formation.tech

# Formation Angular

Romain Bohdanowicz
Twitter : @bioub
http://formation.tech/


# Introduction

# Présentations

- Romain Bohdanowicz
  Ingénieur EFREI 2008, spécialité en Ingénierie Logicielle
- Expérience
  Formateur/Développeur Freelance depuis 2006
  Plus de 8000 heures de formation animées
- Langages
  Expert : HTML / CSS / JavaScript / PHP / Java
  Notions : C / C++ / Objective-C / C# / Python / Bash / Batch
- Certifications
  PHP 5 / PHP 5.3 / PHP 5.5 / Zend Framework 1
- Particularités
  Premier site web à 12 ans (HTML/JS/PHP), Triathlète à mes heures perdues
- Et vous ?
  Langages ? Expérience ? Utilité de cette formation ?


# TypeScript

# TypeScript - Introduction

- TypeScript : JavaScript + Typage statique
  - TypeScript est un langage créé par Microsoft, construit comme un sur-ensemble d'ECMAScript
  - Pour pouvoir exécuter le code il faut le transformer en JavaScript avec un compilateur
  - A quelques exceptions près et selon la configuration, le JavaScript est valide en TypeScript
- Le principal intérêt de TypeScript est l'ajout d'un typage statique

# TypeScript - Installation

- Installation
  - `npm install -g typescript`
- Création d'un fichier de configuration
  - `tsc --init`
- Compilation
  - `tsc`

# TypeScript - Typage statique

- Le principal intérêt de TypeScript est l'introduction d'un typage statique

```ts
const lastName: string = 'Bohdanowicz';
const age: number = 32;
const isTrainer: boolean = true;
```

- Types basiques :
  - *boolean*
  - *number*
  - *string*

# TypeScript - Typage statique

- Avantages
  - Complétion

```
const firstName: string = 'Romain';

function hello(firstName: string): string {
  return `Hello ${firstName.}`;
}
```

```
charAt(pos: number)
charCodeAt(index: number)
concat(... strings: string)
indexOf(searchString: string,
```

  - Détection des erreurs

```
const firstName: string = 'Romain';

function hello(firstName: string): string {
  return `Hello ${firstName}`;
}

hello({
  firstName: 'Romain',
});
```

# TypeScript - Typage statique

- Tableaux

```ts
const firstNames: string[] = ['Romain', 'Edouard'];
const colors: Array<string> = ['blue', 'white', 'red'];
```

- Tuples

```ts
const email: [string, boolean] = ['romain.bohdanowicz@gmail.com', true];
```

- Enum

```ts
enum Choice {Yes, No, Maybe}

const c1: Choice = Choice.Yes;
const choiceName: string = Choice[1];
```

- Never

```ts
function error(message: string): never {
  throw new Error(message);
}
```

# TypeScript - Typage statique

- Any

```ts
let anyType: any = 12;
anyType = "now a string string";
anyType = false;
anyType = {
  firstName: 'Romain'
};
```

- Void

```ts
function withoutReturn(): void {
  console.log('Do someting')
}
```

- Null et undefined

```ts
let u: undefined = undefined;
let n: null = null;
```

# TypeScript - Assertion de type

- Le compilateur ne peut pas toujours déterminer le type adéquat :

```
const formElt = document.querySelector('form.myForm');
const url = formElt.action; // error TS2339: Property 'action' does not exist on
type 'Element'.
```

- Il faut alors lui préciser, 3 syntaxes possibles

```
let formElt = <HTMLFormElement> document.querySelector('form.myForm');
const url = formElt.action;
```

```
let formElt = document.querySelector<HTMLFormElement>('form.myForm');
const url = formElt.action;
```

```
let formElt = document.querySelector('form.myForm') as HTMLFormElement;
const url = formElt.action;
```

# TypeScript - Inférence de type

- TypeScript peut parfois déterminer automatiquement le type :

```
const title = 'First Names';
console.log(title.toUpperCase());

const names = ['Romain', 'Edouard'];
for (let n of names) {
  console.log(n.toUpperCase());
}
```

# TypeScript - Interfaces

- Pour documenter un objet on utilise une interface
  - Anonyme

```
function helloInterface(contact: {firstName: string}) {
  console.log(`Hello ${contact.firstName.toUpperCase()}`);
}
```

  - Nommée

```
interface ContactInterface {
  firstName: string;
}

function helloNamedInterface(contact: ContactInterface) {
  console.log(`Hello ${contact.firstName.toUpperCase()}`);
}
```

# TypeScript - Interfaces

- Les propriétés peuvent être :
  - optionnelles (ici *lastName*)
  - en lecture seule, après l'initialisation (ici *age*)
  - non déclarées (avec les crochets)

```ts
interface ContactInterface {
  firstName: string;
  lastName?: string;
  readonly age: number;
  [propName: string]: any;
}

function helloNamedInterface(contact: ContactInterface) {
  console.log(`Hello ${contact.firstName.toUpperCase()}`);
}
```

# TypeScript - Classes

- Quelques différences avec JavaScript sur le mot clé class
  - On doit déclarer les propriétés
  - On peut définir une visibilité pour chaque membre : *public*, *private*, *protected*

```
class Contact {
  private firstName: string;

  constructor(firstName: string) {
    this.firstName = firstName;
  }

  hello(): string {
    return `Hello my name is ${this.firstName}`;
  }
}

const romain = new Contact('Romain');
console.log(romain.hello()); // Hello my name is Romain
```

# TypeScript - Classes

- Une classe peut
  - Hériter d’une autre classe (comme en JS)
  - Implémenter une interface
  - Être utilisée comme type

```ts
interface Writable {
  write(data: string): void;
}

class FileLogger implements Writable {
  write(data: string): Writable {
    console.log(`Write ${data}`);
    return this;
  }
}
```

# TypeScript - Génériques

- Permet de paramétrer le type de certaines méthodes

```ts
class Stack<T> {
  private data: Array<T> = [];
  push(val: T) {
    this.data.push(val);
  }
  pop(): T {
    return this.data.pop();
  }
  peek(): T {
    return this.data[this.data.length - 1];
  }
}

const strStack = new Stack<string>();
strStack.push('html');
strStack.push('body');
strStack.push('h1');
console.log(strStack.peek().toUpperCase()); // H1
console.log(strStack.pop().toUpperCase()); // H1
console.log(strStack.peek().toUpperCase()); // BODY
```

# TypeScript - Décorateurs

- Permettent l'ajout de fonctionnalités aux classes ou membre d'une classe en annotant plutôt que via du code à l'utilisation
- Norme à l'étude en JavaScript par le TC39
  https://github.com/tc39/proposal-decorators
- Supporté de manière expérimentale en TypeScript
- Pour activer leur support il faut éditer le tsconfig.json ou passer une option au compilateur

```
{
  "compilerOptions": {
    "target": "es5",
    "experimentalDecorators": true
  }
}
```

# TypeScript - Décorateurs

- Décorateur de classes

```
'use strict';

function Freeze(obj) {
  Object.freeze(obj);
}

@Freeze
class MyMaths {
  static sum(a, b) {
    return Number(a) + Number(b);
  }
}

try {
  MyMaths['substract'] = function(a, b) {
    return a - b;
  };
}
catch(err) {
  // Cannot add property substract, object is not extensible
  console.log(err.message);
}
```

# TypeScript - Décorateurs

- Décorateur de propriétés

```
import 'reflect-metadata';

const minLengthMetadataKey = Symbol("minLength");

function MinLength(length: number) {
  return Reflect.metadata(minLengthMetadataKey, length);
}

function validateMinLength(target: any, propertyKey: string): boolean {
  const length = Reflect.getMetadata(minLengthMetadataKey, target, propertyKey);
  return target[propertyKey].length >= length;
}

class Contact {
  @MinLength(7)
  protected firstName;

  constructor(firstName: string) {
    this.firstName = firstName;
  }

  isValid(): boolean {
    return validateMinLength(this, 'firstName');
  }
}

const romain = new Contact('Romain');
console.log(romain.isValid()); // false
```


# Zone.js

# Zone.js - Introduction

- Angular inclus une bibliothèque développée par Google appeler Zone.js
- Zone.js permet d'intercepter automatiquement des callbacks asynchrone et d'exécuter du code avant ou après
- Angular s'en sert principalement dans 3 contextes :
  - Lancer son algo de détection de changement après une requête AJAX, un changement de route ou toute autre opérations asynchrone
  - Intercepter les erreurs pouvant se produire dans les callbacks asynchrones
  - Lors des tests automatisés, marquer la fin du tests à l'issue de l'exécution du code synchrone et asynchrone
- Il est possible de gagner en performance en supprimant Zone.js et en lançant la détection de changement manuellement
- Installation
  `npm install zone.js`

# Zone.js - Exemples

- Exécuter du code après le dernier callback asynchrone

```
require('zone.js');

const myZone = Zone.current.fork({
  onHasTask(delegate, current, target, hasTaskState) {
    if (!hasTaskState.microTask && !hasTaskState.macroTask) {
      console.log('DONE');
    }
  },
});

myZone.run(() => {
  setTimeout(() => {
    console.log('setTimeout 1');
  }, Math.floor(Math.random() * 1001));

  setTimeout(() => {
    console.log('setTimeout 2');
  }, Math.floor(Math.random() * 1001));
});
```

```
setTimeout 1
setTimeout 2
DONE
```

# Zone.js - Exemples

- Exécuter du code avant ou après chaque callback asynchrone

```
require('zone.js');

const myZone = Zone.current.fork({
  onInvokeTask: (parentZoneDelegate, currentZone, targetZone, task) => {
    console.log('async call');
    return parentZoneDelegate.invokeTask(targetZone, task);
    // angular -> $digest
    // dt.detectChanges();
  }
});

myZone.run(() => {
  setTimeout(() => {
    console.log('setTimeout 500ms');
  }, 500);

  setTimeout(() => {
    console.log('setTimeout 800ms');
  }, 800);
});
```

```
async call
setTimeout 500ms
async call
setTimeout 800ms
```

# Zone.js - Exemples

- Intercepter les erreurs dans les callbacks asynchrones

```
require('zone.js');

const myZone = Zone.current.fork({
  onHandleError: (parentZoneDelegate, currentZone, targetZone, error) => {
    console.log(error.message);
  }
});

myZone.run(() => {
  setTimeout(() => {
    throw new Error('Error in async callback : setTimeout 300ms');
  }, 300);

  setTimeout(() => {
    console.log('setTimeout 500ms');
  }, 500);

  setTimeout(() => {
    throw new Error('Error in async callback : setTimeout 800ms');
  }, 800);
});
```

```
Error in async callback : setTimeout 300ms
setTimeout 500ms
Error in async callback : setTimeout 800ms
```


# Angular CLI

# Angular CLI - Introduction

- Angular introduit un programme en ligne de commande permettant d'interagir avec l'application :
  - créer un projet
  - builder
  - lancer le serveur de dev
  - générer du code
  - générer les fichiers de langue
  - …

# Angular CLI - Introduction

- Installation
  - `npm install -g @angular/cli`
- Documentation
  - https://cli.angular.io/
  - https://github.com/angular/angular-cli/wiki
- `ng help`
- `ng help COMMANDE`

# Angular CLI - Création d’un projet

- Création d'un projet
  `ng new CHEMIN_VERS_MON_PROJET`
- Autres options
  - `--skip-commit` : ne fait pas de commit initial
  - `--routing` : créer un module pour les routes (Single Page Application)
  - `--prefix` : change le préfixe des composant (par défaut *app*)
  - `--style` : change type de fichier CSS (*css* par défault ou *sass*, *scss*, *less*, *stylus*)
  - `--service-worker` : ajouter un service worker pour le mode hors-ligne

# Angular CLI - Squelette

- angular.json
  Fichier de configuration du programme *ng*, permet de renommer des répertoires, des fichiers
- e2e
  Test End to End (qui pilotent le navigateur)
- src/app
  Le code source de l'application
- tsconfig.json
  Configuration du compilateur TypeScript
- tslint.json
  Configuration des conventions de code

```
├── README.md
├── angular.json
├── e2e
|   ├── protractor.conf.js
|   ├── src
|   |   ├── app.e2e-spec.ts
|   |   └── app.po.ts
|   └── tsconfig.e2e.json
├── node_modules
├── package.json
├── src
|   ├── app
|   |   ├── app.component.css
|   |   ├── app.component.html
|   |   ├── app.component.ts
|   |   └── app.module.ts
|   ├── assets
|   ├── browserslist
|   ├── environments
|   |   ├── environment.prod.ts
|   |   └── environment.ts
|   ├── favicon.ico
|   ├── index.html
|   ├── karma.conf.js
|   ├── main.ts
|   ├── polyfills.ts
|   ├── styles.css
|   ├── test.ts
|   ├── tsconfig.app.json
|   ├── tsconfig.spec.json
|   └── tslint.json
├── tsconfig.json
└── tslint.json
```

# Angular CLI - Squelette minimal

- src/assets
  Les fichiers statiques non-buildés (images…)
- src/browserslist
  Les navigateurs ciblés (pour autoprefixer)
- src/environments
  Configuration de l’application
- src/index.html — src/main.ts
  Points d’entrées de l’application
- src/polyfills.ts
  Chargement des polyfills (core-js, …)
- src/style.css
  CSS global
- src/karma.conf.js — src/test.ts
  Configuration des tests

```
├── README.md
├── angular.json
├── e2e
|   ├── protractor.conf.js
|   ├── src
|   |   ├── app.e2e-spec.ts
|   |   └── app.po.ts
|   └── tsconfig.e2e.json
├── node_modules
├── package.json
├── src
|   ├── app
|   |   ├── app.component.css
|   |   ├── app.component.html
|   |   ├── app.component.ts
|   |   └── app.module.ts
|   ├── assets
|   ├── browserslist
|   ├── environments
|   |   ├── environment.prod.ts
|   |   └── environment.ts
|   ├── favicon.ico
|   ├── index.html
|   ├── karma.conf.js
|   ├── main.ts
|   ├── polyfills.ts
|   ├── styles.css
|   ├── test.ts
|   ├── tsconfig.app.json
|   ├── tsconfig.spec.json
|   └── tslint.json
├── tsconfig.json
└── tslint.json
```

# Angular CLI - Build

- Compiler l'application Angular
  - `ng build`
- Options intéressantes :
  - `--prod` : minifie le code avec UglifyJS et active les options *--aot*, *--environment=prod*, *--extract-css*, *--build-optimizer*...
  - `--environment=NOM` : permet de charger un fichier de configuration particulier (staging, test...)
  - `--vendor-chunk` : pour que le code de node_modules soit dans un fichier séparé

# Angular CLI - Build

- Gains d’un build avec *--prod*

```
Angular 4 : ng build
Date: 2017-11-02T09:02:41.042Z
Hash: 1d2842c3e0ac46a944f0
Time: 6349ms
chunk {inline} inline.bundle.js, inline.bundle.js.map (inline) 5.83 kB [entry] [rendered]
chunk {main} main.bundle.js, main.bundle.js.map (main) 18.1 kB {vendor} [initial] [rendered]
chunk {polyfills} polyfills.bundle.js, polyfills.bundle.js.map (polyfills) 199 kB {inline} [initial] [rendered]
chunk {styles} styles.bundle.js, styles.bundle.js.map (styles) 11.3 kB {inline} [initial] [rendered]
chunk {vendor} vendor.bundle.js, vendor.bundle.js.map (vendor) 1.98 MB [initial] [rendered]

Angular 5 : ng build
Date: 2017-11-02T09:07:47.401Z
Hash: d1a929eaad03e8e746bb
Time: 4937ms
chunk {inline} inline.bundle.js, inline.bundle.js.map (inline) 5.83 kB [entry] [rendered]
chunk {main} main.bundle.js, main.bundle.js.map (main) 17.9 kB [initial] [rendered]
chunk {polyfills} polyfills.bundle.js, polyfills.bundle.js.map (polyfills) 199 kB [initial] [rendered]
chunk {styles} styles.bundle.js, styles.bundle.js.map (styles) 11.3 kB [initial] [rendered]
chunk {vendor} vendor.bundle.js, vendor.bundle.js.map (vendor) 2.29 MB [initial] [rendered]

Angular 4 : ng build --prod
Date: 2017-11-02T09:03:29.639Z
Hash: cb067f695303856c2315
Time: 6228ms
chunk {0} polyfills.14173651b8ae6311a4b5.bundle.js (polyfills) 61.4 kB {4} [initial] [rendered]
chunk {1} main.f5677287cea9969f6fb6.bundle.js (main) 8.39 kB {3} [initial] [rendered]
chunk {2} styles.d41d8cd98f00b204e980.bundle.css (styles) 0 bytes {4} [initial] [rendered]
chunk {3} vendor.43700a281455e3959c70.bundle.js (vendor) 217 kB [initial] [rendered]
chunk {4} inline.6b5a62abf05dcccf24d7.bundle.js (inline) 1.45 kB [entry] [rendered]

Angular 5 : ng build --prod --vendor-chunk
Date: 2017-11-02T09:10:58.444Z
Hash: cf4dd52226e15e33c748
Time: 11487ms
chunk {0} polyfills.ad37cd45a71cb38eee76.bundle.js(polyfills) 61.1 kB [initial] [rendered]
chunk {1} main.2c7fbf970f7125d9617e.bundle.js (main) 7.03 kB [initial] [rendered]
chunk {2} styles.d41d8cd98f00b204e980.bundle.css (styles) 0 bytes [initial] [rendered]
chunk {3} vendor.719fe92af8c44a7e3dac.bundle.js (vendor) 167 kB [initial] [rendered]
chunk {4} inline.220ce59355d1cb2bcb28.bundle.js (inline) 1.45 kB [entry] [rendered]
```

# Angular CLI - Serveur de développement

- Lancer le serveur de dev
  - `ng serve`
- Options intéressantes :
  - `--port` : changer le port
  - `--target=production` : sert les fichiers dans la config de prod

# Angular CLI - Générateurs

- Générateurs
  Angular CLI contient un certains nombre de générateurs : application, class, component, directive, enum, guard, interface, module, pipe, service, universal, appShell

- Dry run
  Chaque générateur peut se lancer avec l'option *--dry-run* ou *-d* qui va afficher le résultat de la commande sans rien créer, sachant qu'il n'y a pas de retour automatique possible une fois les fichiers créés.

- Afficher la doc d'un générateur
  - `ng help generate NOM_DU_GENERATEUR`

# Angular CLI - Générateurs

- Générer un module
  - `ng generate module CHEMIN_DEPUIS_APP`
  - `ng g m CHEMIN_DEPUIS_APP`
- Autres options
  - `--routing` : génère un 2e module pour les routes
  - `--flat` : ne créé pas de répertoire

# Angular CLI - Générateurs

- Générer un composant
  - `ng generate component CHEMIN_DEPUIS_APP`
  - `ng g c CHEMIN_DEPUIS_APP`
- Autres options
  - `--flat` : ne créé pas de répertoire
  - `--export` : ajoute une entrée dans les exports du module

# Angular CLI - Tests & lint

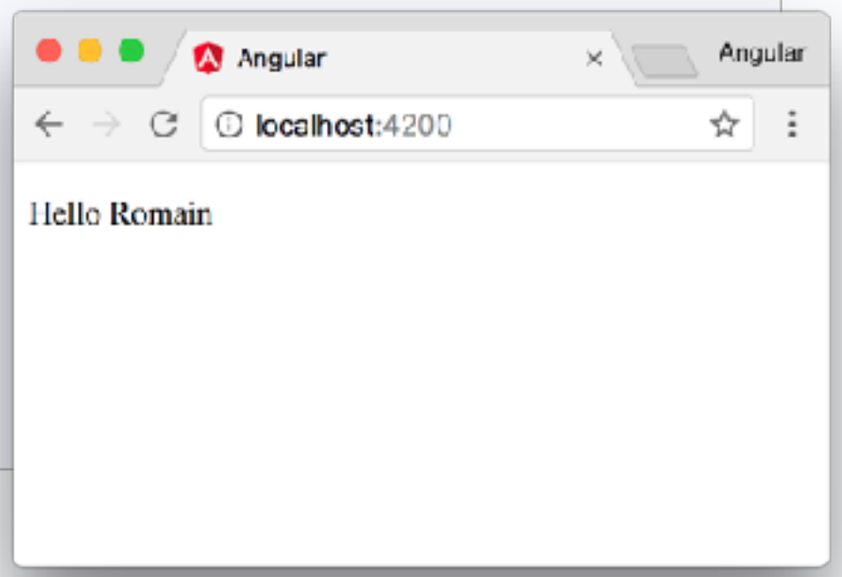
- Lancer les tests Karma + Jasmine
  - `ng test`
- Lancer les tests Protractor
  - `ng e2e`
- Vérifier les conventions de code
  - `ng lint`
  - `ng lint --fix --type-check`


# Composants

# Composants - Introduction

- 2 parties
  - code TypeScript
  - template
- Compilation
  - Les 2 sont compilés dans un code optimisé pour la VM JavaScript
  - Le template peut être compilé en JIT (par le browser) ou en AOT (au moment du build)

```ts
import { Component } from '@angular/core';

@Component({
  selector: 'my-hello',
  template: '<p>Hello {{name}}</p>',
})
export class HelloComponent {
  public name = 'Romain';
}
```



# Composants - Lifecycle Hooks

- Les composants peuvent implémenter les interfaces et leurs méthodes suivantes seront appelées automatiquement :
  - OnChanges / ngOnChanges() : lorsque qu'un changement se produit au niveau d'un input binding.
  - OnInit / ngOnInit() : une fois que e composant reçoit ses propriétés @Input et affiche les premiers input bindings.
  - OnDestroy / ngOnDestroy() : juste avant la destruction du component/directive. Il faut s'y désabonner des Observable ou événement pour éviter les fuites mémoires.
  - DoCheck / ngDoCheck() à chaque lancement de la détection de changement, permet d'identifier des changements que ngOnChanges ne peut détecter.

# Composants - Lifecycle Hooks

- ngAfterContentInit()
  - Respond after Angular projects external content into the component's view / the view that a directive is in.
  - Called once after the first ngDoCheck().
- ngAfterContentChecked()
  - Respond after Angular checks the content projected into the directive/component.
  - Called after the ngAfterContentInit() and every subsequent ngDoCheck().
- AfterViewInit / ngAfterViewInit()
  - Respond after Angular initializes the component's views and child views / the view that a directive is in.
  - Called once after the first ngAfterContentChecked().
- AfterViewChecked / ngAfterViewChecked()
  - Respond after Angular checks the component's views and child views / the view that a directive is in.
  - Called after the ngAfterViewInit and every subsequent ngAfterContentChecked().

# Composants - Lifecycle Hooks

- Exemple

```
import { Component, OnDestroy, OnInit } from '@angular/core';

@Component({
  selector: 'hello-lifecycle',
  template: `
    {{ now | date:'HH:mm:ss' }}
  `,
})
export class LifecycleComponent implements OnInit, OnDestroy {

  public now = new Date();
  private intervalId: number;

  ngOnInit() {
    this.intervalId = setInterval(() => {
      this.now = new Date();
    }, 1000)
  }

  ngOnDestroy() {
    clearInterval(this.intervalId);
  }

}
```

formation.tech

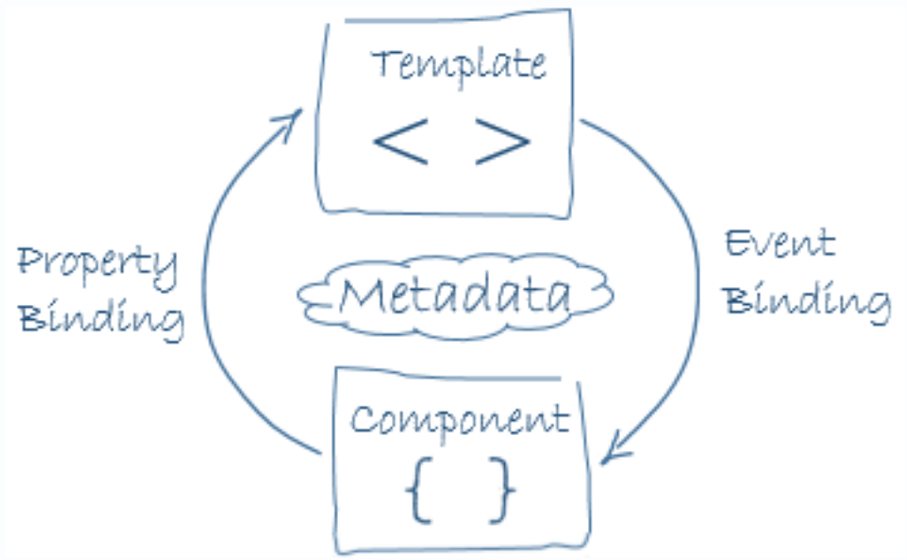
# Templates

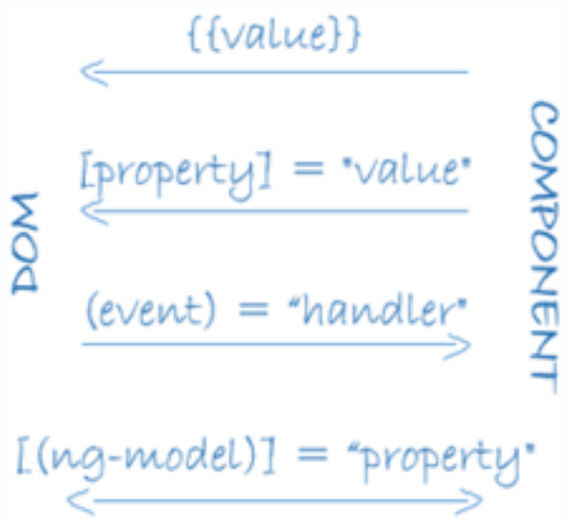
# Templates - Introduction

- Templates
  - Comme dans AngularJS, on décrit l'interface de manière déclarative dans des templates
  - Chaque template est compilé par le compilateur d'Angular, soit en amont (mode AOT pour Ahead Of Time Compilation), soit dans le browser (mode JIT pour Just In Time Compilation)
  - Les templates sont ainsi transformé en du code optimisé pour la VM/Moteur JavaScript

# Templates - Data binding

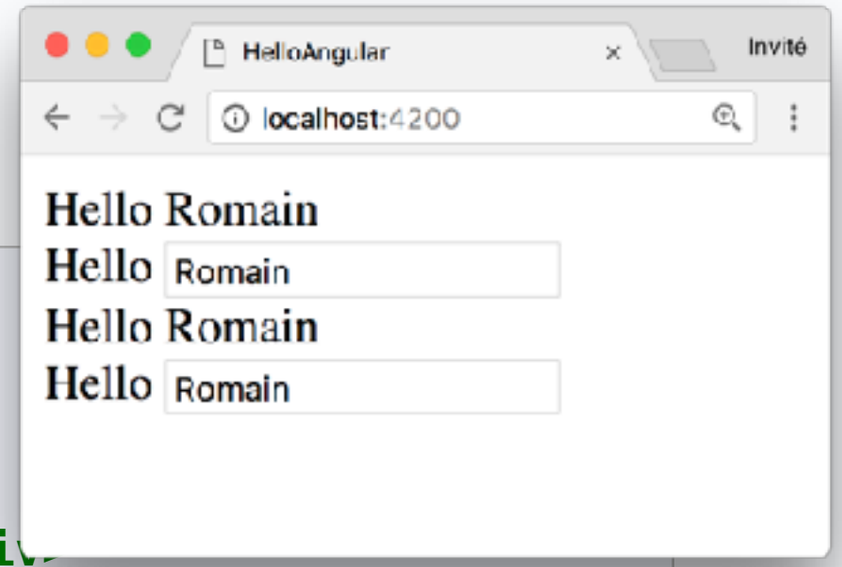
- Data binding
  - Sans data binding ce serait au développeur de maintenir les changements à opérer sur le DOM à chaque événement
  - Dans jQuery par exemple, cliquer sur un bouton peut avoir pour conséquence de rafraîchir une balise, de lancer un indicateur de chargement…
  - Avec Angular le développeur décrit l'état du DOM en fonction de propriétés qui constitue le Modèle, ainsi un événement n'a plus qu'



# Templates - Property Binding

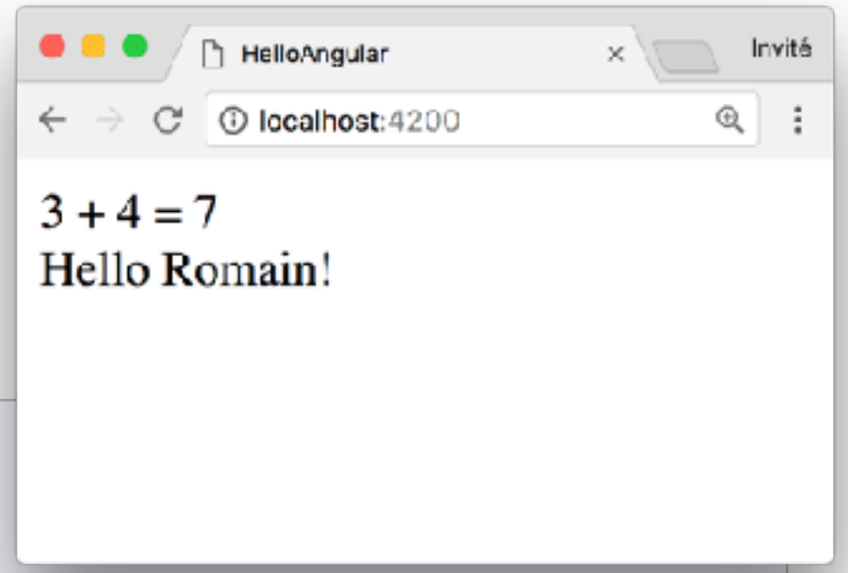
- 2 syntaxes
  Pour synchroniser le DOM avec le modèle (les propriétés publiques du composant dans Angular)
  - bind-nomDeLaPropDuDOM="propDuComposant"
  - [nomDeLaPropDuDOM]="propDuComposant"




```
import { Component } from '@angular/core';

@Component({
selector: 'hello-property-binding',
template: `
  <div>Hello <span bind-textContent="prenom"></span></div>
  <div>Hello <input bind-value="prenom"></div>
  <div>Hello <span [textContent]="prenom"></span></div>
  <div>Hello <input [value]="prenom"></div>
`,
})
export class PropertyBindingComponent {
public prenom = 'Romain';
}
```

# Templates - Expressions

- Dans un property binding il est possible d'utiliser des noms de propriétés ou des expressions, sauf les expressions ayant des effets de bords :
  - affectations (=, +=, -=, ...)
  - new
  - expressions chainées avec ; ou ,
  - incrementation et décrémentation (++ et --)

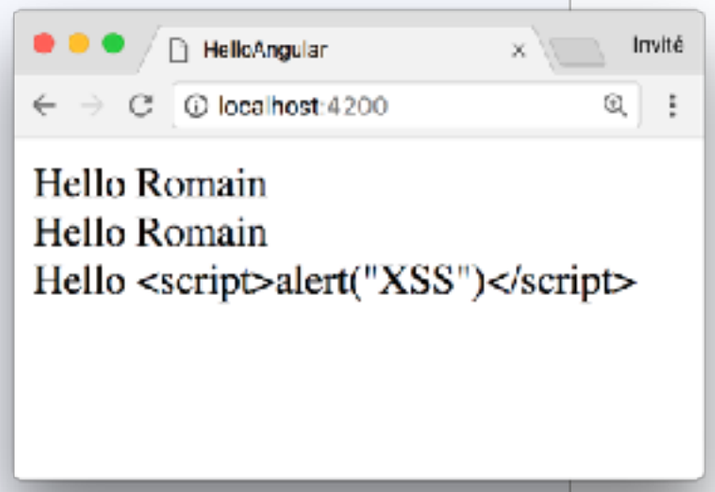



```
import { Component } from '@angular/core';

@Component({
selector: 'hello-prenom',
template: `
  <div>3 + 4 = <span [textContent]="3 + 4"></span></div>
  <div>Hello <span [textContent]="prenom + '!'"></span></div>
`,
})
export class PrenomComponent {
public prenom = 'Romain';
}
```

# Templates - Interpolation

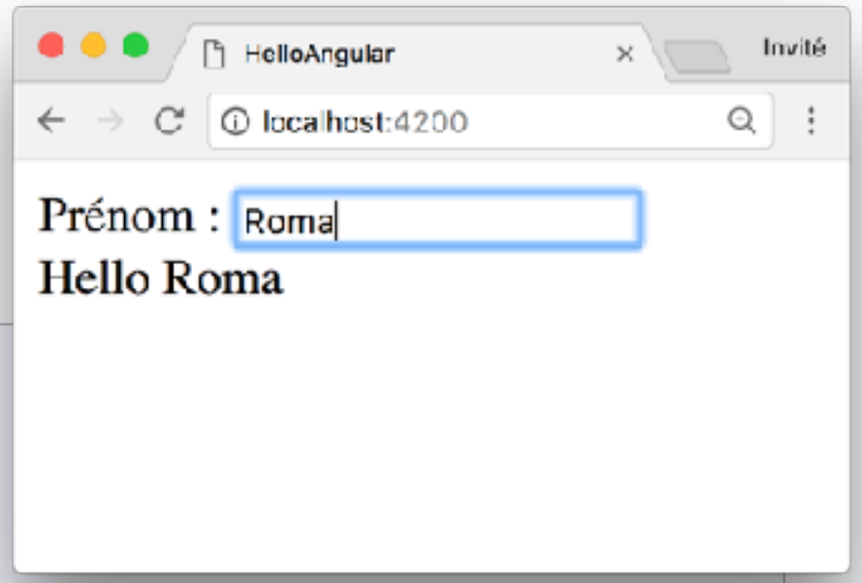
- Interpolation
  - Plutôt que bind-innerHTML sur une balise span, on peut utiliser la syntaxe aux doubles accolades {{ }}
  - A privilégier car cette syntaxe échappe les entrées, évitant ainsi que des balises contenues dans les entrées se retrouvent dans le DOM (faille XSS)

```
import { Component } from '@angular/core';

@Component({
  selector: 'hello-interpolation',
  template: `
<div>Hello <span [innerHTML]="prenom"></span></div>
<div>Hello {{prenom}}</div>
<div>Hello {{xssAttack}}</div>
  `,
})
export class InterpolationComponent {
  public prenom = 'Romain';
  public xssAttack = '<script>alert("XSS")</script>';
}
```



# Templates - Event Binding

- 2 syntaxes
  Pour synchroniser le DOM avec le modèle (les propriétés publiques du composant dans Angular), on utilise des événements
  - on-nomEvent="methodeDuComposant()"
  - (nomEvent)="methodeDuComposant()"



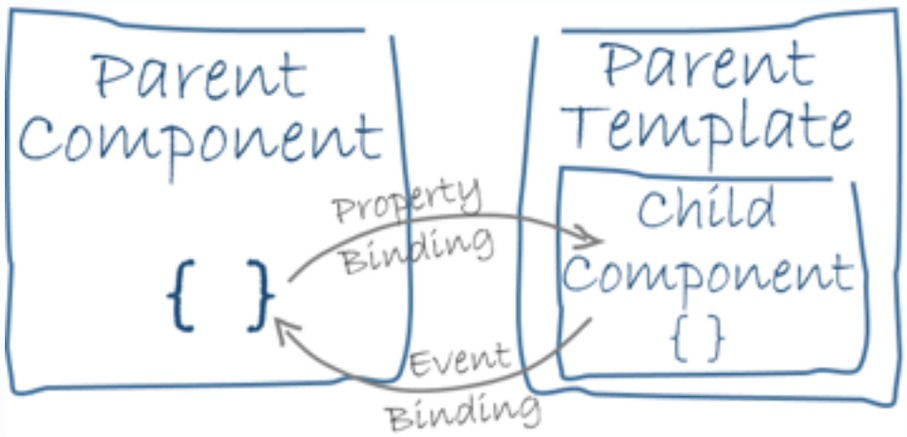

```
import { Component } from '@angular/core';

@Component({
  selector: 'hello-event-binding',
  template: `
<div>Prénom : <input on-input="updatePrenom($event)"></div>
<div>Prénom : <input (input)="updatePrenom($event)"></div>
<div>Prénom : <input (input)="prenom = $event.target.value"></div>
<div>Hello {{prenom}}</div>
  `,
})
export class EventBindingComponent {
  public prenom = '';

  public updatePrenom(e) {
this.prenom = e.target.value;
  }
}
```

# Templates - Two way data-binding

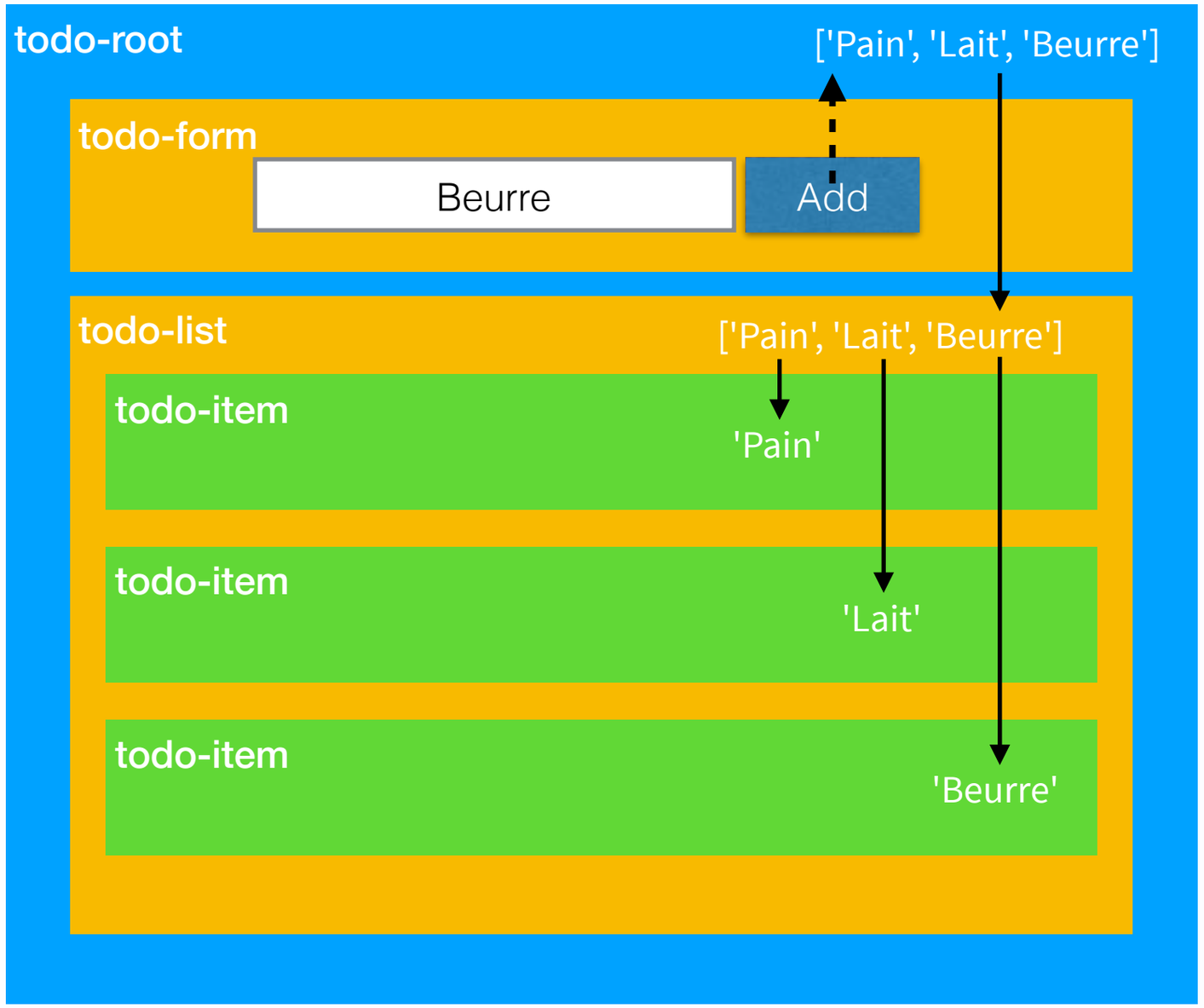
- AngularJS proposait des bindings dans les 2 sens (two way data-binding )
- Angular propose une syntaxe similaire, mais en réalité le binding se fait toujours en 2 temps, d'abord les events bindings, puis les property bindings
- Pour utiliser les 2 sur une ligne il faut :
  - Avoir un property binding
  - Avoir un event binding du même nom suffixé par change et qui affecte $event à la variable lié au property binding
- Exemple
  - `[ngModel]="prenom" (ngModelChange)="prenom = $event"`
  - `[(ngModel)]="prenom"`
- Pour moyen mnémotechnique : [( )] = BANANA IN A BOX

# TODO

- Communication inter-composant



# Template - Exercice




- Créer un nouveau projet TodoList avec prefix todo et style scss
- Créer 3 composants : Form, List, Item
- Créer un tableau dans App
- Passer le tableau à List via un property binding
- Passer chaque élément du tableau à Item via un property binding
- Au submit du form du composant Form, transmettre la valeur saisie au parent via un Event Binding (ajouter au tableau dans App)


# Modules

# Modules - Introduction

- 2 notions de modules
  - NgModule (class décorée avec @NgModule)
  - Module ES6 (import / export de fichiers)
- Jusqu'à la RC d'Angular 2, la notion de NgModule n'existait pas
- Intérêt d'avoir des NgModules :
  - Pouvoir importer un ensemble de composants / directives / pipes…
  - Pour configurer la portée d'un service
  - Permettre de charger des blocs de code par lazy-loading (après le chargement initial)
  - ...

# Modules - Principaux Modules

- Principaux Modules
  - `AppModule` : le module racine
  - `CommonModule` : le module qui inclus toutes les directives Angular de base comme *NgIf*, *NgForOf*, ...
  - `BrowserModule` : exporte *CommonModule* et contient les services permettant le rendu DOM, la gestion des erreurs, la modification des balises *title* ou *meta*...
  - `FormsModule` : le module qui permet la validation des formulaires, la déclaration de la directive *ngModel*...
  - `HttpClientModule` : contient les composants pour les requêtes HTTP
  - `RouterModule` : permet de manipuler des routes (associer des composants à des URL)
- Open-Source
  La première chose à faire après l'installation d'une bibliothèque *Angular* via *npm* sera d'importer un module

# Modules - Déclaration

- Déclaration
  - Pour qu'un composant, directive ou pipe existe dans l'application il faut le déclarer dans un module
  - Ne jamais déclarer 2 fois la même classe dans 2 modules différents

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';
import { HelloComponent } from './hello/hello.component';

@NgModule({
  declarations: [
    AppComponent,
    HelloComponent,
  ],
  imports: [
    BrowserModule
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

# Modules - Erreur courante

- Erreur courante
  Si un composant, directive ou pipe n'est pas déclaré dans un module, ou bien que le module dans lequel il est déclaré n'est pas importé par le module qui l'utilise

```
Uncaught Error: Template parse errors:
'app-title' is not a known element:
1. If 'app-title' is an Angular component, then verify that it is part of this
NgModule.
2. If 'app-title' is a Web Component then add 'CUSTOM_ELEMENTS_SCHEMA' to the
'@NgModule.schemas' of this component to suppress this message.
```

# Modules - Bonnes pratiques

- Bonnes pratiques
  - Créer un module CoreModule global
    Contiendra la déclaration de tous les services mono-instanciés (singleton) et composants, pipes ou directives utilisés uniquement dans le module racine (AppModule)
  - Créer un module SharedModule global
    Contiendra la déclaration de tous les services multi-instanciés et composants, pipes ou directives utilisés dans différents modules

# Best Practices - Linters

# Best Practices - Style Guide

# Best Practices - Build

formation.tech

# Mettre à jour Angular

# Mettre à jour Angular - Introduction

- Angular depuis la v2 respecte le versionnage sémantique
  https://semver.org/
- Avant de migrer vers une version majeure
  - Lire le changelog
    https://github.com/angular/angular/blob/master/CHANGELOG.md
  - Suivre les recommandation de l'Update Guide
    https://angular-update-guide.firebaseapp.com/

# Mettre à jour Angular - Outdated

- Vérifier qu'on est à jour ou non :
  npm outdated

```
MacBook-Pro:AddressBookAngular romain$ npm outdated
Package                                Current   Wanted  Latest  Location
@angular/animations                      4.4.6    4.4.6   5.2.9  address-book-angular
@angular/cli                             1.4.9    1.4.9   1.7.3  address-book-angular
@angular/common                          4.4.6    4.4.6   5.2.9  address-book-angular
@angular/compiler                        4.4.6    4.4.6   5.2.9  address-book-angular
@angular/compiler-cli                    4.4.6    4.4.6   5.2.9  address-book-angular
@angular/core                            4.4.6    4.4.6   5.2.9  address-book-angular
@angular/forms                           4.4.6    4.4.6   5.2.9  address-book-angular
@angular/http                            4.4.6    4.4.6   5.2.9  address-book-angular
@angular/language-service                4.4.6    4.4.6   5.2.9  address-book-angular
@angular/platform-browser                4.4.6    4.4.6   5.2.9  address-book-angular
@angular/platform-browser-dynamic        4.4.6    4.4.6   5.2.9  address-book-angular
@angular/router                          4.4.6    4.4.6   5.2.9  address-book-angular
@ngx-translate/core                      7.2.2    7.2.2   9.1.1  address-book-angular
@types/jasmine                          2.5.54   2.5.54   2.8.6  address-book-angular
@types/node                             6.0.90  6.0.102   9.4.7  address-book-angular
codelyzer                                3.2.2    3.2.2   4.2.1  address-book-angular
core-js                                  2.5.1    2.5.3   2.5.3  address-book-angular
jasmine-core                             2.6.4    2.6.4   3.1.0  address-book-angular
jasmine-spec-reporter                    4.1.1    4.1.1   4.2.1  address-book-angular
karma                                    1.7.1    1.7.1   2.0.0  address-book-angular
karma-chrome-launcher                    2.1.1    2.1.1   2.2.0  address-book-angular
karma-coverage-istanbul-reporter         1.3.0    1.4.2   1.4.2  address-book-angular
karma-jasmine                            1.1.0    1.1.1   1.1.1  address-book-angular
karma-jasmine-html-reporter              0.2.2    0.2.2   1.0.0  address-book-angular
protractor                               5.1.2    5.1.2   5.3.0  address-book-angular
rxjs                                     5.5.1    5.5.7   5.5.7  address-book-angular
ts-node                                  3.2.2    3.2.2   5.0.1  address-book-angular
tslint                                   5.7.0    5.7.0   5.9.1  address-book-angular
typescript                               2.3.4    2.3.4   2.7.2  address-book-angular
zone.js                                 0.8.18   0.8.20  0.8.20  address-book-angular
```

# Mettre à jour Angular - Migrer

- Angular CLI a une commande update depuis la v1.7
- Pour migrer Angular CLI :
  npm i @angular/cli@latest
- Mettre à jour Angular
  ng update
- Vers la prochaine version pour tester son code
  ng update --next

# Mettre à jour Angular - Bonnes pratiques

- Lancer la migration dans une branche, lancer les tests automatisés et/ou refaire des tests manuels
- Avec git, sur les commandes : checkout, pull, merge, rebase
  Lancer npm install
- Peut s'automatiser avec des hooks git
- Utiliser husky pour versionner ces scripts dans package.json
  https://github.com/typicode/husky

formation.tech

# Immuabilité

# Immuabilité - Introduction

- Lors de la modification d'un objet, le changement peut-être muable en modifiant l'objet d'origine ou immuable en créant un nouvel objet
- Les algorithmes de détections de changements préfèreront les changements immuables, ayant ainsi juste à comparer les références plutôt que l'ensemble du contenu de l'objet
- Exemple, en JS les tableaux sont muables, les chaines de caractères immuables

```
const firstName = 'Romain';
firstName.toUpperCase();
console.log(firstName); // Romain

const firstNames = ['Romain'];
firstNames.push('Edouard');
console.log(firstNames.join(', ')); // Romain, Edouard
```

# Immuabilité - Tableaux

- Ajouter à la fin

```
const firstNames = ['Romain', 'Edouard'];

function append(array, value) {
  return [...array, value];
}

const newfirstNames = append(firstNames, 'Jean');
console.log(newfirstNames.join(', ')); // Romain, Edouard, Jean
console.log(firstNames === newfirstNames); // false
```

- Ajouter au début

```
const firstNames = ['Romain', 'Edouard'];

function prepend(array, value) {
  return [value, ...array];
}

const newfirstNames = prepend(firstNames, 'Jean');
console.log(newfirstNames.join(', ')); // Jean, Romain, Edouard
console.log(firstNames === newfirstNames); // false
```

# Immuabilité - Tableaux

- Ajouter à un indice donné

```
const firstNames = ['Romain', 'Edouard'];

function insertAt(array, value, i) {
  return [
    ...array.slice(0, i),
    value,
    ...array.slice(i),
  ];
}

const newfirstNames = insertAt(firstNames, 'Jean', 1);
console.log(newfirstNames.join(', ')); // Romain, Jean, Edouard
console.log(firstNames === newfirstNames); // false
```

# Immuabilité - Tableaux

- Modifier un élément

```
const firstNames = ['Romain', 'Edouard'];

function modify(array, value, i) {
  return [
    ...array.slice(0, i),
    value,
    ...array.slice(i + 1),
  ];
}

const newfirstNames = modify(firstNames, 'Jean', 1);
console.log(newfirstNames.join(', ')); // Romain, Jean
console.log(firstNames === newfirstNames); // false
```

# Immuabilité - Tableaux

- Supprimer un élément

```
const firstNames = ['Romain', 'Edouard'];

function remove(array, i) {
  return [
    ...array.slice(0, i),
    ...array.slice(i + 1),
  ];
}

const newfirstNames = remove(firstNames, 1);
console.log(newfirstNames.join(', ')); // Romain
console.log(firstNames === newfirstNames); // false
```

# Immuabilité - Objet

- Ajouter un élément

```
const contact = {
  firstName: 'Romain',
  lastName: 'Bohdanowicz',
};

function add(object, key, value) {
  return {
    ...object,
    [key]: value
  };
}

const newContact = add(contact, 'city', 'Paris');
console.log(JSON.stringify(newContact));
// {"firstName":"Romain","lastName":"Bohdanowicz","city":"Paris"}
console.log(contact === newContact); // false
```

# Immuabilité - Objet

- Modifier un élément

```
const contact = {
  firstName: 'Romain',
  lastName: 'Bohdanowicz',
};

function modify(object, key, value) {
  return {
    ...object,
    [key]: value
  };
}

const newContact = modify(contact, 'firstName', 'Thomas');
console.log(JSON.stringify(newContact));
// {"firstName":"Thomas","lastName":"Bohdanowicz"}
console.log(contact === newContact); // false
```

# Immuabilité - Objet

- Supprimer un élément

```
const contact = {
  firstName: 'Romain',
  lastName: 'Bohdanowicz',
};

function remove(object, key) {
  const { [key]: val, ...rest } = object;
  return rest;
}

const newContact = remove(contact, 'lastName');
console.log(JSON.stringify(newContact));
// {"firstName":"Romain"}
console.log(contact === newContact); // false
```

# Immuabilité - Immutable.js

- Pour simplifier la manipulation d'objets ou de tableaux immuables, Facebook a créé Immutable.js
- Installation
  `npm install immutable`

# Immuabilité - Immutable.js List

- Ajouter à la fin

```
const immutable = require('immutable');

const firstNames = immutable.List(['Romain', 'Edouard']);

const newfirstNames = firstNames.push('Jean');
console.log(newfirstNames.join(', ')); // Romain, Edouard, Jean
console.log(firstNames === newfirstNames); // false
```

- Ajouter au début

```
const immutable = require('immutable');

const firstNames = immutable.List(['Romain', 'Edouard']);

const newfirstNames = firstNames.unshift('Jean');
console.log(newfirstNames.join(', ')); // Jean, Romain, Edouard
console.log(firstNames === newfirstNames); // false
```

# Immuabilité - Immutable.js List

- Ajouter à un indice donné

```
const immutable = require('immutable');

const firstNames = immutable.List(['Romain', 'Edouard']);

const newfirstNames = firstNames.insert(1, 'Jean');
console.log(newfirstNames.join(', ')); // Romain, Jean, Edouard
console.log(firstNames === newfirstNames); // false
```

# Immuabilité - Immutable.js List

- Modifier un élément

```
const immutable = require('immutable');

const firstNames = immutable.List(['Romain', 'Edouard']);

const newfirstNames = firstNames.set(1, 'Jean');
console.log(newfirstNames.join(', ')); // Romain, Jean
console.log(firstNames === newfirstNames); // false
```

# Immuabilité - Immutable.js List

- Supprimer un élément

```
const immutable = require('immutable');

const firstNames = immutable.List(['Romain', 'Edouard']);

const newfirstNames = firstNames.delete(1);
console.log(newfirstNames.join(', ')); // Romain
console.log(firstNames === newfirstNames); // false
```

# Immuabilité - Immutable.js Map

- Ajouter un élément

```
const immutable = require('immutable');

const contact = immutable.Map({
  firstName: 'Romain',
  lastName: 'Bohdanowicz',
});

const newContact = contact.set('city', 'Paris');
console.log(JSON.stringify(newContact));
// {"firstName":"Romain","lastName":"Bohdanowicz","city":"Paris"}
console.log(contact === newContact); // false
```

# Immuabilité - Immutable.js Map

- Modifier un élément

```
const immutable = require('immutable');

const contact = immutable.Map({
  firstName: 'Romain',
  lastName: 'Bohdanowicz',
});

const newContact = contact.set('firstName', 'Thomas');
console.log(JSON.stringify(newContact));
// {"firstName":"Thomas","lastName":"Bohdanowicz"}
console.log(contact === newContact); // false
```

# Immuabilité - Immutable.js Map

- Supprimer un élément

```
const immutable = require('immutable');

const contact = immutable.Map({
  firstName: 'Romain',
  lastName: 'Bohdanowicz',
});

const newContact = contact.remove('lastName');
console.log(JSON.stringify(newContact));
// {"firstName":"Romain"}
console.log(contact === newContact); // false
```

formation.tech

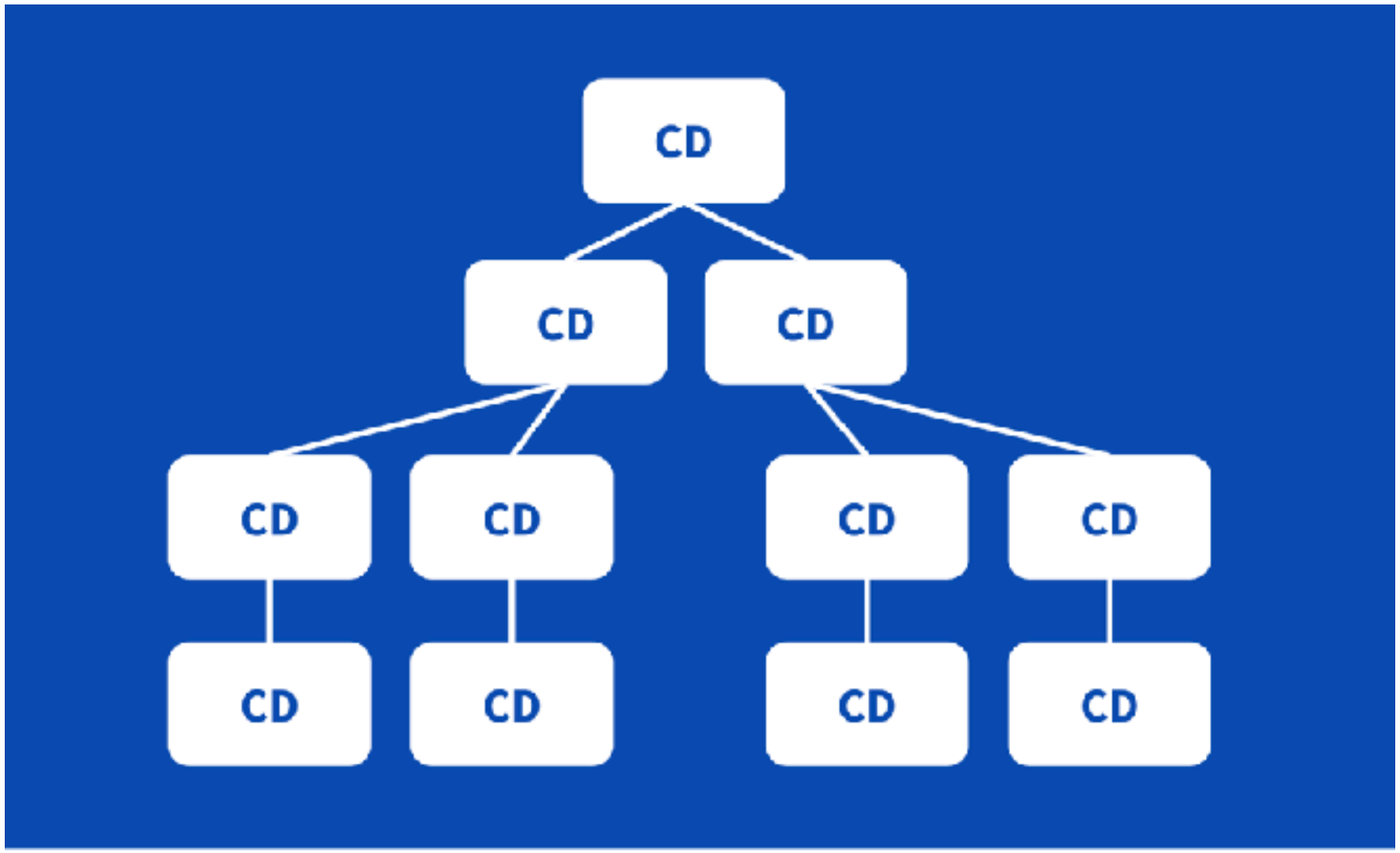
# Détection de changement

# Détection de changement - Introduction

- Dans Angular la détection de changement permet de rafraîchir les property bindings des composants
- On peut lancer les algorithmes de détection de changement de 3 façons :
  - Les events bindings (click), (submit)...
  - Les opérations asynchrones grâce à Zone.js
  - Manuellement via ChangeDetectionRef

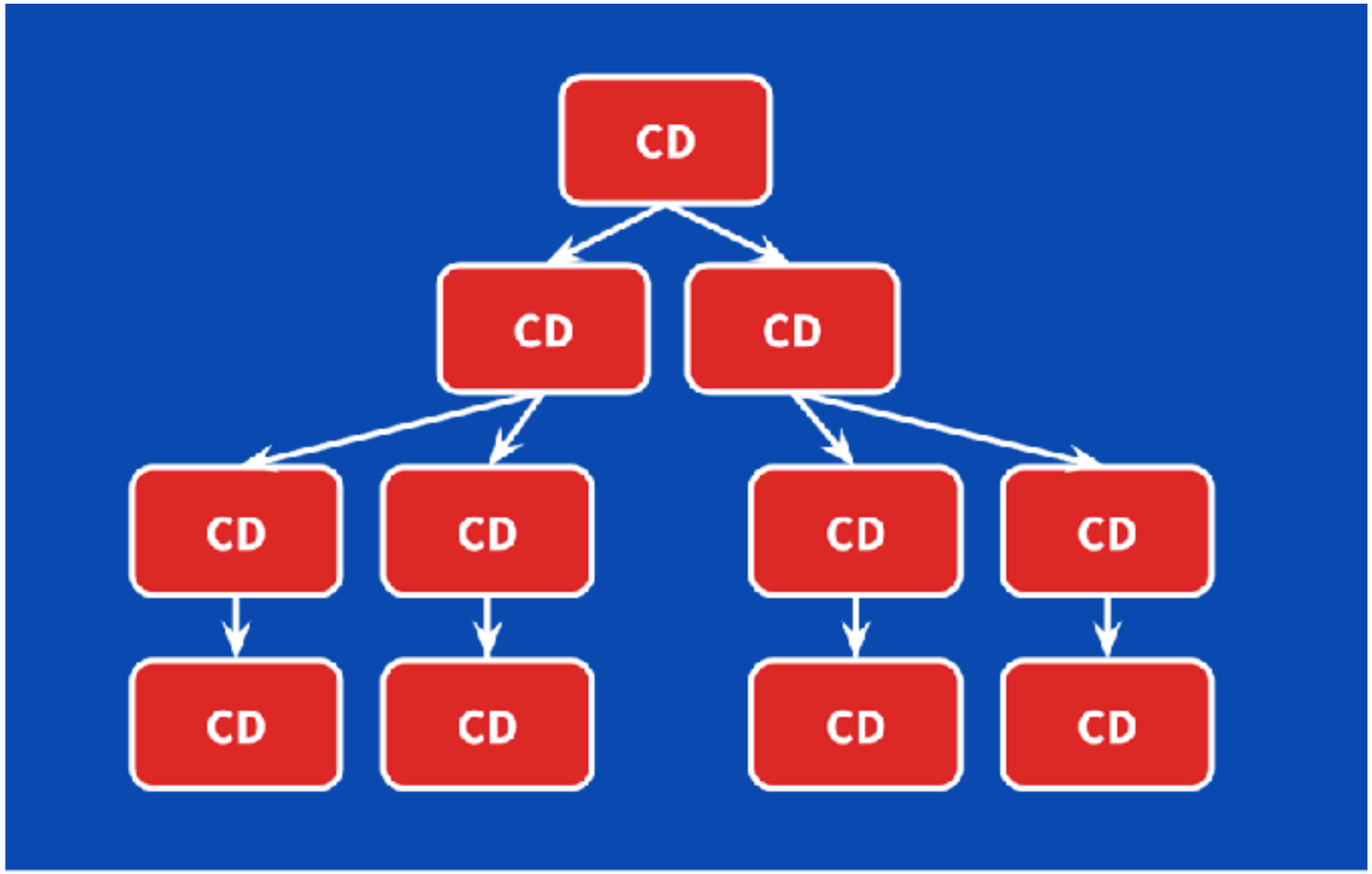
# Détection de changement - Déclenchement

- Chaque composant vient avec son propre algorithme de détection qui est généré au moment de la compilation du template (en JIT ou AOT)
- Le code généré est optimisé pour la VM JavaScript et lance des comparaison du type *ancienneValeur* === *nouvelleValeur* ou *ancienneValeur.prop* === *nouvelleValeur.prop*



# Détection de changement - Lancement

- Lorsque que la détection de changement est lancée, l'ensemble des algorithmes de des composants sont lancés du composant racine vers les composants les plus lointains

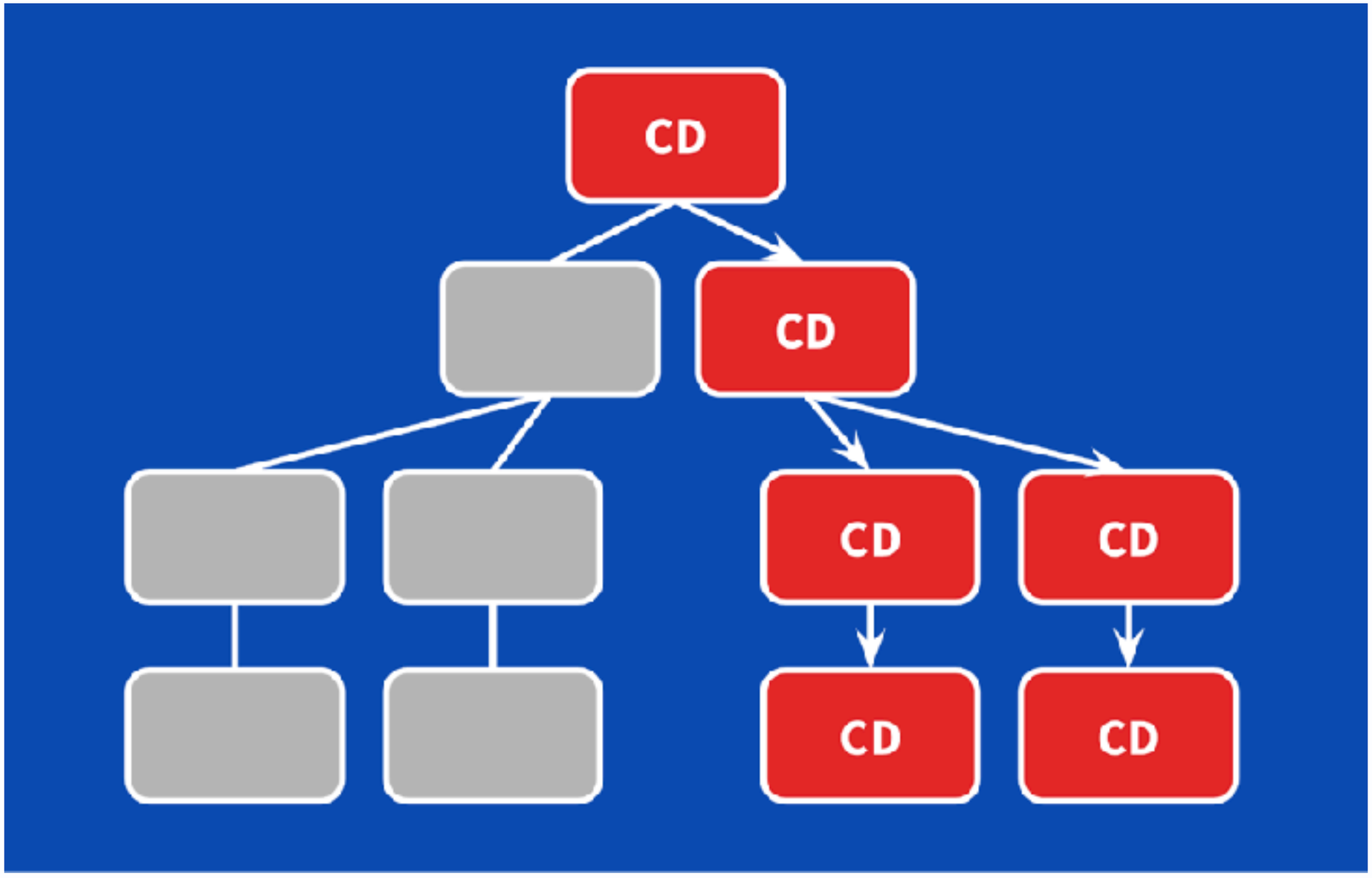


# Détection de changement - Interruption

- Pour éviter que la détection de changement soit lancée sur un composant et ses descendants, on peut :
  - Utiliser la stratégie OnPush
  - Utiliser ChangeDetectionRef (Manuel)

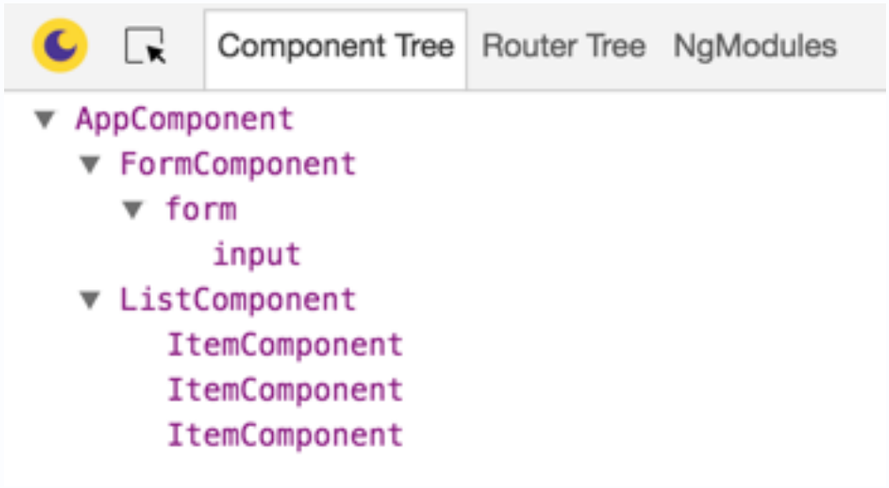
# Détection de changement - OnPush

- OnPush
  - Avec la stratégie OnPush on peut faire en sorte qu'un composant ne dépende que de ses Input (changement immuable)



# Détection de changement - OnPush

- Exemple
  https://gitlab.com/angular-avance/TodoAngular



# Détection de changement - ChangeDetectorRef

- ChangeDetectorRef
  Permet de contrôler soit même la détection de changement

```
export declare abstract class ChangeDetectorRef {
    abstract markForCheck(): void;
    abstract detach(): void;
    abstract detectChanges(): void;
    abstract checkNoChanges(): void;
    abstract reattach(): void;
}
```

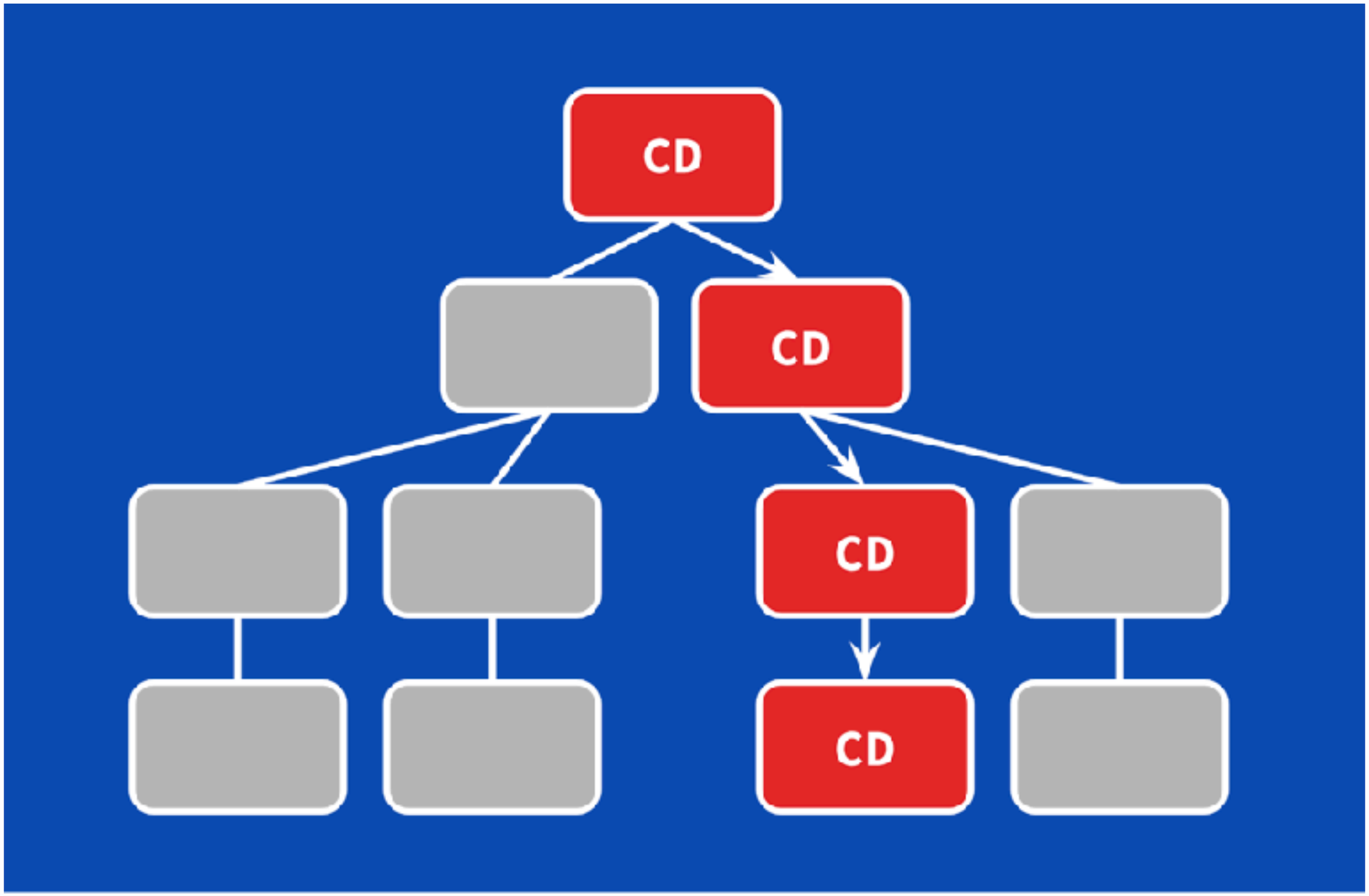
# Détection de changement - ChangeDetectorRef

- markForCheck()
  Permet de tagger le composant et tous ses ancêtres OnPush comme étant à checker

```
import { ChangeDetectionStrategy, ChangeDetectorRef } from '@angular/core';

@Component({
  changeDetection: ChangeDetectionStrategy.OnPush,
})
export class ContactsShowComponent implements OnInit {
  public contact: Contact;

  constructor(private cd: ChangeDetectorRef) {}

  ngOnInit() {
    this.contactService.getAll()
      .subscribe(contact => {
        this.contact = contact;
        this.cd.markForCheck();
      });
  }
}
```

# Détection de changement - ChangeDetectorRef

- markForCheck()



# Détection de changement - ChangeDetectorRef

- detach()
  Désactive la détection du changement de ce composant
- reattach()
  Réactive la détection du changement de ce composant
- detectChanges()
  Lance la détection du changement manuellement

```
import { ChangeDetectionStrategy, ChangeDetectorRef } from '@angular/core';

@Component()
export class ContactsShowComponent implements OnInit {
  public contact: Contact;

  constructor(private cd: ChangeDetectorRef) {}

  ngOnInit() {
    this.cd.detach();
    this.contactService.getAll()
      .subscribe(contact => {
        this.contact = contact;
        this.cd.detectChanges();
      });
  }
}
```

# Détection de changement - Resources

- Angular Change Detection Explained
  https://blog.thoughtram.io/angular/2016/02/22/angular-2-change-detection-explained.html
- Everything you need to know about change detection in Angular
  https://blog.angularindepth.com/everything-you-need-to-know-about-change-detection-in-angular-8006c51d206f
- How to test OnPush components
  https://medium.com/@juliapassynkova/how-to-test-onpush-components-c9b39871fe1e
- JS web frameworks benchmark
  http://www.stefankrause.net/js-frameworks-benchmark7/table.html
-

formation.tech

# Créer sa bibliothèque

# Créer sa bibliothèque - Introduction

- Deux options pour créer sa propre bibliothèque
  - Utiliser @angular/compiler et @angular/compiler-cli
  - Utiliser ng-packagr
- Avantages de ng-packagr
  - Respecte le format Angular Package
    - Créé des versions ESM ES6, ESM ES6, and UMD
    - Code compatible Angular CLI, Webpack, or SystemJS
    - Definitions et metadata(.d.ts, .metadata.json)
    - Points d'entrées secondaires : @my/foo, @my/foo/testing, @my/foo/bar
  - Converti les templates et les styles en inline
  - Fonctionnalités CSS incluses : SCSS, LESS, Stylus, Autoprefixer, PostCSS

# Créer sa bibliothèque - Création

- Créer son propre package

```
├── package-lock.json
├── package.json
├── src
│   ├── index.ts
│   ├── ui-copyright.component.ts
│   └── ui-copyright.module.ts
└── tsconfig.json
```

- Le fichier index.ts exporte tout ce qui doit pouvoir être importé (Modules, Services, Interfaces, Classes, Injectors Tokens...)

```ts
export * from './ui-copyright.module';
```

# Créer sa bibliothèque - Création

- package.json minimal

```
{
  "name": "@formation.tech/ui-copyright",
  "version": "0.0.1",
  "main": "dist/index.js",
  "typings": "dist/index.d.ts",
  "dependencies": {
    "@angular/common": "^5.2.9",
    "@angular/core": "^5.2.9",
    "rxjs": "^5.5.7"
  },
  "devDependencies": {
    "@angular/compiler": "^5.2.9",
    "@angular/compiler-cli": "^5.2.9",
    "typescript": "^2.7.2"
  },
  "scripts": {
    "build": "ngc"
  }
}
```

# Créer sa bibliothèque - Création

- tsconfig.json minimal

```
{
  "compilerOptions": {
    "module": "es2015", // modules exportés en ES6
    "moduleResolution": "node", // rend accessible node_modules si module es2015
    "target": "es5", // code exporté en ES5
    "sourceMap": false, // génère les fichiers map pour le debug ou non
    "declaration": true, // génère les fichiers .d.ts pour la complétion
    "experimentalDecorators": true, // support des décorateurs @Component ...
    "outDir": "dist", // dossier de destination
    "rootDir": "src", // dossier à builder
    "lib": [
      "dom", // reconnait les types du DOM : Console, Node, Document...
      "es2015" // reconnait les types ES6 : Map...
    ]
  }
}
```

- Lancer ensuite le compilateur
  npm run build

# Créer sa bibliothèque - ngPackagr

- Installation
  npm i ngPackager -D
- Ajouter la config au package.json

```
{
  "name": "@formation.tech/ui-horloge",
  "version": "0.0.1",
  "license": "MIT",
  "scripts": {
    "build": "ng-packagr -p package.json"
  },
  "ngPackage": {
    "lib": {
      "entryFile": "public_api.ts"
    },
    "dest": "../lib/ui-horloge"
  }
}
```

- Lancer le build
  npm run build

# Créer sa bibliothèque - ngPackagr

- Package généré

```
├── bundles
│   ├── formation.tech-ui-horloge.umd.js
│   ├── formation.tech-ui-horloge.umd.js.map
│   ├── formation.tech-ui-horloge.umd.min.js
│   └── formation.tech-ui-horloge.umd.min.js.map
├── esm2015
│   ├── formation.tech-ui-horloge.js
│   └── formation.tech-ui-horloge.js.map
├── esm5
│   ├── formation.tech-ui-horloge.js
│   └── formation.tech-ui-horloge.js.map
├── formation.tech-ui-horloge.d.ts
├── formation.tech-ui-horloge.metadata.json
├── node_modules
├── package.json
├── public_api.d.ts
└── src
    ├── app
    │   └── ui-horloge
    │       ├── ui-horloge.component.d.ts
    │       └── ui-horloge.module.d.ts
    └── typings.d.ts
```

# Créer sa bibliothèque - Installation

- Pour installer un paquet on peut :
  - créer un lien symbolique : npm install ../ma-lib
  - déployer sur git : npm install
  - déployer sur npm public ou privé : npm install ma-lib
- Attention depuis Angular 5 il faut ajouter l'option preserveSymlinks au moment du build (en CLI --preserve-symlinks ou dans le .angular-cli.json) en cas d'installation locale

```
{
  "defaults": {
    "build": {
      "preserveSymlinks": true
    }
  }
}
```

# Créer sa bibliothèque - Bonnes pratiques

- Supprimer le node_modules de la bibliothèque après son build
- Ne pas générer les ngfactory.js "angularCompilerOptions": { "skipTemplateCodegen": true }
- Compiler la version la plus basse possible (un module compilé en Angular 5 ne fonctionnera pas dans Angular 4)

# Créer sa bibliothèque - Resources

- How to build and publish an Angular module
  https://medium.com/@cyrilletuzi/how-to-build-and-publish-an-angular-module-7ad19c0b4464
- angular-cli-lib-example
  https://github.com/jasonaden/angular-cli-lib-example
- Building an Angular 4 Component Library with the Angular CLI and ng-packagr
  https://medium.com/@nikolasleblanc/building-an-angular-4-component-library-with-the-angular-cli-and-ng-packagr-53b2ade0701e
- Distributing an Angular Library - The Brief Guide
  http://blog.mgechev.com/2017/01/21/distributing-an-angular-library-aot-ngc-types/


# Angular Universal

# Angular Universal - Introduction

- Permet de faire un rendu côté server (SSR)
- 3 intérêts :
  - Faciliter le référencement (SEO)
  - Améliorer les performances sur mobile et machines peu performantes
  - Afficher la première page plus rapidement
- Angular CLI 1.6+ facilite fortement la mise en place
- 3 moteurs
  - Express (Node.js)
  - Hapi (Node.js)
  - ASP.NET

# Angular Universal - Pièges

- Choses à anticiper
  - Les Web APIs ne seront pas dispo côté serveur (Window, Document, ...) bien qu'ils soient en partie réimplémentés
  - Les requêtes AJAX vont s'exécuter 2 fois, côté serveur puis côté client, créer un cache pour l'éviter
  - Les requêtes AJAX doivent être absolues
  - Eviter ou bannir setTimeout et encore surtout setInterval
  - Certains modules ne fonctionneront pas, ou nécessiteront des changements, ex ngx-translate :

```
import { TranslateLoader, TranslateModule } from '@ngx-translate/core';
import { TranslateHttpLoader } from '@ngx-translate/http-loader';
import { UniversalTranslateLoader } from '@ngx-universal/translate-loader';

// AoT requires an exported function for factories
export function translateFactory(platformId: any, httpClient: HttpClient):
TranslateLoader {
  const browserLoader = new TranslateHttpLoader(httpClient);
  return new UniversalTranslateLoader(platformId, browserLoader, 'dist-server/
assets/i18n');
}
```

# Angular Universal - Test de plate-forme

- Exécuter du code sur une plateforme spécifiquement (client ou serveur)
  - Utiliser l'Injection Token *PLATFORM_ID* et les méthodes *isPlatformBrowser* et *isPlatformServer*

```
import { PLATFORM_ID } from '@angular/core';
import { isPlatformBrowser, isPlatformServer } from '@angular/common';

constructor(@Inject(PLATFORM_ID) private platformId: Object) { ... }

ngOnInit() {
  if (isPlatformBrowser(this.platformId)) {
    // Client only code.
    ...
  }
  if (isPlatformServer(this.platformId)) {
    // Server only code.
    ...
  }
}
```

# Angular Universal - Création

- Rendre son application universelle
  ng generate universal [nom]
  ng generate universal serverApp
- La commande *generate universal* fait des changements dans l'application et créé une seconde application dans le .angular-cli.json
- Pour builder les 2 apps
  ng build --prod && ng build --prod --app server-app --output-hashing=none

# Angular Universal - Serveur Express

- Installer express et le moteur universal pour express
  npm i express @nguniversal/express-engine

```
require('zone.js/dist/zone-node');

const path = require('path');
const express = require('express');
const ngUniversal = require('@nguniversal/express-engine');
const appServer = require('./dist-server/main.bundle');

const app = express();
app.use(express.static(path.resolve(__dirname, 'dist')));
app.engine('html', ngUniversal.ngExpressEngine({
  bootstrap: appServer.AppServerModuleNgFactory
}));
app.set('view engine', 'html');
app.set('views', 'dist');

app.use((req, res) => {
  res.render('index', { req, res });
});

app.listen(8000, () => {
  console.log(`Listening on http://localhost:8000`);
});
```

# Angular Universal - TransferState

- Avec un rendu côté serveur, les requêtes vont s'exécuter 2 fois, côté serveur puis client
- Outre le problème de performance, les données vont "flasher"



# Angular Universal - TransferState

- Configuration
  - Importer ServerTransferStateModule dans AppServerModule
  - Importer BrowserTransferStateModule dans AppModule
  - Dans un Composant :

```
import { makeStateKey, TransferState } from '@angular/platform-browser';
const RESULT_KEY = makeStateKey('contacts');

export class ContactService implements ContactServiceInterface {
  public contacts;

  public getList$(): Observable<Contact[]> {
    if (this.transferState.hasKey(RESULT_KEY)) {
      const res = Observable.of(this.transferState.get<Contact[]>(RESULT_KEY, null));
      this.transferState.remove(RESULT_KEY);
      return res;
    } else {
      this.transferState.onSerialize(RESULT_KEY, () => this.contacts);
      return this.http.get<Contact[]>(`${environment.apiServer}/contacts`)
        .pipe(tap(contacts => this.contacts = contacts));
    }
  }
}
```

# Angular Universal - TransferState

- Voir aussi
  - TransferHttpCacheModule pour se simplifier les transferts HTTP : https://github.com/angular/universal/tree/master/modules/common
  - Pour ngx-translate : https://github.com/ngx-translate/core/issues/754#issuecomment-353616515

# Angular Universal - Resources

- ng-seed/universal
  https://github.com/ng-seed/universal
- Using TransferState API in an Angular v5 Universal App
  https://blog.angularindepth.com/using-transferstate-api-in-an-angular-5-universal-app-130f3ada9e5b
- Angular server-side rendering in Node with Express Universal Engine
  https://medium.com/@cyrilletuzi/angular-server-side-rendering-in-node-with-express-universal-engine-dce21933ddce

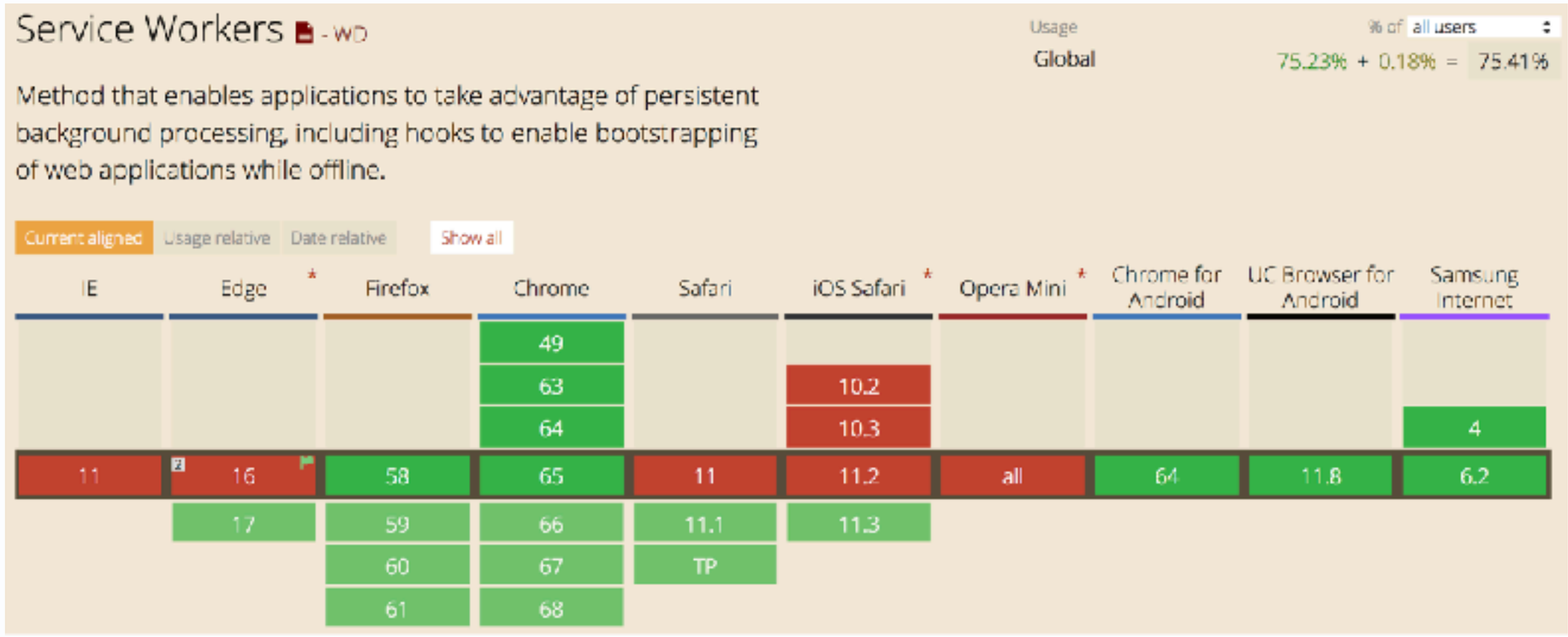

# Progressive Web Apps

# PWA - Introduction

- Intérêt des Progressive Web Apps
  - Un temps de chargement considérablement réduit
  - Une utilisation sans connexion Internet
  - Elles sont Responsive, donc compatibles avec n'importe quel système d'exploitation et n'importe quel support (pc, tablette, mobile)
  - Pas d'installation requise
  - Les PWA sont accessibles depuis une URL ou directement depuis une icône sur l'écran d'accueil du mobile
  - Elles ne prennent pas de place dans la mémoire du mobile
  - Elles sont sécurisées (protocole HTTPS)
  - Une expérience immersive grand écran, semblable aux applications natives.

# PWA - Introduction

- Service Worker vs AppCache
  - Depuis quelques années les navigateurs implémentaient un API appelé AppCache permettant de décrire dans un fichier manifest les resources à garder en cache pour une utilisation hors ligne
  - Un Service Worker permet l'exécution du code dans un thread séparé, améliorant les performances lors d'appel à des données et permettant de les retrouver en mode hors ligne



# PWA - Mise en place

- Installation
  npm i @angular/service-worker
- Angular CLI
  Ajouter dans l'app cliente
  "serviceWorker": true
- Créer un fichier src/ngsw-config.json (peut se générer à partir du CLI ngsw-config)
- Importer dans AppModule

```
ServiceWorkerModule.register(
  '/ngsw-worker.js', {
    enabled: environment.production
  },
),
```

```
{
  "index": "/index.html",
  "assetGroups": [{
    "name": "app",
    "installMode": "prefetch",
    "resources": {
      "files": [
        "/favicon.ico",
        "/index.html"
      ],
      "versionedFiles": [
        "/*.bundle.css",
        "/*.bundle.js",
        "/*.chunk.js"
      ]
    }
  }, {
    "name": "assets",
    "installMode": "lazy",
    "updateMode": "prefetch",
    "resources": {
      "files": [
        "/assets/**"
      ]
    }
  }]
}
```

# PWA - Manifest

- Créer un fichier src/manifest.json (l'ajouter dans les assets dans .angular-cli.json)

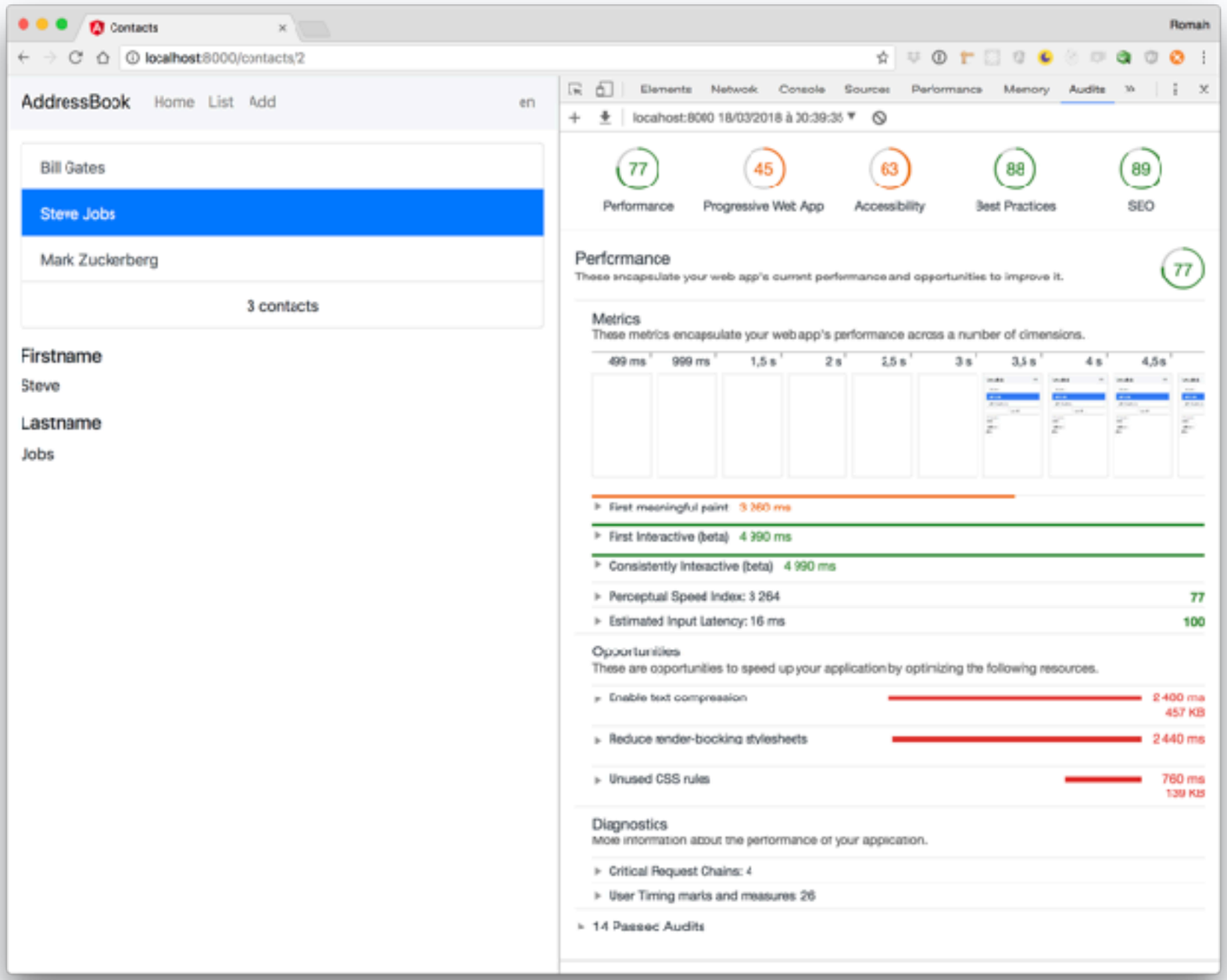
```
{
  "short_name": "Address Book",
  "name": "Address Book Angular Avancé",
  "start_url": "/",
  "theme_color": "#212529",
  "background_color": "#f8f9fa",
  "display": "standalone",
  "orientation": "portrait",
  "icons": [
    {
      "src": "/assets/icons/android-chrome-512x512.png",
      "sizes": "512x512",
      "type": "image/png"
    }
  ]
}
```

- Ajouter au fichier index.html

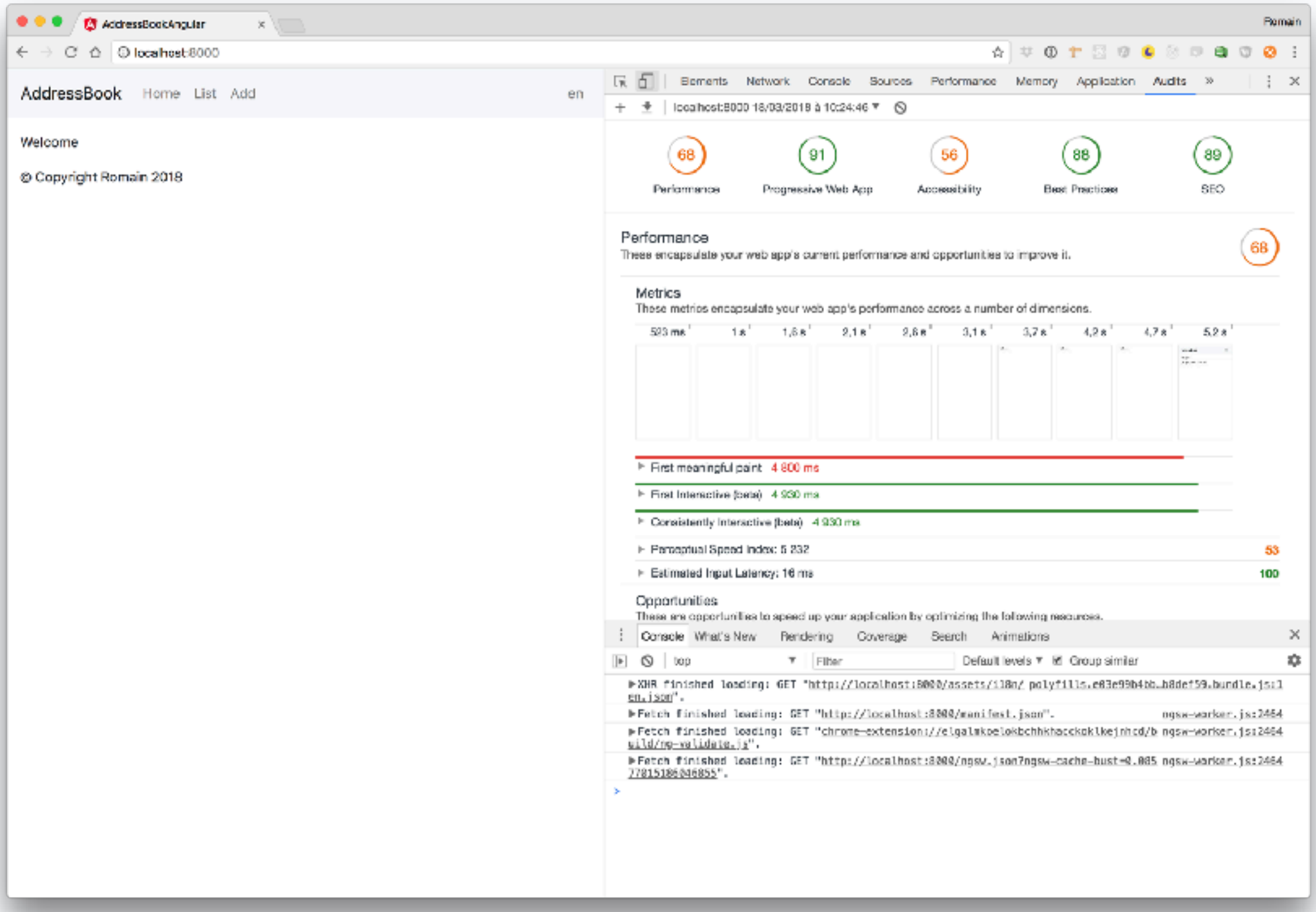
```
<link rel="manifest" href="/manifest.json">
<meta name="theme-color" content="#212529"/>
```

# PWA - Lighthouse

- Lighthouse
  Chrome inclus un outil permettant de savoir si une application fait le nécessaire pour se dire PWA



# PWA - Lighthouse



# PWA - Resources

- Service worker configuration
  https://angular.io/guide/service-worker-config
- Service Workers in Angular With @angular/service-worker
  https://alligator.io/angular/service-workers/


# RxJS

# RxJS - Resources

- What's with the Subjects in RxJS 5
  https://samvloeberghs.be/posts/whats-with-the-subjects-in-rxjs5