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Mettre à jour Angular - Outdated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  npm outdated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MacBook-Pro:AddressBookAngular romain$ npm outdated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

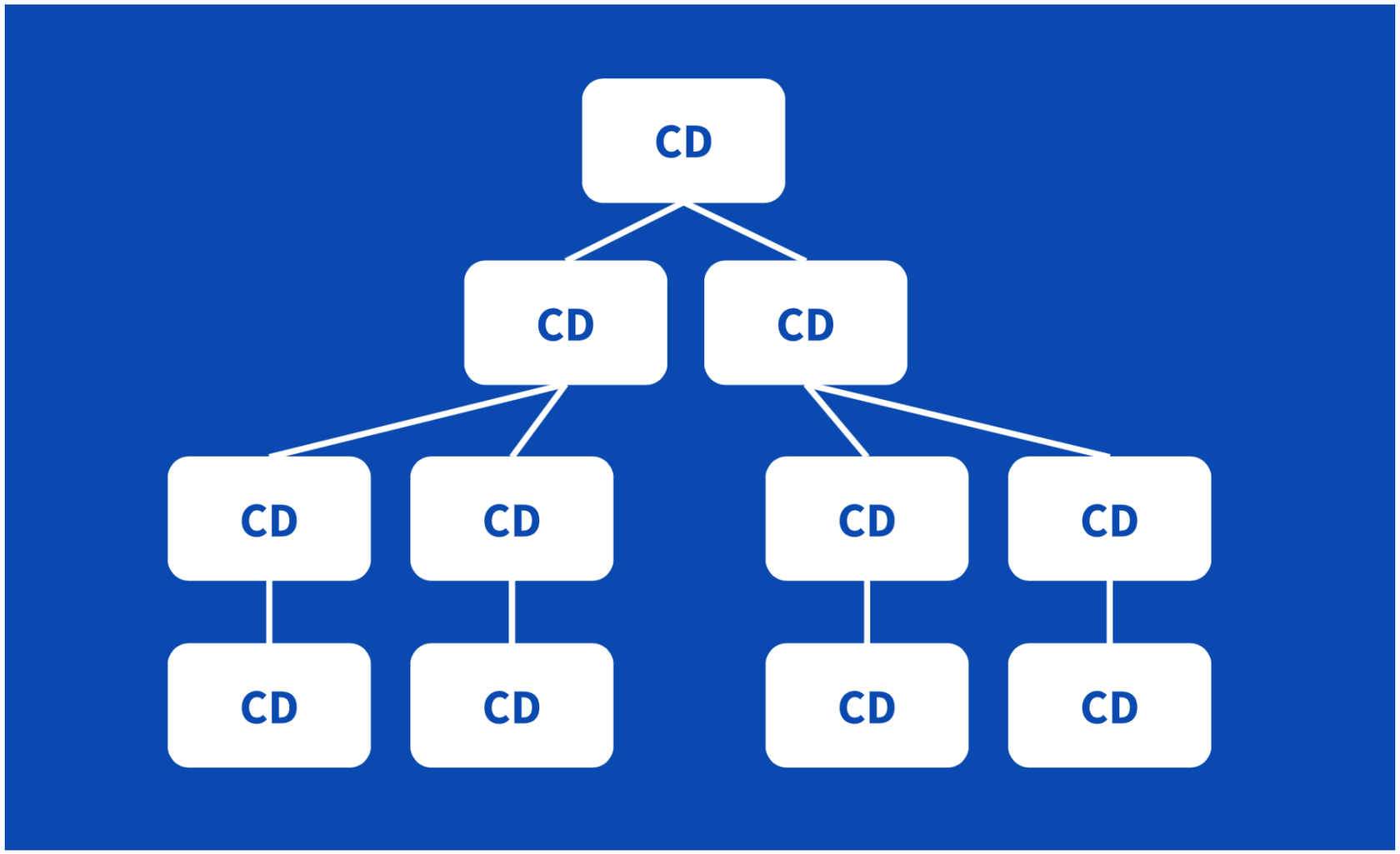
Détection de changement

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

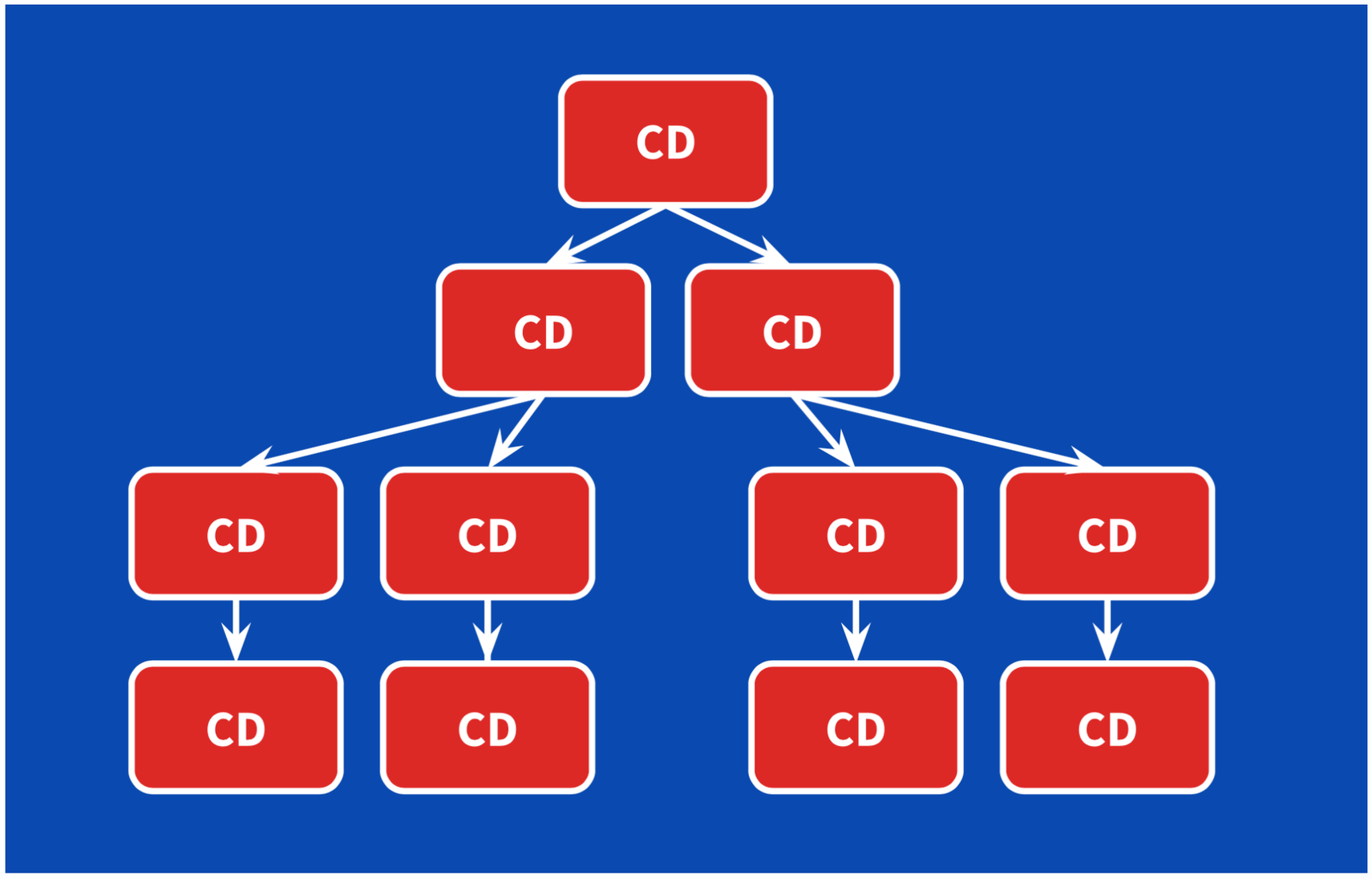
Détection de changement - Déclenchement

- 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

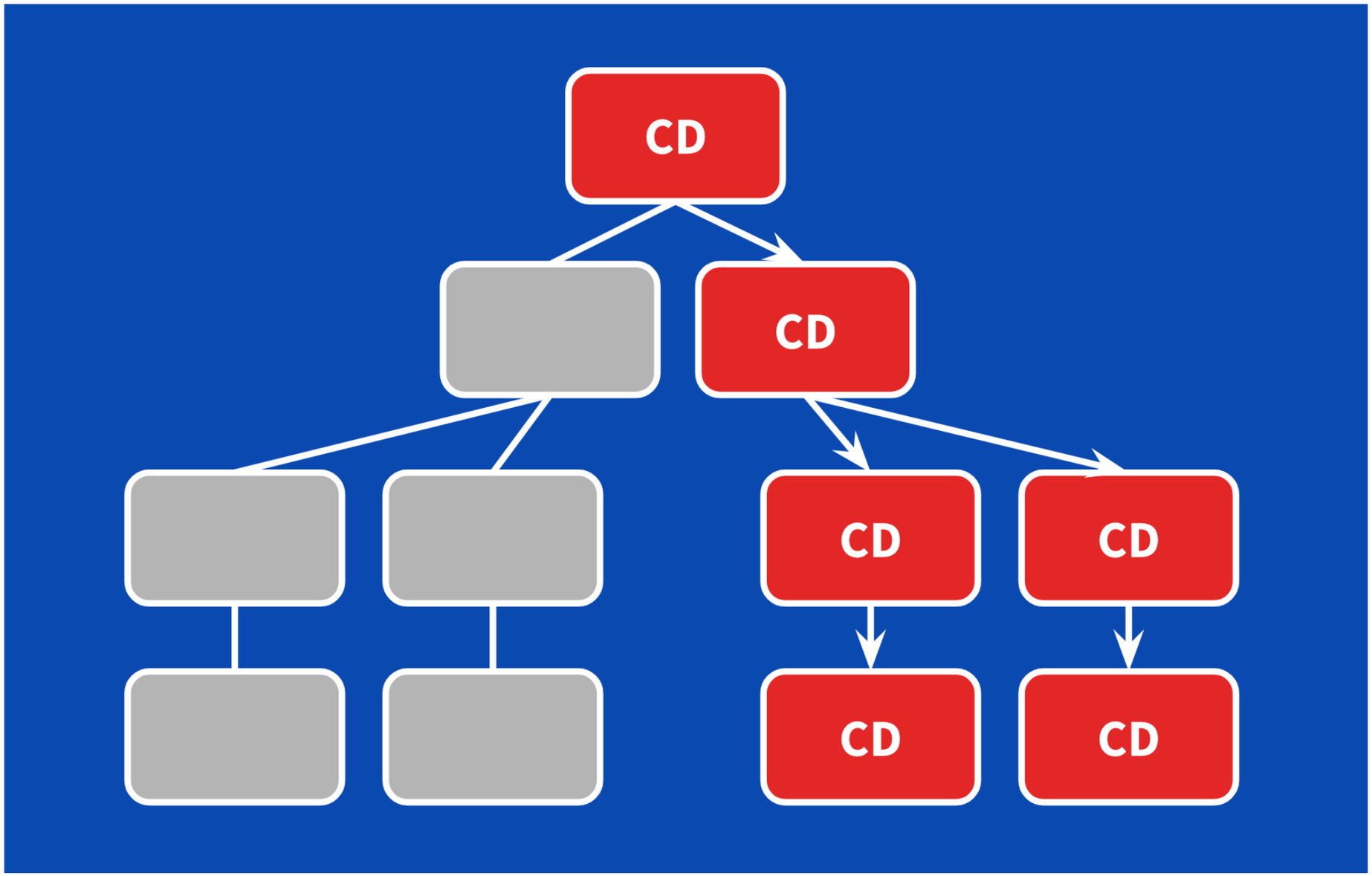


Détection de changement - Interruption

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

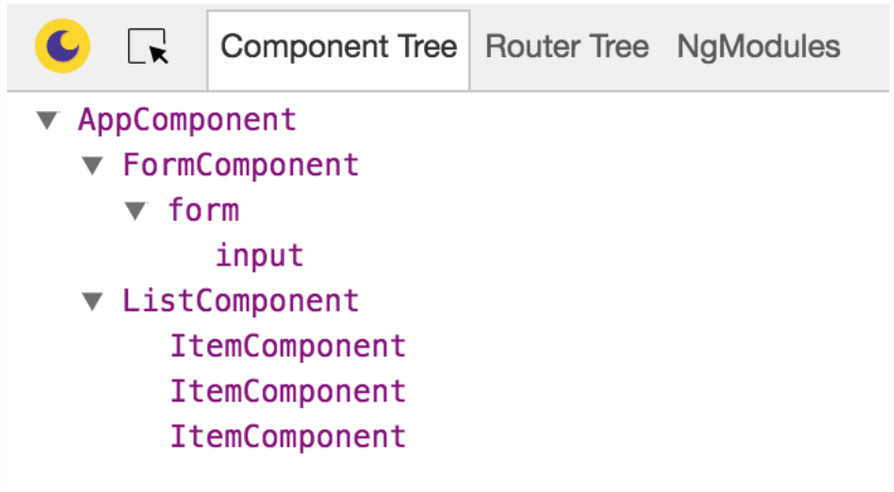
Détection de changement - OnPush

- 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

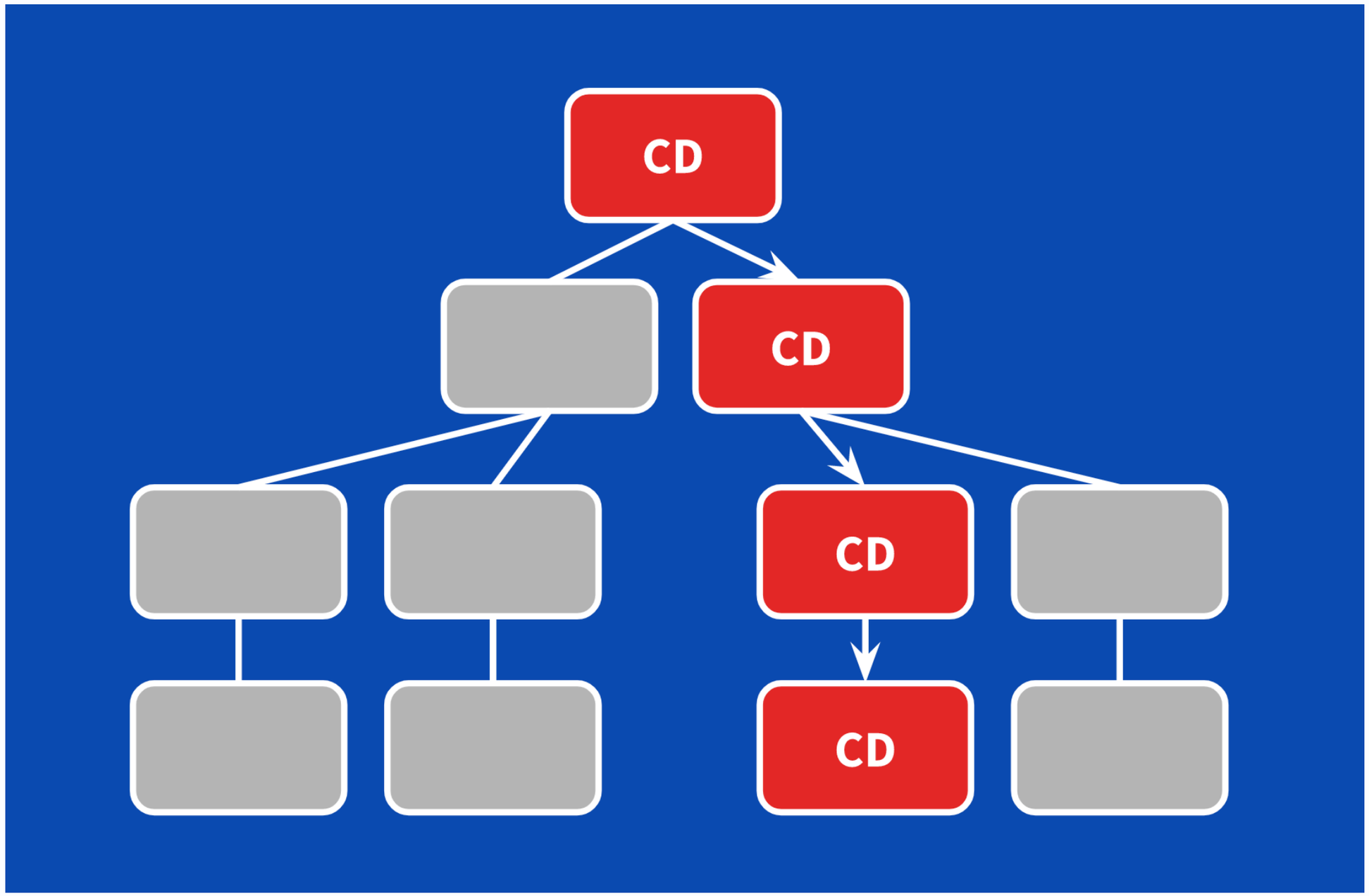
```
export declare abstract class ChangeDetectorRef {
    abstract markForCheck(): void;
    abstract detach(): void;
    abstract detectChanges(): void;
    abstract checkNoChanges(): void;
    abstract reattach(): void;
}
```

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/

formation.tech

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

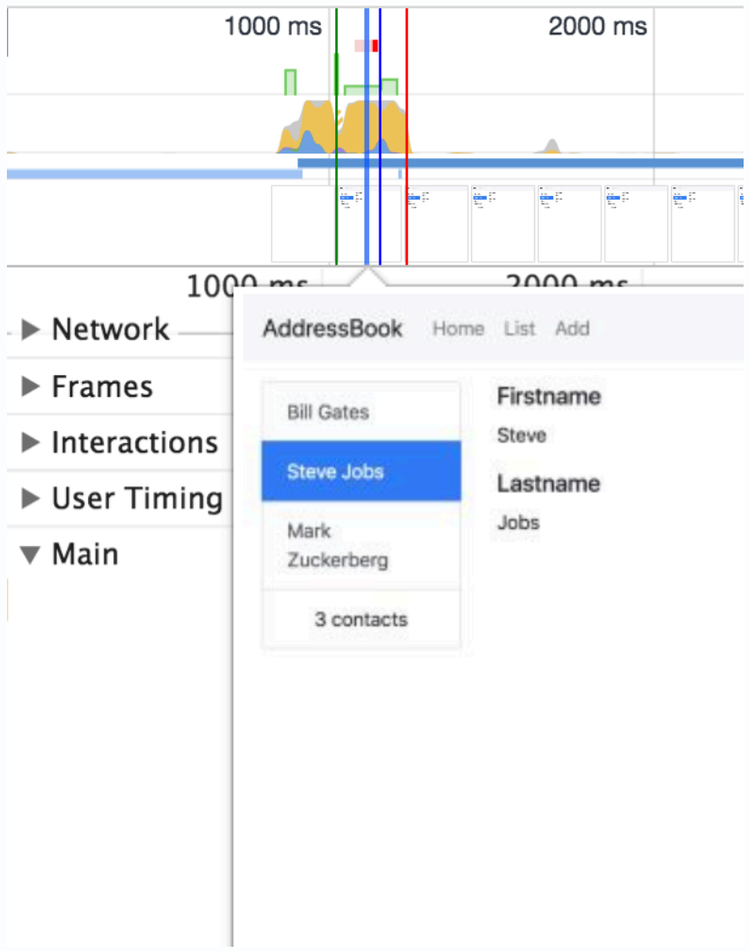
Angular Universal - Serveur Express

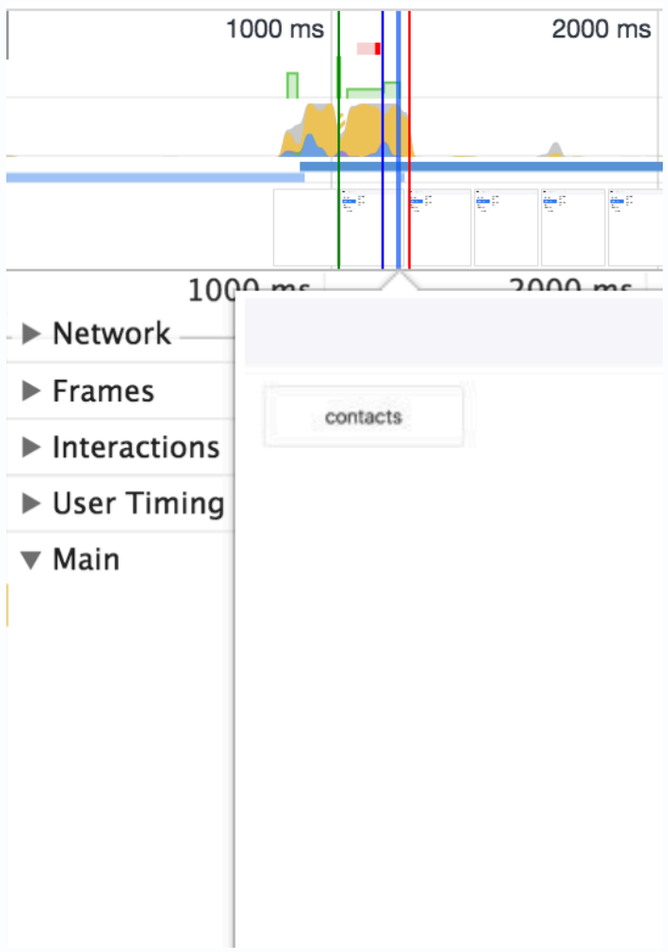
- Installer express et le moteur universal pour express
  npm i express @nguniversal/express-engine

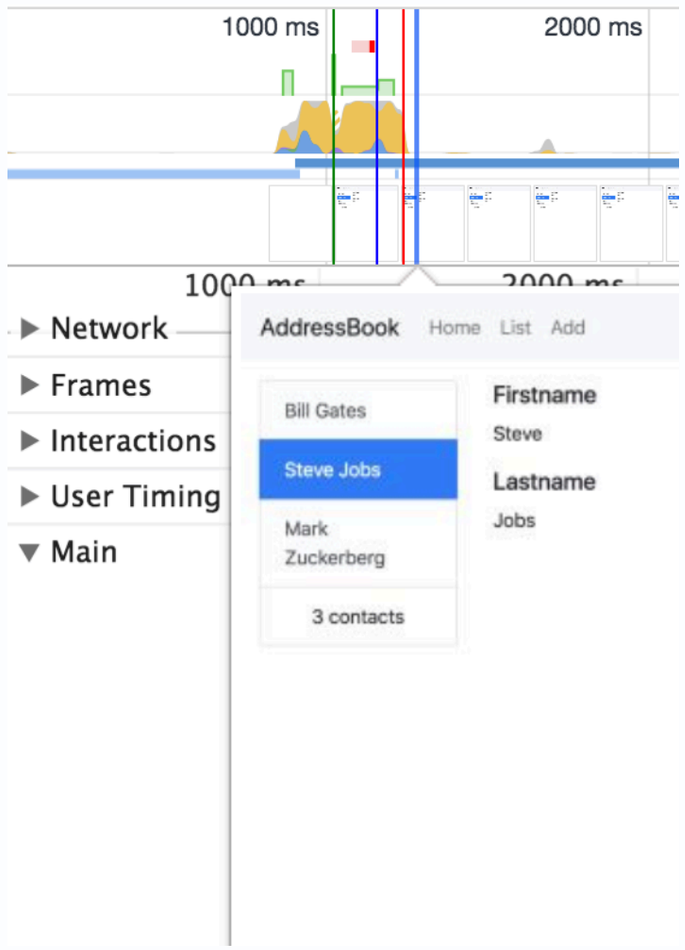
```
require('zone.js/dist/zone-node');

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

formation.tech

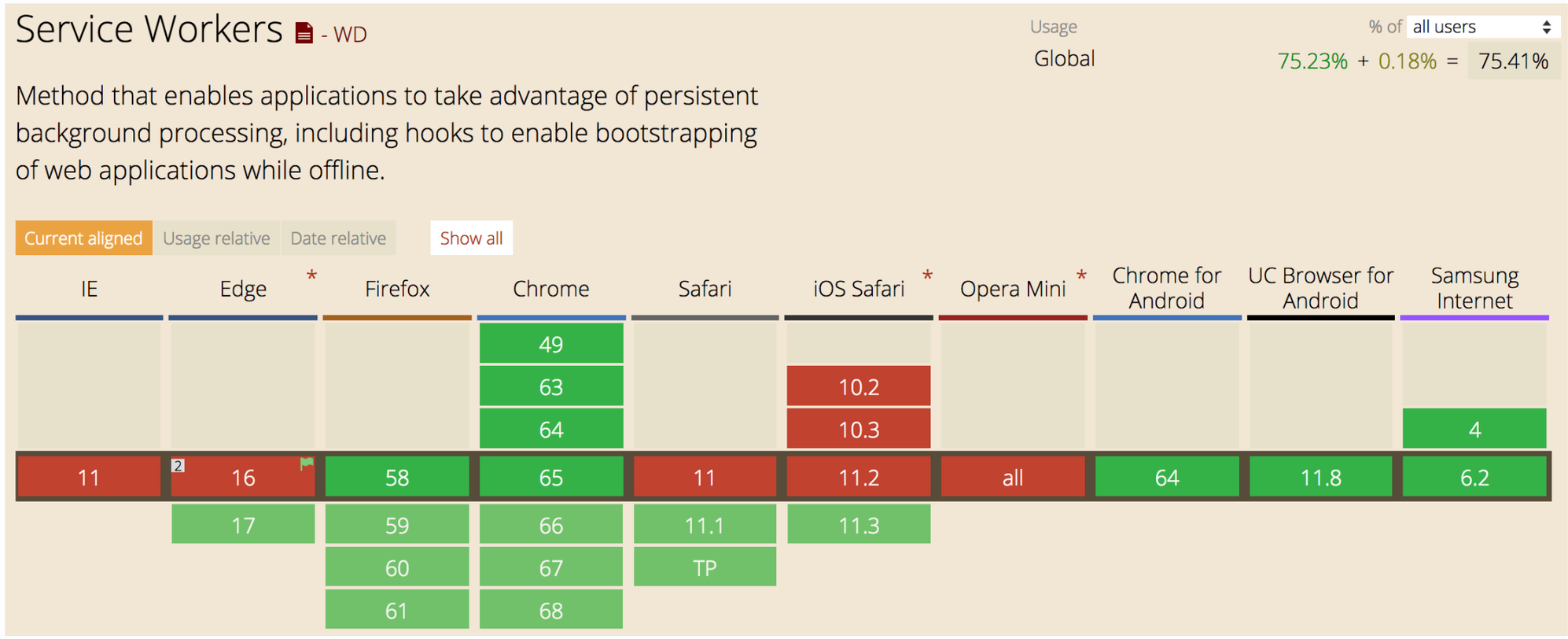
Progressive Web Apps

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)

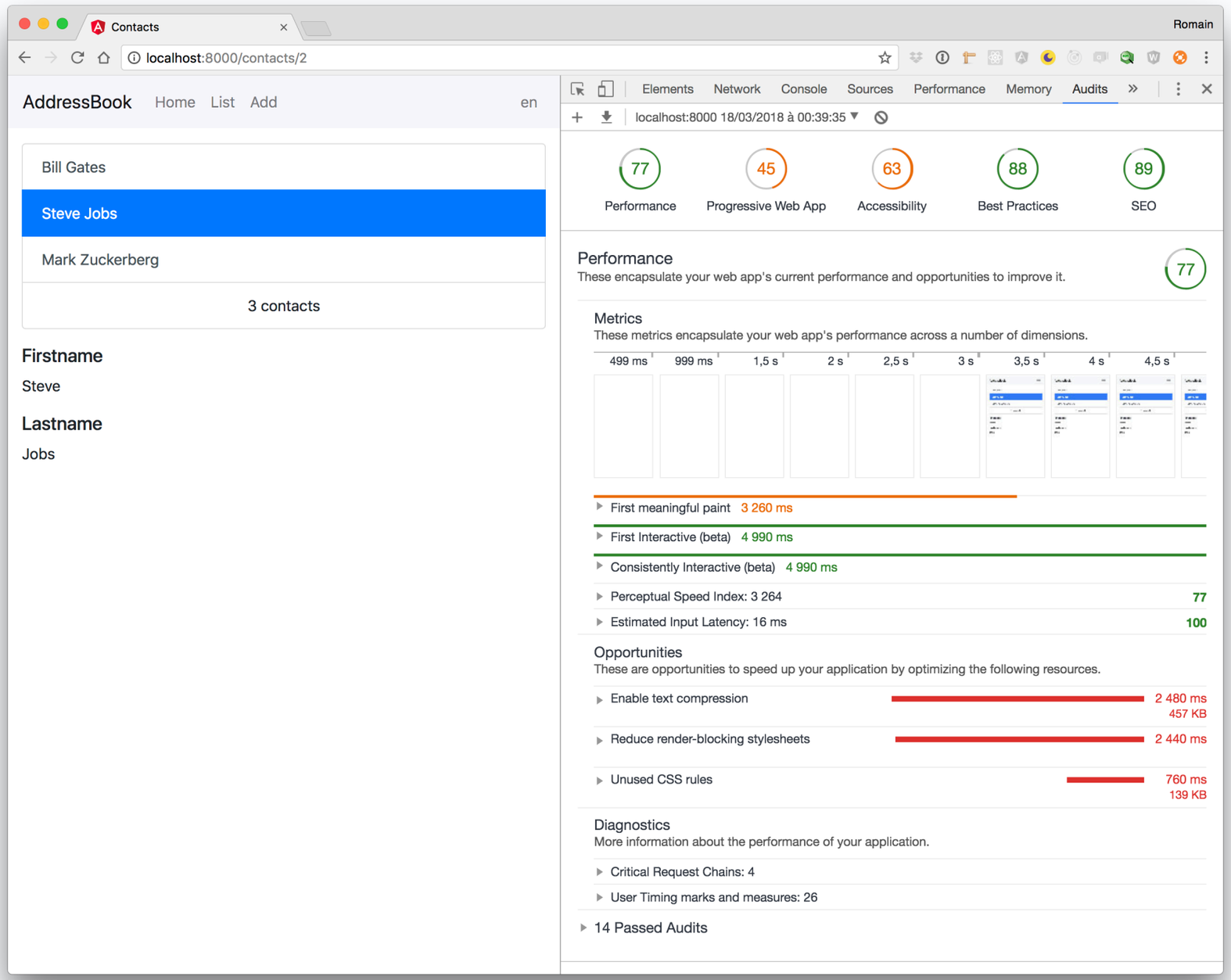
```
{
  "short_name": "Address Book",
  "name": "Address Book Angular Avancé",
  "start_url": "/",
  "theme_color": "#212529",
  "background_color": "#f8f9fa",
  "display": "standalone",
  "orientation": "portrait",
  "icons": [
    {
      "src": "/assets/icons/android-chrome-512x512.png",
      "sizes":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

- Ajouter au fichier index.html

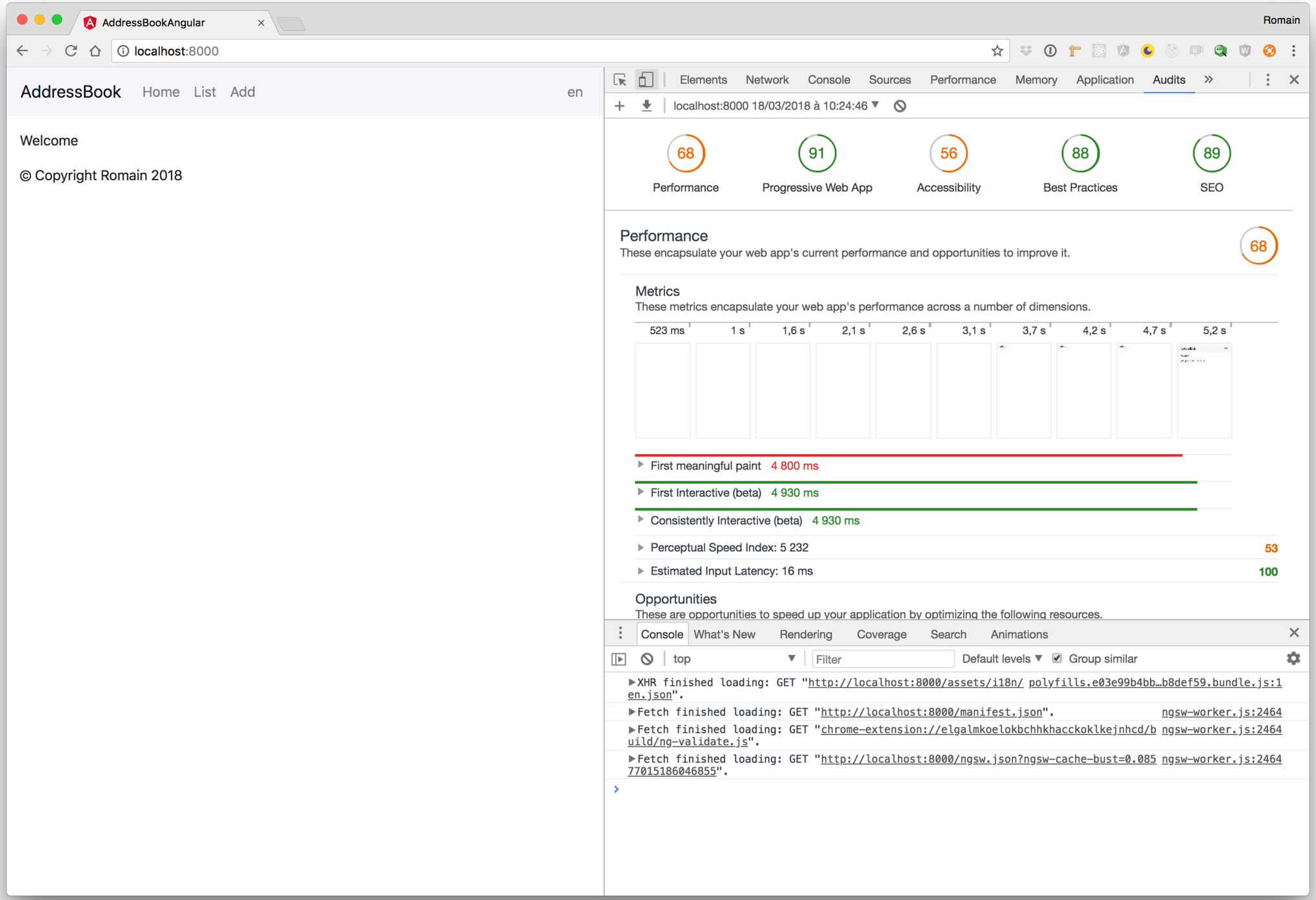
```
<link rel="manifest" href="/manifest.json">
<meta name="theme-color" content="#212529"/>
```

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

-