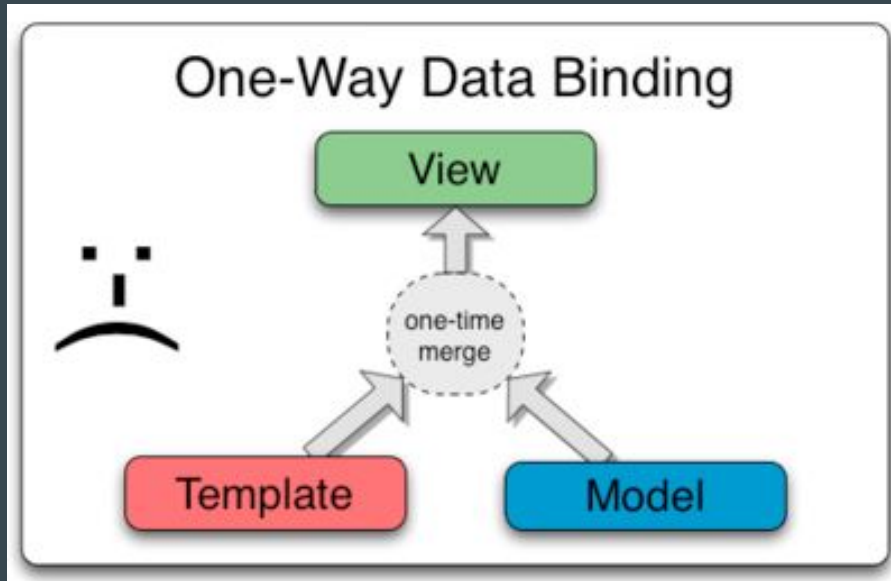
MVC



MVVM



Angular. Data binding



Angular. Instalación y ng-app (I)

- Descargar (<https://angularjs.org/>)
 - a. CDN: <https://ajax.googleapis.com/ajax/libs/angularjs/1.5.6/angular.min.js>
 - b. Bower: `bower install angular#1.5.6`
 - c. Npm: `npm install angular@1.5.6`
- Instalar dependencia en html

```
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.5.6/angular.min.js"></script>
```

Angular. Instalación y ng-app (II)

Directivas

- Son atributos que extiende a los ya existentes en HTML
- Empiezan por el prefijo “ng-”

ng-app

- Directiva usada para inicializar una nueva aplicación AngularJS
- Indica el “root” element de la aplicación
- Solo puede haber una en un html. Si hay más de una, se tiene en cuenta la primera.
- Recibe como parámetro opcional el nombre del módulo que debe cargar

Angular. Instalación y ng-app (III)

Ejemplo1

```
<!doctype html>
<html ng-app>
  <head>
    <script src="https://[...]"></script>
  </head>
  <body>
    <p>Esto es una aplicación Angular</p>
  </body>
</html>
```

Angular. Expresiones

- JavaScript-like code snippets (like!)
- Moustache-like template: `<p>1+2={{1+2}}</p>`

Ejemplo 2

```
<!-- operadores numéricos -->
<p>
  4 + 6 suman {{4+6}}
</p>
```

```
<!-- operadores String -->
<p>
  {{"Hola" + " mundo"}}
</p>
```



```
<!-- operadores numéricos -->
<p>
  4 + 6 suman 10
</p>
```

```
<!-- operadores String -->
<p>
  Hola mundo
</p>
```



Angular. Ejemplo 3

```
<!doctype html>
<html ng-app>
  <head>
    <script src="..."></script>
  </head>
  <body>
    <div>
      <label>Nombre:</label>
      <input type="text" ng-model="name"
placeholder="Escribe tu nombre..."
      <br>
      <h1>Hola {{name}}!</h1>
    </div>
  </body>
</html>
```

```
<!doctype html>
<html ng-app>
  <head>
    <script src="https:[...]"></script>
    <script type="text/javascript">
      $(function() {
        var name = $('#name');
        var welcome = $('#welcome');
        name.keyup((function () {
          welcome.text('Hello ' + name.val() + '!');
        }));
      })
    </script>
  </head>
  <body>
    <div>
      <label>Nombre:</label>
      <input id="name" type="text" placeholder="
Escribe tu nombre..."
      <br>
      <h1 id="welcome">Hello !</h1>
    </div>
  </body>
</html>
```

Angular. Built-in directives

Directivas propias de Angular.

Complementan los tags html existentes.

- ng-app
- ng-init
- ng-model
- ng-repeat
- ng-show/ng-hide
- ...

Angular. Built-in directives

Directivas propias de Angular.

Complementan los tags html existentes.

- ng-app
- **ng-init**
- ng-model
- ng-repeat
- ng-show/ng-hide
- ...

Se utiliza para evaluar una expresión en el “scope” actual o inicializa datos de la aplicación (!!!)

Ejemplo 2

```
<p ng-init="a=4;b=2;">
    {{a}} + {{b}} = {{a+b}}
</p>
```

Angular. Built-in directives

Directivas propias de Angular.

Complementan los tags html existentes.

- ng-app
- ng-init
- **ng-model**
- ng-repeat
- ng-show/ng-hide
- ...

Enlaza (binding) las vistas y el controlador mediante el scope

Ejemplo 4

```
<div ng-init="name='CARLOS'">
  <label>Nombre:</label>
  <input type="text" ng-model="name"
placeholder="Escribe tu nombre...">
  <br>
  <h1>Hola {{name}}!</h1>
</div>
```

Angular. Built-in directives

Directivas propias de Angular.

Complementan los tags html existentes.

- ng-app
- ng-init
- ng-model
- **ng-repeat**
- ng-show/ng-hide
- ...

Instancia un “template” por cada “documento” de una “colección”. Cada elemento != scope

Ejemplo 5

```
<div ng-init="hotels=[{name:'Winterfell Palace', rating:'5'}, {name:'Hostal the Wall', rating:'2'}, {name:'Volantis Plaza', rating:'4'}]">
  <ul>
    <li ng-repeat="hotel in hotels">
      <span>{{hotel.name}}</span>
      <span>{{hotel.rating}}</span>
    </li>
  </ul>
</div>
```

Angular. Built-in directives

Directivas propias de Angular.

Complementan los tags html existentes.

- ng-app
- ng-init
- ng-model
- ng-repeat
- **ng-show/ng-hide**
- ...

Muestra u oculta un elemento html

Ejemplo 6.1 y 6.2

```
<ul>
  <li ng-repeat="hotel in hotels"
      ng-show="hotel.rating>3">
    <span>{{hotel.name}}</span>
    <span>{{hotel.rating}}</span>
  </li>
</ul>
```


Angular. Filtros

- Se utilizan para transformar datos
- Pueden añadirse utilizando ‘|’ después de una expresión
 - `{{ datos | filtro:opciones* }}`
- “Built-in Filters”
 - currency
 - date
 - filter
 - json
 - limitTo
 - lowercase
 - number
 - orderBy
 - uppercase

Moneda

```
{{ number | currency : symbol : fractionsize }}
```

Data

```
{{ date | date : format : timezone }}
```

Minúsculas

```
{{ string | lowercase }}
```

Números

```
{{ string | number : fractionsize }}
```

Ordenar

```
{{ array | orderBy : expression : reverse }}
```

Filter

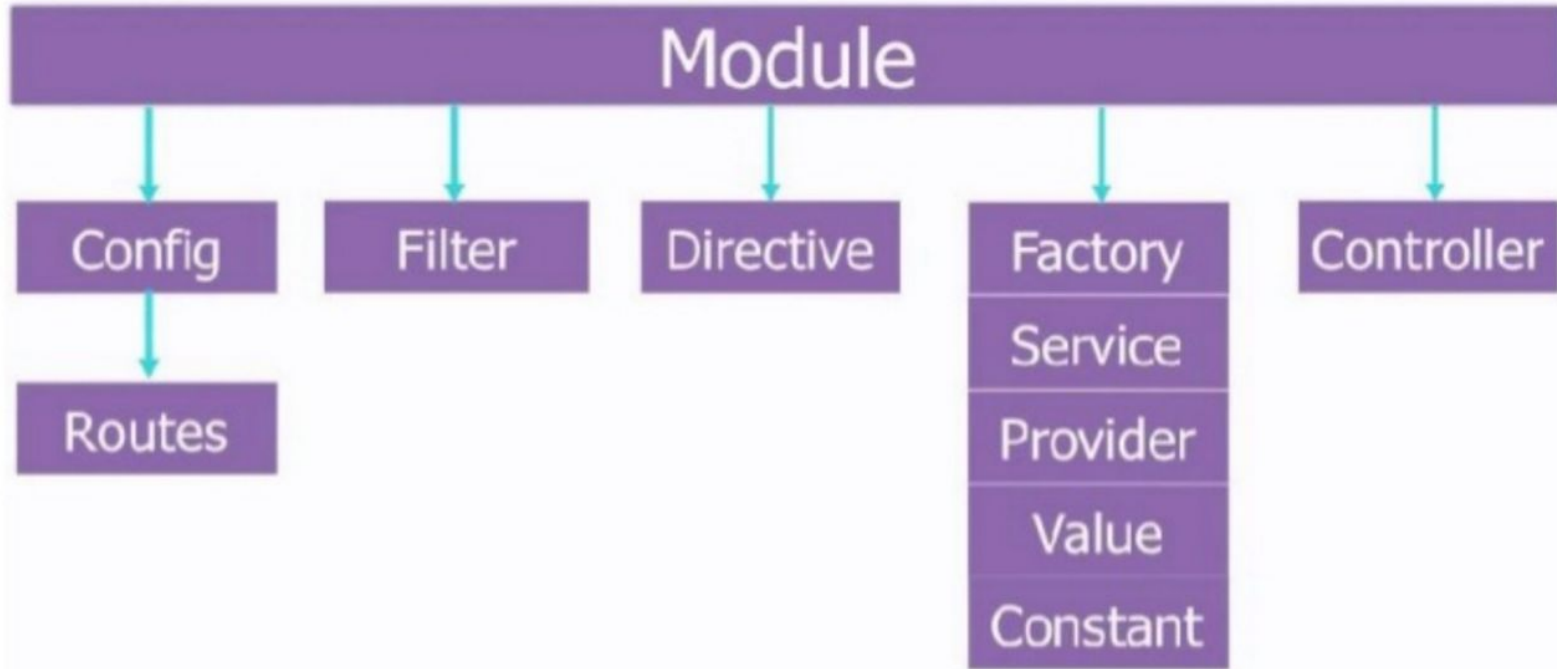
```
{{arrayexpression | filter : expression:comparator}}
```

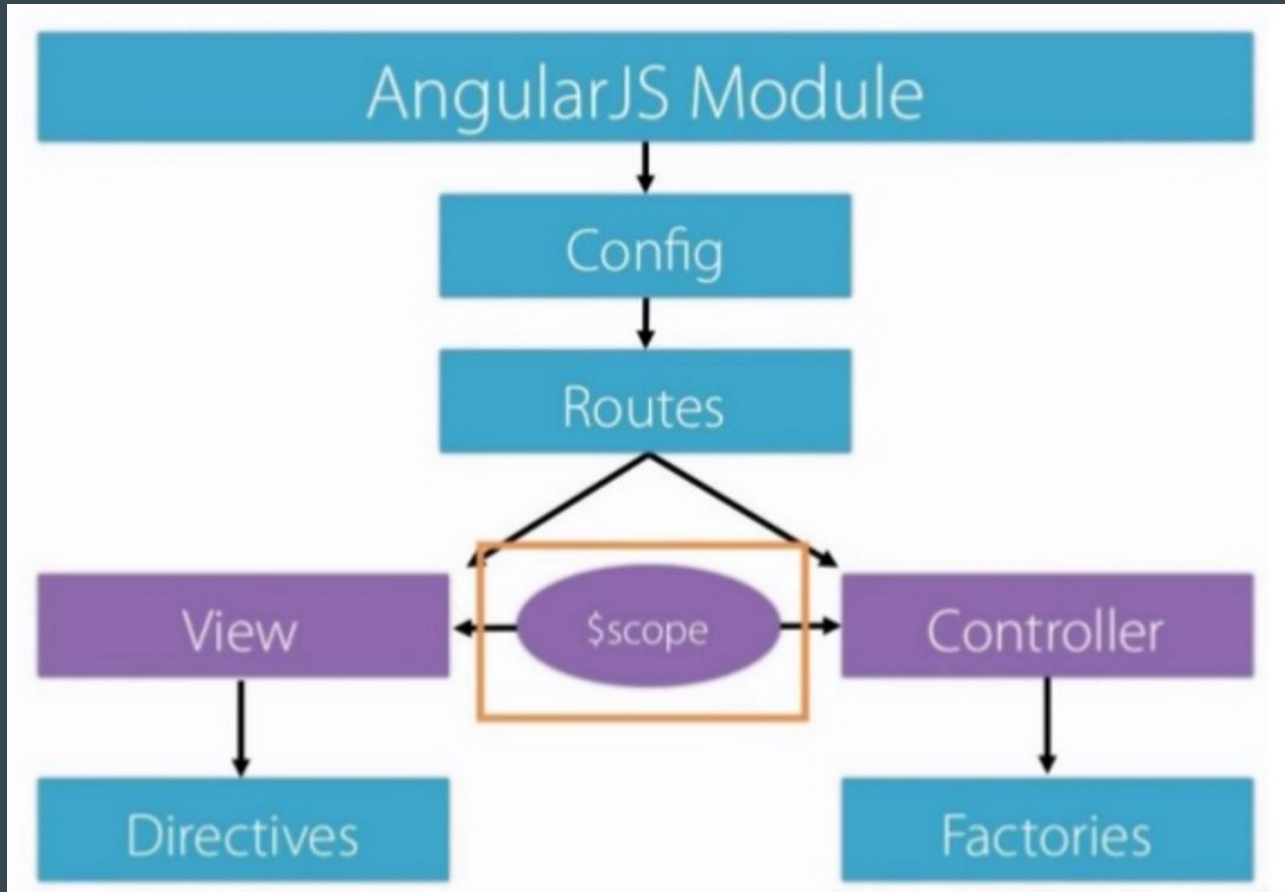
*Ejemplo 7

Angular. Módulos (I)

- Forma de organizar aplicaciones
- Cada módulo puede contener diferentes partes de la aplicación
 - Controladores
 - Servicios
 - Filtros
 - Directivas
 - etc.
- Sitio donde se declaran las dependencias
- Encapsula y aísla funcionalidades en un único paquete reutilizable en otras aplicaciones
- Mantenimiento, legibilidad, testabilidad

Modules are Containers for AngularJS Components






Angular. Módulos (II)

Declarar un módulo

```
{function() {  
  'use strict';  
  
  angular.module('app', []);  
  
}}();
```



AngularJS

Nombre
del módulo

Dependencias

Utilizar un módulo (JS)

```
angular.module('app');
```



AngularJS

Nombre
del módulo
que vamos a
usar

Angular. Módulos (III)

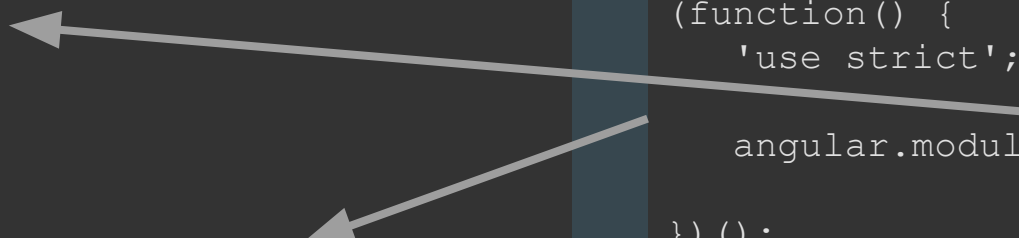
Utilizar módulo en HTML (ng-app) (Ejemplo 8)

```
<!doctype html>
<html ng-app="app">
  <head>
    [...]
  </head>
  <body>
    [...]
    <script src="app.module.js"></script>
  </body>
</html>
```

```
/*app.module.js*/
(function() {
  'use strict';

  angular.module('app');

})();
```



Angular. Controladores (I)

- Componentes que se encargan de gestionar la lógica de una vista
- Se definen funciones y valores
- A los controladores se le pueden inyectar dependencias

Crear un controlador (JS)

```
angular
  .module('app')
  .controller('AppController', AppController);
```

Nombre
del módulo al
que inyectamos
el controlador

Nombre del controlador

Función con el código del controlador

Angular. Controladores (II)

Indicar el controlador que usamos en las vistas (HTML)

```
<!doctype html>
<html ng-app="app">
  <head>
    [...]
  </head>
  <body ng-controller="AppController as appCtrl">
    [...]
    <input type="text" ng-model="appCtrl.search" placeholder="Buscar...">

    <script src="app.module.js"></script>
    <script src="app.controller.js"></script>
  </body>
</html>
```


Angular. Un poco de orden

- Index.html -> contiene las vistas
 - Indicamos que el elemento HTML es una aplicación angular y que carga el módulo “app”
 - Importamos AngularJS
 - Importamos estilos
 - Declaramos que todo el body se gestionará por el controlador “AppController” al que llamaremos “appCtrl”
 - Mostramos datos del controlador
 - Importamos nuestros ficheros .js **al final del body(*)**
- app.module.js -> creamos el módulo “app”
- app.controller.js -> creamos el controlador del módulo “app”
- app.css -> estilos css

Veamos el ejemplo 9

(*) [Yahoo's Best Practices for Speeding Up Your Web Site: Put Scripts at the Bottom.](#)

Estructura y herramientas

- Estructura por componentes
- Estructura por funcionalidad
- Herramientas
 - bower
 - npm
 - gulp
 - ...

Estructura y herramientas. Estructura por componentes

- Estructuramos el código agrupando componentes de un mismo tipo en un mismo directorio
- Recuerda un MVC
- Fácil de entender
- Proyectos pequeños
- Poco escalable (>10 controladores)
- Poco reutilizable

Angular dispone de un proyecto de ejemplo que sigue un modelo parecido a este: <https://github.com/angular/angular-seed>

```
app/  
----- controllers/  
----- mainController.js  
----- otherController.js  
----- directives/  
----- mainDirective.js  
----- otherDirective.js  
----- services/  
----- userService.js  
----- itemService.js  
----- js/  
----- bootstrap.js  
----- jquery.js  
----- app.js  
views/  
----- mainView.html  
----- otherView.html  
----- index.html
```

Estructura y herramientas. Estructura por funcionalidad

- Estructuramos el código agrupando todos los componentes relacionados con una misma funcionalidad en un mismo directorio
- Más difícil de leer al principio
- Proyectos grandes
- Más fácil de mantener y escalar
- Más reutilizable

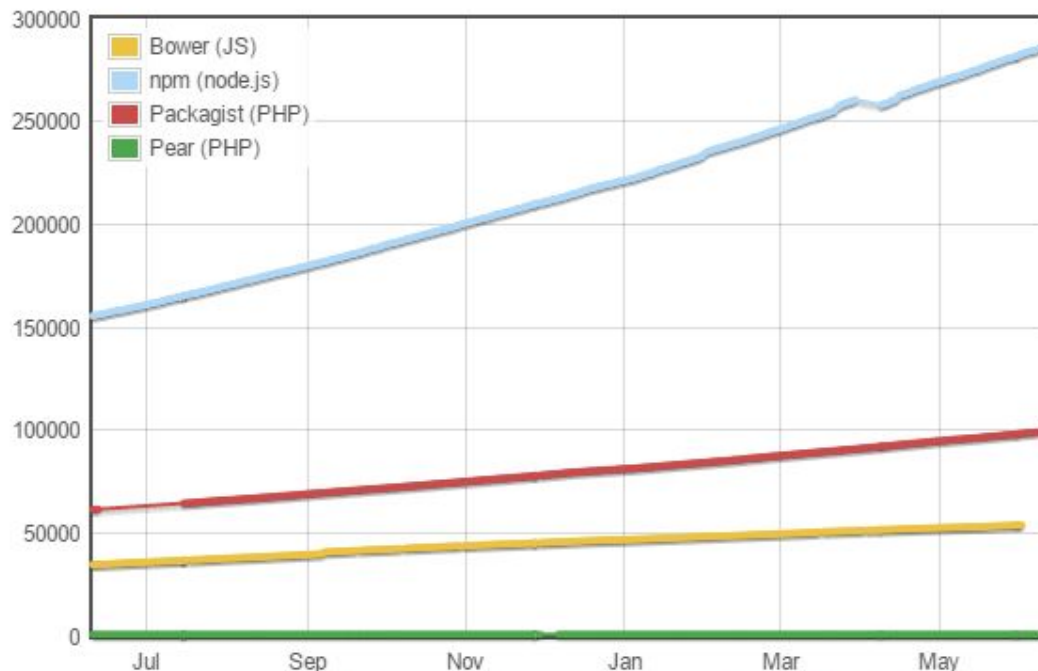
John Pappas propone un generador de proyectos que sigue un modelo similar a este: <https://github.com/johnpapa/generator-hottowel>

```
app/
----- shared/    // componentes reutilizables
----- sidebar/
----- sidebarDirective.js
----- sidebarView.html
----- article/
----- articleDirective.js
----- articleView.html
----- components//cada componente pequeña app. angular
----- home/
----- homeController.js
----- homeService.js
----- homeView.html
----- blog/
----- blogController.js
----- blogService.js
----- blogView.html
----- app.module.js
----- app.routes.js
assets/
----- img/
----- css/
----- js/ //JavaScript que no sean Angular
----- libs/ //Librerías de terceros (jQuery, etc.)
index.html
```

Estructura y herramientas. Herramientas

- Implementar proyectos grandes implica realizar muchas acciones repetitivas durante la fase de desarrollo.
- Existen herramientas para agilizar nuestros flujos de trabajo y automatizar procesos
- NPM (node.js)
 - Es un gestor de paquetes javascript.
 - Pensado para gestionar paquetes utilizados en node (backend). npm+browserify para usar front-end
 - package.json
- Bower
 - Igual que NPM pero orientado a front-end
- Gulp
 - Herramienta para automatizar tareas (minify, uglify, browser reload, build, serve, etc.)

*Ver gráfico y ejemplo 10



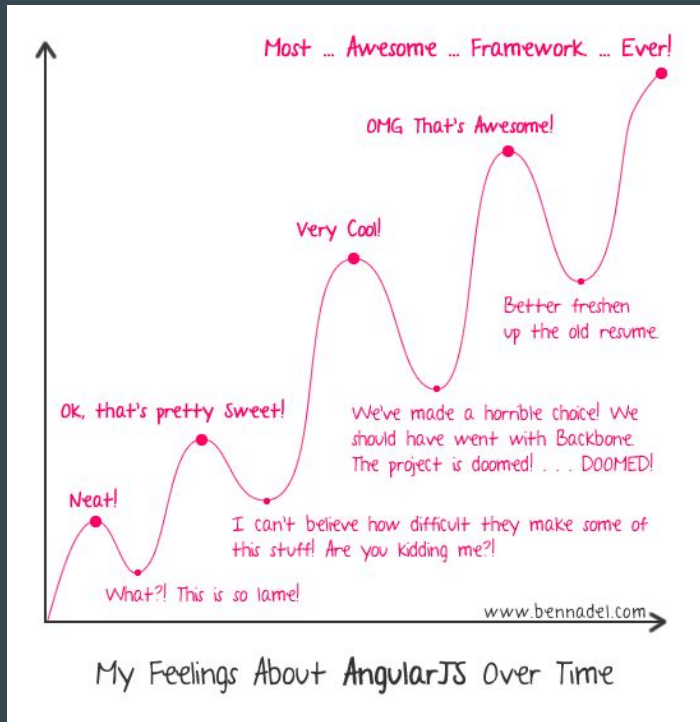
Include

- ☒ Bower (JS)
- ☒ npm (node.js)
- ☒ Packagist (PHP)
- ☒ Pear (PHP)

	Jun 3	Jun 4	Jun 5	Jun 6	Jun 7	Jun 8	Jun 9	Avg Growth
<u>Bower (JS)</u>								37/day
<u>npm (node.js)</u>	283657	283973	284256	284552	284997	285462	285893	373/day
<u>Packagist (PHP)</u>	98609	98676	98772	98885	98984	99103	99223	102/day
<u>Pear (PHP)</u>	602	602	602	602	602	602	602	0/day
<u>Rubygems.org</u>	119126	119158	119180	119221	119280	119344	119392	44/day

Conclusiones

- Ideal para aplicaciones basadas en REST API
- Ideal para SPAs
- Sistema de data binding muy completo y potente
- Ayuda a estructurar proyectos javascript
- Modular (inyección de dependencias)
- HTML extensible (directivas)
- Re-aprovechable
- “Testeable”
- Creciente demanda de AngularJS
- Gran comunidad
- Respaldado por Google
- Curva de aprendizaje lenta
- Rendimiento(*)



(*) <http://www.bennadel.com/blog/2439-my-experience-with-angularjs---the-super-heroic-javascript-mvw-framework.htm>

Enlaces de interés

- Enlaces oficiales
 - <https://angularjs.org/>
 - <https://docs.angularjs.org/guide>
 - <https://docs.angularjs.org/api>
 - <https://docs.angularjs.org/tutorial>
- Cursos online
 - <https://www.codeschool.com/courses/shaping-up-with-angular-js>
 - <https://www.codeschool.com/courses/staying-sharp-with-angular-js>
- Manuales/tutoriales
 - <http://www.desarrolloweb.com/manuales/manual-angularjs.html>
 - <http://www.w3schools.com/angular/default.asp>
- Buenas prácticas
 - <https://github.com/johnpapa/angular-styleguide>
 - <https://scotch.io/tutorials/angularjs-best-practices-directory-structure>

AngularJS

...

¿Preguntas?