¿...Y SI VAMOS UN PASO ADELANTE USANDO **ANGULARJS** + **ES2015**?

Victor Cataño

UN POCO DE HISTORIA

- Fue desarrollado en Mayo de 1995 bajo el nombre **Mocha** por Brendan Eich.
- En Septiembre fue renombrado como **LiveScript.**
- En Diciembre del mismo año fue renombrado a **JavaScript.**

UN POCO DE HISTORIA

- En 1996 es llevado a la estandarización ECMA. así que ECMA en ese momento es la especificación y Javascript es una implementación.
- 1997 ECMA-262 (ECMAScript)
- 1998 ECMAScript 2
- 1999 ECMAScript 3

UN POCO DE HISTORIA

- En 2005 Mozilla y Macromedia comienzan a trabajar en ECMAScript 4, rico en nuevas funcionalidades, es un cambio importante desde ECMAScript 3.
- este nuevo estándar tiene opositores, y es descartado.

- El arreglo entre las partes da la llegada al estándar ECMAScript 3.1

UN POCO DE HISTORIA

- En 2009 las partes opuestas, después de reunirse en Oslo, aclaran ambigüedades de la version 3, se define el modo estricto del lenguaje, entre otras nuevas características. y lanzan el estándar ECMAScript 5.

UN POCO DE HISTORIA

- En 2015 llega ECMAScript 6, conocido también como ECMAScript Harmony, con cambios significativos en la sintaxis para escribir aplicaciones complejas, incluyendo clases y módulos en otras nuevas características.
- los nombres de las nuevas versiones serán basadas en los años de lanzamiento, así que ES6 es ES2015 y ES7 deberá ser ES2016.

ALGUNAS NOVEDADES

HERENCIA ES5

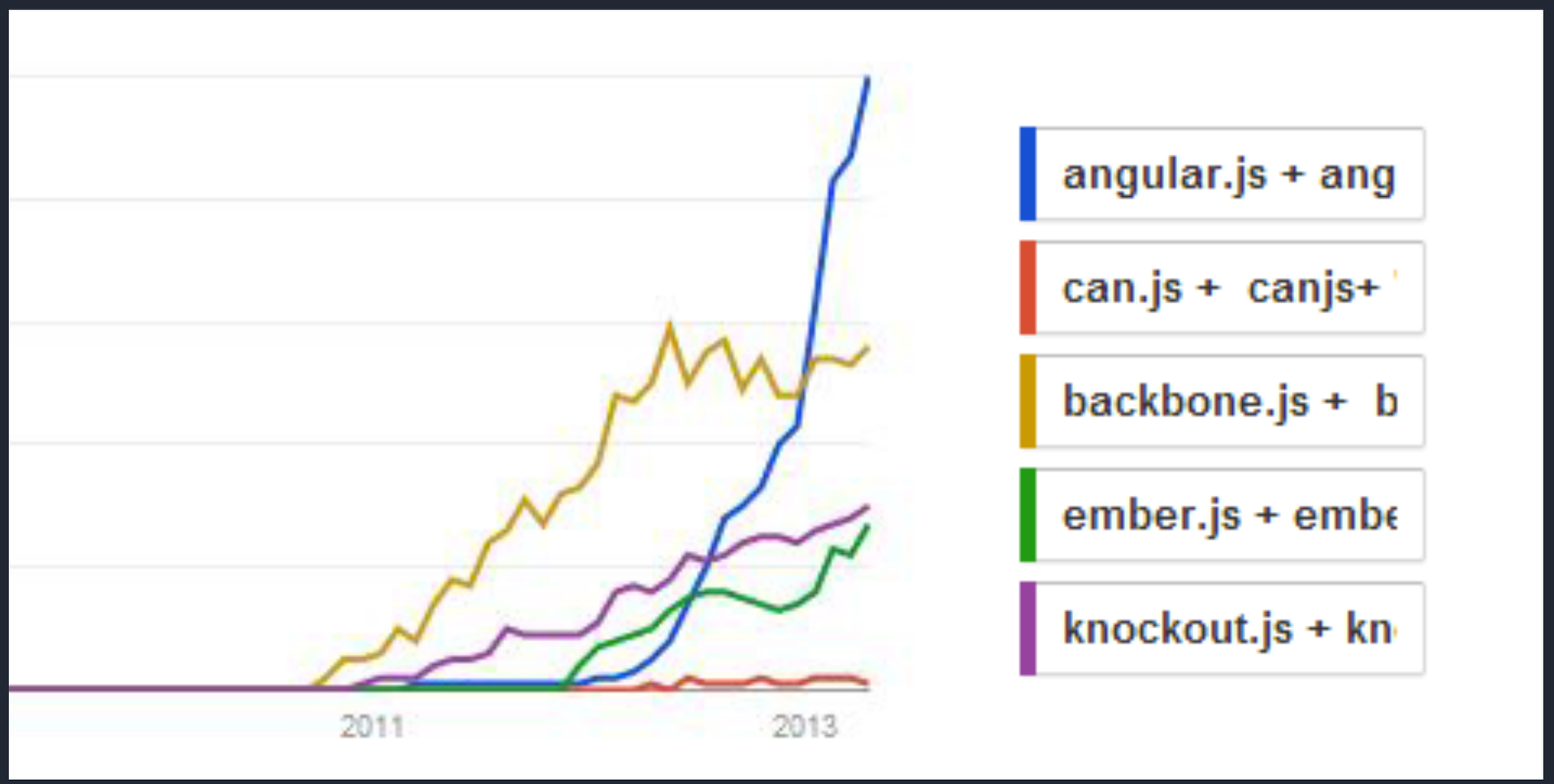
```
//Vehicle
function Vehicle (name) {
    this._name = name;
}

Object.defineProperty(Vehicle.prototype, 'name', {
    get: function () { return this._name; },
    set: function (value) { this._name = value }
});


// Car
function Car (name) {
    Vehicle.call(this, name);
}

Car.prototype = Object.create(Vehicle.prototype);
Car.prototype.constructor = Car;

Car.prototype.move = function () {
    console.log(this.name + ' is spinning wheels...');
}
```

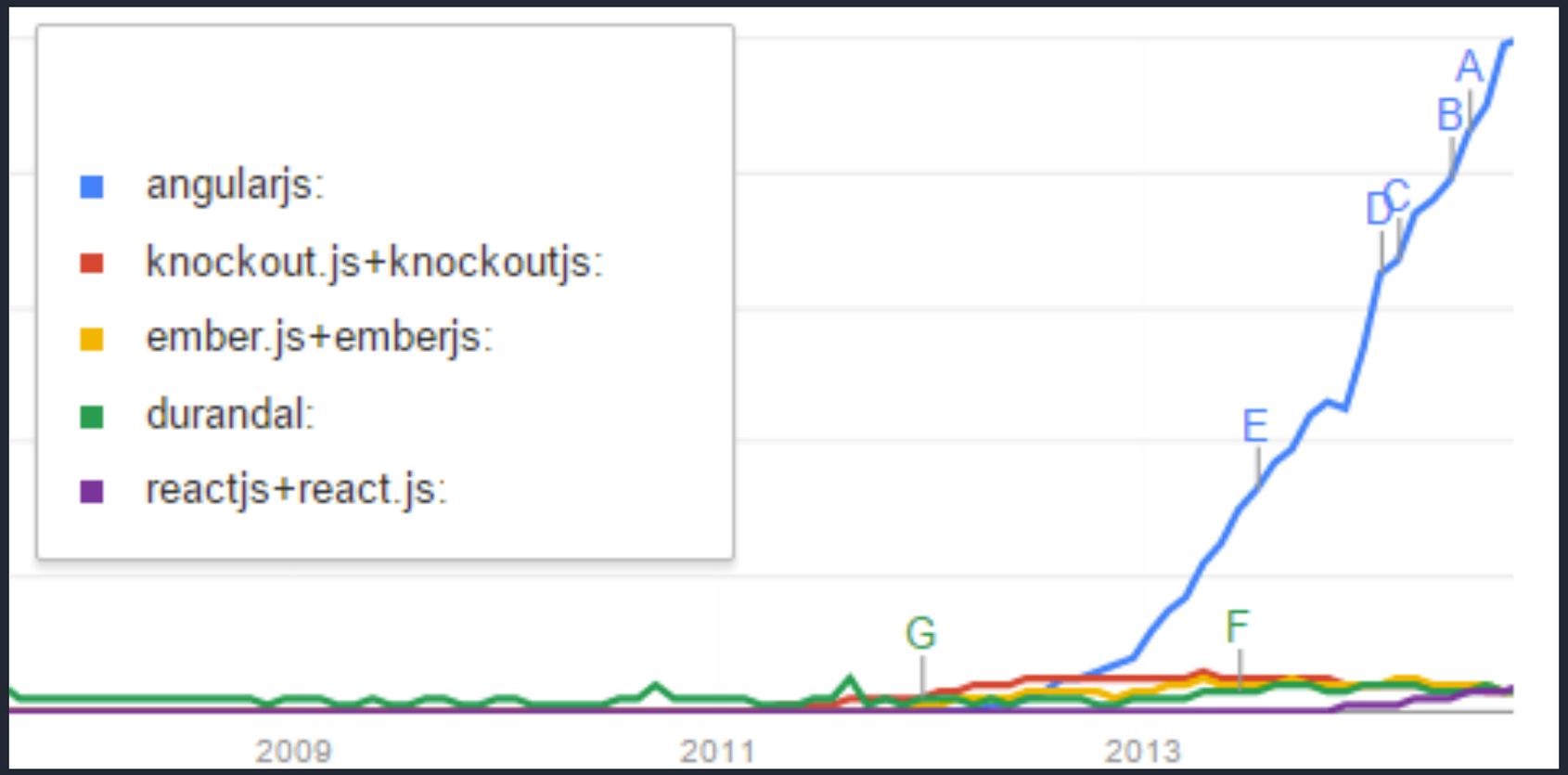
HERENCIA ES6

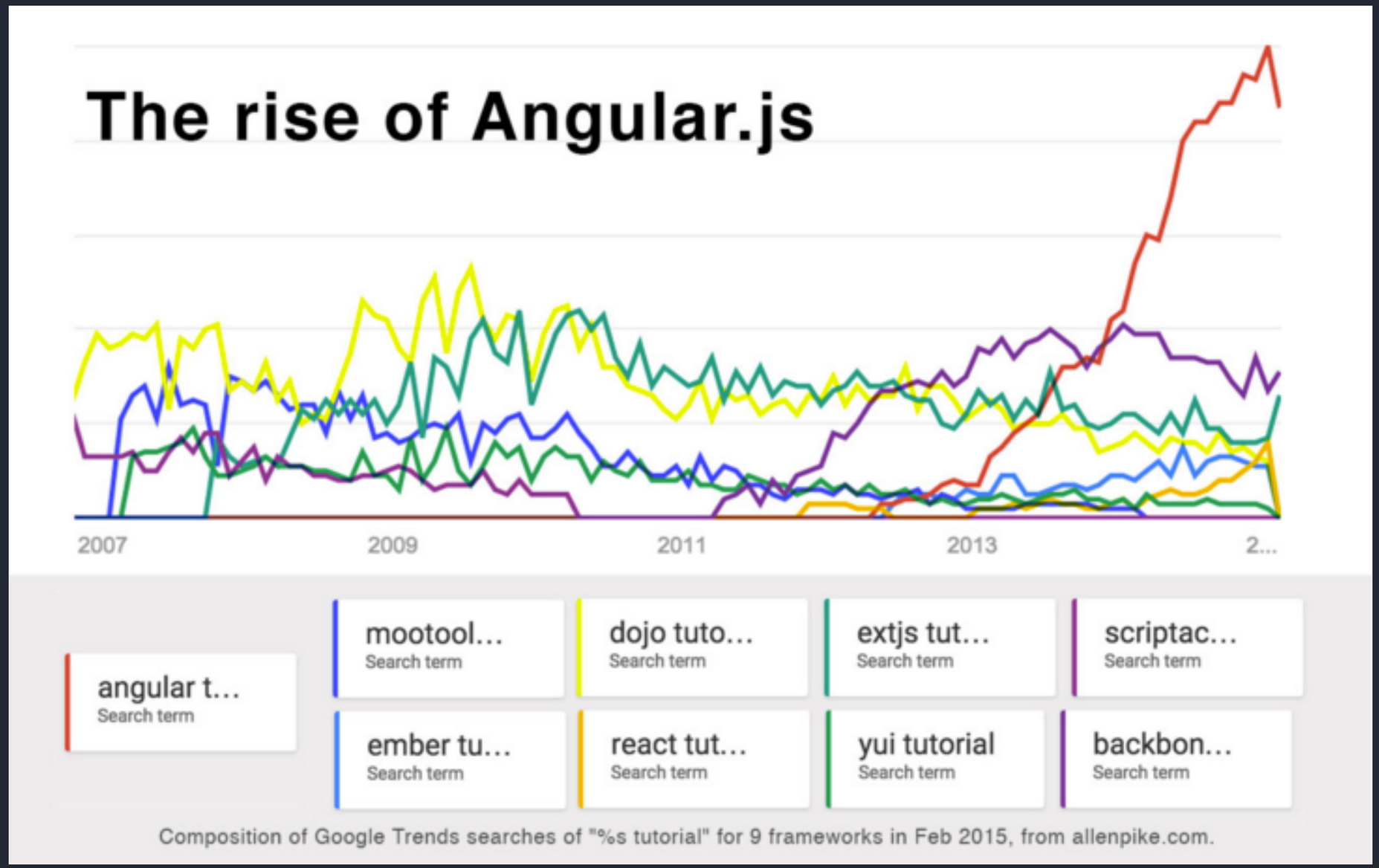
```
class Vehicle {

    constructor (name) {
        this._name = name;
    }

    get name () {
        return this._name;
    }
}
export { Vehicle }
```

```
import { Vehicle } from './Vehicle';

class Car extends Vehicle {

    move () {
        console.log(this.name + ' is spinning wheels...')
    }
}
export { Car }
```

¿POR QUÉ ANGULARJS?


angular.js + ang
can.js + canjs+
backbone.js + b
ember.js + embe
knockout.js + kn
2011
2013


angularjs:
knockout.js+knockoutjs:
ember.js+emberjs:
durandal:
reactjs+react.js:
A
B
C
D
E
F
G
2009
2011
2013



Composition of Google Trends searches of "%s tutorial" for 9 frameworks in Feb 2015, from allenpike.com.

¿POR QUE COMENZAR A IMPLEMENTAR ES6 CON ANGULAR?

- ES6 ya esta en su version final (ya están trabajando en ES7).
- AngularJS 1.x seguirá usándose mucho más tiempo.
- Las herramientas de compilación lo hacen fácil.
- Su definición se ajusta muy bien a AngularJS.
- Las características de ES6 ayudan a desarrollar en menor tiempo.
- Ayudar a propagar el nuevo estándar.

¿QUE NECESITAMOS PARA COMENZAR?

TRANSPILERS

Nos ayudan a compilar nuestro código ES6 a ES5
interpretable en todos los navegadores modernos

- Babel
- Traceur

¿PERO COMO ES UN CONTROLADOR EN ANGULAR?

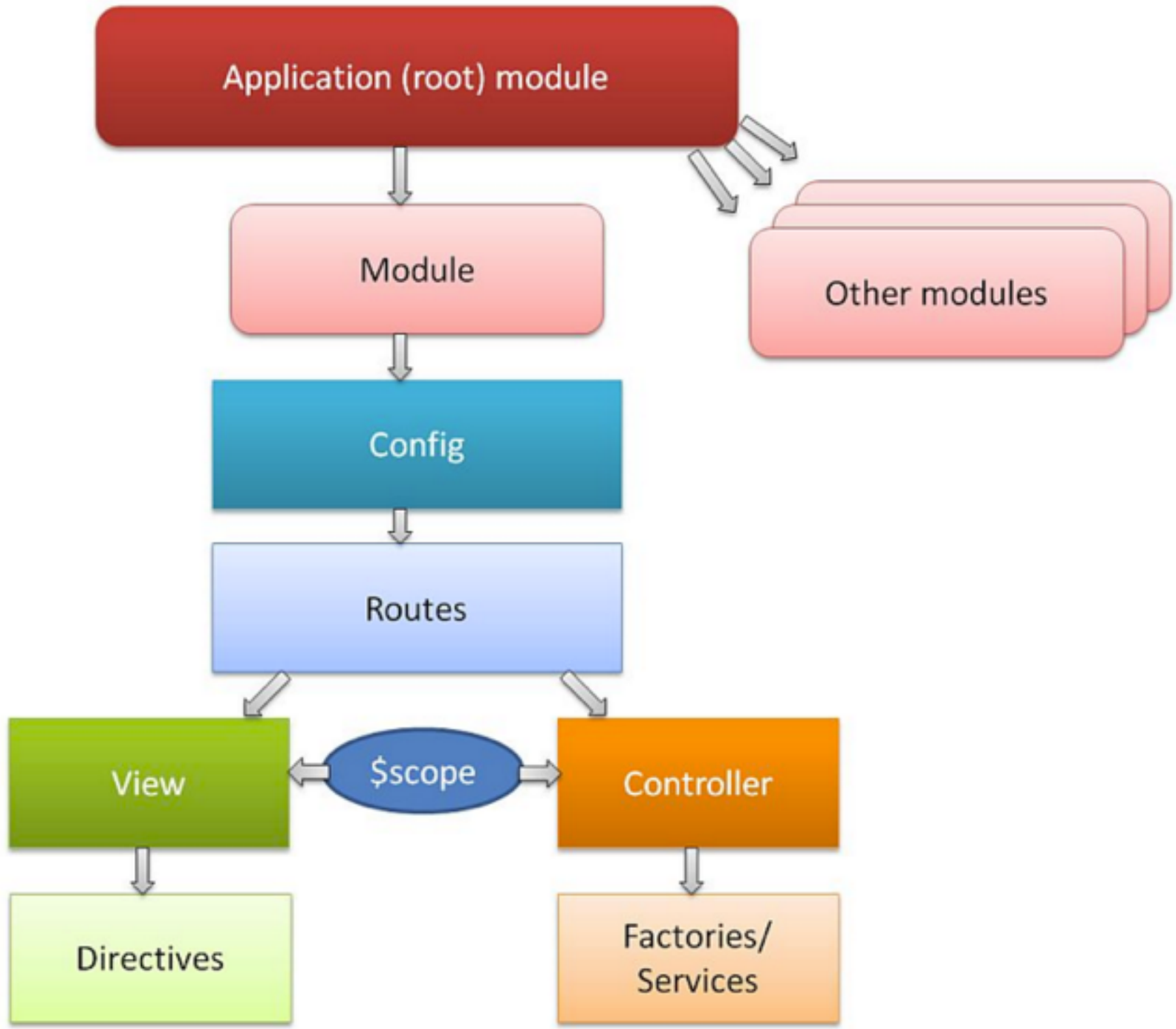
CONTROLADOR ES6

```
class MainController {

    constructor(searchService) {
        this.searchService = searchService;
    }

    search () {
        this.searchService
            .fetch(this.searchTerm)
            .then(response => {
                this.items = response.data.items;
            });
    }
}
export { MainController }
```

IMPORTANTE REFERENCIAR LOS NUEVOS MODULOS

```
import { MainController } from './MainController';
import { SearchService } from './SearchService';

angular
    .module('app', [])
    .controller('mainController', MainController)
    .service('searchService', SearchService);
```

PARA NO PERDER DE VISTA



Arquitectura base de una aplicación Angular