Enhance Web Development

Angular 2

L'écosystème Web

- Frontend spectrum
- Les modules côté client
- Webpack
- ES6
- TypeScript
- Typings
- Angular 2 avec ES6 & ES5

THE FRONT-END SPECTRUM

Package Managers

Front-End Performance

JS Frameworks

JS Preprocessors

Process Automation

Code Quality

Build Tools

Testing

Back-End

Core

Text Editors

UI Frameworks

Responsible Web Design

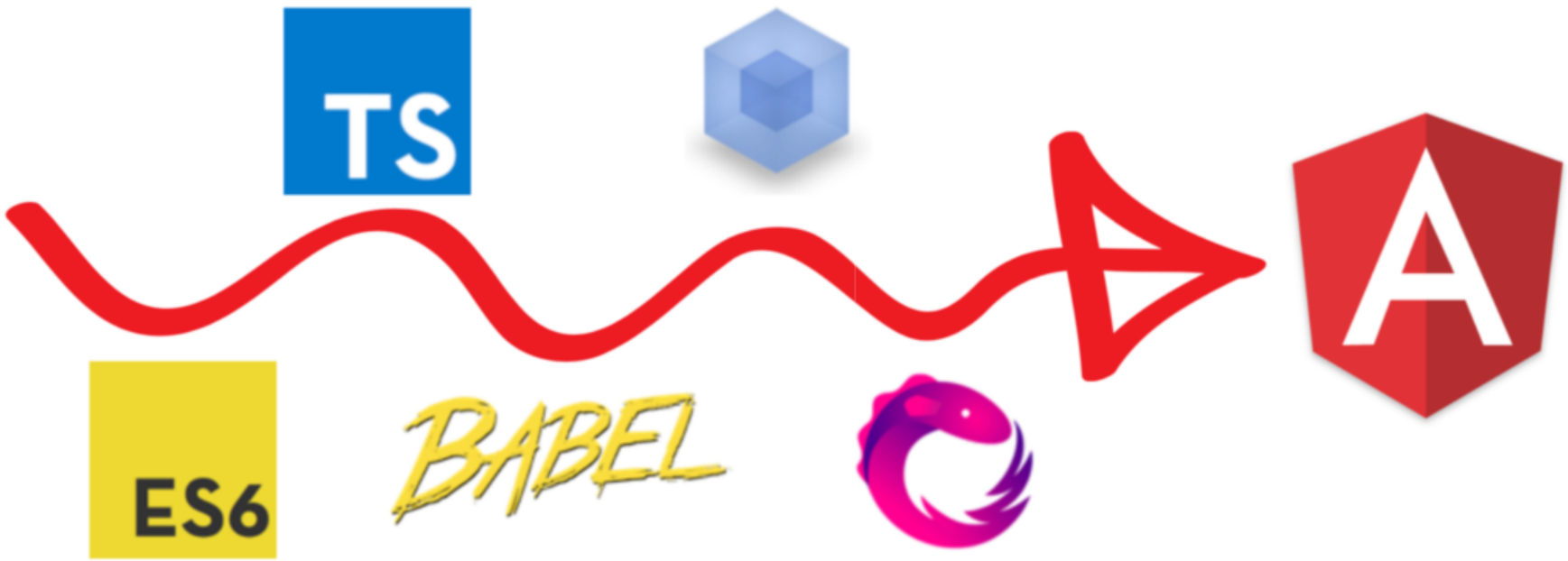
Templating

Browser Refreshing

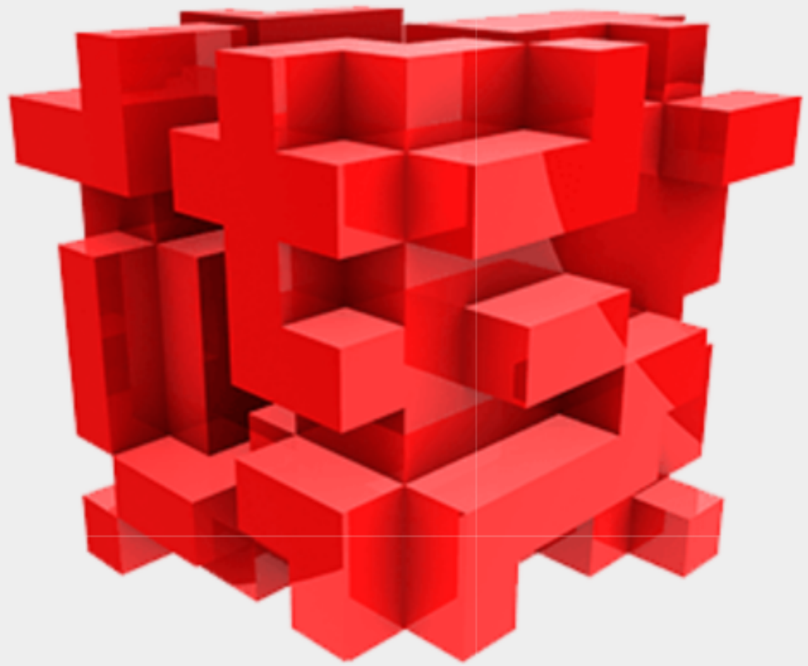
CSS Preprocessors

OOCSS & Style Guides

Version Control


TS
ES6
BABEL

Les modules côté client



Module Loading

- Non-obligatoire mais fortement conseillé !
- Organisation clair et précise
- Evite les conflits de namespace
- Evite d'insérer une balise <script src...> pour chaque inclusion
- Les modules ne sont pas supports pas les navigateurs donc il faut transpiler le module en pseudo module (wrapped function)

Les outils

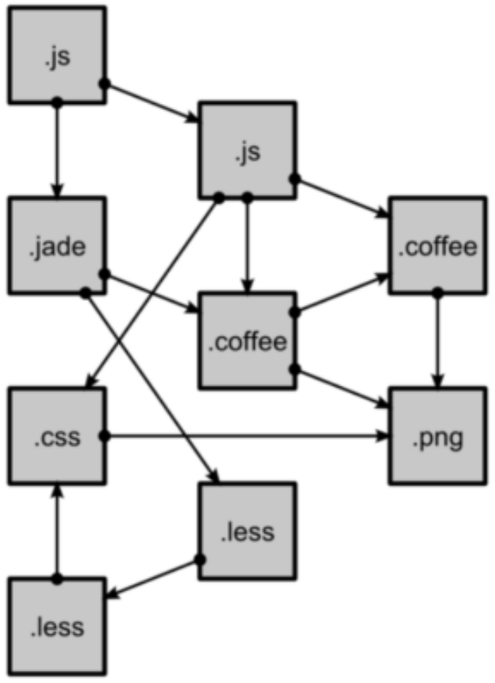
SystemJS

webpack

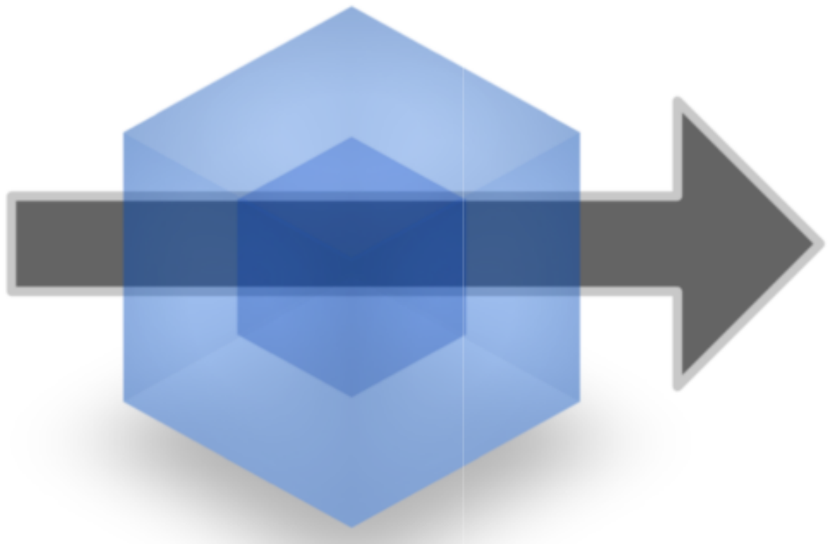
MODULE BUNDLER

Webpack

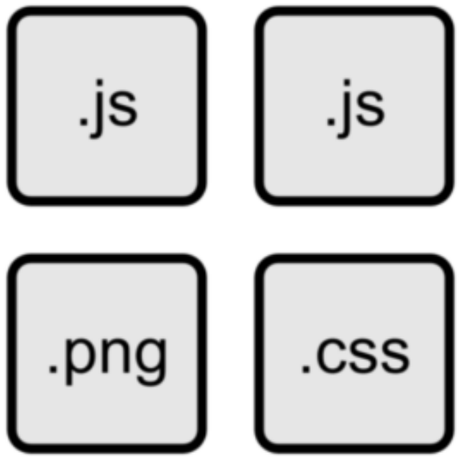
- Un des plus connus des modules loader
- Permet d'inclure n'importe quel type de fichier
- Très complet, peut remplacer gulp, grunt pour tout les scripts de build.



modules
with dependencies



webpack
MODULE BUNDLER



static
assets


ES6

ES6

- ES6/ES2015 est le dernier standard pour Javascript
- Beaucoup de nouvelle fonctionnalité comme :

  module system, new array methods, **classes**, multi-line templates, **arrow functions**, spread operator, et bien plus !

```
class Point {
        constructor(x = 0, y = 0) {
            this.x = x;
            this.y = y;
        }

        toString() {
                return `(${this.x}, ${this.y})`;
        }
}
class NamedPoint extends Point {
        constructor(name = "defaultName", x = 0, y = 0) {
                super(x, y)
                this.name = name;
        }

        toString() {
                return `NamedPoint: ${this.name} ${super.toString()}`
        }
}
```

Classes & Inheritance

```
let nums = [1,2,3,4,5,6,7];
let odds = nums.filter(i => i%2);
 // [1,3,5,7]
let evens = nums.filter(i => i%2 === 0);
 // [2,4,6]
```

```
let forEachAdd = (arr, add) => arr.map(i => i+add)
forEachAdd(num, 2) // [3,4,5,6,7,8,9]

forEachAdd(num, 4) // [5,6,7,8,9,10,11]
```

Arrow functions

Et bien plus encore

- es6-features.org
- Exploring ES6 de Axel Rauschmayer
- Blog Axel Rauschmayer

TypeScript

- TypeScrit est construit au desus de ES6
- Language compilé => erreur avant le runtime
- Inclut les dernières versions d'ES6 (contrairement au navigateurs)
- @<Decorateur ou Annotations>
- Typage statique

```
interface IPoint {
        x: number;
        y: number
}
class Point implements IPoint {
        constructor(
                public x: number = 0,
                public y: number = 0
        ) {}

        toString() {
                return `(${this.x}, ${this.y})`;
        }
}
class NamedPoint extends Point {
        constructor(
                public name: string = "defaultName",
                public x: number = 0,
                public y: number = 0
        ) {
                super(x, y)
        }

        toString() {
                return `NamedPoint: ${this.name} ${super.toString()}`
        }
`
```

Classes, Inheritance & Static types

S'amuser en ligne

- TypeScript playground
- typescript.org

Typings

The TypeScript Definition Manager

Typings

- Typings est un paquet NPM pour gérer les définitions de types pour les 3rd party
- Installation : *npm i --save-dev typings* ou en global
- Installation d'une définition de typage :

  typings install -f <package> --ambient --save
- Fichier de configuration : *typings.json*
- Github : https://github.com/typings/typings

Angular 1

- Première version : 2009
- Créer par Miško Hevery
- 1.3m développeur utilise la version 1.X

Angular 2

- Annoncée en 2014
- RC disponible en mai 2016
- Actuellement : 2.0.0-rc2
- Déjà 360k développeurs
- Utilisation des bonnes pratique AngularJS 1.X

Ok et ? sinon pouquoi Angular 2 ?


ES6

Angular 2 + ES6

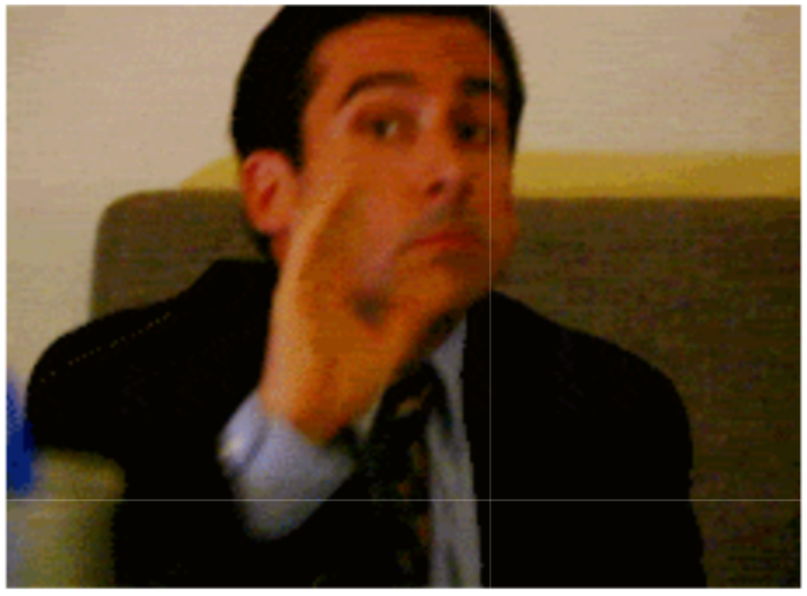
- Pratiquement identique à la version TypeScript sans le typage et les annotations
- Besoin d'écrire l'injection de dépendance à la main (fastidieux)
- Changer le compilateur Typescript avec Babel (ou compiler avec TypeScript sans les features TypeScript)
- Babel a des fonctionnalités expérimentales (à l'inverse de TypeScript)


ES6
5

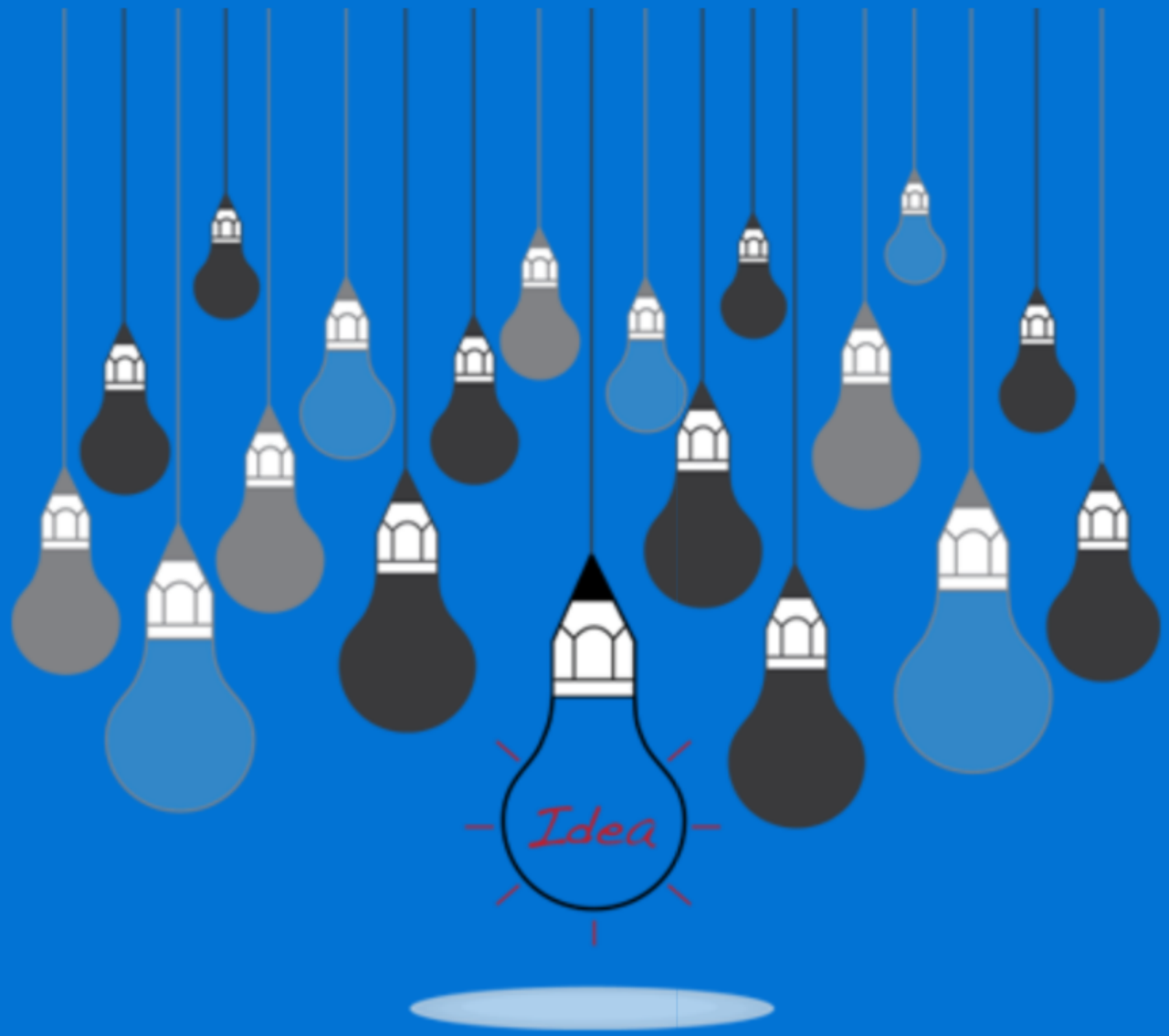
Angular 2 + ES5

- Support natif
- Aucun système de module n'est disponible (use IIFE ou des libraires tierce : RequireJS, CommonJS, SystemJS, etc.)
- Expose un namespace global *ng*
- Pas besoin de build ou de compilateur
- Pas de fichiers de config (typings, typescript, etc.)
- Pas de documentation :D

Mon conseil : a ne pas faire ;-)



Demo time !



Défi

Créer un projet à partir de zéro !

1. Initialiser NPM
2. Installer les dépendances nécessaires
3. Initialiser les fichier de configurations de tslint, tsconfig, webpack et index.html
4. Ecrire une application simple (hello world)
5. Lancer le serveur de dév webpack