ORSYS formation

Formation ANGULAR Développement Avancé

Animé par Michel BOCCIOLESI

Table des matières

Application finale réalisée

- https://dev.webjs.fr/1-Angular/website2

- Projet source à télécharger – cahier d'exercices : https://dev.webjs.fr/1-Angular/CAHIERS/

Framework Orienté ou Progressif

Framework ou librairies ?

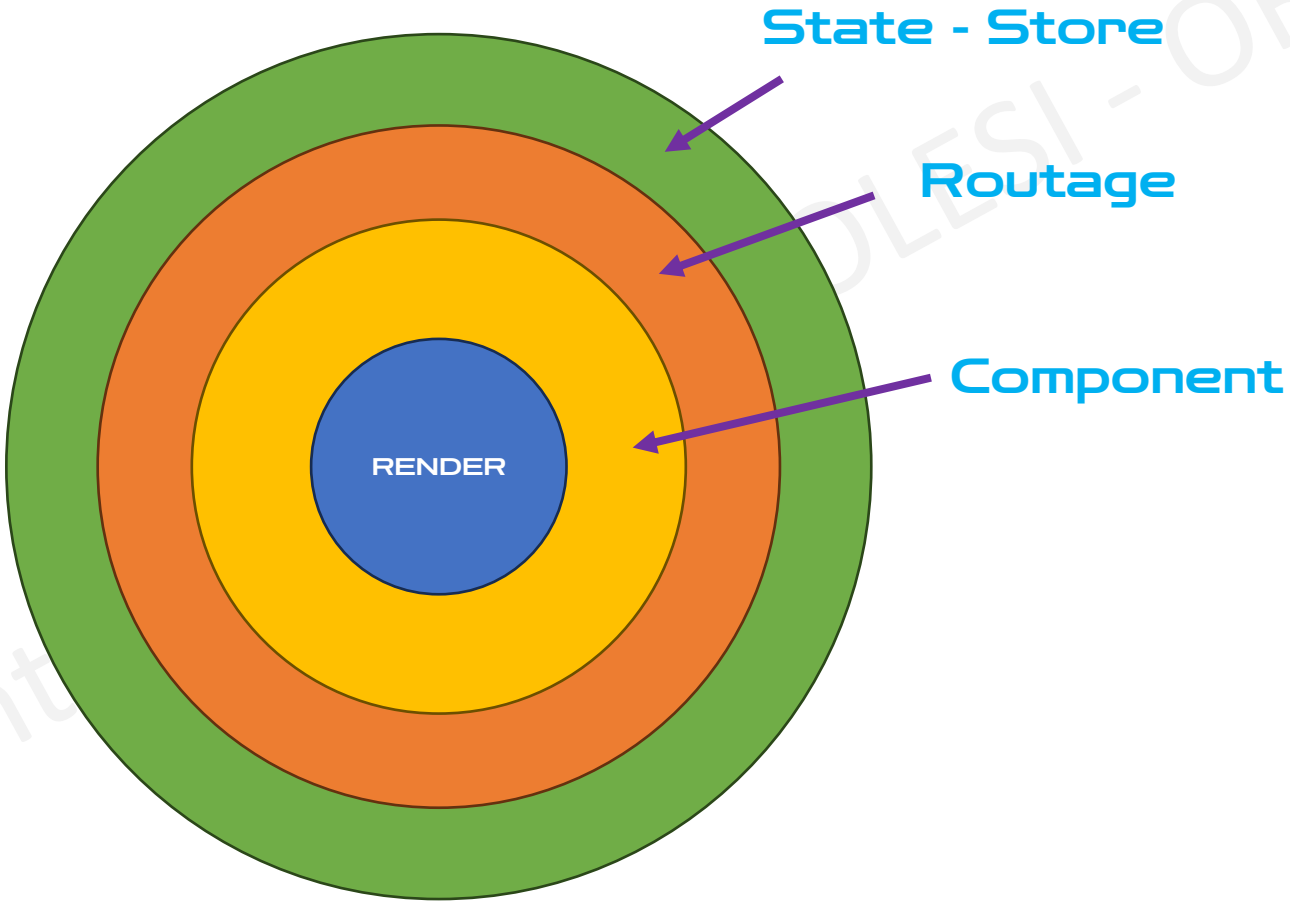
Dans le monde du développement Front, il existe plusieurs catégories de frameworks :

- Angular (2016) est un framework « orienté »
- React (2013) est une librairie
  mais peut être piloté et utilisé par d'autres framework (« NextJS » ou « Vite »)
- Vue (2014) est un framework « progressif [1]» .
  Il peut cependant être utilisé en tant que librairie. 🤔
  Il possède ses propres outils de création npm create vue ou Vite [2]

[1] *Le projet peut être simple initialement et peut facilement évoluer vers un projet beaucoup plus complexe, grâce à sa modularité*

[2] *Vite a été développé en **2020 par Evan You** le créateur de **Vue, Vite et Vitest***

Framework progressif



Histoire du Web

- 2003 : Création du WHATWG
- 2007 : HTML5 et les API JS – le tout 1er smartphone est présenté
- 2010 : Responsive Web Design
- 2011 : Bibliothèque CSS Bootstrap
- 2015 : le développement Front devient très important, les frameworks, le webpack sont de plus en plus utilisés

EcmaScript : la normalisation de Javascript

- 1999 : version 3
- 2009 : version 5 (la 4 n'avait existé)
- 2015: version 6
- 2016
- 2017
- 2018
- 2019
- ES Next

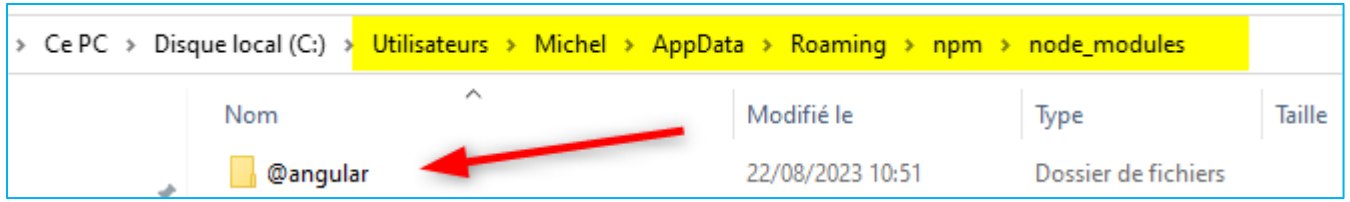
Installation et configuration

Installation et configuration

https://nodejs.org
https://angular.dev/
https://www.npmjs.com/package/@angular/cli

- npm install **-g** @angular/cli (global dans le path utilisateur)

  /home/michel (linux)



Node.JS

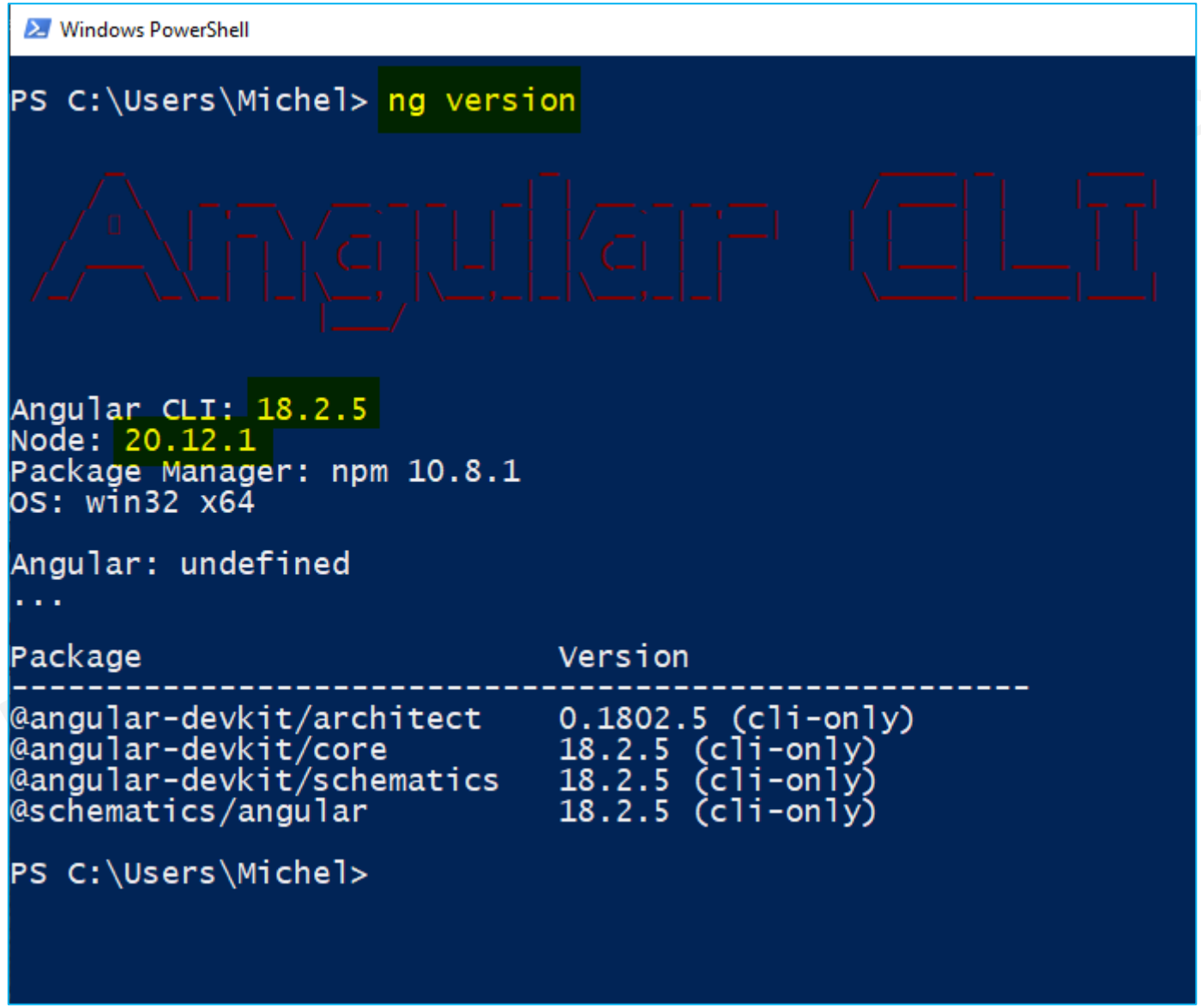
Tous ces frameworks utilisent NODEJS (2009), un serveur javascript développé par Google.

NodeJS permet de travailler son projet Web en tant que WEBPACK.

C'est beaucoup plus facile de structurer, d'architecturer un projet avec un webpack.

NB : Même en Vanilla, on peut développer avec un Webpack. La toute 1ère étape est de créer un projet avec la commande **npm init** (-y)

https://nodejs.org/fr



Depuis la version 18, le mode standalone est proposé par défaut, si vous souhaitez utiliser le mode module 🙄 ajouter l'option - - standalone=false

```
ng new projet-avec-module --standalone=false
```

```
PS C:\Users\Michel> ng new projet-standalone-ng-18
? Which stylesheet format would you like to use? CSS                [ https://developer.mozilla.org/docs/Web/CSS
? Do you want to enable Server-Side Rendering (SSR) and Static Site Generation (SSG/Prerendering)? no
```

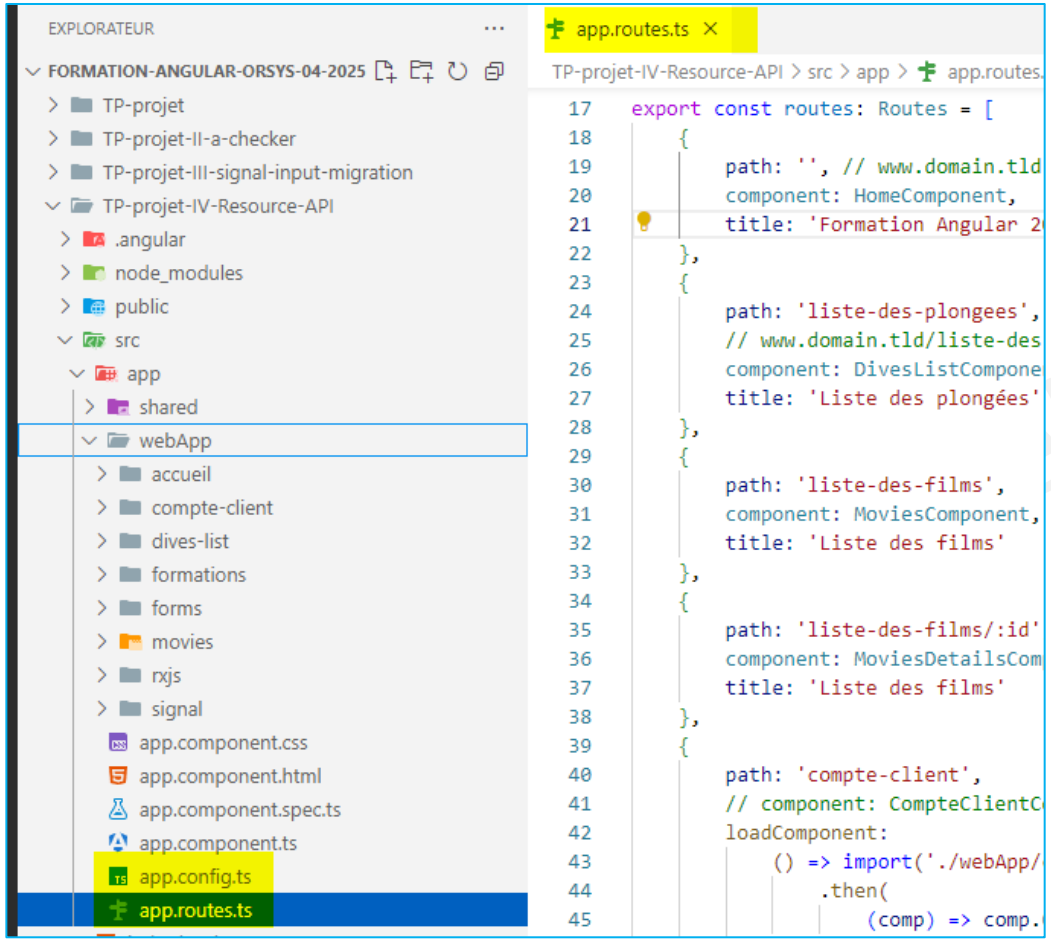
```
PS C:\Users\Michel> npx @angular/cli@14 new projet-angular-14
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? CSS
```

- npx @angular/cli@14 new nomDuProjet

```
PS C:\Users\Michel\projet-angular-14> ls


    Répertoire : C:\Users\Michel\projet-angular-14


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        30/09/2024     10:37                .vscode
d-----        30/09/2024     10:38                node_modules
d-----        30/09/2024     10:37                src
-a----        30/09/2024     10:37            600 .browserslistrc
-a----        30/09/2024     10:37            274 .editorconfig
-a----        30/09/2024     10:37            548 .gitignore
-a----        30/09/2024     10:37           2977 angular.json
-a----        30/09/2024     10:37           1434 karma.conf.js
-a----        30/09/2024     10:38         479274 package-lock.json
-a----        30/09/2024     10:37           1050 package.json
-a----        30/09/2024     10:37           1070 README.md
-a----        30/09/2024     10:37            287 tsconfig.app.json
-a----        30/09/2024     10:37            863 tsconfig.json
-a----        30/09/2024     10:37            333 tsconfig.spec.json


PS C:\Users\Michel\projet-angular-14>
```

Les versions marquantes 15-20

Version 15 : Le mode Standalone



Version 15 (16 novembre 2022)

- Plus besoin des modules
- Le composant est autonome
- Il importe tout ce dont il a besoin
- Le routage est beaucoup plus simple (un seul fichier à la racine du projet)
- L'application ne bootstrappe plus un module mais un composant « point d'entrée »

Les versions marquantes Angular

- Version 16 (12 mai 2023)

- Les Signals
- Change Detection Change
- Encore besoin de Zone.JS ?
- Avenir de RXJS ?

Angular 17 - La renaissance !

Version 17 (8 novembre 2023) https://angular.dev/

- Le control flow est totalement revu
*ngIf et *ngFor sont remplacés par @if et @for

- Webpack est remplacé par ESBUILD (ng server et nb build iront 2 fois plus vite !

- SSR par défaut (Server Side Rendering = > SEO …)

- Lazy Loading (Vues différées) @defer @placeholder

- Signal est abouti en tant qu'API de réactivité

Angular 18

Version 18 (23 mai 2024) https://angular.dev/

- Utilisation des signaux entre composants
- hydratation dans Angular DevTools
- 18.1 : @let 💫 @let movies = movies$ | async | filter ..

Angular 19

19 novembre 2024
https://angular.dev/reference/releases

- LinkedSignal
  https://angular.dev/guide/signals/linked-signal#
- Ressource
  https://angular.dev/guide/signals/resource#resource-loaders

New Refactoring -> Input & Output -> Signal Use

Angular 19

- New Refactoring -> @ViewChild => viewChild Signal

```
ng g @angular/core:signal-queries-migration
```

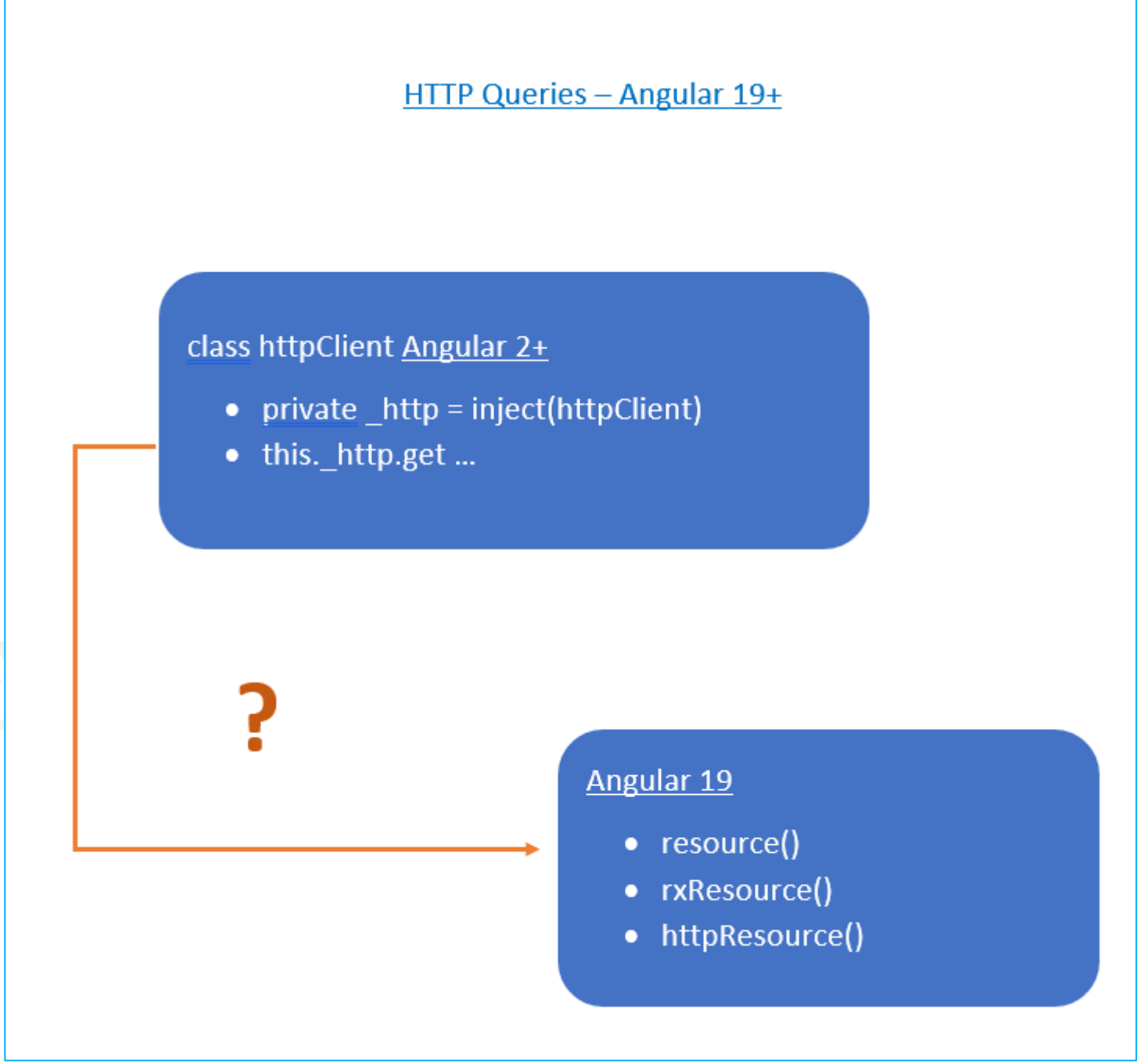
- New Refactoring -> Input & Output -> Signal Use

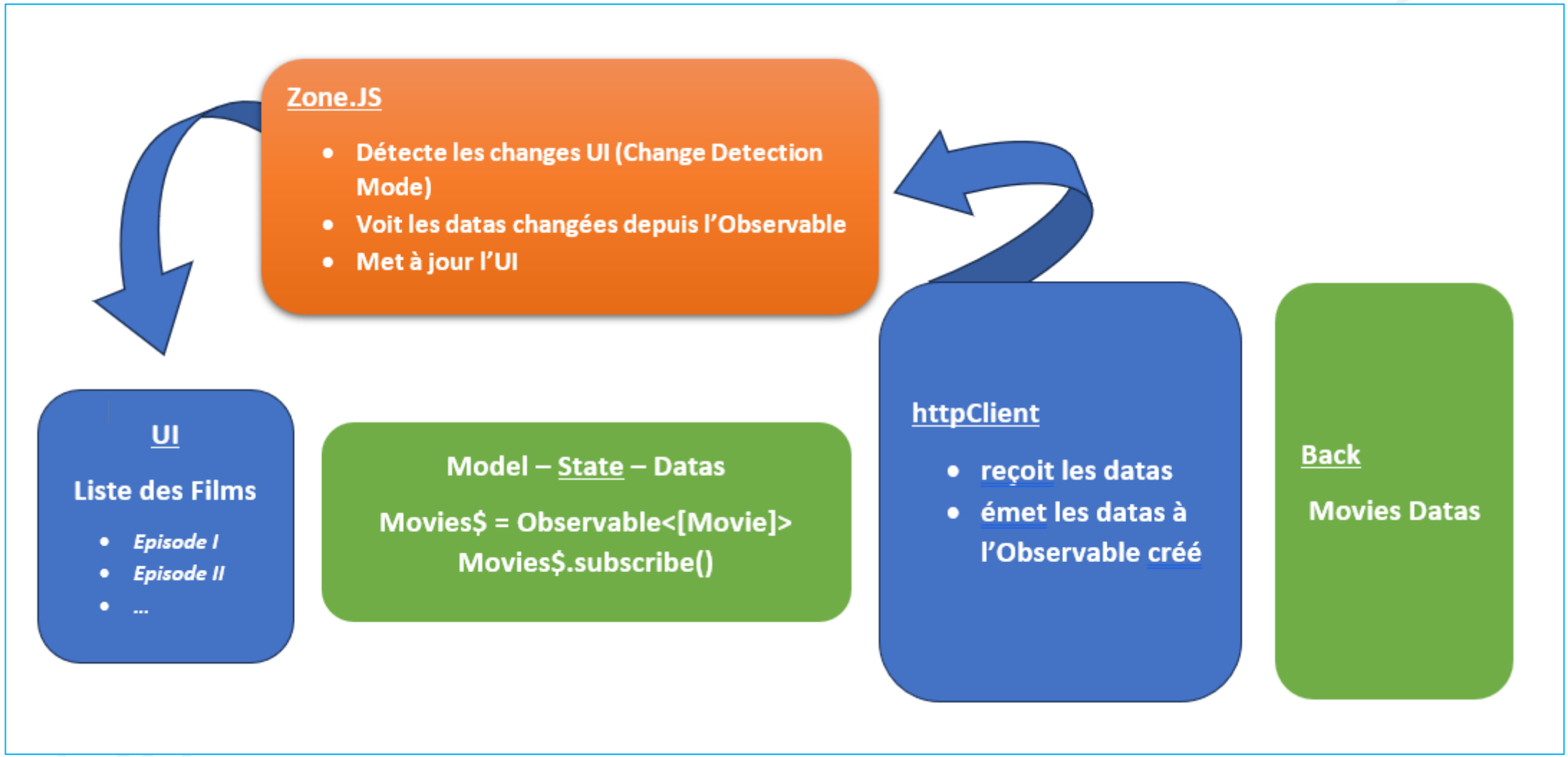
```
ng g @angular/core:signal-input-migration
```

```
ng g @angular/core:output-migration
```

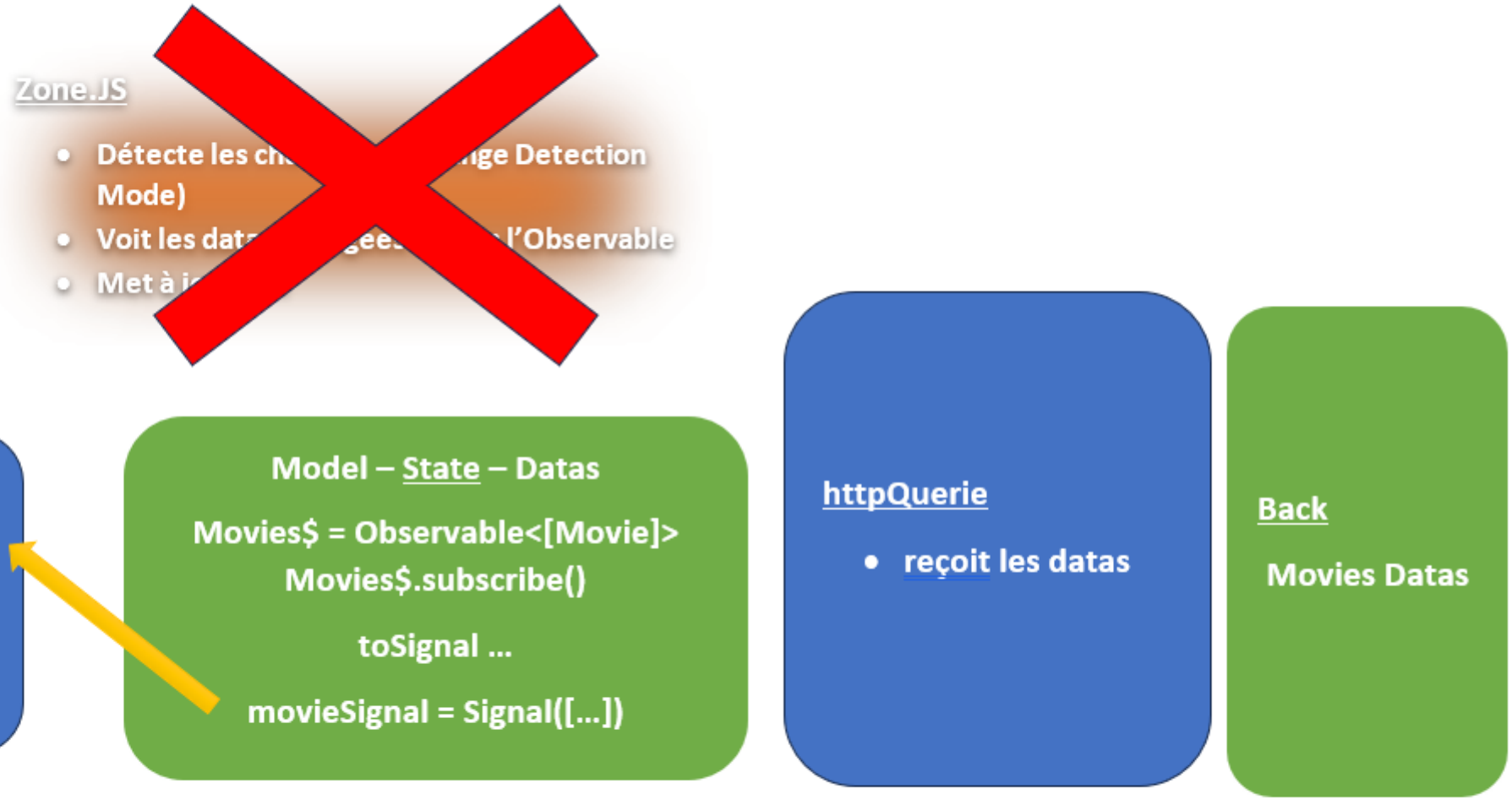
Angular 19 – API Resource

API Resource Pourquoi ?



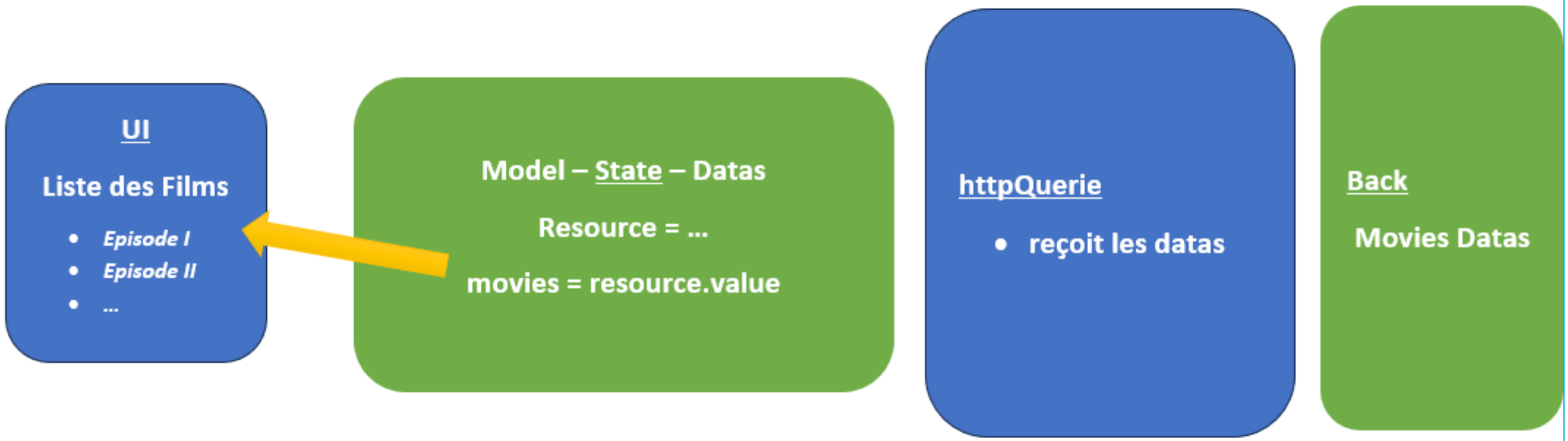

Zone.JS
Détecte les changes UI (Change Detection Mode)
Voit les datas changées depuis l'Observable
Met à jour l'UI
UI
Liste des Films
Episode I
Episode II
...
Model – State – Datas
Movies$ = Observable<[Movie]>
Movies$.subscribe()
httpClient
reçoit les datas
émet les datas à l'Observable créé
Back
Movies Datas

SIGNAL & RESOURCE API



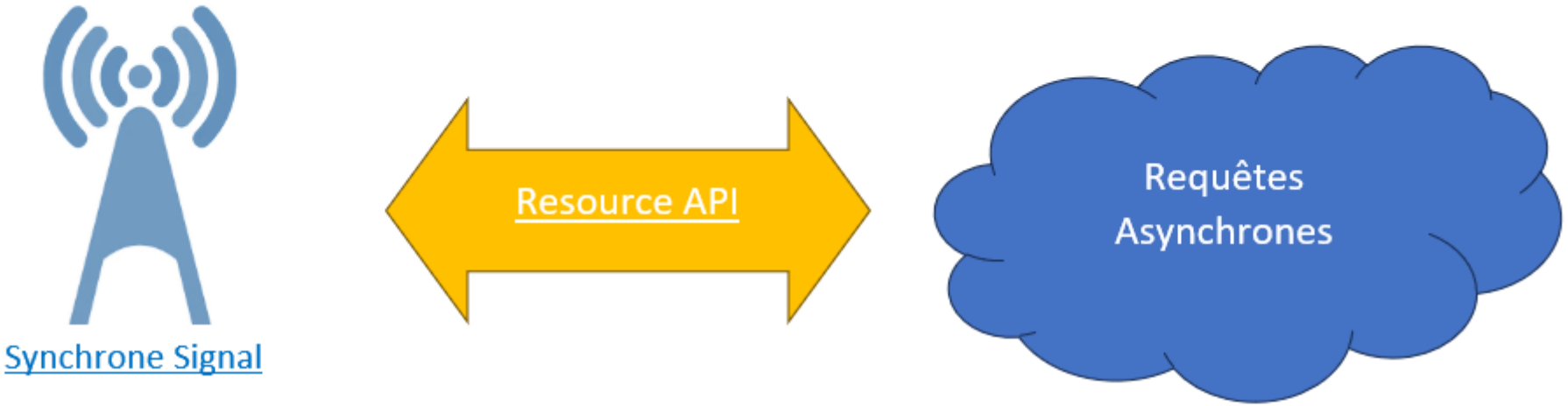
Le signal (SYNCHRONE) n'utilise plus Zone.JS et peut mettre à jour directement l'UI. Mais Angular 19+ a introduit une solution encore plus pratique (Resource API), qui est une sorte de pont entre la requête asynchrone http et le signal qui est synchrone ! Plus besoin de Subscription, les datas peuvent être reçues directement par le resource signal.

SIGNAL & RESOURCE API



Le signal (SYNCHRONE) n'utilise plus Zone.JS et peut mettre à jour directement l'UI. Mais Angular 19+ a introduit une solution encore plus pratique (Resource API), qui est une sorte de pont entre la requête asynchrone http et le signal qui est synchrone ! Plus besoin de Subscription, les datas peuvent être reçues directement par le resource signal.

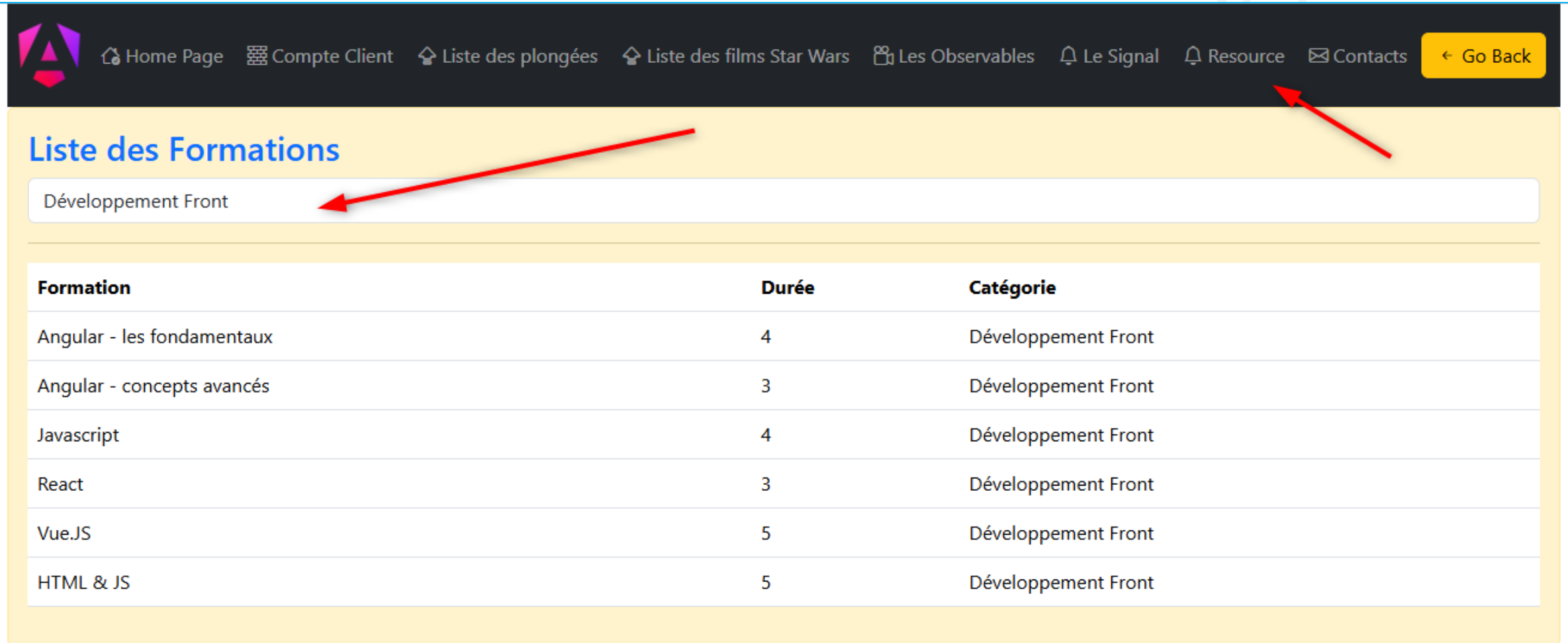
Resource API



Le but de Resource API est de récupérer directement les datas dans un signal, sans utiliser Zone.JS. Resource API offre 3 méthodes d'utilisation au choix :

- resource() : utilise le fetch de javascript et les promesses
- rxResource() : utilise la programmation RXJS, les operators et les Observables (map, mergeMap, forkJoin)
- httpResource() : Angular 19.2 : plus facile et plus souple, utilise une syntaxe de Query http,

TP : Mise en Pratique



formations.service.ts

src > app > shared > services > formations.service.ts > FormationsService

```ts
import { computed, inject, Injectable, resource, signal } from '@angular/cc
import { HttpClient, HttpErrorResponse } from '@angular/common/http';
import { FORMATIONS_API_URL } from '@shared/env/globals';
import { Formation } from '@shared/interfaces/Formation';
import { setErrorMessage } from '@shared/errors/error-http-msg';


@Injectable({ providedIn: 'root' })

export class FormationsService {

    private _url = FORMATIONS_API_URL;
    public formationModels = signal<string[]>([
        'Programmation', 'Développement Front', 'Développement Back', 'Fror
    ]);
    public formationSelected = signal<string>('');
    // ------------------------------------------

    // 1- resource() avec paramètres

    private formationsResource = resource({
        request: this.formationSelected,
        loader: (param) => fetch(`${this._url}?category=${param.request}`)
            .then(
                (res: any) => {
                    console.log(res);
                    return res.json() as Promise<Formation[]>

                }

            )
    });

    public formations = this.formationsResource.value

    public error = computed(
        () => this.formationsResource.error() as HttpErrorResponse
    )
    public errorMessage = computed(
        () => setErrorMessage(this.error(), 'Formations'));

    public isLoading = this.formationsResource.isLoading;

}
```

formations.component.ts

src > app > webApp > formations > formations.component.ts > FormationsComponent

```ts
import { AfterViewChecked, Component, inject } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { FormationsService } from '@app/shared/services/formations.service';

@Component({
  selector: 'app-formations',
  imports: [FormsModule],
  templateUrl: './formations.component.html',
  styleUrl: './formations.component.css'
})
export class FormationsComponent {

  private _FormationsService = inject(FormationsService);

  // Signals envoyés au Template HTML
  public formationModels = this._FormationsService.formationModels
  public formationsSelected = this._FormationsService.formationSelected

  public formations = this._FormationsService.formations
  public errorMessage = this._FormationsService.error;
  public isLoading = this._FormationsService.isLoading;


}
```

Le template HTML

formations.component.html

src > app > webApp > formations > formations.component.html > div.container.alert.alert-warning > div

Go to component

```
<div class="container alert alert-warning">
    <h3 class="text-primary">Liste des Formations</h3>

    @if (isLoading()) {
    <div>... loading formations</div>
    }
    @else if (errorMessage()){
    <div style='color: red'>An error occurred: {{ errorMessage() }}</div>
    }
    @else {
    <div>
        <select [(ngModel)]="formationsSelected" class="form-control">

            <option value="" selected>Choisir une formation</option>

            @for(formation of formationModels(); track formation) {
            <option>{{ formation}}</option>
            }

        </select>
    </div>

    <hr>

```

formations.component.html

src > app > webApp > formations > formations.component.html > div.container.alert.ale

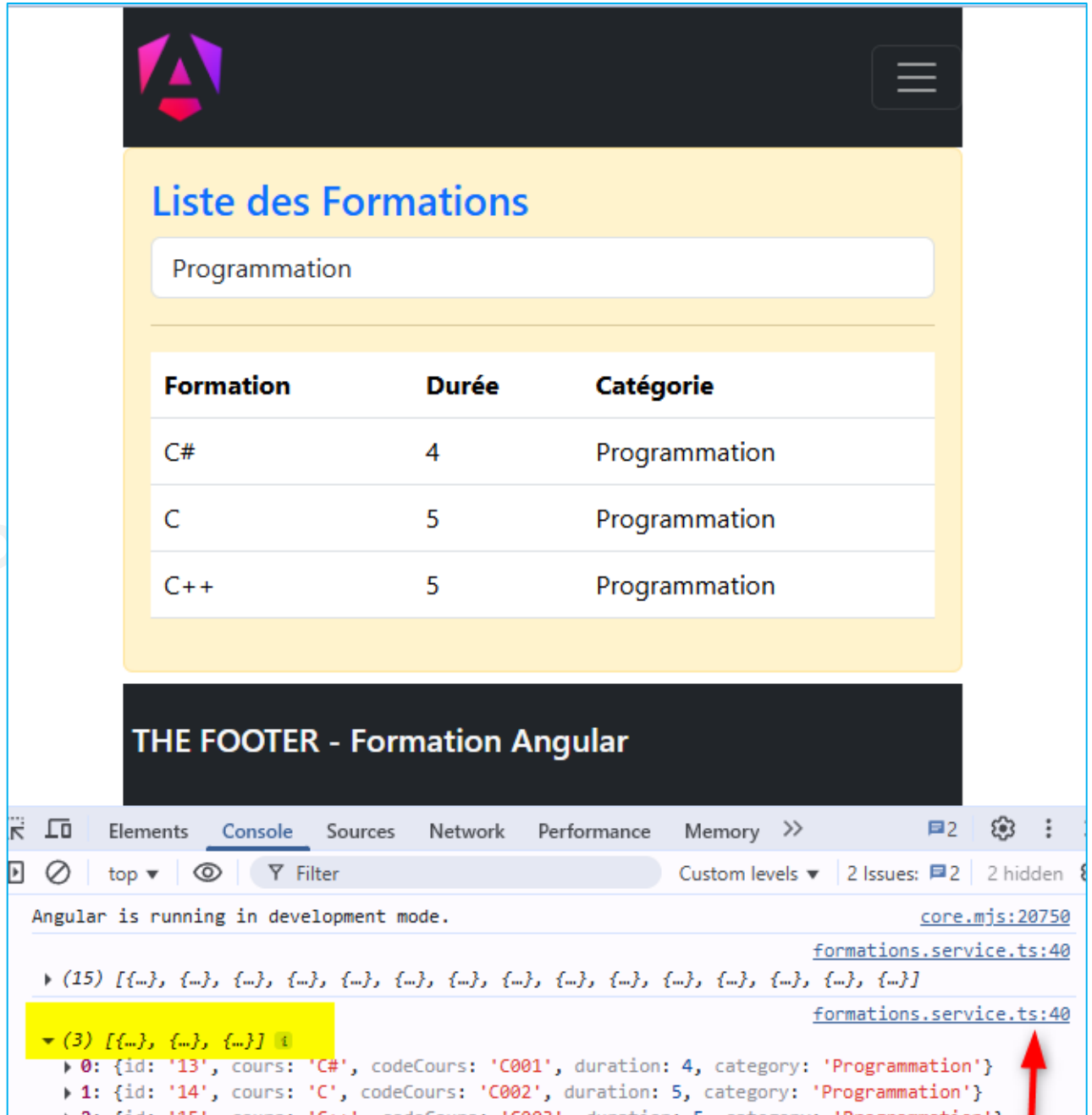
```
<div class="container alert alert-warning">

    <hr>

    @if (formations()) {
    <table class="table ">
        <thead>
            <tr>
                <th scope="col">Formation</th>
                <th scope="col">Durée</th>
                <th scope="col">Catégorie</th>
            </tr>
        </thead>

        <tbody>
            @for (f of formations(); track f) {
            <tr>
                <td> {{ f.cours }} </td>
                <td> {{ f.duration }} </td>
                <td> {{ f.category }} </td>
            </tr>
            }
        </tbody>
    </table>
    }
    @else {
    <div>Pas de formations</div>
    }

    }
</div>
```

Avec RxResource (Observables et opérateurs RXJS)

formations.service.ts

src > app > shared > services > formations.service.ts > FormationsService

```ts
@Injectable({ providedIn: 'root' })

export class FormationsService {

    private _url = FORMATIONS_API_URL;
    public formationModels = signal<string[]>([
        'Programmation', 'Développement Front', 'Développement Back', 'Front CSS
    ]);
    public formationSelected = signal<string>('');
    // ------------------------------------------

    // 1- resource() avec paramètres :
    // private formationsResource = resource({
    //     request: this.formationSelected,
    //     loader: (param) => fetch(`${this._url}?category=${param.request}`)
    //         .then(
    //             (res: any) => {
    //                 console.log(res);
    //                 return res.json() as Promise<Formation[]>;
    //             }
    //         )
    // });

    // 2- rxResource() avec paramètres
    private _http = inject(HttpClient)
    private formationsResource = rxResource({
        request: this.formationSelected,
        loader: (param) => this._http.get<Formation>(`${this._url}?category=${pa
            .pipe(
                tap(
                    (valObs) => console.log(valObs)
                )
            )
    });

    // ------------------------------------------------
    public formations = this.formationsResource.value

    public error = computed(
        () => this.formationsResource.error() as HttpErrorResponse
    )
```



Avec httpResource (Angular 19.2)

formations.service.ts | package-lock.json

src > app > shared > services > formations.service.ts > FormationsService > formationsResource

```
12  export class FormationsService {
34      // private _http = inject(HttpClient)
35      // private formationsResource = rxResource({
36      //     request: this.formationSelected,
37      //     loader: (param) => this._http.get<Formation>(`${this._url}?categ
38      //         .pipe(
39      //             tap(
40      //                 (valObs) => console.log(valObs)
41      //             )
42      //         )
43      // });
44
45      // 3- httpResource() avec paramètres (Juste avec l'URL 😉)
46      // private formationsResource = httpResource<Formation>(
47      //     () => `${this._url}?category=${this.formationSelected()}`);
48
49      // 4- httpResource() avec paramètres dans un OBJET
50      private formationsResource = httpResource<Formation>({
51
52          url: this._url,
53          method: 'get',
54          headers: { accept: 'application/json' },
55          params: { category: this.formationSelected(), }
56      })
57
58      // ------------------------------------------------
59      public formations = this.formationsResource.value
60
61      public error = computed(
62          () => this.formationsResource.error() as HttpErrorResponse
63      )
64      public errorMessage = computed(
65          () => setErrorMessage(this.error(), 'Formations'));
66
67      public isLoading = this.formationsResource.isLoading;
68
69  }
70
```

Angular 20

28 Mai 2025
https://angular.dev/reference/releases

```
PS C:\Users\Michel> ng new NG20
√ Do you want to create a 'zoneless' application without zone.js (Developer Preview)? Yes
√ Which stylesheet format would you like to use? CSS              [ https://developer.mozilla.org/docs/Web/CSS
√ Do you want to enable Server-Side Rendering (SSR) and Static Site Generation (SSG/Prerendering)? Yes
CREATE NG20/angular.json (2470 bytes)
CREATE NG20/package.json (1150 bytes)
CREATE NG20/README.md (1526 bytes)
CREATE NG20/tsconfig.json (1026 bytes)
```

Zoneless

app.config.ts M

TP19 > src > app > app.config.ts > appConfig > providers

```ts
import { ApplicationConfig, provideZoneChangeDetection } from '@angular/
import { provideRouter } from '@angular/router';

import { routes } from './app.routes';

export const appConfig: ApplicationConfig = {
  providers: [
    provideZoneChangeDetection({ eventCoalescing: true }),
    provideRouter(routes)
  ]
};

```

app.config.ts M

TP20 > src > app > app.config.ts > ...

```ts
import { ApplicationConfig, provideBrowserGlobalErrorListeners
import { provideRouter } from '@angular/router';

import { routes } from './app.routes';

export const appConfig: ApplicationConfig = {
  providers: [
    provideBrowserGlobalErrorListeners(),
    //  provideZoneChangeDetection({ eventCoalescing: true }),
    provideZonelessChangeDetection(), // NG20
    provideRouter(routes)
  ]
};

```

app.component.ts M

TP19 > src > app > app.component.ts > ...

```ts
import { NgFor } from '@angular/common';
import { afterRender, Component, signal } from '@angular/core';


@Component({
  selector: 'app-root',
  imports: [NgFor],
  templateUrl: './app.component.html',
  styleUrl: './app.component.css'
})
export class AppComponent {
  // 1-props
  public title: string = 'Les Images';
  public imagesArray: string[] = ['avatar1.jpg', 'avatar2.jpg', 'avatar3
  // Zoneless
  public titleSignal = signal<string>('Vive le Signal');


  constructor() {
    afterRender(
      () => {
        // news NG20
        console.log('AfterRender =>', this.title);
      }
    );
```

app.ts M

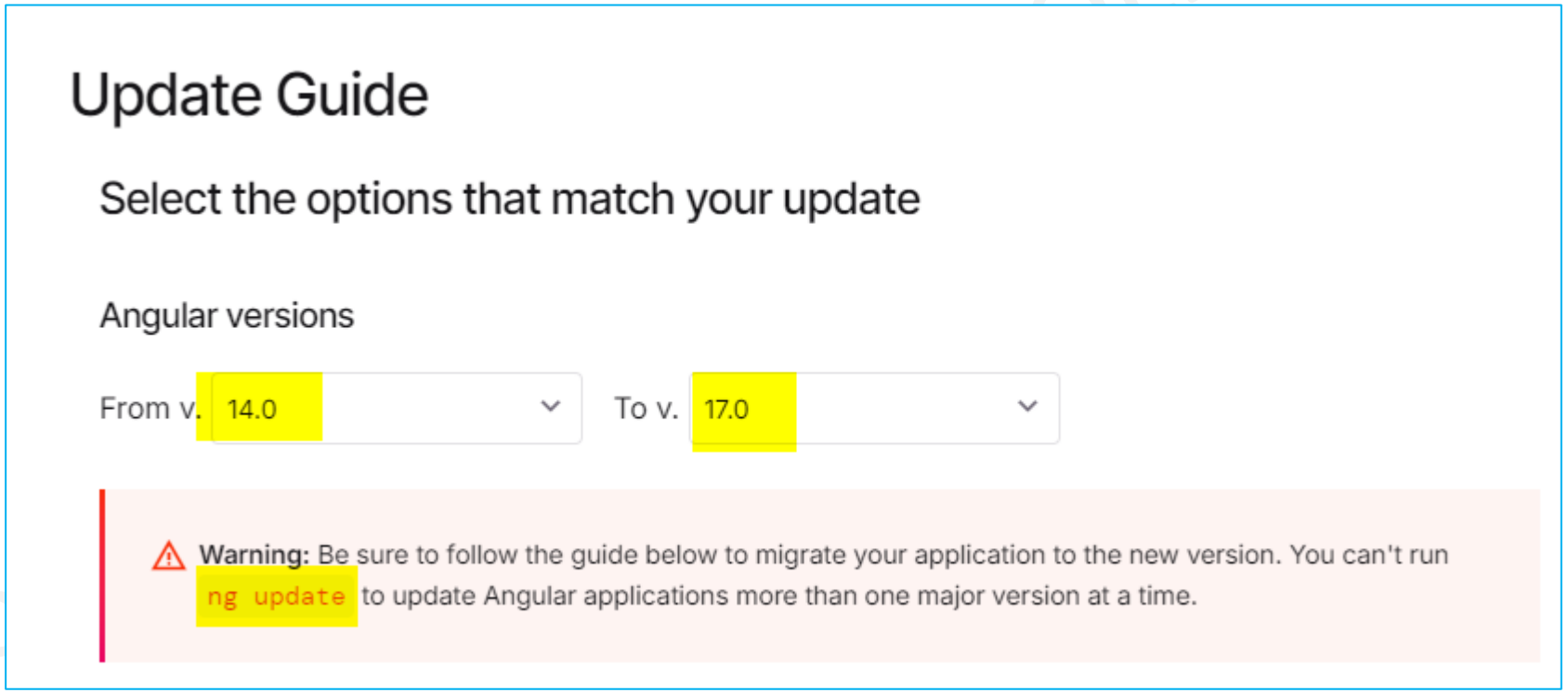
TP20 > src > app > app.ts > App > constructor > afterEveryRender() callback

```ts
import { NgFor } from '@angular/common';
import {  afterEveryRender, Component, signal } from '@angular


@Component({
  selector: 'app-root',
  imports: [NgFor],
  templateUrl: './app.html',
  styleUrl: './app.css'
})
export class App  {
  // 1-props
  public title: string = 'Les Images';
  public imagesArray: string[] = ['avatar1.jpg', 'avatar2.jpg'
  // Zoneless
  public titleSignal = signal<string>('Vive le Signal');


  constructor(){
    afterEveryRender(
      () => {
        // news NG20
        console.log('AfterEveryRender =>', this.title);
      }
    );
```

Les structural directives *ngIf *ngFor *ngSwitch sont « deprecated »
« Removed » in V22 😬

Migration entre versions majeures

- Pour mettre à jour une version majeure d'Angular 😌, suivons les conseils d'Angular 😊
- https://angular.dev/update-guide
- ng update @angular/core @angular/cli (upgrade d'une version)
- ng update @ngrx/store



Pour mettre à jour une version majeure d'Angular 😌, suivons les conseils d'Angular 😊

Guide to update your Angular application v14.0 → v17.0 for basic applications

Before you update

You don't need to do anything before moving between these versions.

Update to the new version

Review these changes and perform the actions to update your application.

- [ ] Make sure that you are using a supported version of node.js before you upgrade your application. Angular v15 supports node.js versions: 14.20.x, 16.13.x and 18.10.x. Read further
- [ ] Make sure that you are using a supported version of TypeScript before you upgrade your application. Angular v15 supports TypeScript version 4.8 or later. Read further
- [ ] In the application's project directory, run `ng update @angular/core@15 @angular/cli@15` to update your application to Angular v15.
- [ ] In your application's `tsconfig.json` file, remove `enableIvy`. In v15, Ivy is the only rendering engine so `enableIvy` is not required.
- [ ] Make sure that all `ActivatedRouteSnapshot` objects have a `title` property. In v15, the `title` property is a required property of `ActivatedRouteSnapshot`. Read further
- [ ] In v15, `relativeLinkResolution` is not configurable in the Router. It was used to opt out of an earlier bug fix that is now standard. Read further
- [ ] Update instances of `TestBed.inject()` that use an `InjectFlags` parameter to use an `InjectOptions` parameter. The `InjectFlags` parameter of `TestBed.inject()` is deprecated in v15. Read further
- [ ] Using `providedIn: 'any'` for an `@Injectable` or `InjectionToken` is deprecated in v15. Read further

PROBLÈMES SORTIE CONSOLE DE DÉBOGAGE TERMINAL PORTS

```
PS C:\Users\Formateur\Desktop\TP-Standalone-ANY-10-2024-Solutions> ng update @angular/core@19 @angular/cli@19
The installed Angular CLI version is outdated.
Installing a temporary Angular CLI versioned 19.0.1 to perform the update.
Using package manager: npm
Collecting installed dependencies...
Found 40 dependencies.
Fetching dependency metadata from registry...
    Updating package.json with dependency @angular-devkit/build-angular @ "19.0.1" (was "18.2.7")...
    Updating package.json with dependency @angular/cli @ "19.0.1" (was "18.2.7")...
    Updating package.json with dependency @angular/compiler-cli @ "19.0.0" (was "18.2.6")...
    Updating package.json with dependency @angular/localize @ "19.0.0" (was "18.2.6")...
    Updating package.json with dependency ng-packagr @ "19.0.1" (was "18.2.1")...
    Updating package.json with dependency typescript @ "5.6.3" (was "5.4.5")...
    Updating package.json with dependency @angular/animations @ "19.0.0" (was "18.2.6")...
    Updating package.json with dependency @angular/common @ "19.0.0" (was "18.2.6")...
    Updating package.json with dependency @angular/compiler @ "19.0.0" (was "18.2.6")...
    Updating package.json with dependency @angular/core @ "19.0.0" (was "18.2.6")...
    Updating package.json with dependency @angular/forms @ "19.0.0" (was "18.2.6")...
    Updating package.json with dependency @angular/platform-browser @ "19.0.0" (was "18.2.6")...
    Updating package.json with dependency @angular/platform-browser-dynamic @ "19.0.0" (was "18.2.6")...
    Updating package.json with dependency @angular/router @ "19.0.0" (was "18.2.6")...
    Updating package.json with dependency @angular/service-worker @ "19.0.0" (was "18.2.6")...
    Updating package.json with dependency zone.js @ "0.15.0" (was "0.14.10")...
UPDATE package.json (1768 bytes)
✓ Cleaning node modules directory
✓ Installing packages
** Executing migrations of package '@angular/cli' **

❯ Update '@angular/ssr' import paths to use the new '/node' entry point when 'CommonEngine' is detected.
  Migration completed (No changes made).

❯ Update the workspace configuration by replacing deprecated options in 'angular.json' for compatibility with the latest Angular CLI changes.
  Migration completed (No changes made).

** Optional migrations of package '@angular/cli' **

This package has 1 optional migration that can be executed.
Optional migrations may be skipped and executed after the update process, if preferred.

 Select the migrations that you'd like to run (Press <space> to select, <a> to toggle all, <i> to invert selection, and <enter> to proceed)
❯◉ [use-application-builder] Migrate application projects to the new build system. (https://angular.dev/tools/cli/build-system-migration)
```

```
UPDATE src/app/webApp/tests/pipes/see-more.pipe.ts (682 bytes)
  Migration completed (46 files modified).

› Updates ExperimentalPendingTasks to PendingTasks.
  Migration completed (No changes made).

** Optional migrations of package '@angular/core' **

This package has 1 optional migration that can be executed.
Optional migrations may be skipped and executed after the update process, if preferred.

 Select the migrations that you'd like to run (Press <space> to select, <a> to toggle all, <i> to invert selection, and <enter> to proceed)
❯◯ [provide-initializer] Replaces `APP_INITIALIZER`, `ENVIRONMENT_INITIALIZER` & `PLATFORM_INITIALIZER` respectively with `provideAppInitializer`, `provideEnvironmentInitializer` & `providePlatformInitializer`.
```

Webpack

Le webpack est un environnement de développement NODEJS, composé de plusieurs fichiers de configuration permettant de « customiser » le développement (serve) et le build (compilation)

- package.json
- angular.json
- tsconfig.json
- répertoire node_modules *

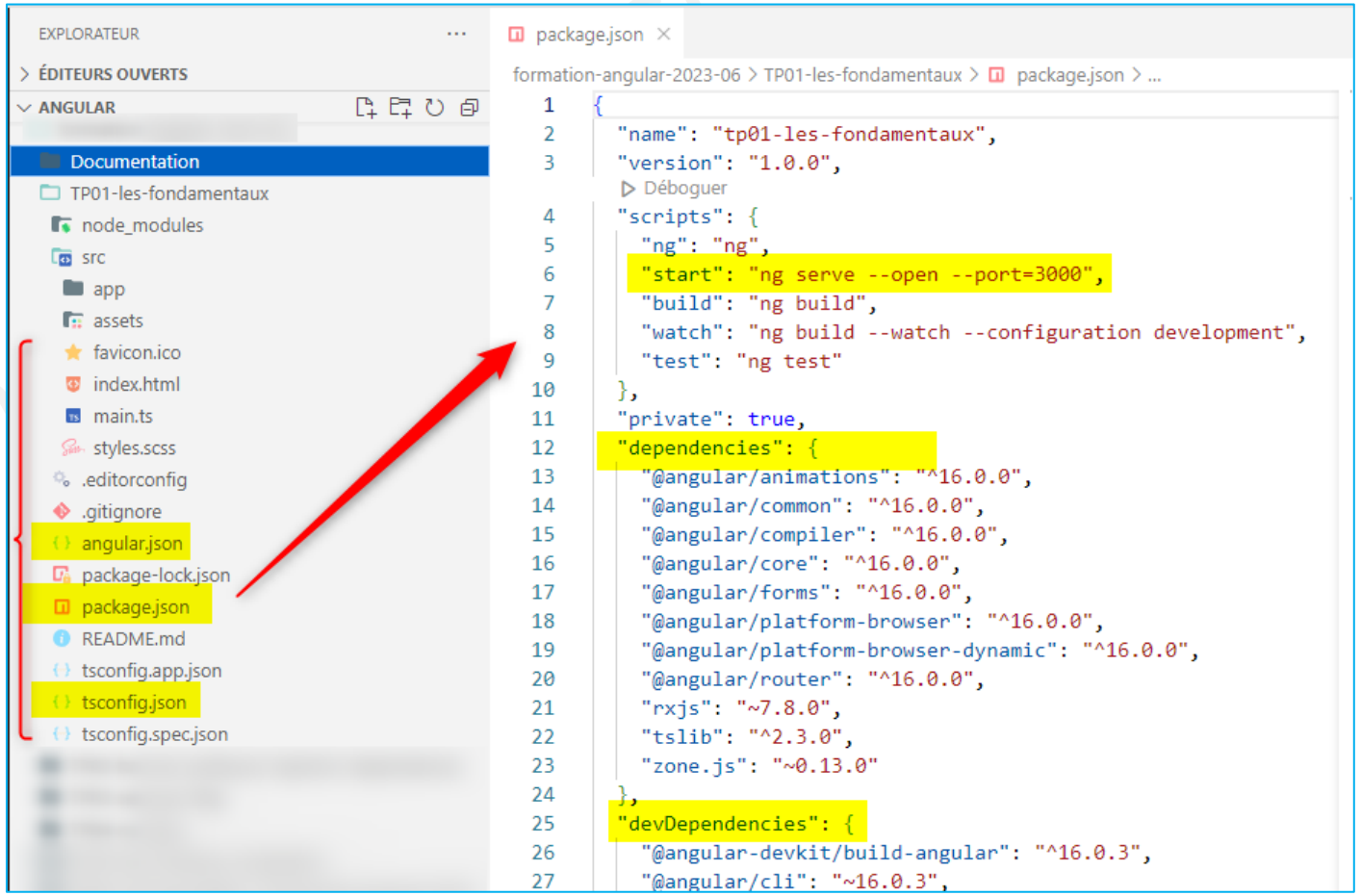
- node_modules * contient toutes les librairies (dépendances) nécessaires au projet webpack.

Il y a au moins 32k fichiers dans un simple projet Angular.

npm init -y : crée un projet webpack

npm update : met à jour les paquets selon les règles du package.json

^ : mise à jour minor + patch
~ : mise à jour patch

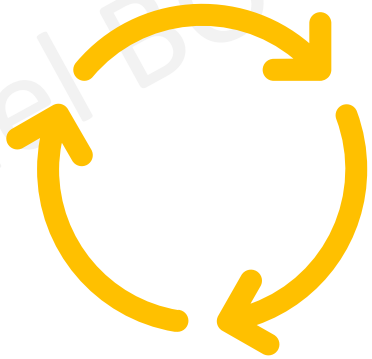


Créer un Webpack

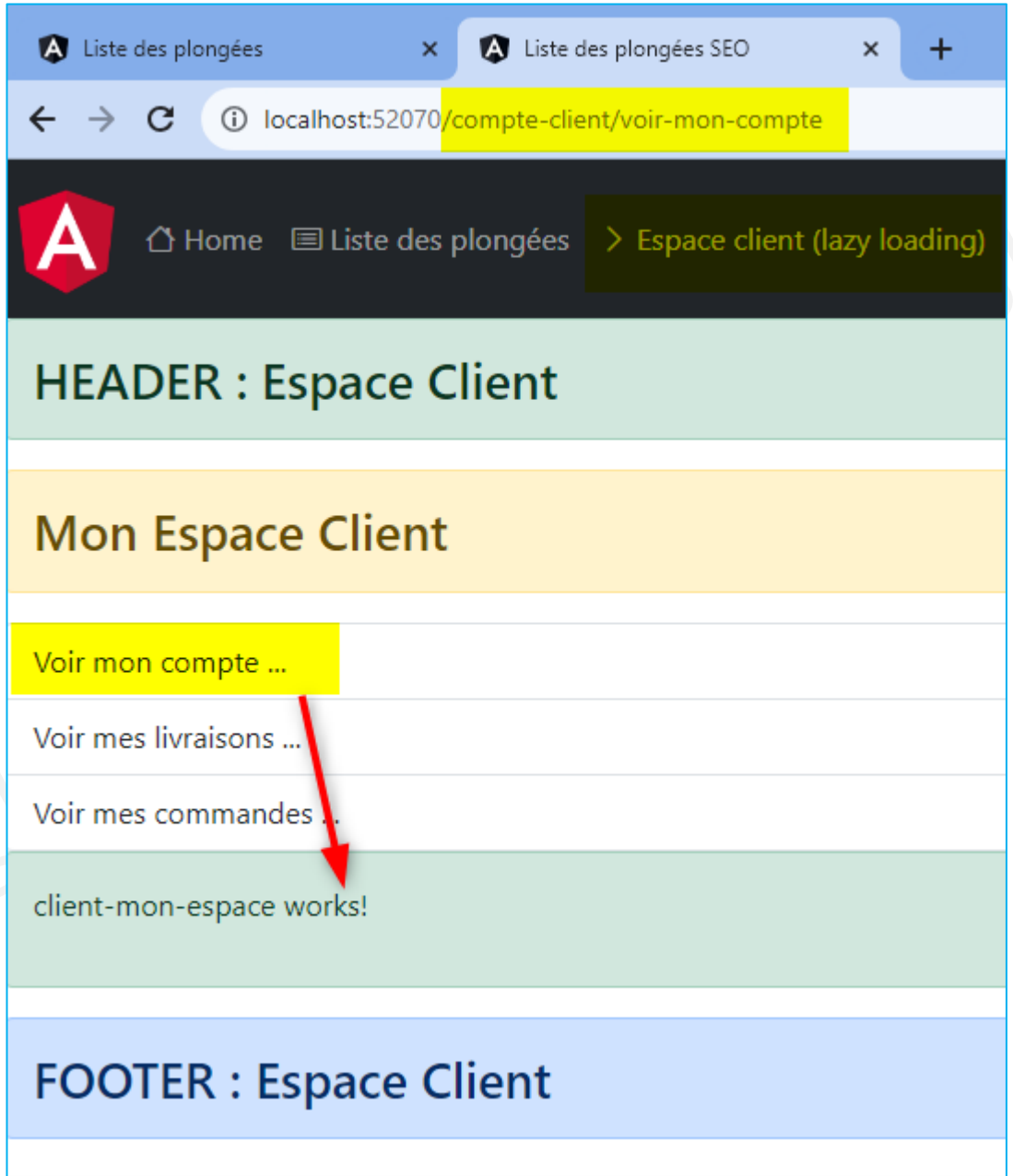
```
PS C:\Users\Formateur> npm init -y
Wrote to C:\Users\Formateur\package.json:

{
  "name": "formateur",
  "version": "1.0.0",
  "description": "ok",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "json-server": "^1.0.0-beta.3"
  },
  "devDependencies": {},
  "keywords": []
}
```

```
PS C:\Users\Formateur> npm create vite
Need to install the following packages:
create-vite@5.5.5
Ok to proceed? (y) y
√ Project name: ... vite-project
? Select a framework: » - Use arrow-keys. Return to submit.
>   Vanilla
    Vue
    React

    Lit
    Svelte
    Solid
    Qwik
    Angular
    Others
```

Routage Avancé
Lazy Loading
Optimisation du build

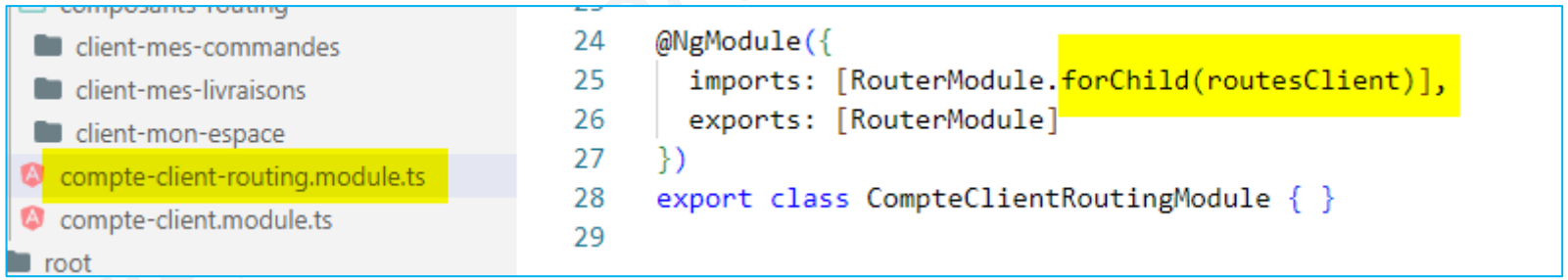


Routing : Concepts avancés – Le Lazy Loading



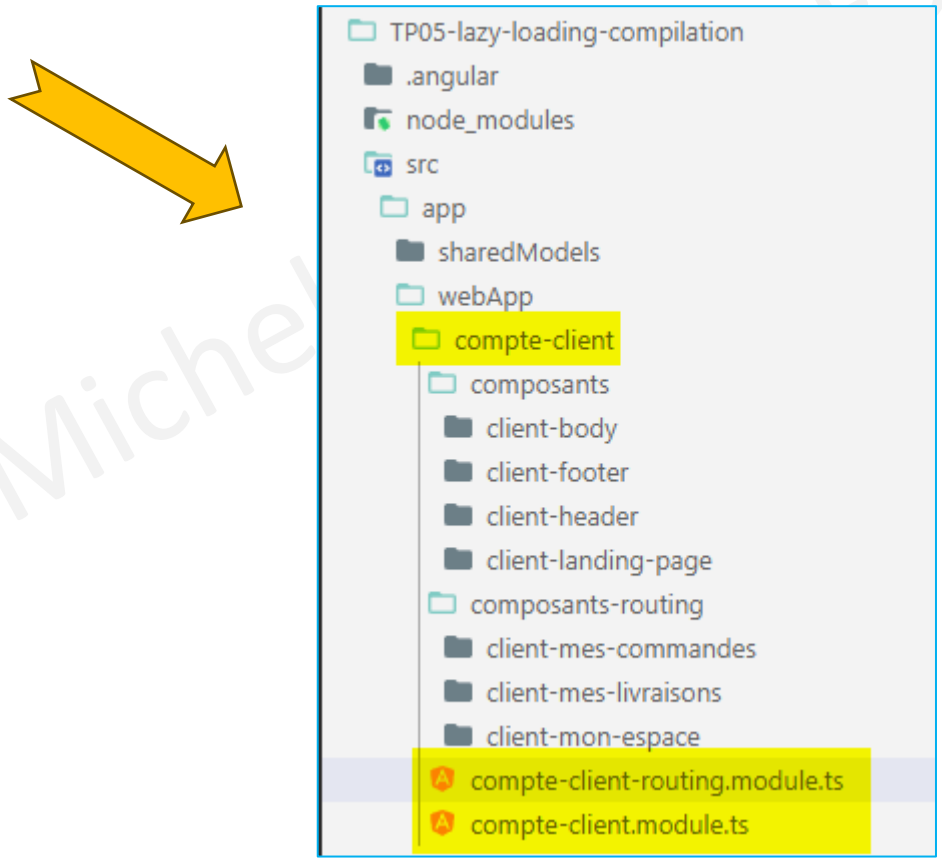
Routing : Concepts avancés – Le Lazy Loading

Créer le module compte-client avec l'option « --routing »
Cette option crée un module de routage spécifique
forChild



Routing : Concepts avancés – Le Lazy Loading

TD : déployer une architecture de module et composants « compte-client » comme ci-dessous



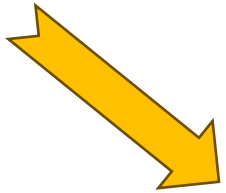
Routing : Concepts avancés – Le Lazy Loading

Le Lazy Loading permet de charger tout un module (compilé dans un fichier extrait du main.js), seulement quand l'utilisateur sollicitera ce module par une action de navigation.
Les fichiers de build (compilation) sont donc dégroupés et le projet final « buildé » est optimisé.

```
PS C:\Angular...ation-angular-2023-06\TP05-lazy-loading-compilation> npm run build

> tp02-bonnes-pratiques-injection-dependances@0.0.0 build
> ng build --stats-json

✓ Browser application bundle generation complete.
✓ Copying assets complete.
✓ Generating index html...1 rules skipped due to selector errors:
  legend+* -> Cannot read properties of undefined (reading 'type')
✓ Index html generation complete.

Initial Chunk Files   | Names         |  Raw Size | Estimated Transfer Size
styles.32ab14ebec5beb97.css | styles  | 265.62 kB |                29.18 kB
main.9281e4b71d326fdf.js    | main    | 246.63 kB |                66.78 kB
scripts.118140219e48fbd3.js | scripts | 142.96 kB |                41.36 kB
polyfills.ed4228387a31b6d8.js | polyfills |  33.05 kB |             10.64 kB
runtime.6565837a760cd7e6.js | runtime   |   2.73 kB |              1.26 kB

                      | Initial Total | 690.99 kB |               149.22 kB

Lazy Chunk Files      | Names                              |  Raw Size | Estimated Transfer Size
49.43219d5090c1d8db.js | webApp-compte-client-compte-client-module |  1.36 kB |              435 bytes
```

Routing : Concepts avancés – Le Lazy Loading

Afin de bien analyser ces customisations « tuning » de compilation, mise en place d'une stratégie de lecture de statistiques avec le « webpack-bundle-analyzer »
https://www.npmjs.com/package/webpack-bundle-analyzer



package.json

P05-lazy-loading-compilation > package.json > ...

```
5     "ng": "ng",
6     "start": "ng serve --open --port=3000",
7     "build": "ng build --stats-json",
8     "watch": "ng build --watch --configuration development --stats-json",
9     "test": "ng test",
10    "json-server": "json-server --watch src/assets/json/dives.json -p 3001",
11    "analyze":"webpack-bundle-analyzer dist/stats.json"
12  },
```

Le Lazy Loading : Mise en pratique
Etape #1 (no Lazy Loading)

client-landing-page.component.html

```
<div class="alert alert-primary">
    <app-client-header></app-client-he
    <hr>
    <app-client-body></app-client-body
    <hr>
    <app-client-footer></app-client-fo
</div>
```

client-body.component.html

```
<ul class="list-group">
    <li class="list-group-item">
        <a class="nav-link"
        [routerLink]="['mon-espace']"
        [routerLinkActive]="['lien-actif']"
        >Mon espace Client</a>
    </li>
    <li class="list-group-item">
        <a class="nav-link"
        routerLink="mes-commandes"
        [routerLinkActive]="['lien-actif']"
        >Mes commandes</a>
    </li>
    <li class="list-group-item">
        <a class="nav-link"
        [routerLink]="['mes-livraisons']"
        [routerLinkActive]="['lien-actif']"
        >Mes livraisons</a>
    </li>
</ul>

<p class="alert alert-success">
    <router-outlet></router-outlet>
</p>
```

accueil.module.ts

```
// import { CompteClientModule } from '../../formation/compte-client/
@NgModule({
  declarations: [
    LandingPageComponent,
    HeaderComponent,
    BodyComponent,
    FooterComponent
  ],
  imports: [
    CommonModule,
    RouterModule,
    // nos modules
    FilmsModule,
    FormsReactiveModule,
    // CompteClientModule,
    RxjsModule
  ],
  exports:[
    LandingPageComponent,
    HeaderComponent,
```

compte-client-routing.module.ts

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { ClientMonEspaceComponent } from './composants-routing/client-mon-espace/client-mon-espace.c
import { ClientCommandesComponent } from './composants-routing/client-commandes/client-commandes.com
import { ClientLivraisonsComponent } from './composants-routing/client-livraisons/client-livraisons.

export const routesClient: Routes = [
// const routesClient: Routes = [

  { path: 'mon-espace', component: ClientMonEspaceComponent },
  { path: 'mes-commandes', component: ClientCommandesComponent },
  { path: 'mes-livraisons', component: ClientLivraisonsComponent },
];
```

app-routing.module.ts

```
import { ClientLandingPageComponent } from './webApp/formation/compte
// ------------------------------------------------
import { routesClient}  from  "./webApp/formation/compte-client/comp

const routes: Routes = [
  { path: '', component: BodyComponent },
  { path: 'accueil', component: BodyComponent },
  { path: 'liste-des-films', component: ListeDesFilmsComponent },

  // 1ère étape : sans lazy loading
  // raccrocher un module qui a son propre système de routage
  {
    path: 'compte-client',
    component: ClientLandingPageComponent,
    children: routesClient
  },
```

Le Lazy Loading : Mise en pratique
Etape #2 (Lazy Loading)

client-landing-page.component.html

```html
<div class="alert alert-primary">
    <app-client-header></app-client-he
    <hr>
    <app-client-body></app-client-body
    <hr>
    <app-client-footer></app-client-fo
</div>
```

client-body.component.html

```html
<ul class="list-group">
    <li class="list-group-item">
        <a class="nav-link"
        [routerLink]="['mon-espace']"
        [routerLinkActive]="['lien-actif']"
        >Mon espace Client</a>
    </li>
    <li class="list-group-item">
        <a class="nav-link"
        routerLink="mes-commandes"
        [routerLinkActive]="['lien-actif']"
        >Mes commandes</a>
    </li>
    <li class="list-group-item">
        <a class="nav-link"
        [routerLink]="['mes-livraisons']"
        [routerLinkActive]="['lien-actif']"
        >Mes livraisons</a>
    </li>
</ul>

<p class="alert alert-success">
    <router-outlet></router-outlet>
</p>
```

accueil.module.ts

Formation-ANY-07-2023 > TP03-lazy-loading > src > app > webApp > root > accueil > accueil.module.ts > ...

```ts
// import { CompteClientModule } from '../../formation/compte-client/compte-client.

@NgModule({
  declarations: [
    LandingPageComponent,
    HeaderComponent,
    BodyComponent,
    FooterComponent
  ],
  imports: [
    CommonModule,
    RouterModule,
    // nos modules
    FilmsModule,
    FormsReactiveModule,
    // CompteClientModule,
    RxjsModule
```

compte-client-routing.module.ts

Formation-ANY-07-2023 > TP03-lazy-loading > src > app > webApp > formation > compte-client > compte-client-routing.module.ts > routesClient

```ts
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { ClientMonEspaceComponent } from './composants-routing/client-mon-espace/client-mon-espace.c
import { ClientCommandesComponent } from './composants-routing/client-commandes/client-commandes.com
import { ClientLivraisonsComponent } from './composants-routing/client-livraisons/client-livraisons.

// export const routesClient: Routes = [
const routesClient: Routes = [

  { path: 'mon-espace', component: ClientMonEspaceComponent },
  { path: 'mes-commandes', component: ClientCommandesComponent },
  { path: 'mes-livraisons', component: ClientLivraisonsComponent },
];

@NgModule({
  imports: [RouterModule.forChild(routesClient)],
  exports: [RouterModule]
```

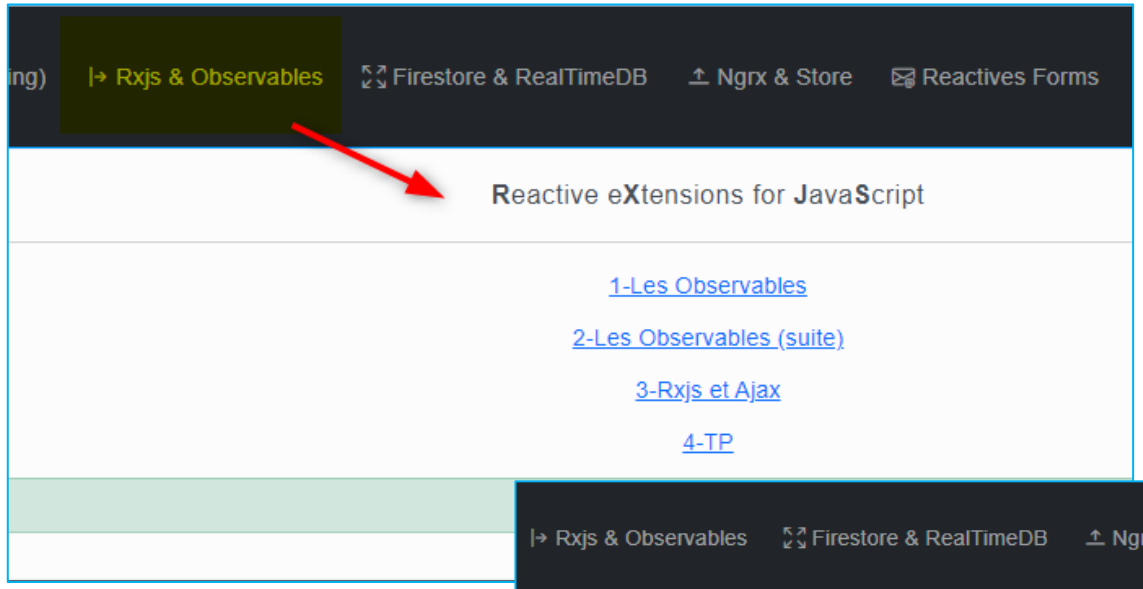
app-routing.module.ts

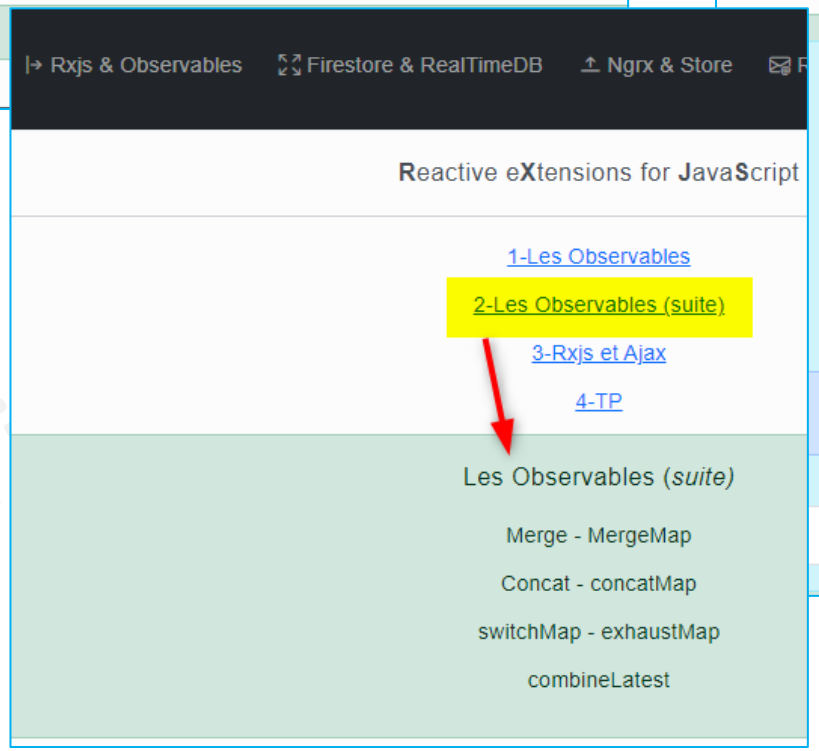
Formation-ANY-07-2023 > TP03-lazy-loading > src > app > app-routing.module.ts > routes > data > preload

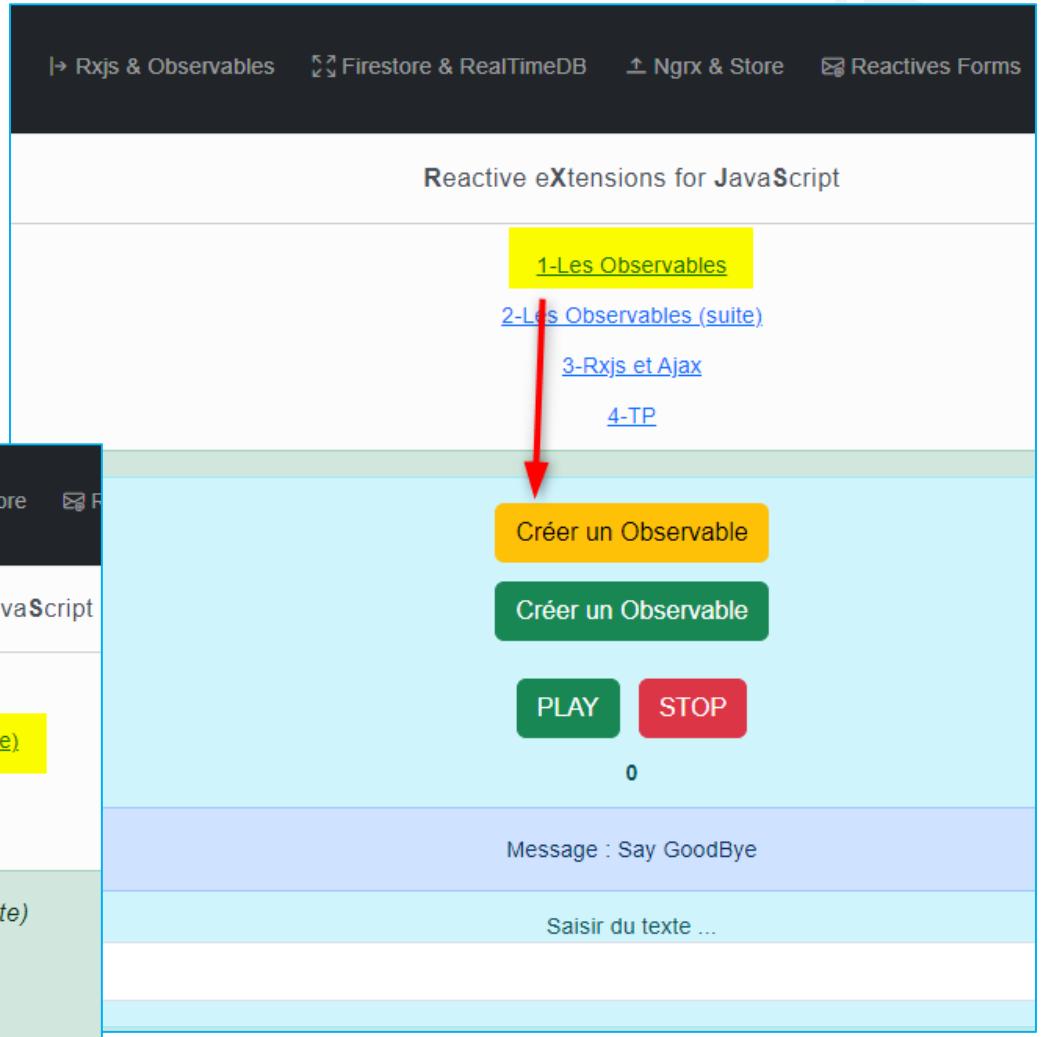
```ts
// 2ème étape : Avec lazy loading (promesses ES2015 😐)
// {
//   path: 'compte-client',
//   component: ClientLandingPageComponent,
//   loadChildren:
//   () => import('./webApp/formation/compte-client/compte-client.module')
//   .then(
//     (m) => {
//       return m.CompteClientModule
//     }
//   )
// },

// 3ème étape : Avec Lazy Loading (ES2017 : Async/await => asynchrone 😉)
{
  path: 'compte-client',
  component: ClientLandingPageComponent,
  loadChildren:
  async () => (await import('./webApp/formation/compte-client/compte-client.module'
  // syntaxic sugar
    data :{
      preload:true
    }

},
```

TP : Déployer une arborescence en Lazy Loading

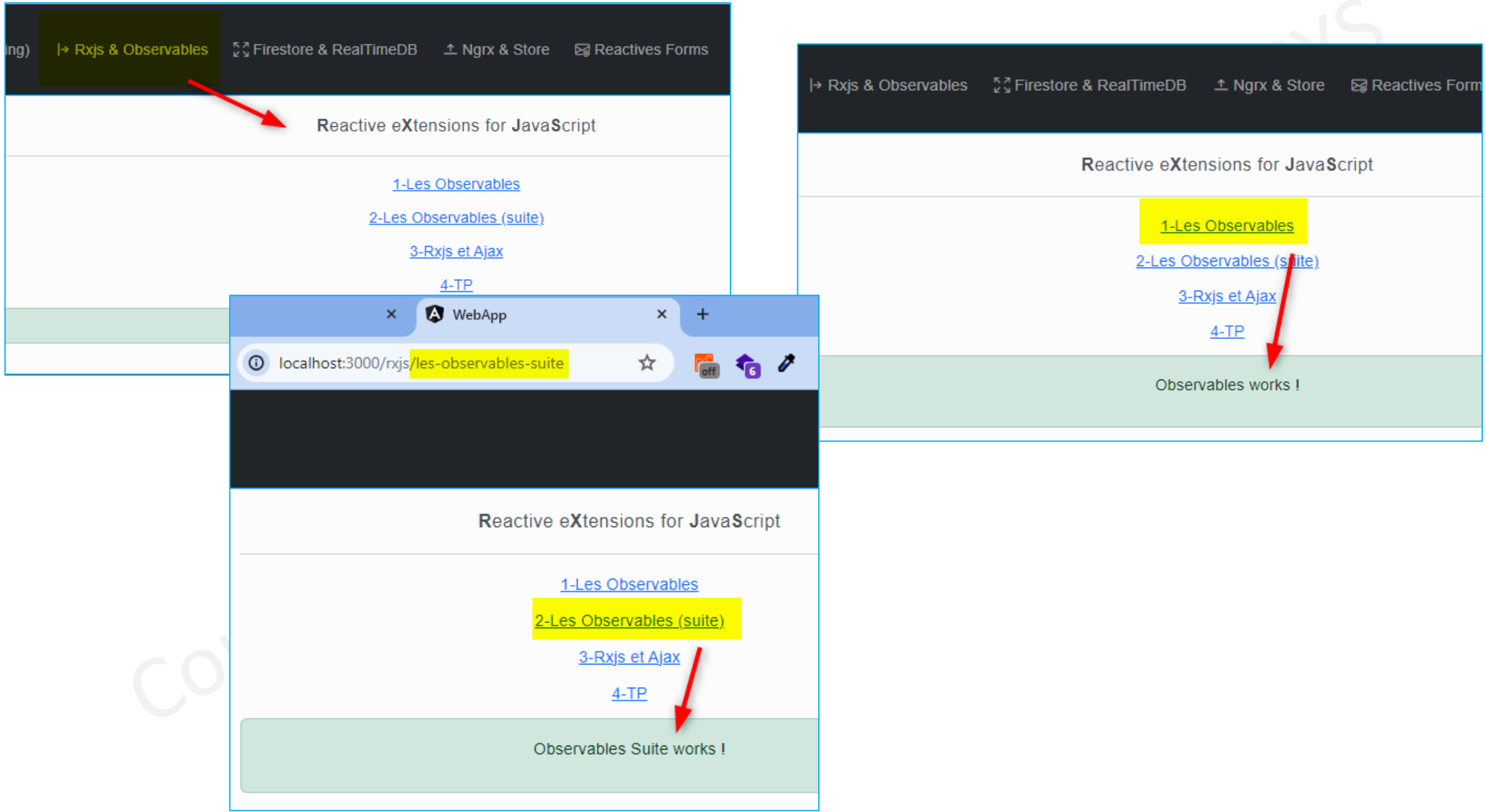
TP : déployer l'arborescence en Lazy Loading RXJS



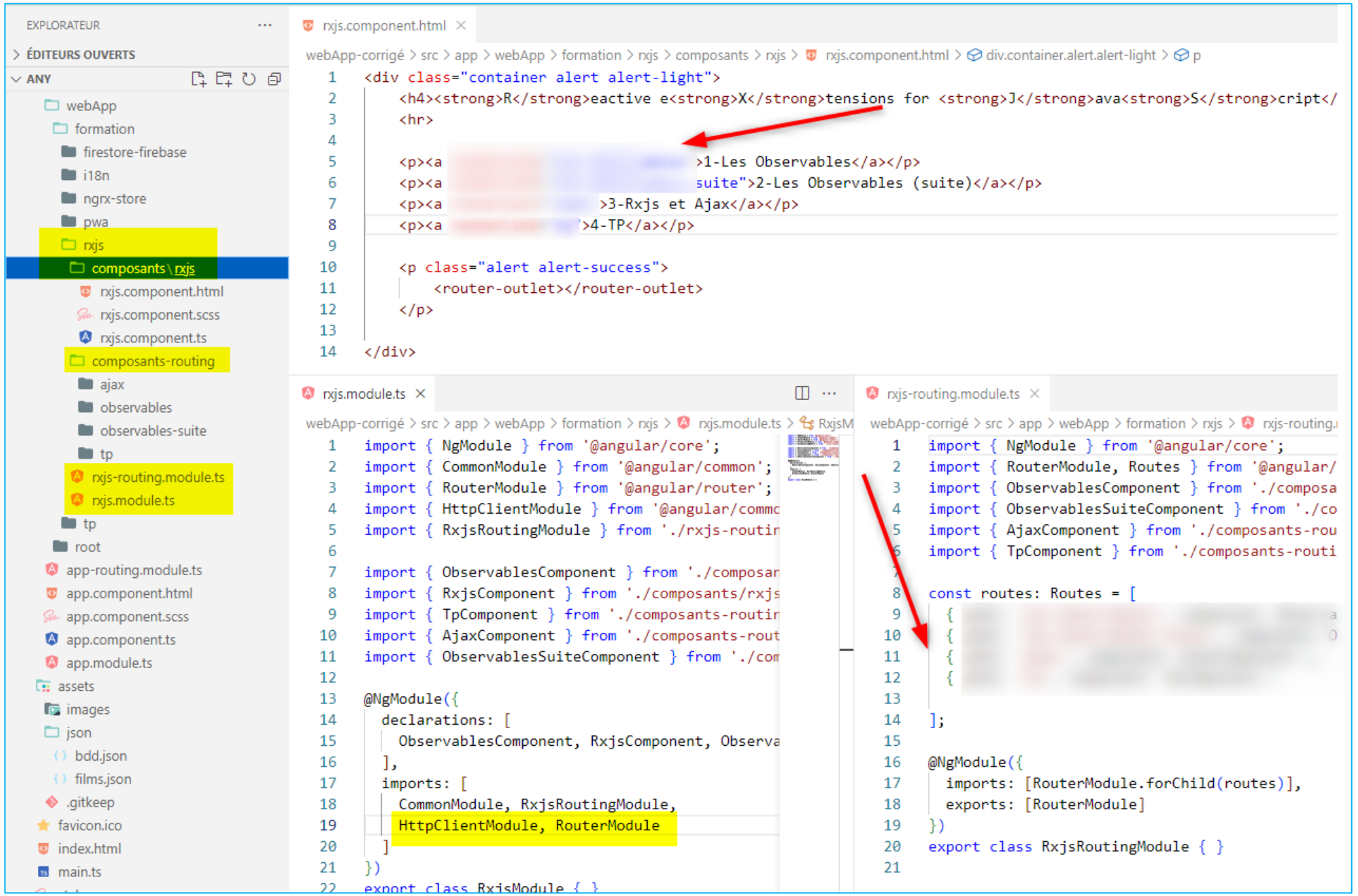




TP : déployer l'arborescence en Lazy Loading RXJS



TP : déployer l'arborescence en Lazy Loading RXJS



app.config.ts

src > app > app.config.ts > appConfig > providers

```ts
import { routes } from './app.routes';
import { provideAnimationsAsync } from '@angular/platform-browser/animations/async';
import { provideHttpClient } from '@angular/common/http';

export const appConfig: ApplicationConfig = {
  providers: [
    provideZoneChangeDetection({ eventCoalescing: true }),
    provideRouter(routes),
    provideAnimationsAsync(),
    provideHttpClient()]
};

```

app.routes.ts

src > app > app.routes.ts > ...

```ts
import { Routes } from '@angular/router';
import { HomeComponent } from './webApp/accueil/components/home/home.component';
import { CommandesComponent } from './webApp/compte-client/components-routing/commandes/commandes.component';
import { LivraisonsComponent } from './webApp/compte-client/components-routing/livraisons/livraisons.component';
import { SignalsComponent } from './webApp/signals/components/signals/signals.component';
import { DivesListComponent } from './webApp/dives-list/components/dives-list/dives-list.component';
import { ContactsComponent } from './webApp/contacts/components/contacts/contacts.component';
import { Page404Component } from './shared/shared-components/page404.component';
import { RxjsComponent } from './webApp/rxjs/components/rxjs/rxjs.component';


export const routes: Routes = [
    { path: '', component: HomeComponent, title: 'Angular 18 - StandAlone Components' },
    { path: 'liste-des-plongees', component: DivesListComponent, title: 'Liste des plongees' },
    { path: 'les-signals-angular-17', component: SignalsComponent, title: 'Les signals Angulars 17 - trop Cool' },
    { path: 'les-observables', component: RxjsComponent, title: 'Rxjs et les observables' },

```

```ts
export const routes: Routes = [
    { path: '', component: HomeComponent, title: 'Angular 18 - StandAlone Components' },
    { path: 'liste-des-plongees', component: DivesListComponent, title: 'Liste des plongees' },
    { path: 'les-signals-angular-17', component: SignalsComponent, title: 'Les signals Angulars 17 - trop Cool' },
    { path: 'les-observables', component: RxjsComponent, title: 'Rxjs et les observables' },

    // Avec le Lazy Loading Component 😉
    {
        path: 'compte-client', title: 'Mon espace client - Formation Angular',
        loadComponent:
            () => import('./webApp/compte-client/components/compte-client/compte-client.component')
                .then(
                    (c) => c.CompteClientComponent
                ),
        children: [
            { path: 'mes-commandes', component: CommandesComponent },
            { path: 'mes-livraisons', component: LivraisonsComponent },
        ]
```

TP : Tester le mode lazy Loading en compilation

Y’a-t-il vraiment un fichier « lazy chunk file » créé ?

La réactivité en Javascript

Réactivité de Javascript

Javascript n’est pas du tout réactif … 🥵

Finalement quel Problème doit on résoudre ?
Tous les Frameworks modernes essaient de résoudre ce même problème

La notion de réactivité en Front End

	A	B
1	Montant HT	150
2	Tva	0,2
3	Montant TTC	=B1*B2+B1

index.html × script.js ×

```
1  let montantHT=150;
2  const TVA=0.2;
3  let total=montantHT*TVA + montantHT;
4  console.log(total);
5
6  let montantHt=200;
7  console.log('Est ce réactif ? : ', total, '🤔😣');
8
```

Console ×

```
180
Est ce réactif ? :  180 🤔😣
```

La notion de réactivité en Front End

Comment implémenter cela dans nos applis JS ? (Google Sheets l'a déjà fait avec du « *fichier Excel* » en JS)

3 Méthodes pour implémenter la réactivité dans les Frameworks JS :

1- Le fonctionnement de base d'Angular : **Value Based** :

L'état actuel d'un composant est stockée dans une zone mémoire précise (store, composant.ts, local Storage, etc …)
le Framework utilise le mode « **Detection Change**[1] **»** ou **« *Dirty Checking*[2]** » pour savoir si la valeur a changé car il ne peut pas l'observer, il parcourt l'ensemble des composants pour vérifier les différents états et mettre à jour le DOM. La librairie actuelle* Zone.js fait ce travail de vérification.

Avantages : aucunes connaissances du « **core** » **Zone**.js n'est requise et c'est extrêmement simple à utiliser pour le dév. Le DOM est automatiquement MAJ.

Inconvénients: Performances 😘 🤯, des quantités de traitements sont faits pour mettre à jour les Vues HTML. Pour améliorer cela, il faut devenir expert en **Zone** et utiliser le **change detector** et le **onPush** !

1 : Pour Angular
2 : utilisé dans d'autres framework

La notion de réactivité en Front End

3 Méthodes pour implémenter la réactivité dans les Frameworks JS :

2- La méthode **Observable Based**:

Cette méthode est beaucoup plus performante que la « Value based » mais demande une solide investigation et des temps d'intégration et de compréhension conséquents.

La notion d'abonnement permet de mettre à jour le DOM uniquement lorsqu'on en aura besoin !

*Rappel :
1 « Observable » est un Objet qui émet une séquence de valeurs
1 « Observer » observe l'observable et réagit à l'arrivée de **nouvelles** valeurs

« Subject » et « Behaviour Subject (avec valeur initiale) » sont en même temps Observables et Observers :
Le « Subject » est multidiffusion par rapport à l'observable qui est monodiffusion

La notion de réactivité en Front End

3 Méthodes pour implémenter la réactivité dans les Frameworks JS :

3- La méthode **Signal()**:

Introduit dans la version 16, validé avec la version 17, le signal va révolutionner la réactivité dans Angular !

Le signal représente l'état d'un composant, d'une propriété , d'une valeur spécifique.

Lorsque le signal est mis à jour, les parties de l'application qui dépendent de ce signal peuvent être informées et modifier leurs propres données.
(mécanisme d'optimisation que l'on reconnaît dans le Store)

Lorsque une valeur change, le signal émet une information et l'application peut se mettre à jour de manière sélective.

Reactive eXtensions for JavaScript
Les Observables

RXJS et les OBSERVABLES

- 1 Observable est un objet qui émet une suite (une séquence) de valeurs (datas) dans le temps.
  Exemple : une requête HTTP

- On pourrait comparer cette suite de valeurs émises aux valises et bagages qui défilent sur un tapis roulant

- Chaque bagage a une destination mais va peut être croiser d'autres bagages qui n'ont pas la même destination mais empruntent le même tapis
- Ces valeurs peuvent être reçues par un poste de tri, interceptées, regroupées en fonction de critères et réaiguillées dynamiquement.
- On peut également stopper le tapis roulant

- Cela offre beaucoup de souplesse dans le traitement des données et la gestion des requêtes asynchrones, la gestion des multiples interactions utilisateurs

RXJS et les OBSERVABLES

https://rxjs-dev.firebaseapp.com/api

Observable = 1 objet typé qui émet des valeurs dans le temps ==> forme une séquence de valeurs ou un Stream ou un flux de datas émises

Subscriber = 1 abonné qui avec la méthode .subscribe() peut recevoir ces valeurs émises(Stream)

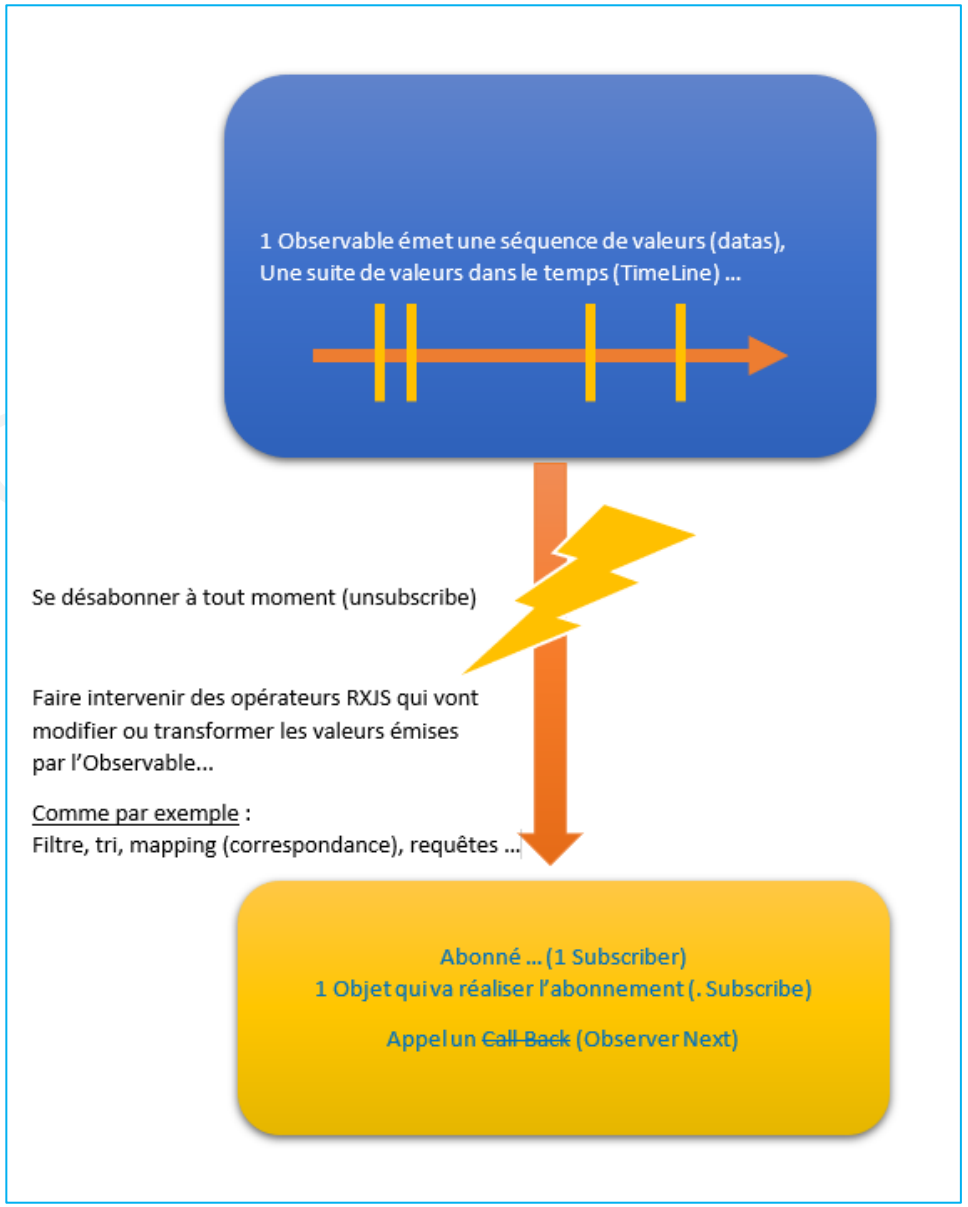
Opérateurs : des fonctions tap, map, merge, filter qui vont pouvoir modifier/transformer ce flux de valeurs émises par l'observable

Dans le subscribe => Les observers (fcts de call back) qui se définissent dans le subscribe : Next - Error – Complete

La gestion des erreurs est mieux optimisée encore avec les Observables car on peut se désabonner !

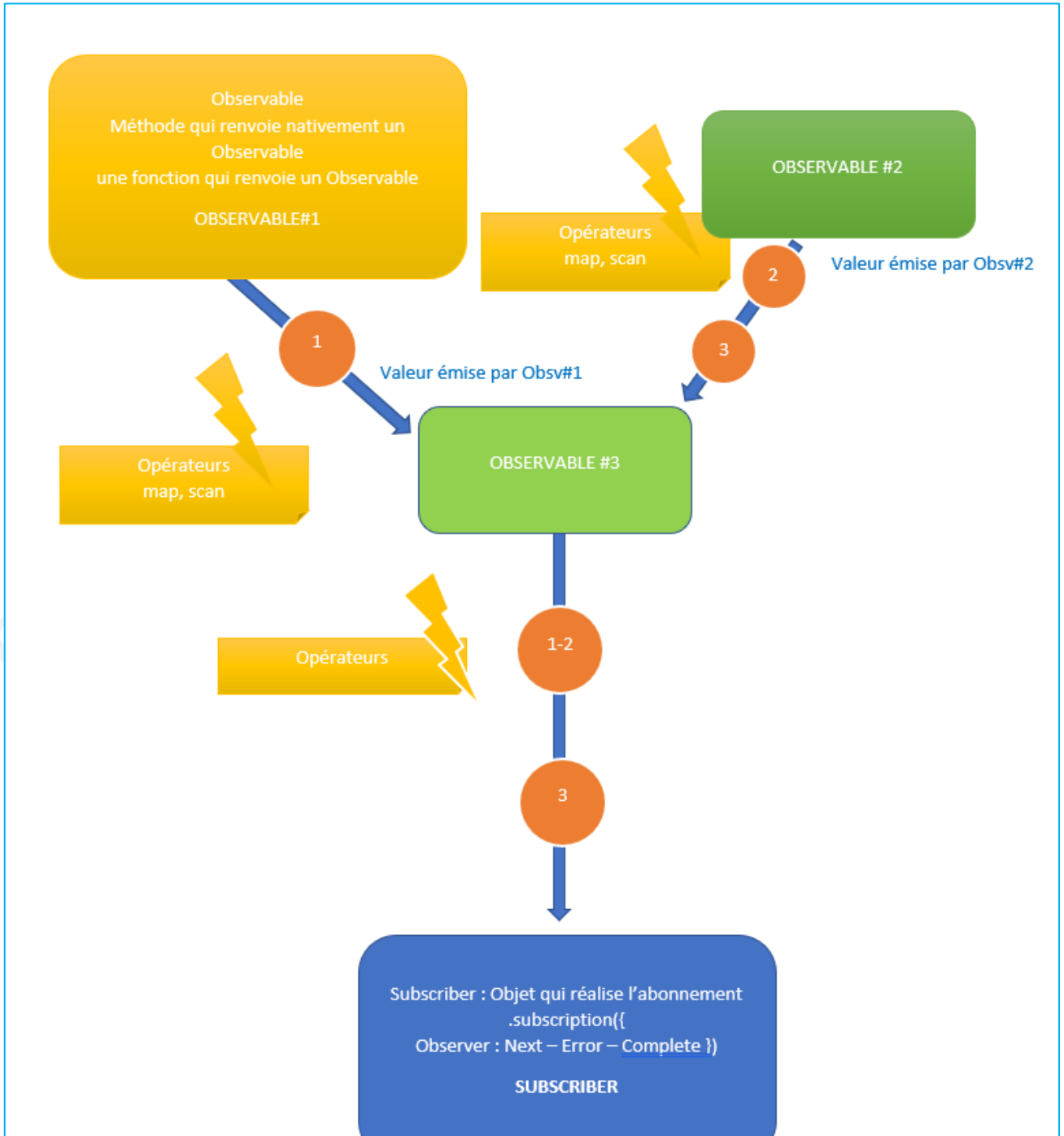
Les séquences de valeurs émises sont facilement annulables !

On peut facilement les recomposer ou les recombiner pour former une nouvelle séquence de valeurs !

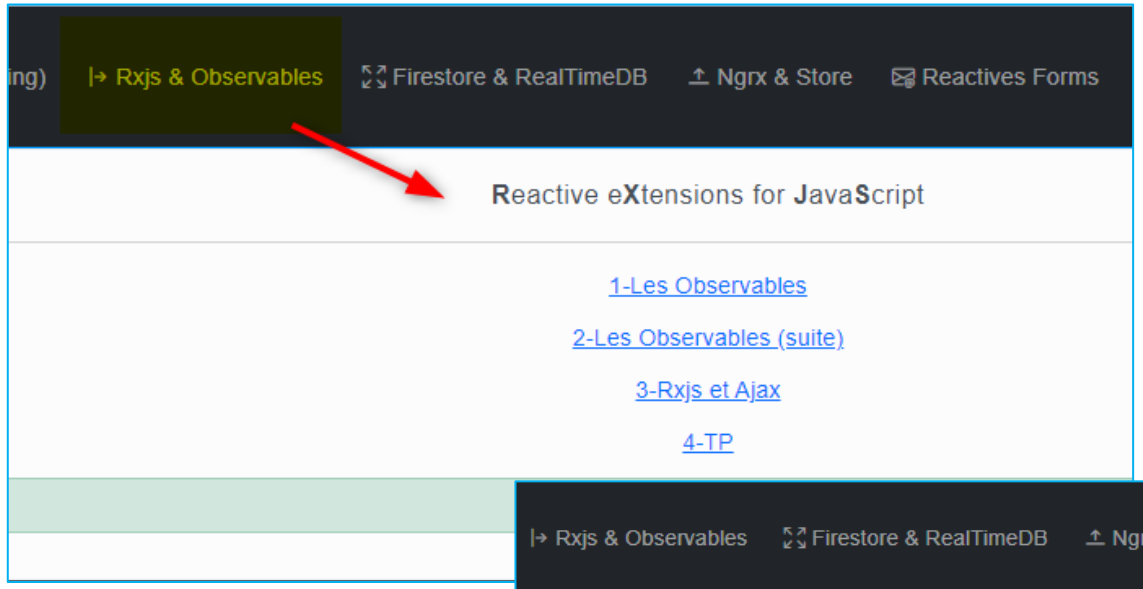


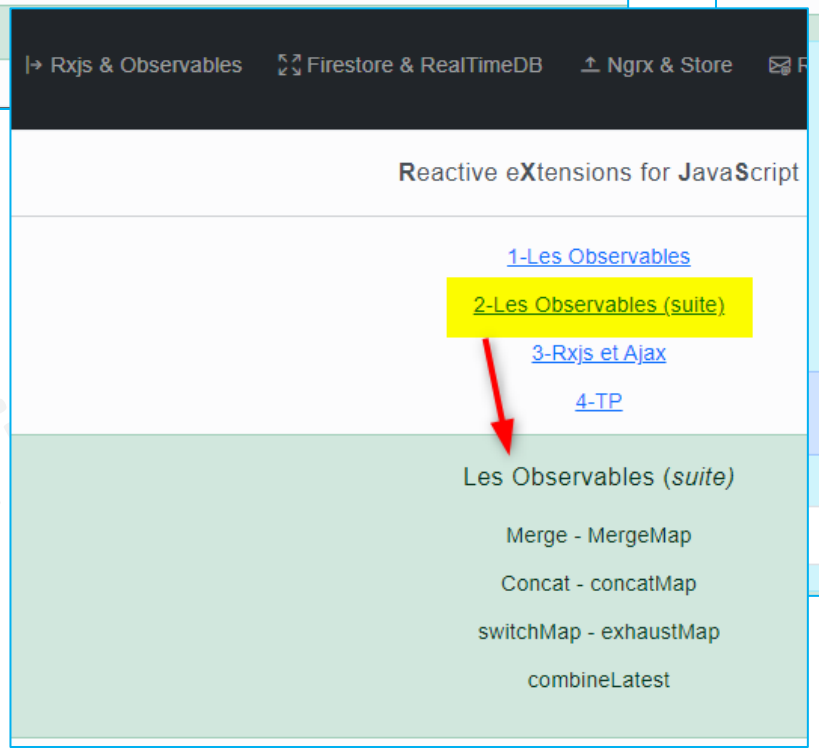
RXJS et les OBSERVABLES

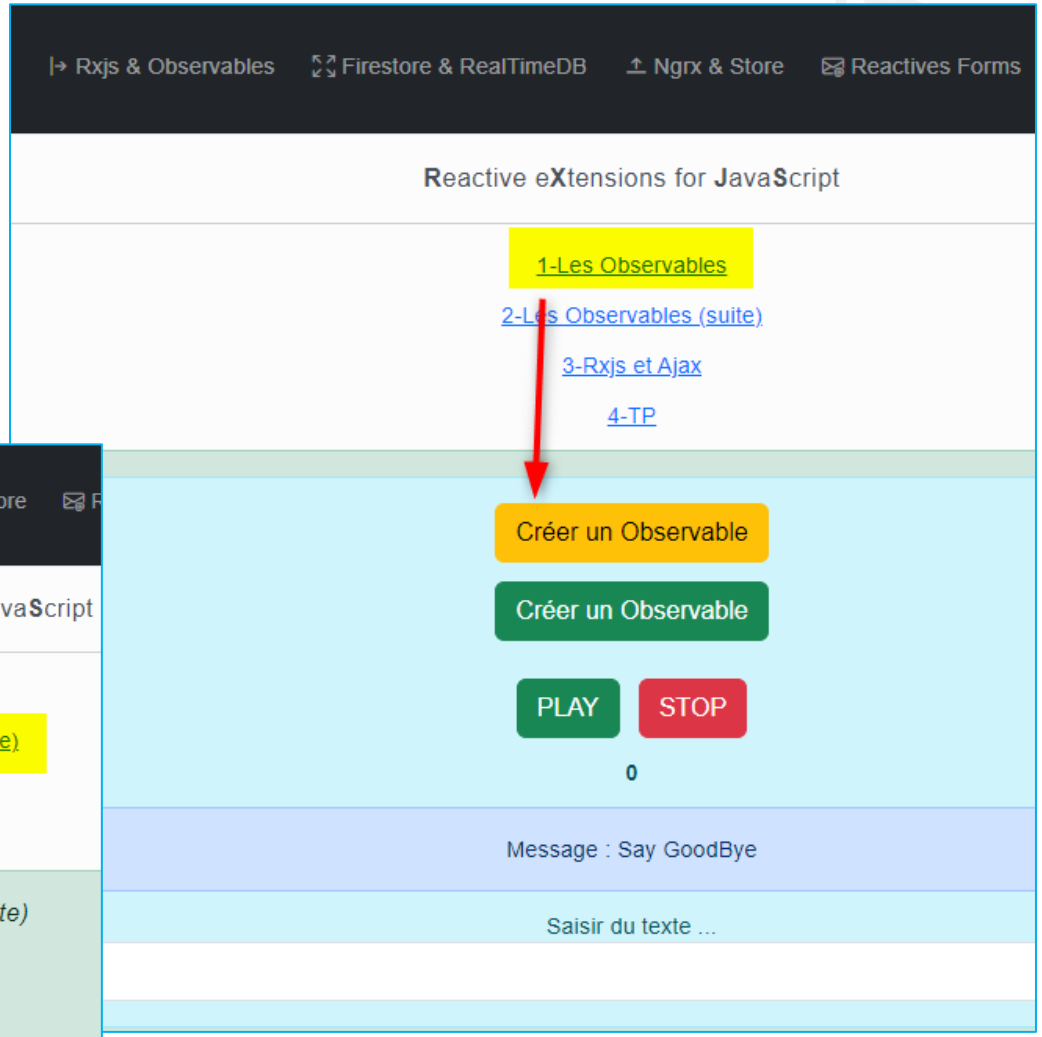
On peut facilement les recomposer
ou les recombiner pour former
une nouvelle séquence de valeurs !



TP : déployer l'arborescence en Lazy Loading RXJS





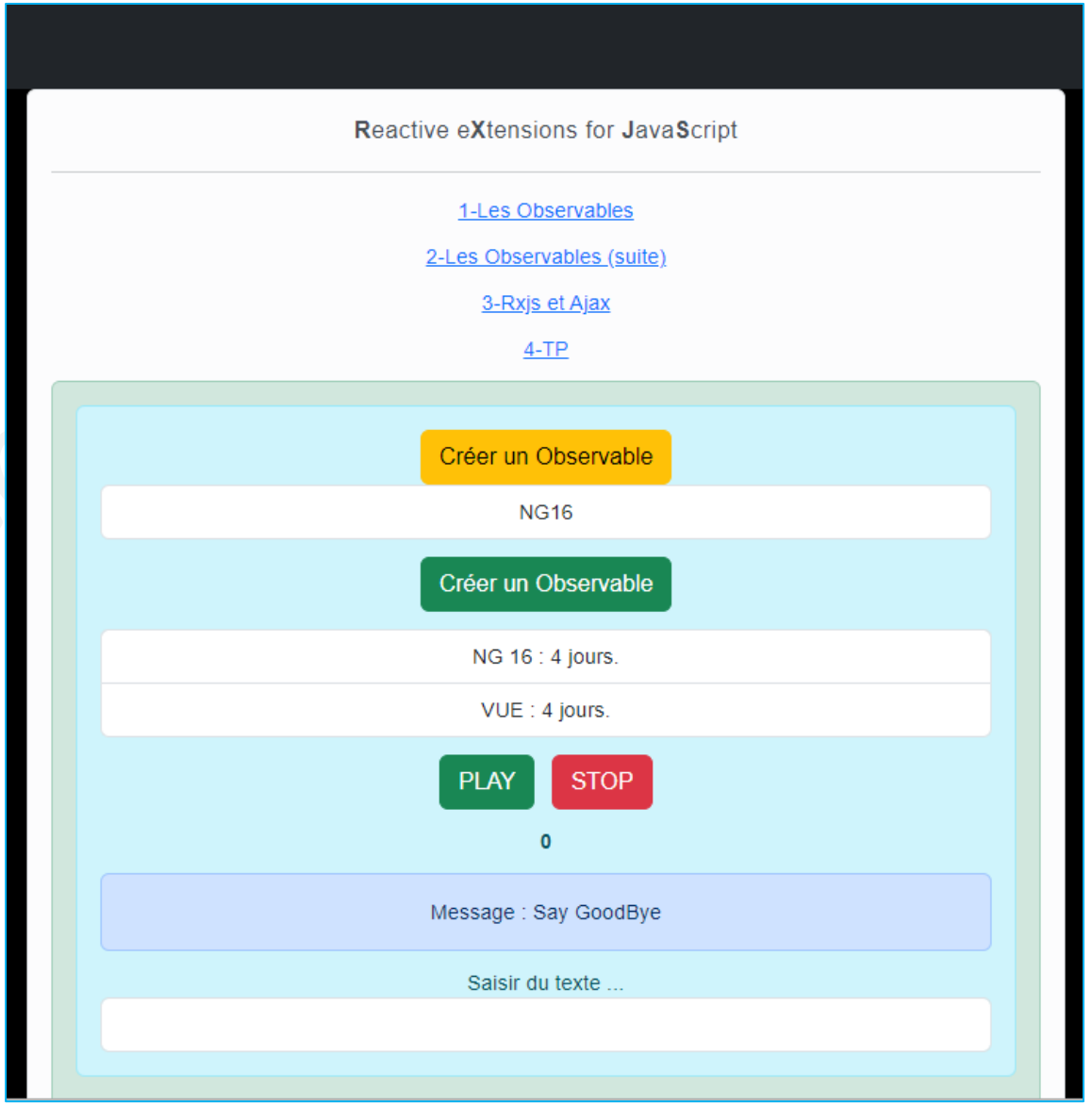


Exercice #1

observables.component.html

p-corrigé > src > app > webApp > formation > rxjs > composants-routing > observables > observables.component.html > div.alert.ale

```
<div class="alert alert-info">
    <button class="btn btn-warning" (click)="createObservable1()">Créer un Observable</button>
    <ul #formations1 class="list-group"></ul>
    <p></p>

    <button class="btn btn-success" (click)="createObservable2()">Créer un Observable</button>
    <p></p>

    <ul #formations2 class="list-group"></ul>
    <p></p>

    <button class="btn btn-success" (click)="play()">PLAY</button>
     
    <!-- <button class="btn btn-danger" (click)="stop()">STOP</button> -->
    <button class="btn btn-danger" #stop >STOP</button>

    <p></p>
    <p><strong>{{temps}}</strong></p>

    <p></p>
    <p class="alert alert-primary">Message : {{ infos$ | async }}</p>

    <p></p>
    <label for="">Saisir du texte ...</label>
    <input type="text" class="form-control" #texte>
</div>
```



Exercice #1

observables.component.ts

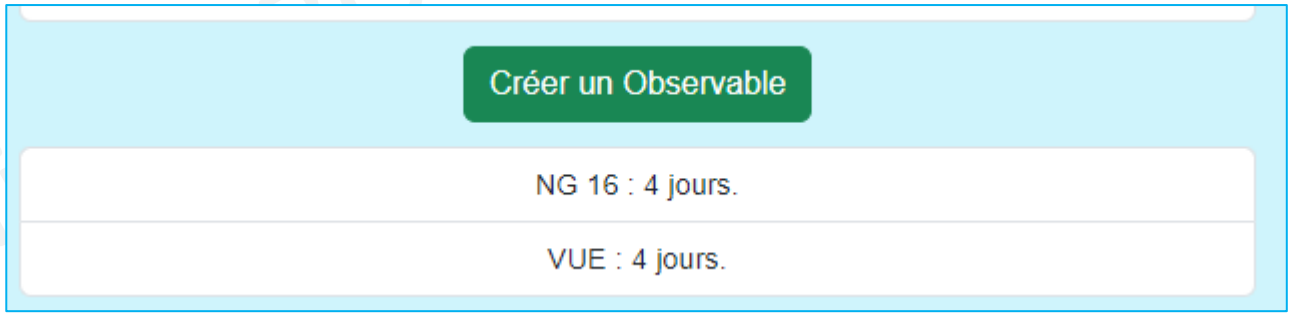
```
// Création de l'observable
const formation$: Observable<string> = from(formationsArray);
console.log(formation$);

this.subscription = formation$
  .pipe(
    first()
  ).subscribe({
    // la notion d'Observer :  next  | error | complete
    next:
      (formation: string) => {
        console.log(formation);
        const eltLi = <HTMLElement>document.createElement('li');
        eltLi.innerHTML = formation;
        eltLi.className = 'list-group-item';
        this.eltUl1.nativeElement.appendChild(eltLi);
      },
    // error
    error:
      (e) => {
        console.error(e);
      },
    //  complete
    complete:
      () => {
        console.warn('Complete');
      }
  });
}
```

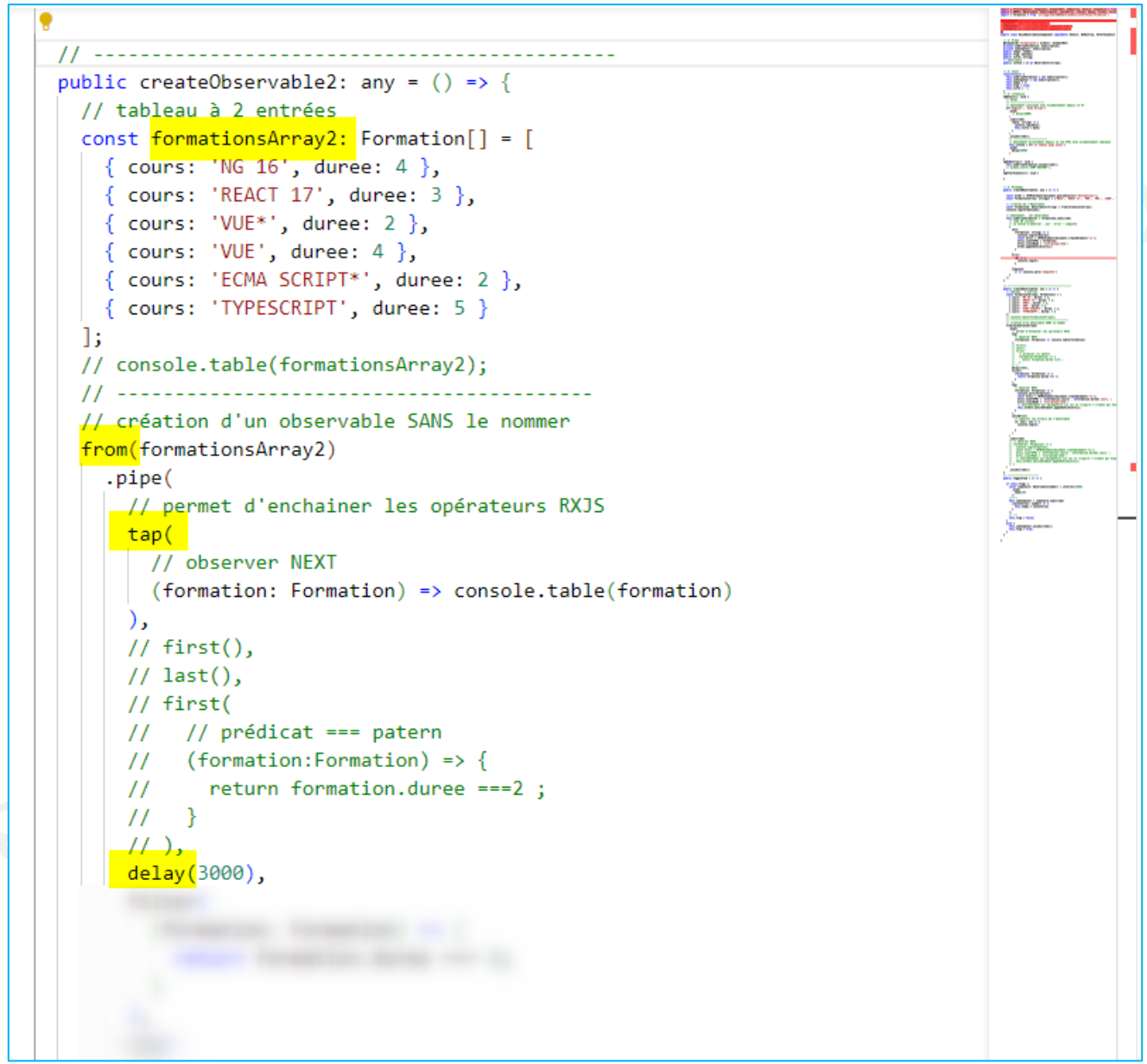
Exercice #1

observables.component.ts

TP : créer l'observable associée à la méthode createObservable2

- seules les formations de 4 jours seront affichées
- gérer l'observer Next dans le tap





```
// ---------------------------------------------
public createObservable2: any = () => {
  // tableau à 2 entrées
  const formationsArray2: Formation[] = [
    { cours: 'NG 16', duree: 4 },
    { cours: 'REACT 17', duree: 3 },
    { cours: 'VUE*', duree: 2 },
    { cours: 'VUE', duree: 4 },
    { cours: 'ECMA SCRIPT*', duree: 2 },
    { cours: 'TYPESCRIPT', duree: 5 }
  ];
  // console.table(formationsArray2);
  // ----------------------------------------
  // création d'un observable SANS le nommer
  from(formationsArray2)
    .pipe(
      // permet d'enchainer les opérateurs RXJS
      tap(
        // observer NEXT
        (formation: Formation) => console.table(formation)
      ),
      // first(),
      // last(),
      // first(
      //   // prédicat === patern
      //   (formation:Formation) => {
      //     return formation.duree ===2 ;
      //   }
      // ),
      delay(3000),
```

Exercice #2

Observables-suite.component

Les fonctions de regroupement et de recombinaison de séquences

```
ngOnInit() {

  // Merge : une simple émission en parallèle
  // ----------------------------------------------
  const obsvervableA$ = interval(1000)
    .pipe(take(3), map(
      (valObs) => `ObsA ${valObs}`
    ));
  const obsvervableB$ = interval(1000)
    .pipe(take(3), map(
      (valObs) => `ObsB ${valObs}`
    ));

  merge(obsvervableA$, obsvervableB$)
    // émission en parrallèle
    .subscribe(
      (valObs) => console.log('merge', valObs)
    );

  // Concat : une émission non plus en parallèle mais séquentielle
  // le 1er observable doit avoir terminé ou envoyer complete
  // avant que le second puisse etre interprété
  // ----------------------------------------------

  concat(obsvervableA$, obsvervableB$)
    // émission  après complete
    .subscribe(
      (valObs) => console.warn('concat', valObs)
    );
```

```
merge ObsA 0
merge ObsB 0
▶ concat ObsA 0
merge ObsA 1
merge ObsB 1
▶ concat ObsA 1
merge ObsA 2
merge ObsB 2
▶ concat ObsA 2
▶ concat ObsB 0
▶ concat ObsB 1
▶ concat ObsB 2
```

Exercice #2

Observables-suite.component

Les fonctions de regroupement et de recombinaison de séquences

```
map + mergeMap :  ▶ {id: 1, cours: 'Angular - les fondamentaux', codeCours: 'AGU', duration: 4}
map + mergeMap :  ▶ {id: 2, cours: 'Angular - concepts avancés', codeCours: 'ANY', duration: 3}
map + mergeMap :  ▶ {id: 3, cours: 'Javascript', codeCours: 'DHL', duration: 4}
mergemap :  ▶ {id: 1, cours: 'Angular - les fondamentaux', codeCours: 'AGU', duration: 4}
mergemap :  ▶ {id: 2, cours: 'Angular - concepts avancés', codeCours: 'ANY', duration: 3}
mergemap :  ▶ {id: 3, cours: 'Javascript', codeCours: 'DHL', duration: 4}
```

```
// ------------------------------------------------
// map - mergeAll - mergeMap - concatmap)
// ------------------------------------------------
const API_URL ='http://localhost:3001/formations';

of(1, 2, 3)
  .pipe(
    map(valObs => this._http.get<any[]>(`${API_URL}/${valObs}`)),
    mergeAll()
  )
  .subscribe(
    (valObs) => console.log('map + mergeMap : ', valObs)
  )
// ------------------------------------------------
of(1, 2, 3)
  .pipe(
    mergeMap(valObs => this._http.get<any[]>(`${API_URL}/${valObs}`))
  )
  .subscribe(
    (valObs) => console.log('mergemap : ', valObs)
  );
```

Exercice #2

Observables-suite.component

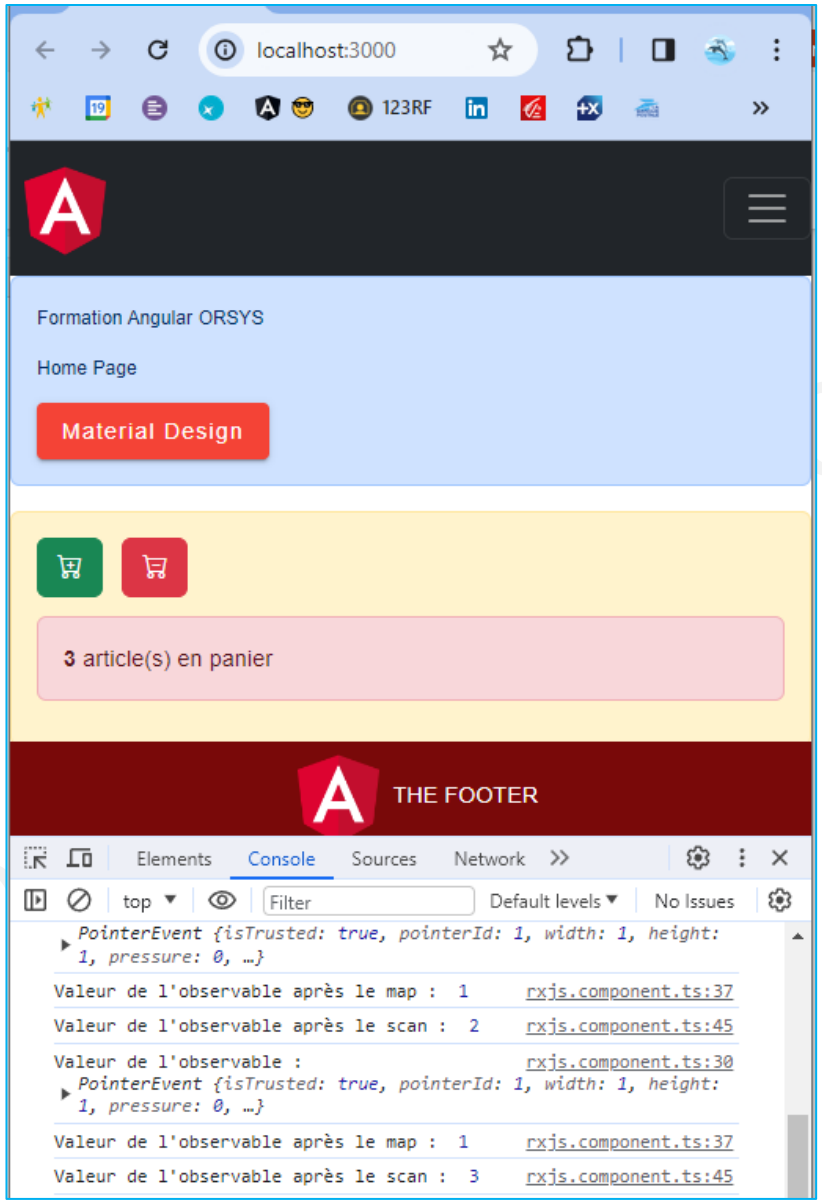
Les fonctions de regroupement et de recombinaison de séquences

```
switchmap :  {id: 3, cours: 'Javascript', codeCours: 'DHL', duration: 4}
exhaustmap :  {id: 1, cours: 'Angular - les fondamentaux', codeCours: 'AGU', duration: 4}
```

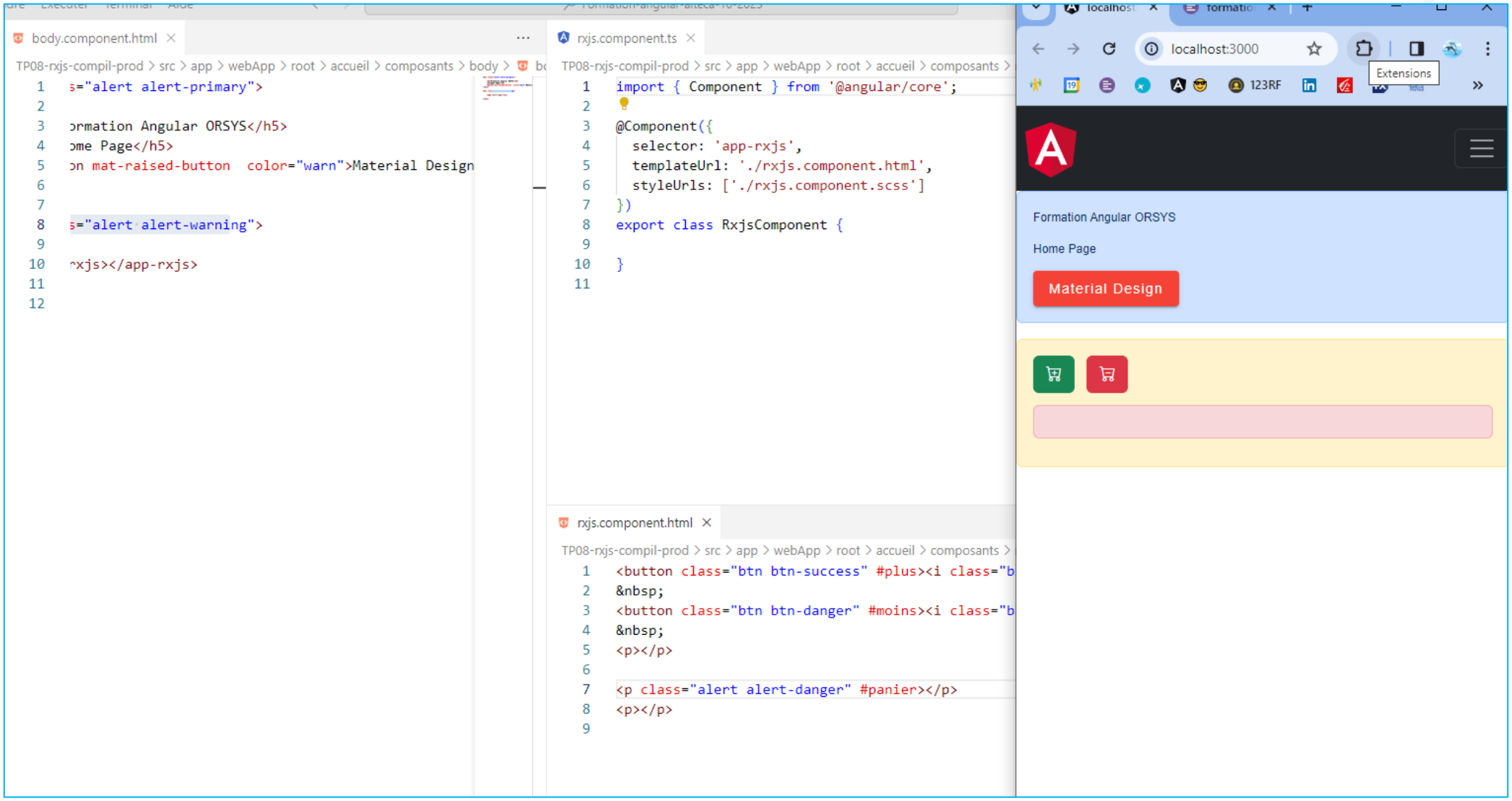
```
// ------------------------------------------------
of(1, 2, 3)
  .pipe(
    switchMap(valObs => this._http.get<any[]>(`${API_URL}/${valObs}`))
  )
  .subscribe(
    (valObs) => console.warn('switchmap : ', valObs)
  );

of(1, 2, 3)
  .pipe(
    exhaustMap(valObs => this._http.get<any[]>(`${API_URL}/${valObs}`))
  )
  .subscribe(
    (valObs) => console.warn('exhaustmap : ', valObs)
  );
```

Intéressant : les requêtes précédentes ou suivantes sont stoppées

TP (Regroupement avec combinelatest)



RXJS et les OBSERVABLES

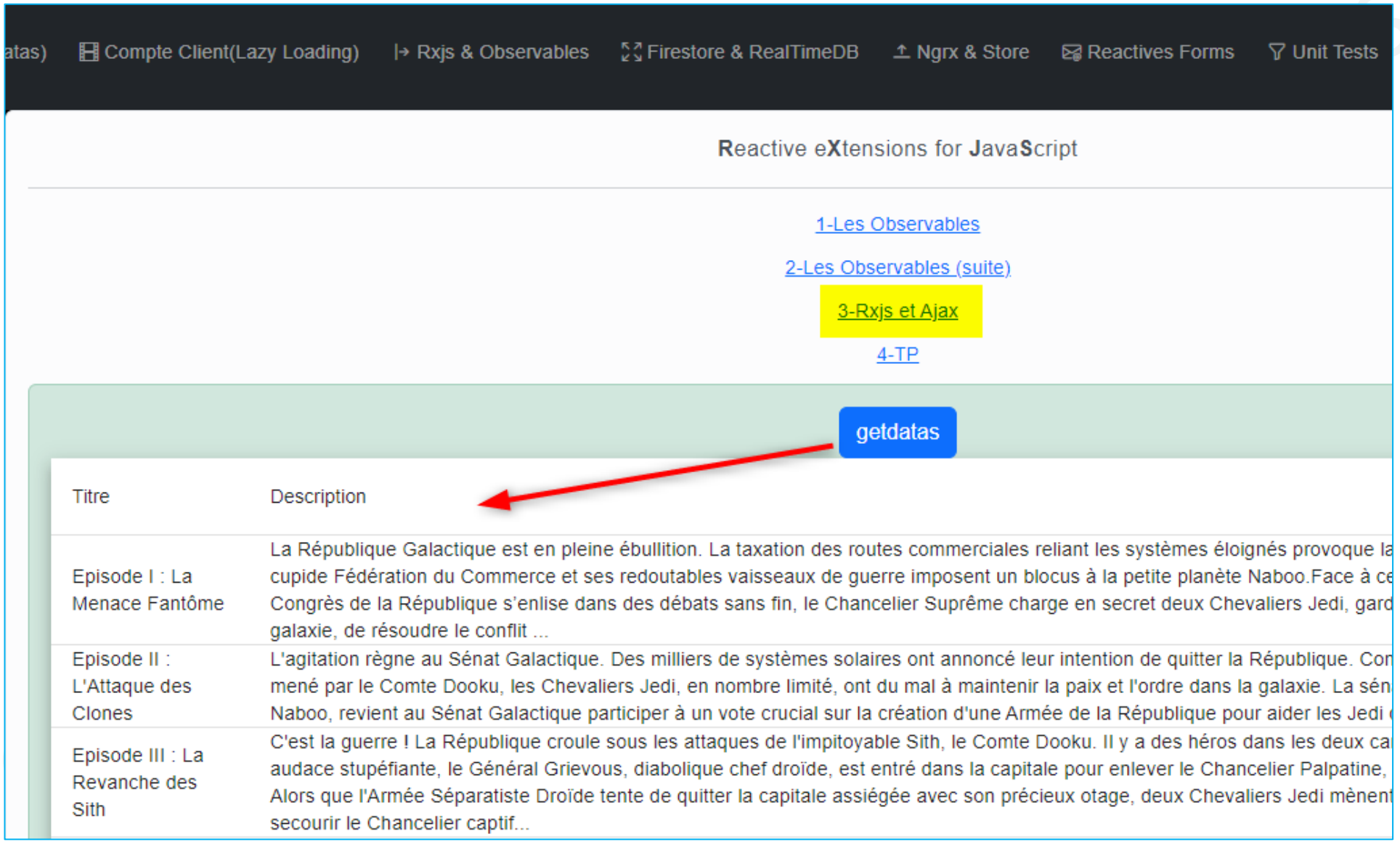


```
// ----------------------------------------
// création de l'observable #1 : PLUS
// ----------------------------------------
const plus$ = fromEvent(this.eltPlus.nativeElement, 'click')
  .pipe(
    startWith(0),
    tap(
      (valObs) => console.log(`Valeur de l'observable : `, valObs)
      // output : PointerEvent {isTrusted: true  ...
    ),
    map(
      () => { return 1 }
    ),
    tap(
      (valObs) => console.log(`Valeur de l'observable après le map : `,
      valObs)
      // output : 1
    ),
    scan(
      (accu: number, nelleValeur: number) => { return accu + nelleValeur }
    ),
    tap(
      (valObs) => {
        console.log(`Valeur de l'observable après le scan : `, valObs);
        // output : 2 - 3 - 4 - 5 ....
        this.eltMoins.nativeElement.style.display = 'inline';
      }
    )
  )
// .subscribe(
//   (valObs) => this.eltPanier.nativeElement.innerHTML = `<strong>${valObs}
</strong> article(s) en panier`
// );
```

```
// ----------------------------------------
// création de l'observable #2 : MOINS
// ----------------------------------------

const moins$ = fromEvent(this.eltMoins.nativeElement, 'click')
  .pipe(
    startWith(0),
    map(
      () => { return 1 }
    ),
    scan(
      (accu: number, nelleValeur: number) => { return accu + nelleValeur }
    )
  )
// .subscribe(
//   (valObs) => this.eltPanier.nativeElement.innerHTML = `<strong>${valObs}
</strong> article(s) en panier`
// );
```

```
// ----------------------------------------------------------------
// Re-Composition des 2 observables : Combinaison - Fusion(merge)
// ----------------------------------------------------------------

combineLatest([plus$, moins$])
  .pipe(
    map(
      ([valObs1, valObs2]) => {
        return valObs1 - valObs2;
      }
    )
  )
  .subscribe(
    (valObs) => {
      this.eltPanier.nativeElement.innerHTML = `<strong>${valObs}</strong>
      article(s) en panier`;

      if (valObs === 0) {
        // this.eltMoins.nativeElement.setAttribute('disabled','true');
        this.eltMoins.nativeElement.style.display = 'none';
      }
    }
  );
```

TP (Requêtes Ajax : forkJoin)



TP (Requêtes Ajax : forkJoin)

ajax.component.html | rxjs.module.ts

webApp-corrigé > src > app > webApp > formation > rxjs > composants-routing > ajax > ajax.component.html

```html
<button class="btn btn-primary" (click)="getDatas()">getdatas</button>

<table mat-table [dataSource]="films" class="mat-elevation-z8">

    <ng-container matColumnDef="title">
        <th mat-header-cell *matHeaderCellDef> Titre </th>
        <td mat-cell *matCellDef="let element"> {{element.title}} </td>
    </ng-container>


    <ng-container matColumnDef="desc">
        <th mat-header-cell *matHeaderCellDef> Description </th>
        <td mat-cell *matCellDef="let element"> {{element.desc}} </td>
    </ng-container>


    <tr mat-header-row *matHeaderRowDef="displayedColumns"></tr>
    <tr mat-row *matRowDef="let row; columns: displayedColumns;"></tr>

</table>
```

```typescript
// ----------------------------------------------------------
public getDatas = () => {

  const URL1 = 'http://localhost:3001/films1';
  const URL2 = 'http://localhost:3001/films2';

  forkJoin({
    rst1:ajax.getJSON(URL1) as Observable<Films>,
    rst2:ajax.getJSON(URL2) as Observable<Films>
  })
  .pipe(
    tap(
      (datas:any) => {
        this.films = datas.rst1.concat(datas.rst2);
        console.table(this.films)
      }
    )
  )
  //-------------------------------------
  //  forkJoin({
  //    rst1:this._http.get(URL1) as Observable<Films>,
  //    rst2:this._http.get(URL2) as Observable<Films>
  // })
  // .pipe(
  //   tap(
  //     (datas:any) => {
  //       this.films = datas.rst1.concat(datas.rst2);
  //       console.table(this.films)
  //     }
  //   )
  // )
  .subscribe()
}
```

Subject et BehaviourSubject

Un simple Observable est un objet en lecture seule. Il n'est pas possible d'invoquer l'événement ou le changement de valeur …

Exemple : le client HTTP utilise un *Observable* pour exposer l'événement à une requête HTTP.

movies.service.ts

src > app > shared > services > movies.service.ts > MovieService > getMovies$

```ts
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { map, Observable, tap } from 'rxjs';
import { Movie } from '../class/movie';

// décorateur
@Injectable({  providedIn: 'root'})

export class MovieService {
    // const (injection de dépendances)
    constructor(private _http: HttpClient) { }

    // méthodes
    public getMovies$(): Observable<Movie[]> {

        // const API_URL = 'http://localhost:3001/films1';
        const API_URL = 'https://dev.webjs.fr/JSON_API/films.json';

        return this._http.get<Movie[]>(API_URL)
            .pipe(tap((responseHTTP: any) => { console.log('Depuis le service : ', responseHTTP); }),
                // delay(4000),
                map((datas: Movie[]) => {
                    return datas.filter(
                        (data: Movie) => {
                            // return data.id==5;
                            return data.id >= 1;
                        }
                    )
                }))
    }
}
```

```html
<div class="row alert alert-primary">
    <div class="col-md-3" *ngFor="let objfilm of Movies$ | async">
        <div class="card">
```

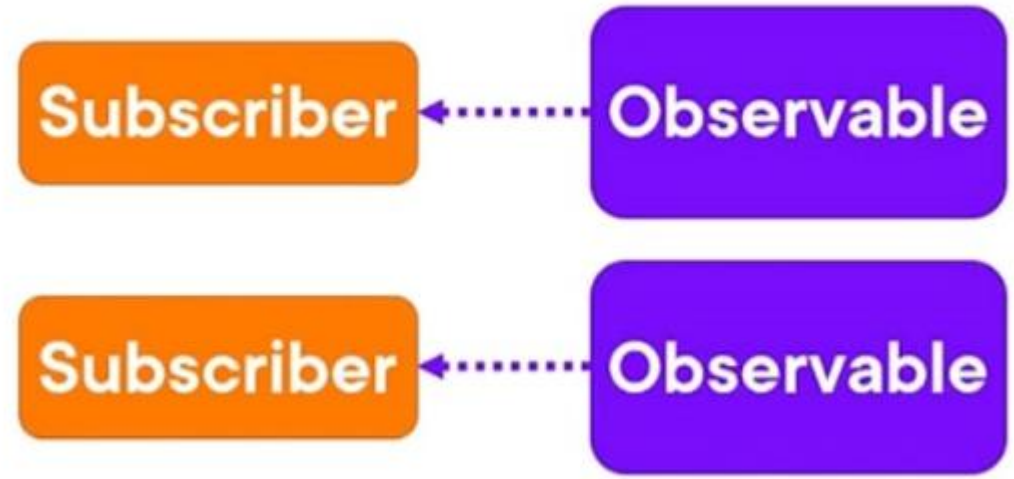
```ts
export class MovieBehaviourService {


    private moviesSubject = new BehaviorSubject<Movie[]>([]);
    public movies = this.moviesSubject.asObservable();

    // asObservable permet de retourner un observable à partir d'un Subject
    // En empêchant les composants abonnés d'appeler la méthode next sur l'observable retourné
    // Ainsi uniquement le Subject intial peut être utilisé pour diffuser de nouvelles séquences de valeurs
    // Et pourra notifier les abonnés des MAJ 😊

    constructor(private _http: HttpClient) { }

    public getMovies() {

        // const API_URL = 'http://localhost:3001/films1';
        const API_URL = 'https://dev.webjs.fr/JSON_API/films.json';

        this._http.get<Movie[]>(API_URL)

            .pipe(
                catchError(
                    () => {
                        this.moviesSubject.error('Une erreur est survenue !');
                        return [];
                    }),
                map(
                    (movies) => { return movies }
                )
            )
            .subscribe(
                (movies) => this.moviesSubject.next(movies)
                // next met à jour l'état courant de l'instance de Behaviour Subject
                // la différence entre Subject et behaviourSubject est le fait que behaviour intialise une 1ère valeur par défaut
                // cette valeur est renvoyée aux composants qui s'abonneet même après le 1er envoi de datas !
                // même si les composants s'abonnent plus tard !
```

A l'opposé et contrairement aux *Observables*, les Subjects peuvent émettre des événements (valeurs) aux « observers » abonnés à l'aide de la fonction next().

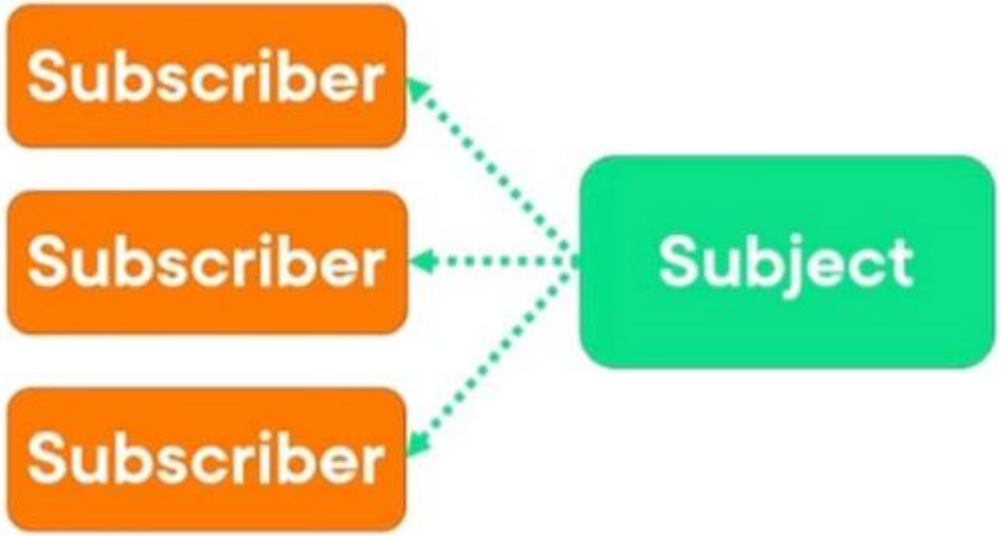
Les modifications crées dans le *Subject*, avec la fonction next(), pourront être écoutées avec la fonction subscribe() ou le pipe async !.

asObservable peut « **déguiser un *Subject* en *Observable* »** pour masquer son comportement de *Subject* et ne montrer que sa capacité à s'abonner aux modifications.

UNICAST VS MULTICAST



L'Observable est normalement UNICAST



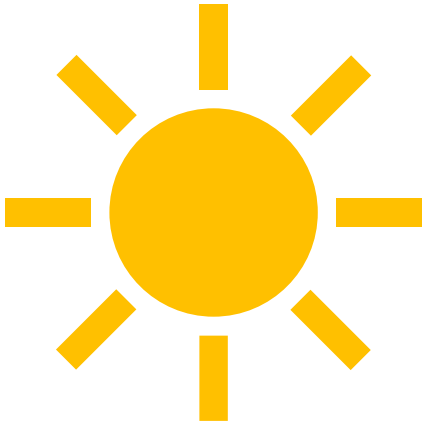
Le Subject est MULTICAST

UNICAST VS MULTICAST

L’observable unicast ne peut être écouté que par un seul subscriber (l’observer) . Si un nouvel observer vient écouter, un nouvel Observable est créé et les anciennes valeurs sont … perdues

L’observable multicast peut être écouté par plusieurs subscribers (observers) . Si un nouvel observer vient écouter, les valeurs déjà émises peuvent être reçues 😊

Hot VS Cold Observables



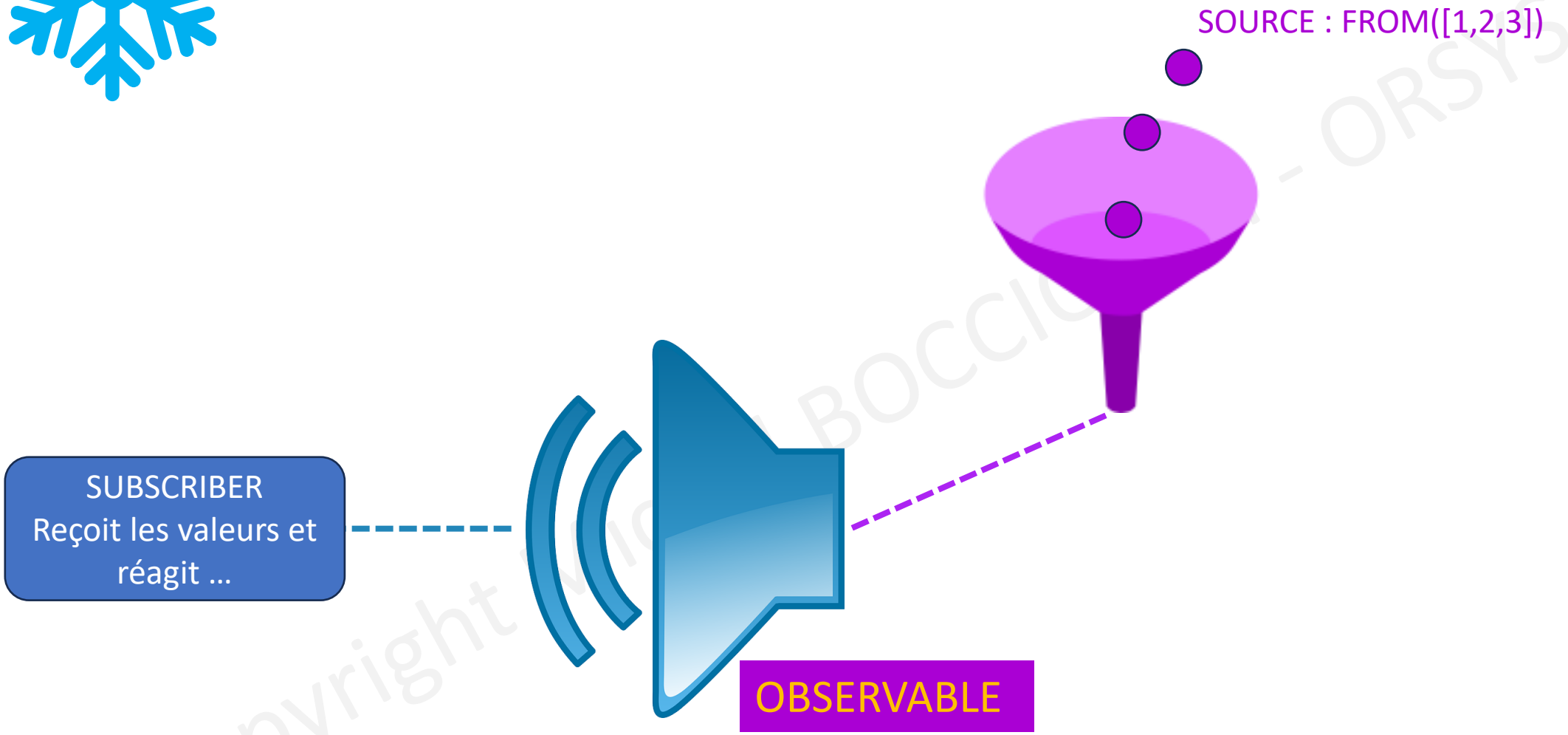
La création de l'observable active la source des valeurs qui peuvent être émises sans subscribers ! L'Hot Observable est généralement MULTICAST

```
productSubject = new Subject<number>();
this.productSubject.next(19);
```

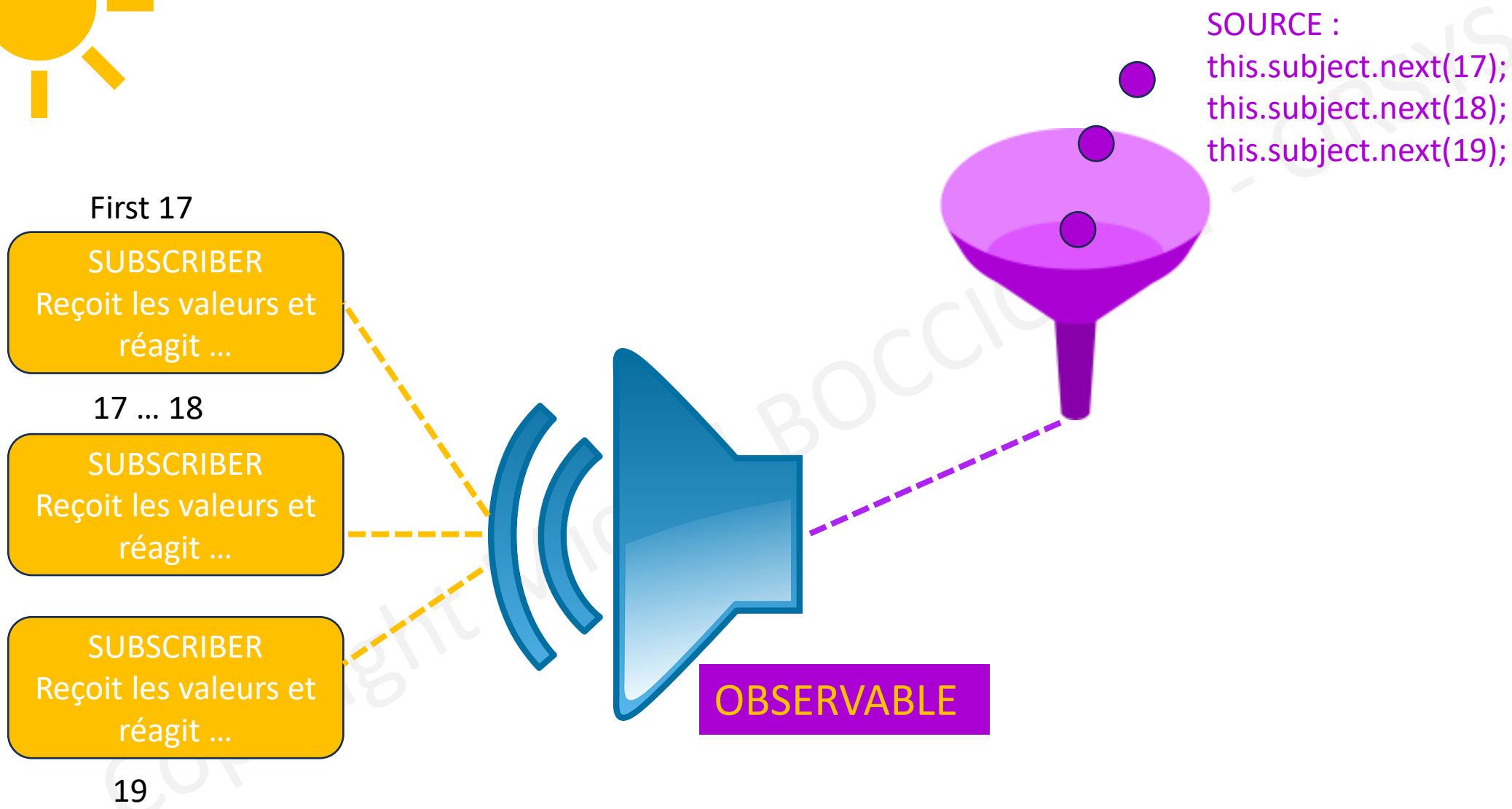
N'émet rien tant qu'il n'y a pas de subscribers donc d'abonnement (.subscribe | async) ! L'abonnement active une Source UNICAST

```
Movies$ =this._http.get<Movies[]>(_url)
```

COLD UNICAST



HOT MULTICAST



Le Signal

La notion de réactivité en Front End

Finalement quel Problème doit on résoudre ?
Tous les Frameworks modernes essaient de résoudre ce même problème

	A	B
1	Montant HT	150
2	Tva	0,2
3	Montant TTC	=B1*B2+B1

index.html × script.js ×

```
1  let montantHT=150;
2  const TVA=0.2;
3  let total=montantHT*TVA + montantHT;
4  console.log(total);
5
6  let montantHt=200;
7  console.log('Est ce réactif ? : ', total, '🤔😠');
8
```

Console ×

```
180
Est ce réactif ? :  180 🤔😠
```

La notion de réactivité en Front End

Comment implémenter cela dans nos applis JS ? (Google Sheets l'a déjà fait avec du « *fichier Excel* » en JS)

3 Méthodes pour implémenter la réactivité dans les Frameworks JS :

1- La méthode **Value Based** :

L'état actuel d'un composant est stockée dans une zone mémoire précise (store, composant.ts, local Storage, etc …)
le Framework utilise le mode « **Detection Change**[1] **»** ou **« *Dirty Checking*[2]** » pour savoir si la valeur a changé car il ne peut pas l'observer, il parcourt l'ensemble des composants pour vérifier les différents états et mettre à jour le DOM. La librairie actuelle* Zone.js fait ce travail de vérification.

Avantages : aucunes connaissances du « **core** » **Zone**.js n'est requise et c'est extrêmement simple à utiliser pour le dév. Le DOM est automatiquement MAJ.

Inconvénients: Performances 😘 🤯, des quantités de traitements sont faits pour mettre à jour les Vues HTML. Pour améliorer cela, il faut devenir expert en **Zone** et utiliser le **change detector** et le **onPush** !

1 : Pour Angular
2 : utilisé dans d'autres framework

La notion de réactivité en Front End

3 Méthodes pour implémenter la réactivité dans les Frameworks JS :

2- La méthode **Observable Based**:

Cette méthode est beaucoup plus performante que la « Value based » mais demande une solide investigation et des temps d'intégration et de compréhension conséquents.

La notion d'abonnement permet de mettre à jour le DOM uniquement lorsqu'on en aura besoin !

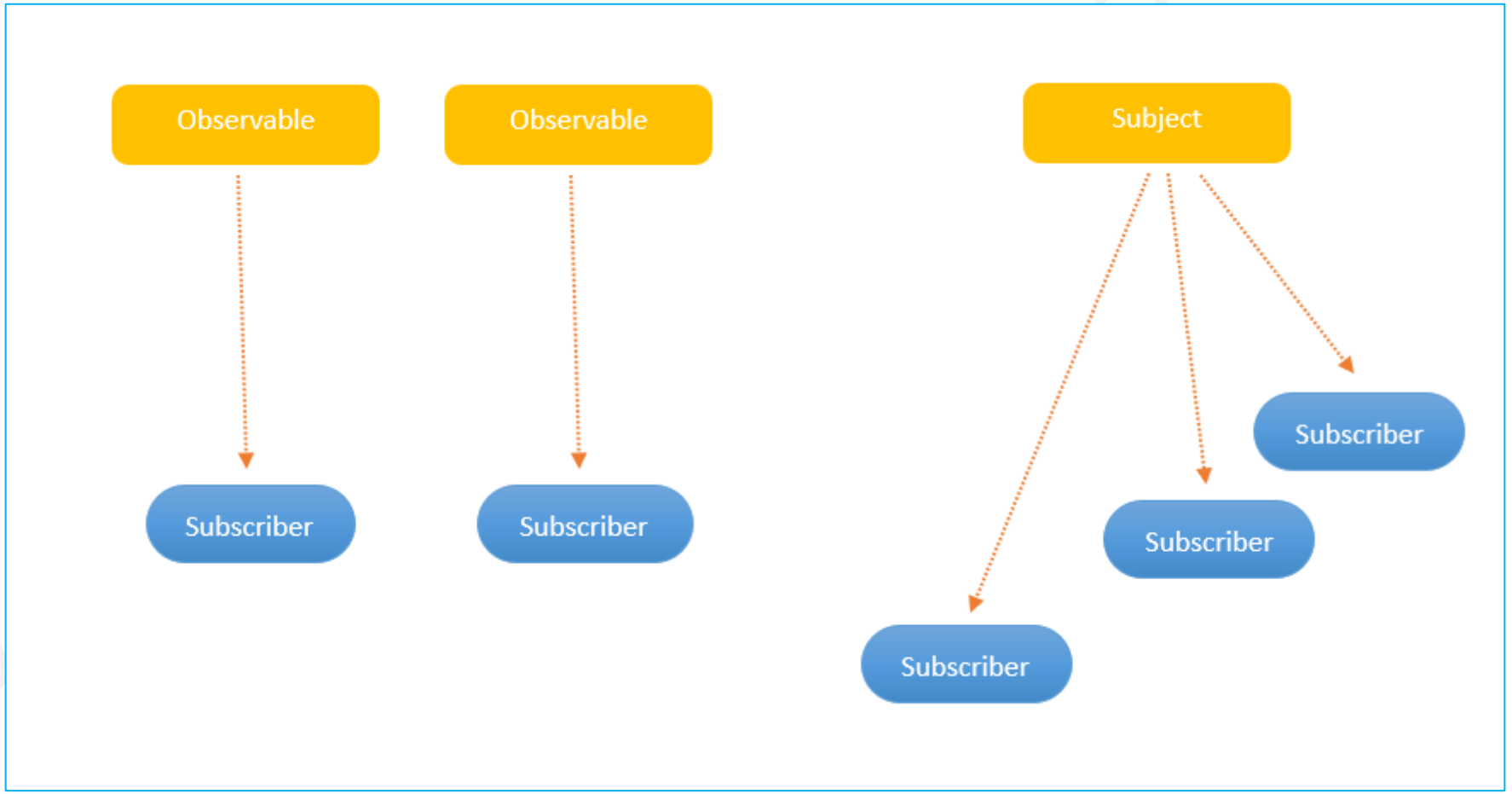
*Rappel :
1 « Observable » est un Objet qui émet une séquence de valeurs
1 « Observer » observe l'observable et réagit à l'arrivée de **nouvelles** valeurs

« Subject » et « Behaviour Subject (avec valeur initiale) » sont en même temps Observables et Observers :
Le « Subject » est multidiffusion par rapport à l'observable qui est monodiffusion

La notion de réactivité en Front End

Le « Subject » est multidiffusion par rapport à l'observable qui est monodiffusion, il faut user de complexité pour recevoir les valeurs du même Observable par plusieurs « Subscribers » 🙄

Plusieurs Subscriber peuvent s'attacher au Subject et recevoir les valeurs émises. 😊



Angular 17 – Le Signal!

Le Signal

Meilleures performances en termes d'éxécution !
Le signal réduit la complexité du parcours de l'arbre des composants parents-enfants

La vérification du changement d'affichage dans le composant, devient de plus en plus fine et étroitement liée au composant concerné. La granularité du « Change Detection Mode » est nettement améliorée.

Zone.js utilisé depuis le début d'Angular pour détecter les changements dans les composants va être progressivement arrêté ! (pour rappel, le principe d'une Zone permet d'éxécuter du code dans le contexte Angular et hors contexte Angular) 😳

Le signaux peuvent interagir avec RXJS et apporter de la complémentarité !
On peut passer facilement d'un Observable à un signal et vice –versa

toSignal() toObservable()

Le Signal

Le signal représente l'état d'un composant, d’une propriété , d’une valeur spécifique.

Lorsque le signal est mis à jour, les parties de l'application qui dépendent de ce signal peuvent être informées et modifier leurs propres données. (mécanisme d’optimisation que l’on reconnaît dans le Store)

Lorsque une valeur change, le signal émet une information et l’application peut se mettre à jour de manière sélective.

https://angular.dev/guide/signals

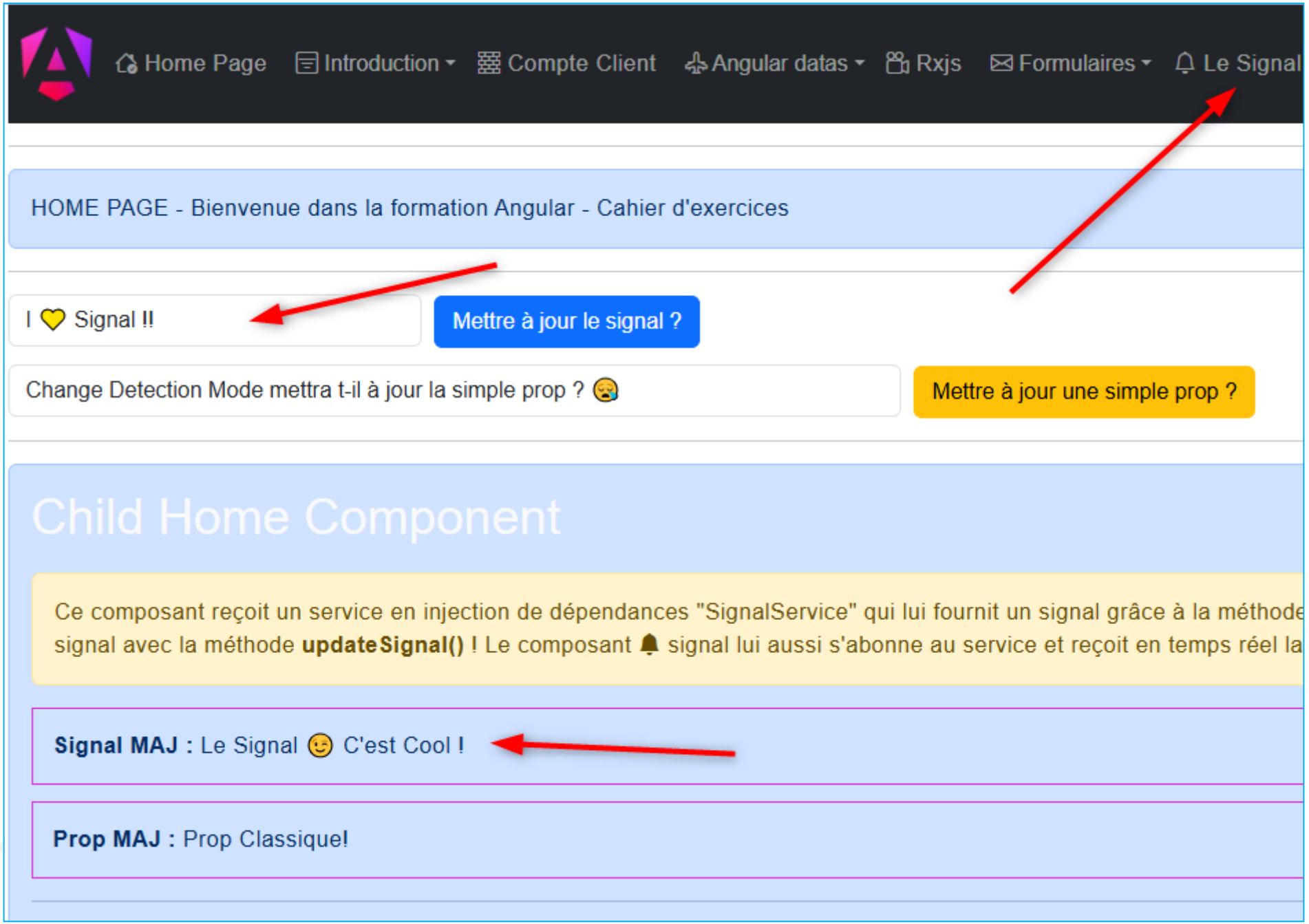
1- Le signal permet de réduire le nombre de mise à jour effectuées lors du "Change Detection Mode" d'Angular !
La réactivité est performante ! Plus besoin de Zone.js ! Zone va ou est en train d'être abandonné ! ✨🎈
Interopérabilité avec RXJS : méthodes toSignal() et toObservable()

2- Le signal est une valeur réactive utilisée pour représenter un état ou une donnée changeante dans une application.

- signal() : crée une propriété de type signal
- set : écrase la valeur du signal et en crée une nouvelle
- update: prend la dernière valeur du signal et la transforme
- computed : permet de calculer des valeurs dérivées à partir du signal
- effects : observer les changements de valeur du signal et crée un "effet d'observation call bacjk d'action"

2 étoiles attribuée(s)

Note : 4/10 😉 2 (étoiles attribuée(s))


Home Page
Introduction
Compte Client
Angular datas
Rxjs
Formulaires
Le Signal
HOME PAGE - Bienvenue dans la formation Angular - Cahier d'exercices
I Signal !!
Mettre à jour le signal ?
Change Detection Mode mettra t-il à jour la simple prop ?
Mettre à jour une simple prop ?
Child Home Component
Ce composant reçoit un service en injection de dépendances "SignalService" qui lui fournit un signal grâce à la méthode
signal avec la méthode updateSignal() ! Le composant signal lui aussi s'abonne au service et reçoit en temps réel la
Signal MAJ : Le Signal C'est Cool !
Prop MAJ : Prop Classique!

Signal Angular 16+

Je suis un composant enfant (*ou très éloigné* ...), je reçois (*m'abonne* 😉) au **service signalService** en injection de dépendances et peut accéder avec beaucoup plus de réactivités aux datas !

Signal : Le Signal 😉 C'est Cool !

Le Signal Angular

- La primitive Signal()
- Computed
- Méthode Set() Update() Mutate()

```
//signal

public signalNb = signal(3);
// la primitive signal = lecture ou en ecriture

// calculé à partir d'autre signaux
public infos = computed(
  () => `${this.signalNb()} étoiles attribuée(s) !`    )

//viewChildren
@ViewChild('plus') eltPlus!: ElementRef<HTMLElement>;
@ViewChild('moins') eltMoins!: ElementRef<HTMLElement>;

ngAfterViewInit() {
  // -------------------------
  fromEvent(this.eltPlus.nativeElement, 'click').pipe(
    tap( () => { this.signalNb.update( (valSignal) => valSignal + 1 )} )
  ).subscribe()
  // -------------------------
  fromEvent(this.eltMoins.nativeElement, 'click').pipe(
    tap( () => { this.signalNb.update( (valSignal) => valSignal - 1 ) } )
  ).subscribe()
  }
}
```

Les Formulaires

Les formulaires Angular (avec Matérial Design)

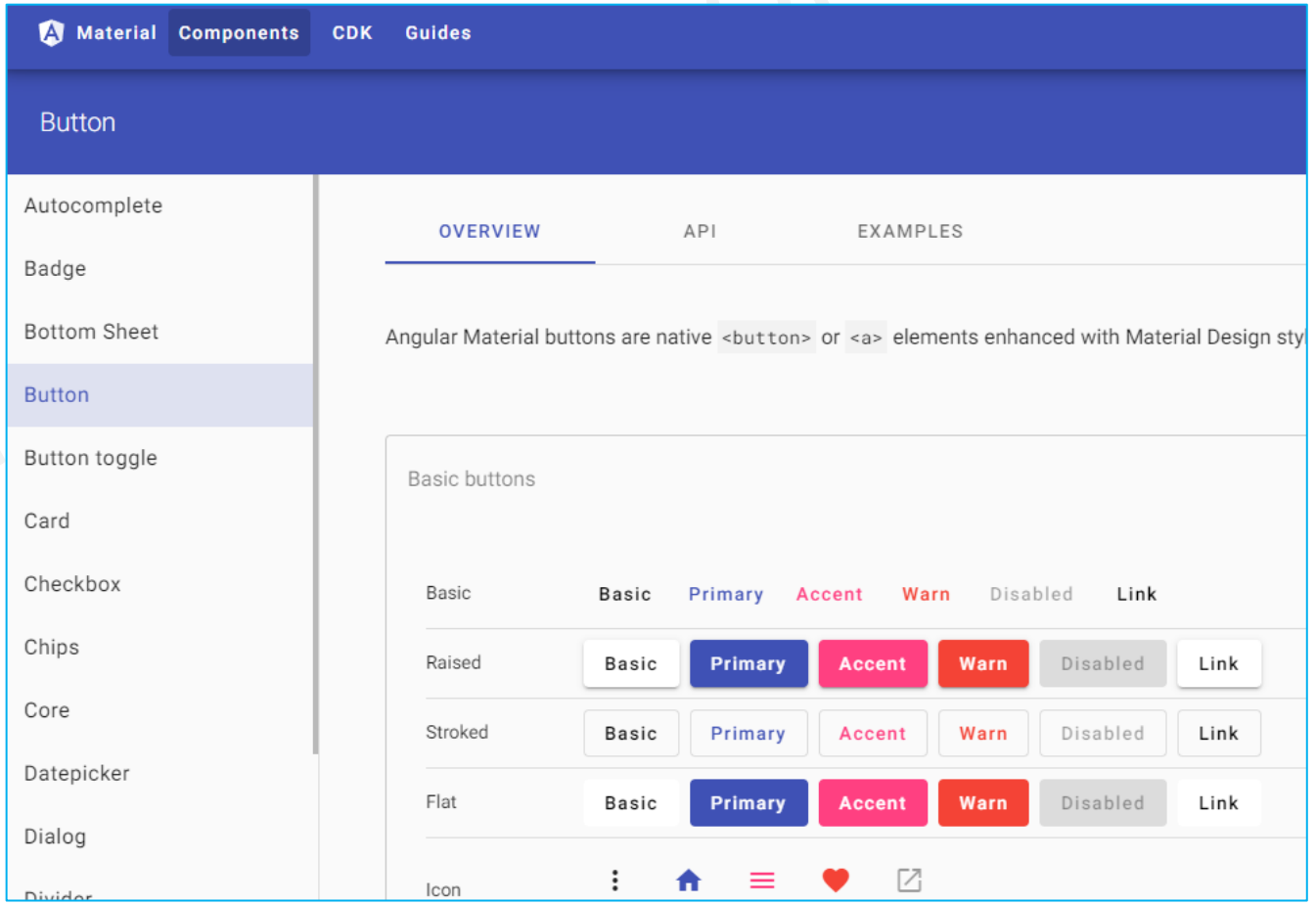
- https://material.angular.io/components/categories
- https://fonts.google.com/icons

HTML TS CSS

```
<section>
  <div class="example-label">Basic</div>
  <div class="example-button-row">
    <button mat-button>Basic</button>
    <button mat-button color="primary">Primary</button>
    <button mat-button color="accent">Accent</button>
    <button mat-button color="warn">Warn</button>
    <button mat-button disabled>Disabled</button>
    <a mat-button href="https://www.google.com/" target="_blank">Link</a>
  </div>
```

HTML TS CSS

```
<mat-form-field>
  <mat-label>Input</mat-label>
  <input matInput>
</mat-form-field>
<mat-form-field>
  <mat-label>Select</mat-label>
  <mat-select>
    <mat-option value="one">First option</mat-option>
    <mat-option value="two">Second option</mat-option>
  </mat-select>
</mat-form-field>
<mat-form-field>
  <mat-label>Textarea</mat-label>
  <textarea matInput></textarea>
</mat-form-field>
```



Les formulaires Angular (avec Matérial Design)

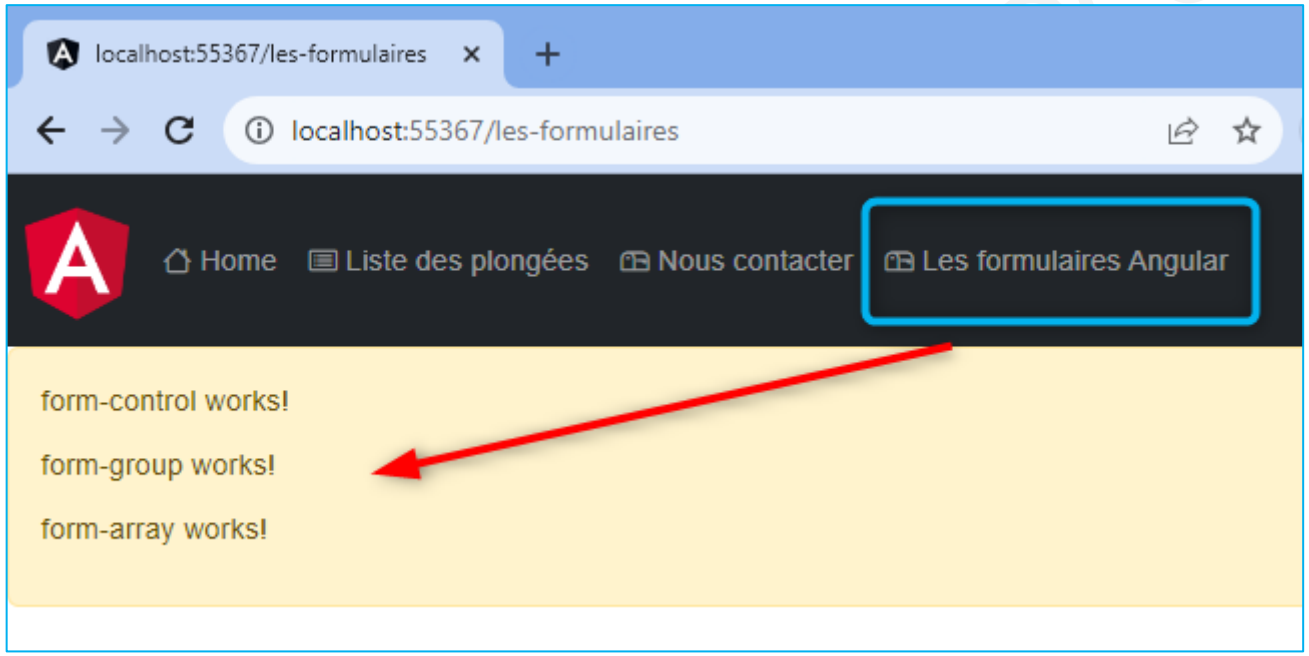
ng add @angular/material

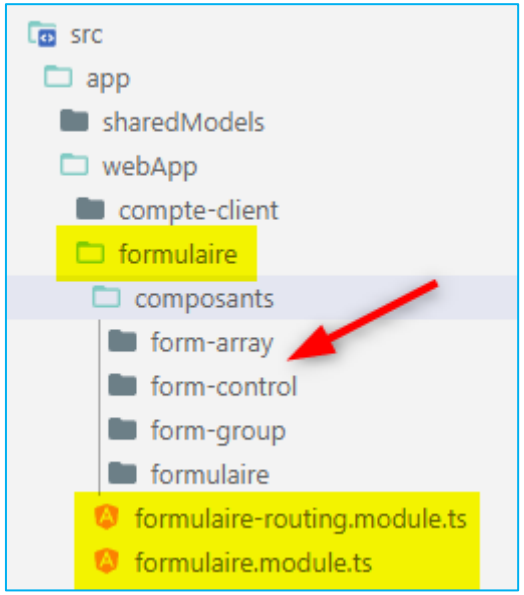
ng add @angular/localize
ng add @angular/pwa

Les formulaires Angular (avec Matérial Design)

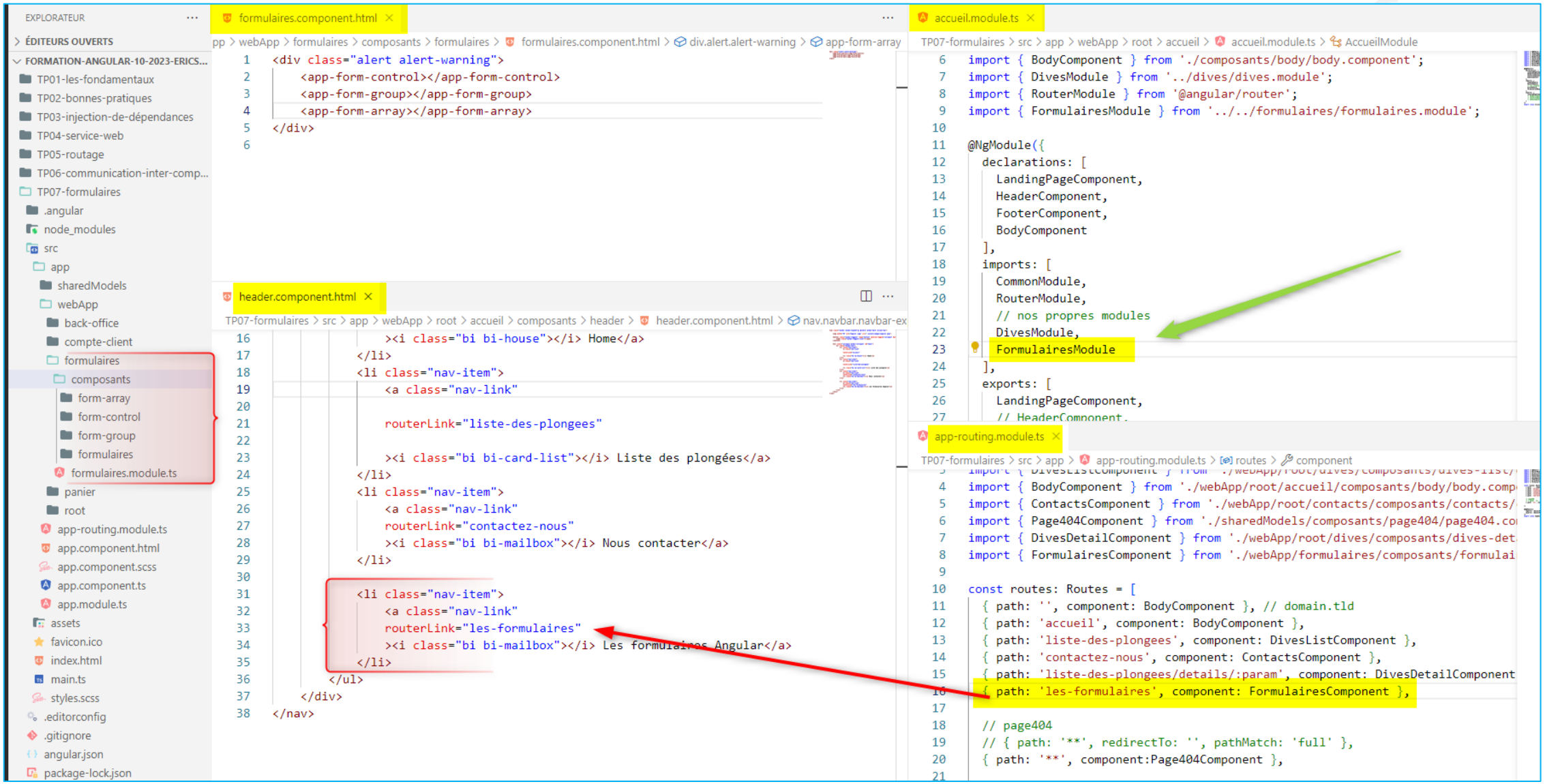
https://dev.webjs.fr/1-Angular/ressources-formation/material.module.ts
https://dev.webjs.fr/1-Angular/ressources-formation/material-all.module.ts

TP : mise en place d'une architecture de projet avec le module « formulaires » comme illustré ci-dessous.
Ressources en ligne (*corrigé en ligne*) :
https://dev.webjs.fr/1-Angular/ressources-formation/reactive-forms.zip



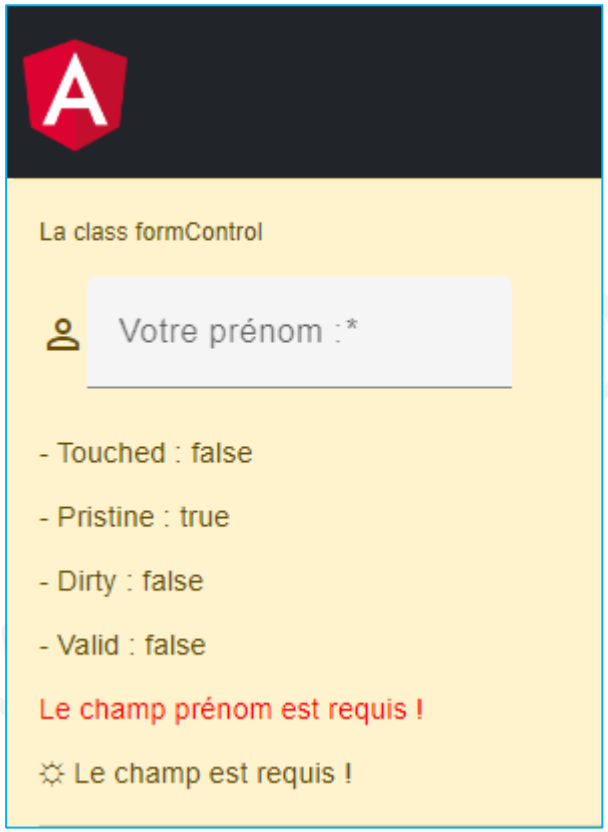


Les formulaires Angular (avec Matérial Design)



Les formulaires Angular

- Mise en lumière des différences importantes entre le « Template Driven Form » et le « Reactive Form »
- Choix de l'utilisation
- Contexte d'utilisation – validation – contrôle de saisies de données – traitement des données saisies



La Class form-control

form-control.component.ts

TP06-formulaires > src > app > webApp > formulaires > composants > form-control > form-control.c

```ts
import { Component } from '@angular/core';
import { FormControl, Validators } from '@angular/forms';

@Component({
  selector: 'app-form-control',
  templateUrl: './form-control.component.html',
  styleUrls: ['./form-control.component.scss']
})
export class FormControlComponent {

  // props
  public prenom:FormControl<string | null>;

  // const
  constructor() {
    this.prenom = new FormControl(
      //val par défaut
      '',
      // Validators
      [
        Validators.required,
        Validators.minLength(3),
        Validators.maxLength(10)
      ]
    );
  }

}

```

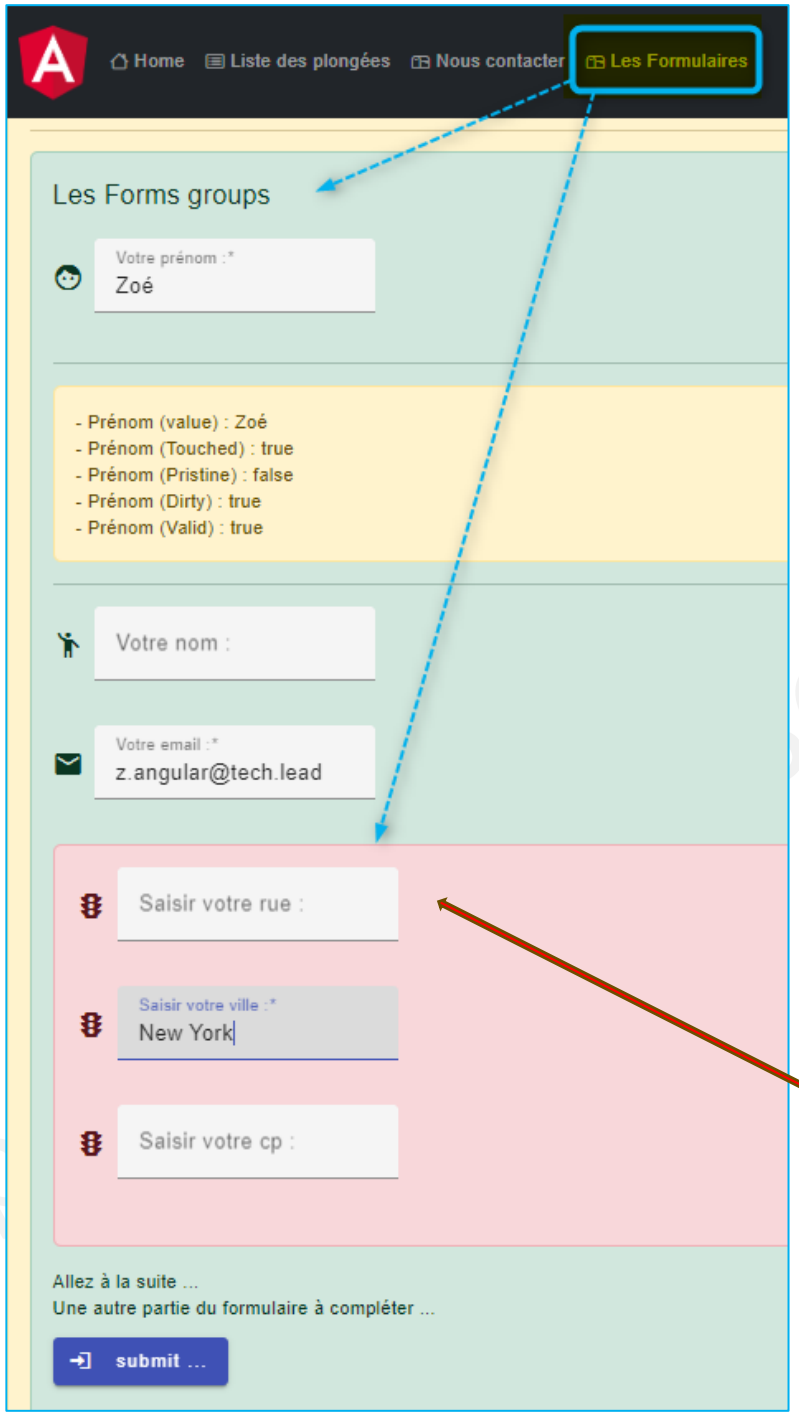
form-control.component.html

TP06-formulaires > src > app > webApp > formulaires > composants > form-control > form-control.component.html

```html
<h5>La class formControl</h5>

    <mat-icon>perm_identity</mat-icon>

    <mat-form-field>
        <mat-label>Votre prénom :</mat-label>
        <input type="text" matInput [formControl]="prenom">
    </mat-form-field>

<!-- tout ce qui suit peut être utilisé autant dans les "templates driven forms" c

<div>
    <p>- Touched  : {{prenom.touched}}</p>
    <p>- Pristine  : {{prenom.pristine}}</p>
    <p>- Dirty  : {{prenom.dirty}}</p>
    <p>- Valid  : {{prenom.valid}}</p>
</div>


<p [hidden]="!prenom.touched && !prenom.valid" style="color:lightgreen">
    Le champ prénom est validé !
</p>

<p [hidden]="prenom.touched || prenom.valid" style="color:red">
    Le champ prénom est requis !
</p>

<div *ngIf="prenom.touched && prenom.valid; then blockOk else blockNotOk"></div>

    <ng-template #blockOk>✿ Le champ est validé !</ng-template>
    <ng-template #blockNotOk>✿ Le champ est requis !</ng-template>

<hr>

```

La Class form-control

La class form-group



form-group.component.html

TP06-formulaires > src > app > webApp > formulaires > composants > form-group > form-group.component.html > div.a

```
1  <div class="alert alert-success">
2
3      <h2>Les Forms groups</h2>
4      <form [formGroup]="monForm">
5          <!-- <form [formGroup]="monForm" (submit)="onSubmit()"> -->
6
7          <mat-icon id="icone-prenom">face</mat-icon> 
8          <mat-form-field>
9              <mat-label>Votre prénom :</mat-label>
10             <input type="text" matInput formControlName="prenom">
11         </mat-form-field>
12         <hr>
13
14         <div class="alert alert-warning">
15             - Prénom (value) : {{monForm.controls['prenom'].value}}
16             <br> - Prénom (Touched) : {{monForm.controls['prenom'].touched}}
17             <br> - Prénom (Pristine) : {{monForm.controls['prenom'].pristine}}
18             <br> - Prénom (Dirty) : {{monForm.controls['prenom'].dirty}}
19             <br> - Prénom (Valid) : {{monForm.controls['prenom'].valid}}
20         </div>
21         <hr>
22
23         <p></p>
24         <mat-icon>emoji_people</mat-icon> 
25         <mat-form-field>
26             <mat-label>Votre nom :</mat-label>
27             <input type="text" matInput formControlName="nom">
28         </mat-form-field>
29         <p></p>
30
31         <p></p>
32         <mat-icon>email</mat-icon> 
33         <mat-form-field>
34             <mat-label>Votre email :</mat-label>
35             <input type="text" matInput formControlName="email">
36         </mat-form-field>
37         <p></p>
38
39         <div formGroupName="adresse" class="alert alert-danger">
40             <div>
41                 <mat-icon>traffic</mat-icon> 
42                 <mat-form-field>
43                     <mat-label>Saisir votre rue :</mat-label>
44                     <input matInput type="text" formControlName="rue">
45                 </mat-form-field>
46                 <p></p>
```

La Class formGroup

```ts
export class FormGroupComponent {

  public monForm: FormGroup;

  constructor(
    private _post: PostService,
    private _fb: FormBuilder) {

    this.monForm = this._fb.group({
      // collection des controls
      prenom: [
        // val par défaut définie comme(as) une string ou null)
        // 'valeur_saisie' as string | null,
        null as string | null,
        [
          Validators.required,
          Validators.minLength(5)
        ]
      ],
      // ------------
      nom: [
        null as string | null,
        Validators.required
      ],
      email: [null as string | null,
      [
        Validators.pattern('^[a-z0-9._%+-]+@[a-z0-9.-]+\\.[a-z]{2,}$'),
        Validators.required
      ]
      ],
      adresse: this._fb.group({
        rue: [null as string | null, Validators.required],
        ville: [null as string | null, Validators.required],
        cp: ['']
      })
    });
  }
```

```ts
  // méthodes
  public onSubmit = () => {
    console.log('group (form) principal : ', this.monForm.value);
    console.log('sous-group (form) adresse : ', this.monForm.controls['adresse'].value);
    this._post.postForm(this.monForm);
  }
```

TP :

1- récupérer le HTML : https://dev.webjs.fr/1-Angular/ressources-formation/form-group.component.zip

2- Mettre en place la validation TS

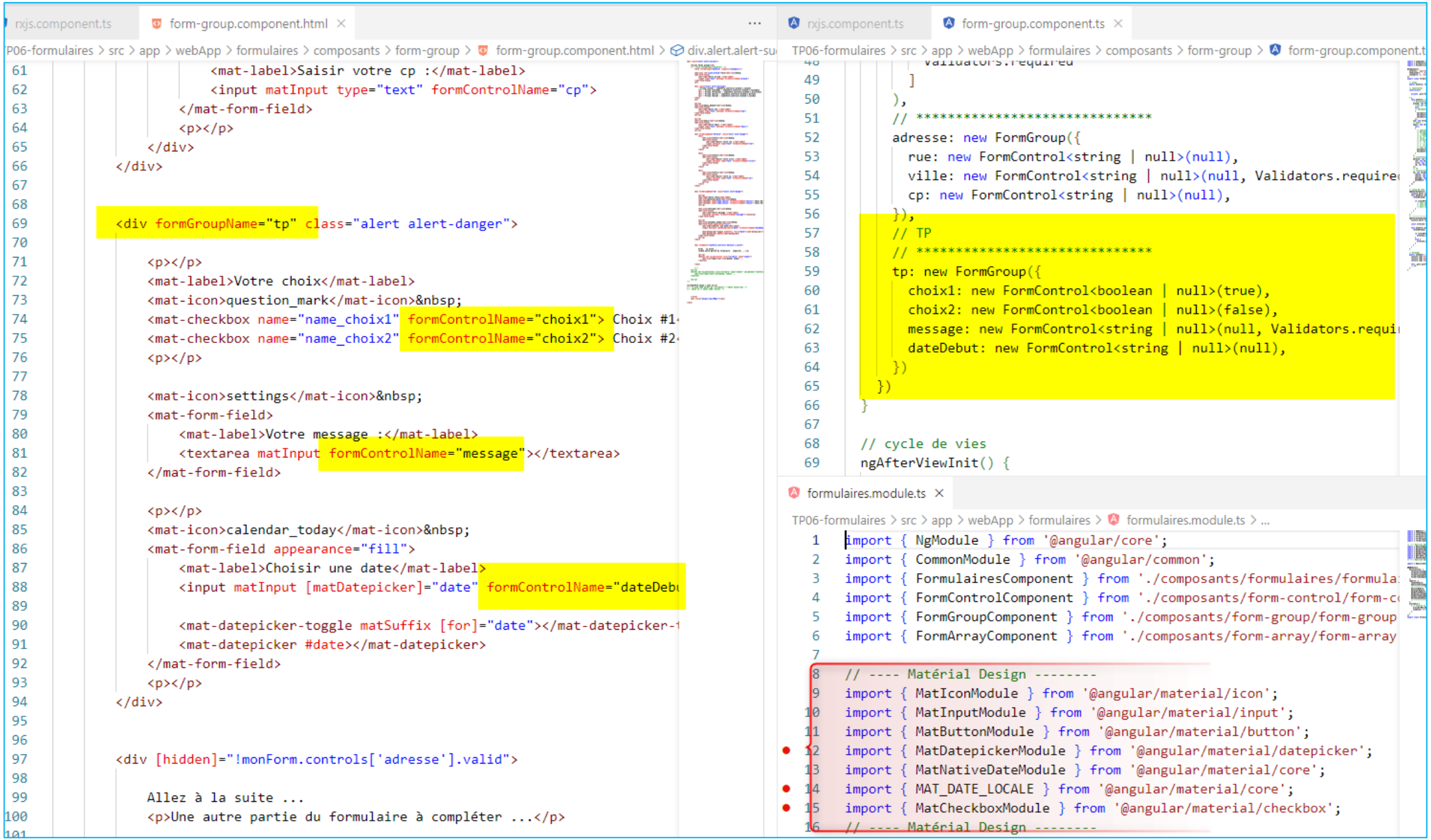
3- Créer le service « post.service.ts » qui poste les datas sur un json-server

```
"json-server": "json-server --watch src/assets/json/bdd.json -p 3001"
```

Base de données JSON : https://dev.webjs.fr/1-Angular/ressources-formation/json.zip

Corrigé en ligne : https://dev.webjs.fr/1-Angular/ressources-formation/post.service.zip

TP : ajouter un sous-form-group (choix1, choix2, message, dateDebut



rxjs.component.ts | form-group.component.html

TP06-formulaires > src > app > webApp > formulaires > composants > form-group > form-group.component.html > div.alert.alert-su

```
61                  <mat-label>Saisir votre cp :</mat-label>
62                  <input matInput type="text" formControlName="cp">
63              </mat-form-field>
64              <p></p>
65          </div>
66      </div>
67
68
69      <div formGroupName="tp" class="alert alert-danger">
70
71          <p></p>
72          <mat-label>Votre choix</mat-label>
73          <mat-icon>question_mark</mat-icon> 
74          <mat-checkbox name="name_choix1" formControlName="choix1"> Choix #1
75          <mat-checkbox name="name_choix2" formControlName="choix2"> Choix #2
76          <p></p>
77
78          <mat-icon>settings</mat-icon> 
79          <mat-form-field>
80              <mat-label>Votre message :</mat-label>
81              <textarea matInput formControlName="message"></textarea>
82          </mat-form-field>
83
84          <p></p>
85          <mat-icon>calendar_today</mat-icon> 
86          <mat-form-field appearance="fill">
87              <mat-label>Choisir une date</mat-label>
88              <input matInput [matDatepicker]="date" formControlName="dateDebu
89
90              <mat-datepicker-toggle matSuffix [for]="date"></mat-datepicker-t
91              <mat-datepicker #date></mat-datepicker>
92          </mat-form-field>
93          <p></p>
94      </div>
95
96
97      <div [hidden]="!monForm.controls['adresse'].valid">
98
99          Allez à la suite ...
100         <p>Une autre partie du formulaire à compléter ...</p>
101
```

rxjs.component.ts | form-group.component.ts

TP06-formulaires > src > app > webApp > formulaires > composants > form-group > form-group.component.t

```
49          ]
50        ),
51        // ******************************
52        adresse: new FormGroup({
53          rue: new FormControl<string | null>(null),
54          ville: new FormControl<string | null>(null, Validators.require
55          cp: new FormControl<string | null>(null),
56        }),
57        // TP
58        // ******************************
59        tp: new FormGroup({
60          choix1: new FormControl<boolean | null>(true),
61          choix2: new FormControl<boolean | null>(false),
62          message: new FormControl<string | null>(null, Validators.requi
63          dateDebut: new FormControl<string | null>(null),
64        })
65      })
66    }
67
68    // cycle de vies
69    ngAfterViewInit() {
```

formulaires.module.ts

TP06-formulaires > src > app > webApp > formulaires > formulaires.module.ts > ...

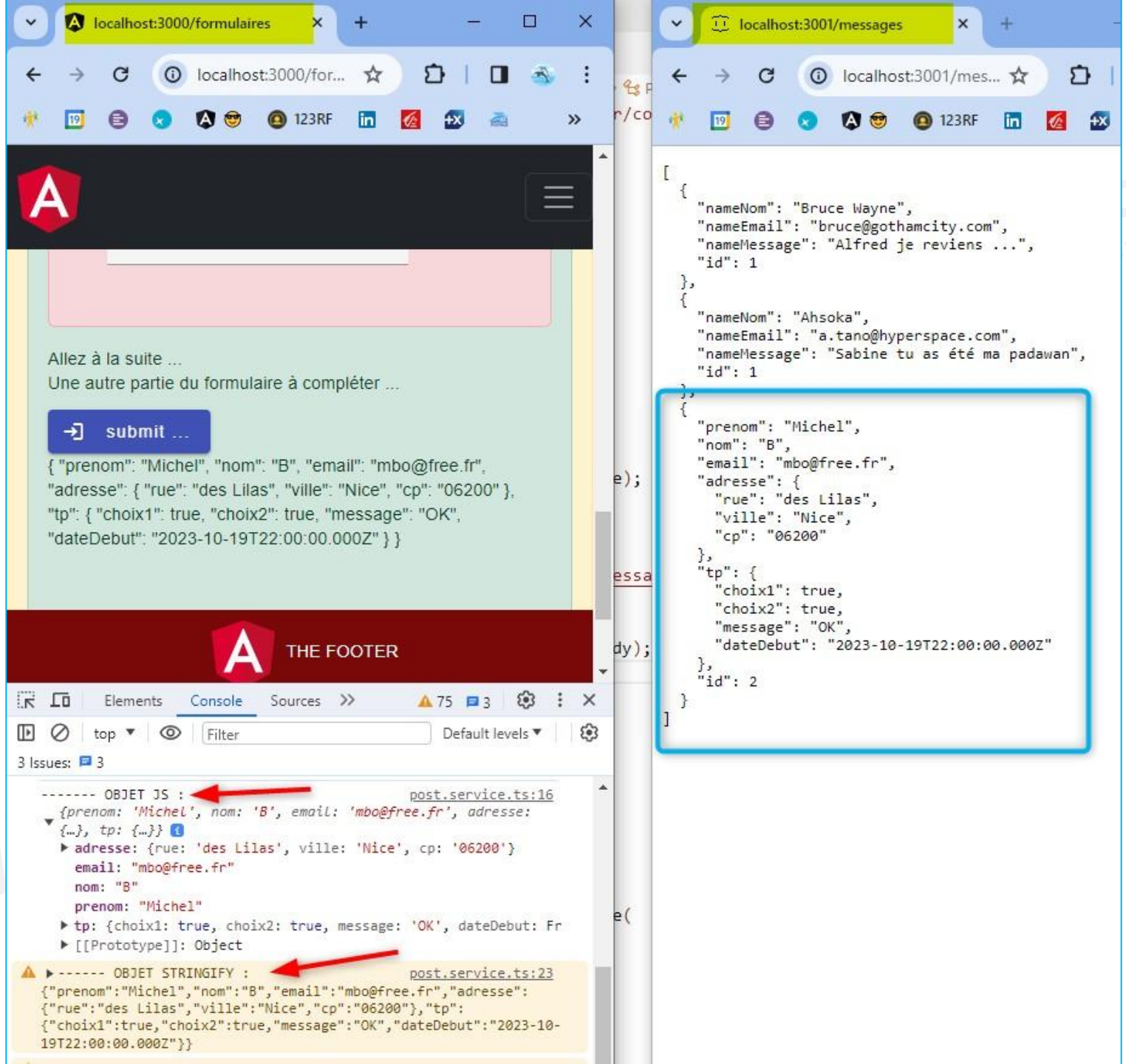
```
1   import { NgModule } from '@angular/core';
2   import { CommonModule } from '@angular/common';
3   import { FormulairesComponent } from './composants/formulaires/formula
4   import { FormControlComponent } from './composants/form-control/form-c
5   import { FormGroupComponent } from './composants/form-group/form-group
6   import { FormArrayComponent } from './composants/form-array/form-array
7
8   // ---- Matérial Design --------
9   import { MatIconModule } from '@angular/material/icon';
10  import { MatInputModule } from '@angular/material/input';
11  import { MatButtonModule } from '@angular/material/button';
12  import { MatDatepickerModule } from '@angular/material/datepicker';
13  import { MatNativeDateModule } from '@angular/material/core';
14  import { MAT_DATE_LOCALE } from '@angular/material/core';
15  import { MatCheckboxModule } from '@angular/material/checkbox';
16  // ---- Matérial Design --------
```

La Class formGroup

```
        // + : 1 ou plusieurs fois
        // * : 0 ou plusieurs fois
        // {val min , val max } : pour répéter
        Validators.pattern('^[a-z0-9._-]+@[a-z0-9.-]+\\.[a-z]{2,}$'),
        Validators.required
      ]
    ),
    // ******************************
    adresse: new FormGroup({
      rue: new FormControl<string | null>(null),
      ville: new FormControl<string | null>(null),
      cp: new FormControl<string | null>(null),
    })
  });

}
  // méthodes
  public onSubmit = () => {

    console.log('Group (form) principal : ', this.monForm.value);
    console.log('Sous-Group (form) adresse : ', this.monForm.controls['adresse'].value);

}
}
```

```
50              <p></p>
51          </div>
52          <div>
53              <mat-icon>tra
54              <mat-form-fie
55                  <mat-labe
56                  <input ma
57              </mat-form-fi
58
59              <p></
60          </div>
61      </div>
62
63
64      <p></p>
65
66      <button mat-raised-bu
67          <mat-icon>
68      </button>
69
70      <p></p>
71
72  </form>
73
```

```
1 Issue: 1

[webpack-dev-server] Server started: Hot Module Replacement disabled, Live Reloading enabled, Progress disabled, Overlay enabled.   index.js:485

Angular is running in development mode.   core.mjs:25499

Group (form) principal :   form-group.component.ts:61
{prenom: 'mbo', nom: 'gfgh', email: 'mbo@free.fr', adresse: {…}}
  adresse:
    cp: "3"
    rue: "2"
    ville: "2"
    [[Prototype]]: Object
  email: "mbo@free.fr"
  nom: "gfgh"
  prenom: "mbo"
  [[Prototype]]: Object

Sous-Group (form) adresse :   form-group.component.ts:62
{rue: '2', ville: '2', cp: '3'}
  cp: "3"
  rue: "2"
  ville: "2"
  [[Prototype]]: Object
```


localhost:3000/formulaires
localhost:3000/for...
Allez à la suite ...
Une autre partie du formulaire à compléter ...
submit ...
{ "prenom": "Michel", "nom": "B", "email": "mbo@free.fr",
"adresse": { "rue": "des Lilas", "ville": "Nice", "cp": "06200" },
"tp": { "choix1": true, "choix2": true, "message": "OK",
"dateDebut": "2023-10-19T22:00:00.000Z" } }
THE FOOTER
Elements
Console
Sources
3 Issues: 3
------- OBJET JS :
post.service.ts:16
{prenom: 'Michel', nom: 'B', email: 'mbo@free.fr', adresse: {…}, tp: {…}}
adresse: {rue: 'des Lilas', ville: 'Nice', cp: '06200'}
email: "mbo@free.fr"
nom: "B"
prenom: "Michel"
tp: {choix1: true, choix2: true, message: 'OK', dateDebut: Fr
[[Prototype]]: Object
------ OBJET STRINGIFY :
post.service.ts:23
{"prenom":"Michel","nom":"B","email":"mbo@free.fr","adresse":{"rue":"des Lilas","ville":"Nice","cp":"06200"},"tp":{"choix1":true,"choix2":true,"message":"OK","dateDebut":"2023-10-19T22:00:00.000Z"}}
localhost:3001/messages
localhost:3001/mes...
[
{
"nameNom": "Bruce Wayne",
"nameEmail": "bruce@gothamcity.com",
"nameMessage": "Alfred je reviens ...",
"id": 1
},
{
"nameNom": "Ahsoka",
"nameEmail": "a.tano@hyperspace.com",
"nameMessage": "Sabine tu as été ma padawan",
"id": 1
},
{
"prenom": "Michel",
"nom": "B",
"email": "mbo@free.fr",
"adresse": {
"rue": "des Lilas",
"ville": "Nice",
"cp": "06200"
},
"tp": {
"choix1": true,
"choix2": true,
"message": "OK",
"dateDebut": "2023-10-19T22:00:00.000Z"
},
"id": 2
}
]

form-group.component.ts

src > app > webApp > formulaires > composants > form-group > form-group.component.ts > FormGroupComponent > constructor

```ts
  public monForm: FormGroup;

  // constructor
  constructor(

    private _post:PostService

  ) {
    this.monForm = new FormGroup({
      // déclaration de tous les controls de ce formulaire (formGroup)
```

form-group.component.ts

> src > app > webApp > formulaires > composants > form-group > form-group.component.ts > FormGroupComponent > onSubmit

```ts
    )
  }

  // Méthodes
  public onSubmit = () => {
    console.log('Groupe principal : ', this.monForm.value);
    console.log('Sous Groupe : ', this.monForm.controls['adresse'].value);
    console.log('Sous Groupe : ', this.monForm.controls['tp'].value);

    this._post.postForm(this.monForm);
    // TP
  }
}
```

bdd.json

TP06-formulaires > src > assets > json > bdd.json > ...

```json
      "photo": "https://dev.webjs.fr/images/fond5.jpg"
    }
  ],
  "messages": [
    {
      "nameNom": "Bruce Wayne",
      "nameEmail": "bruce@gothamcity.com",
      "nameMessage": "Alfred je reviens ...",
      "id": 1
```

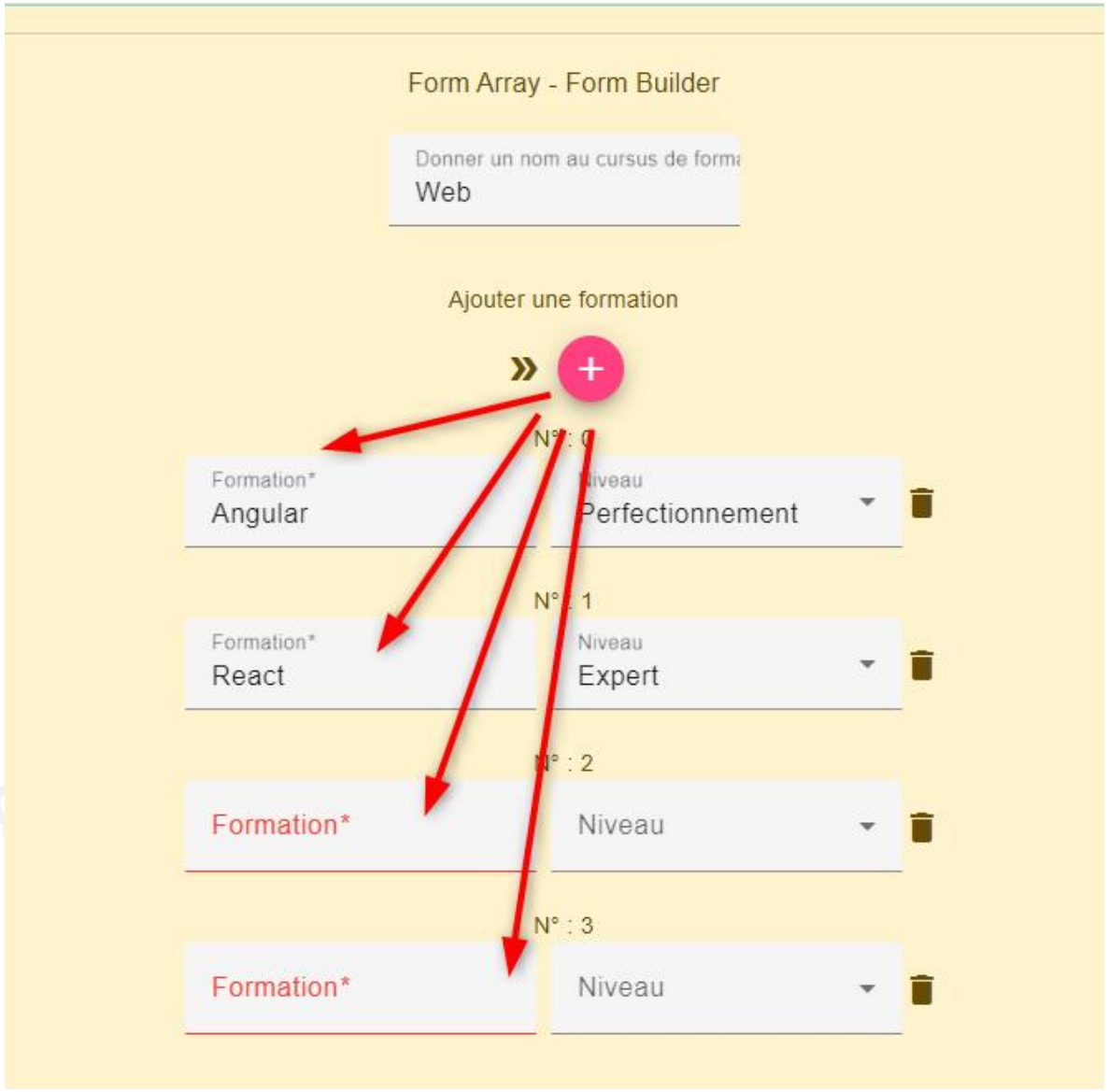
post.service.ts

TP06-formulaires > src > app > sharedModels > services > post.service.ts > PostService > p

```ts
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { FormGroup } from '@angular/forms';

@Injectable({
  providedIn: 'root'
})


export class PostService {

  constructor(private _http: HttpClient) { }

  // méthodes
  public postForm = (form: FormGroup) => {
    console.log('------- OBJET JS : ', form.value);

    // --- Post sur un serveur json
    // 1-url
    const url: string = 'http://localhost:3001/messages';
    // 2-body
    const body = JSON.stringify(form.value);
    console.warn('------ OBJET STRINGIFY : ', body);

    // 3-headers
    const myHeaders = new HttpHeaders({
      'Content-Type': 'application/json',
      'Access-Control-Allow-Origin': '*' // CORS
    });
    const options = { headers: myHeaders };
    //--------------------

    // envoi avec POST
    this._http.post(url, body, options).subscribe(
      // (res: any) => console.log(res);
    );
  }
}
```

La Class formArray

Récupérer les ressources en ligne :
https://dev.webjs.fr/1-Angular/ressources-formation/reactive-forms.zip



La Class formArray

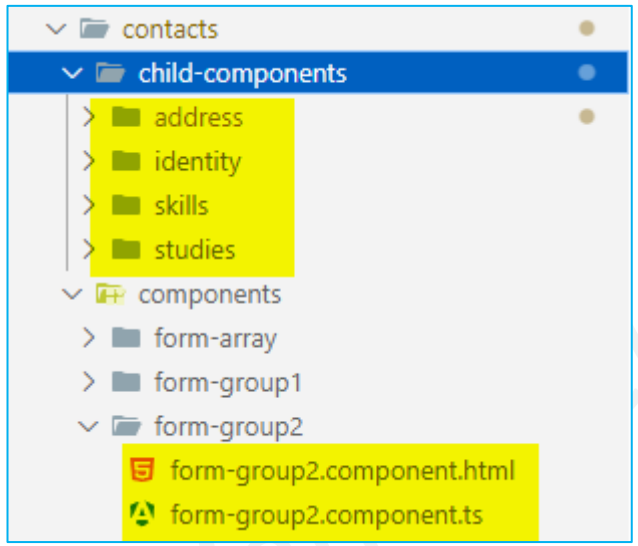
Récupérer les ressources en ligne :
https://dev.webjs.fr/1-Angular/ressources-formation/reactive-forms.zip

```ts
export class FormArrayComponent {

  public cursus: FormGroup;
// ----------------------------------------
  constructor(private _fb: FormBuilder) {

    this.cursus = this._fb.group({
      nomCursus: '',
      // définition du 2nd champ qui va être un tableau(ensemble) de formulaire
      formationsArray: this._fb.array([])
    });
  }
  // un getter qui va nous retourner le formationsArray
  get getFormations(): FormArray {
    return this.cursus.get('formationsArray') as FormArray;
  }

  // --------- Méthodes ------------------
  public addFormation = () => {
    this.getFormations.push(this.newFormation());
  }
  // méthode qui crée un nouveau formgroup
  // avec la formation et le niveau

  private newFormation = (): FormGroup<any> => {
    return this._fb.group({  // définition du formGroup
      formation: [null as string | null, Validators.required],
      niveau: ''
    });
  }
  // -------------------------
  public delFormation = (i: number) => {
    this.getFormations.removeAt(i);
  }
```

```html
<h3>Form Array - Form Builder</h3>

<form [formGroup]="cursus">
    <p>
        <mat-form-field>
            <mat-label>Donner un nom au cursus de formation :</mat-label>
            <input matInput type="text" formControlName="nomCursus">
        </mat-form-field>
    </p>
    <p>Ajouter une formation </p>
    <p>
        <mat-icon>double_arrow</mat-icon> 
        <button mat-mini-fab type="button" (click)="addFormation()">
            <mat-icon>add</mat-icon>
        </button>
    </p>

    <div formArrayName="formationsArray">
        <div *ngFor="let formation of getFormations.controls; let i=index">
            N° : {{i}}
            <div [formGroupName]="i">

                <mat-form-field>
                    <mat-label>Formation</mat-label>
                    <input matInput type=" text" formControlName="formation">
                </mat-form-field> 

                <mat-form-field>
                    <mat-label>Niveau</mat-label>
                    <mat-select formControlName="niveau">
                        <mat-option value="niveau1">Niveau 1</mat-option>
                        <mat-option value="niveau2">Perfectionnement</mat-option>
                        <mat-option value="niveau3">Expert</mat-option>
                    </mat-select>
                </mat-form-field>

                <mat-icon (click)="delFormation(i)">delete</mat-icon>
            </div>
        </div>
    </div>
</form>
```

TP

Améliorer le form-group réalisé en 1ère partie de formation. Le but étant d'avoir un formulaire principal qui appelle des child-components.



Le composant principal passera en Input [parentFormGroup] le nom du FormGroup

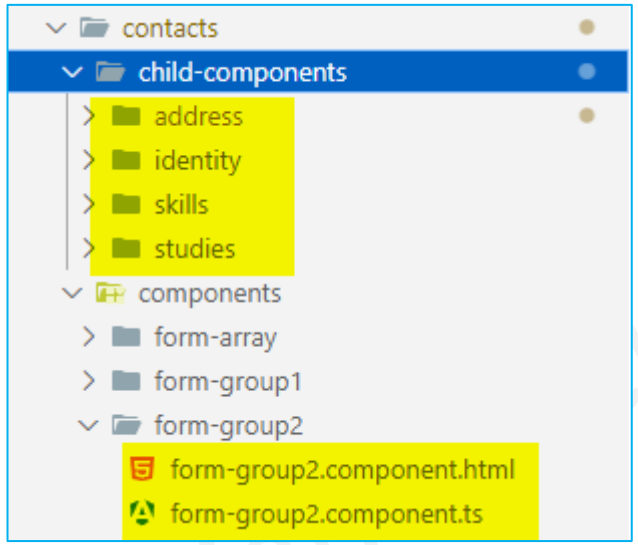
form-group2.component.html

src > app > webApp > contacts > components > form-group2 > form-group2.component.html

```
1   <div class="container">
7     <hr>
8     <form [formGroup]="monForm" (submit)="onSubmit()">
9
10      <div class="alert alert-light">
11        <app-identity [parentFormGroup]="monForm" #identity />
12      </div>
13
14      <!-- ----- -->
15
16      <div [hidden]="!monForm.controls['identity'].valid">
17        <div class="alert alert-light">
18          <app-address [parentFormGroup]="monForm" #address />
19        </div>
20      </div>
21
22      <!-- ----- -->
23
24      <div [hidden]="!monForm.controls['address'].valid">
25      <div class="alert alert-light">
26        <app-studies [parentFormGroup]="monForm" #studies />
27      </div>
28      </div>
29
30      <!-- ----- -->
31
32      <div [hidden]="!monForm.controls['studies'].valid">
33      <div class="alert alert-light">
34        <app-skills [parentFormGroup]="monForm" #skills />
35      </div>
36      </div>
37
38      <!-- ----- -->
```

TP

Améliorer le form-group réalisé en 1ère partie de formation. Le but étant d'avoir un formulaire principal qui appelle des child-components.

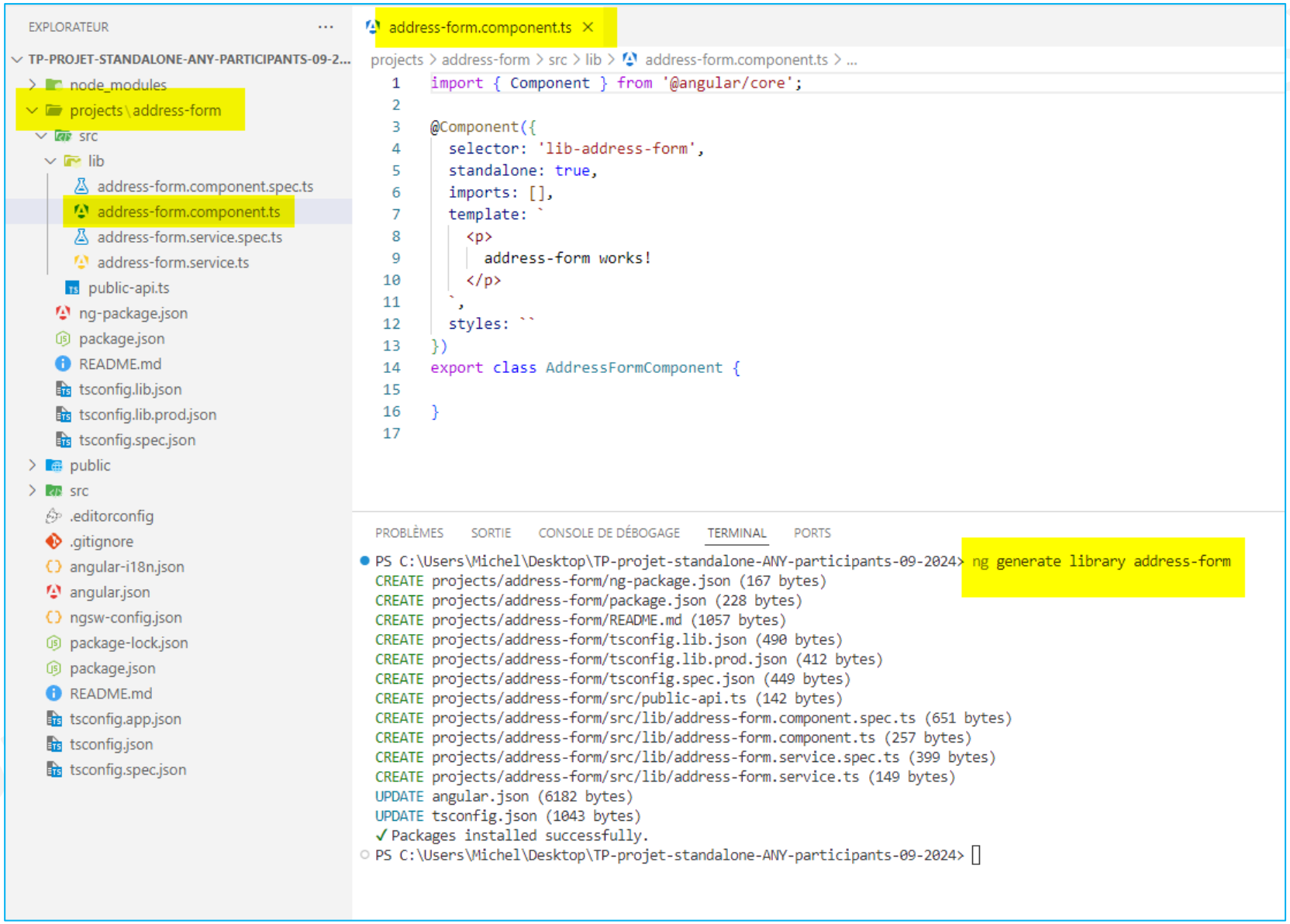



```ts
// 1-props
public monForm: FormGroup;
@ViewChild('identity') eltIdentity!:any;
@ViewChild('adress') eltAddress!:any;

// 2-const
constructor(private _fb:FormBuilder, _service: PostService) {
  this.monForm = this._fb.group({
})
}
public onSubmit = () => {
  console.log(this.monForm.value);
  console.log(this.monForm.controls['identity'].value);
  console.log(this.monForm.controls['address'].value);
  // this._service.postForm(this.monForm);
}
}
```

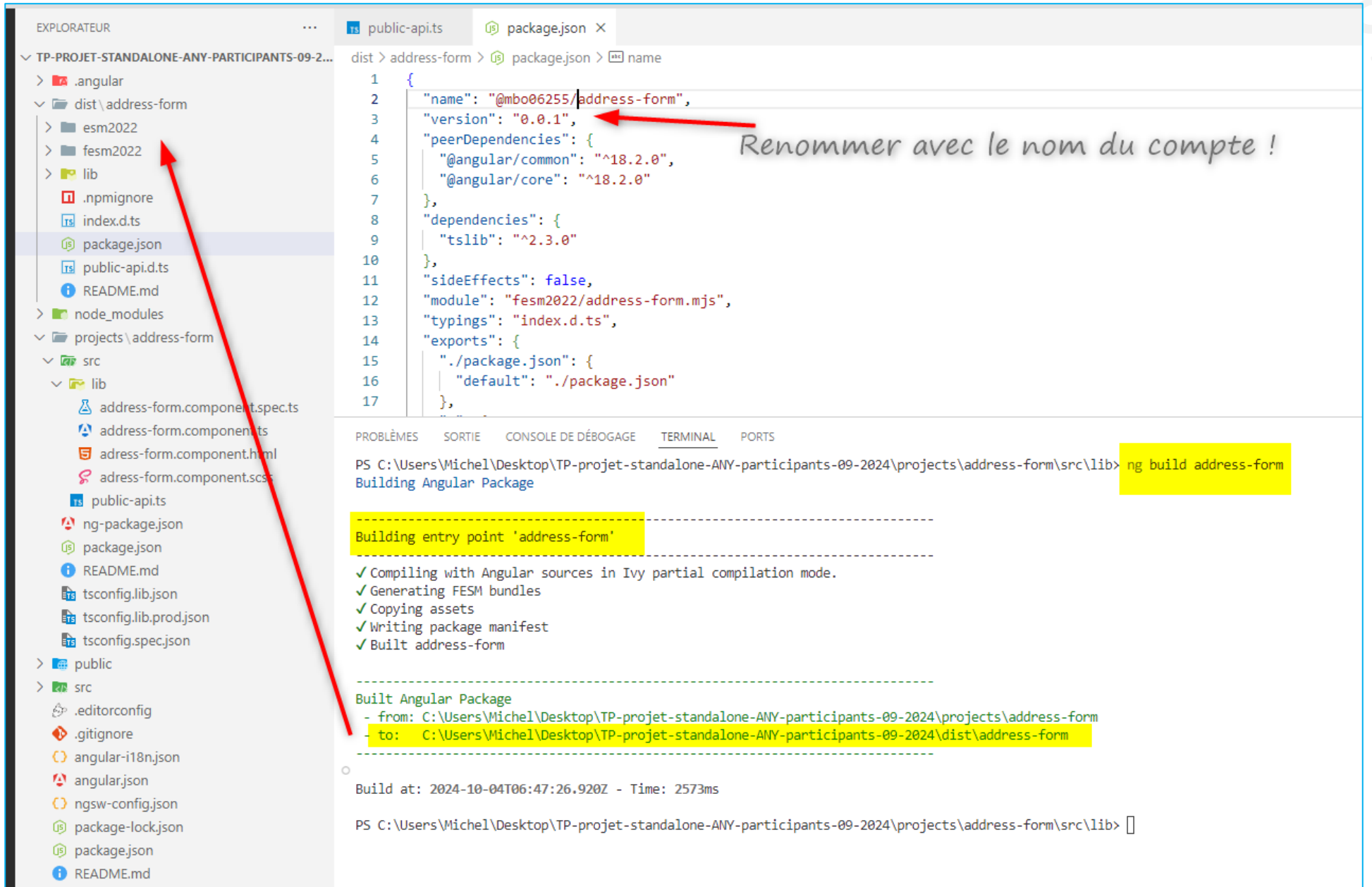
Le composant principal passera en Input [parentFormGroup] le nom du FormGroup

TP2

Compiler le form-group « address » afin de le réexploiter facilement dans ce projet ou d'autres projets.
Phase 1 : Compilation et publication sur NODE.JS

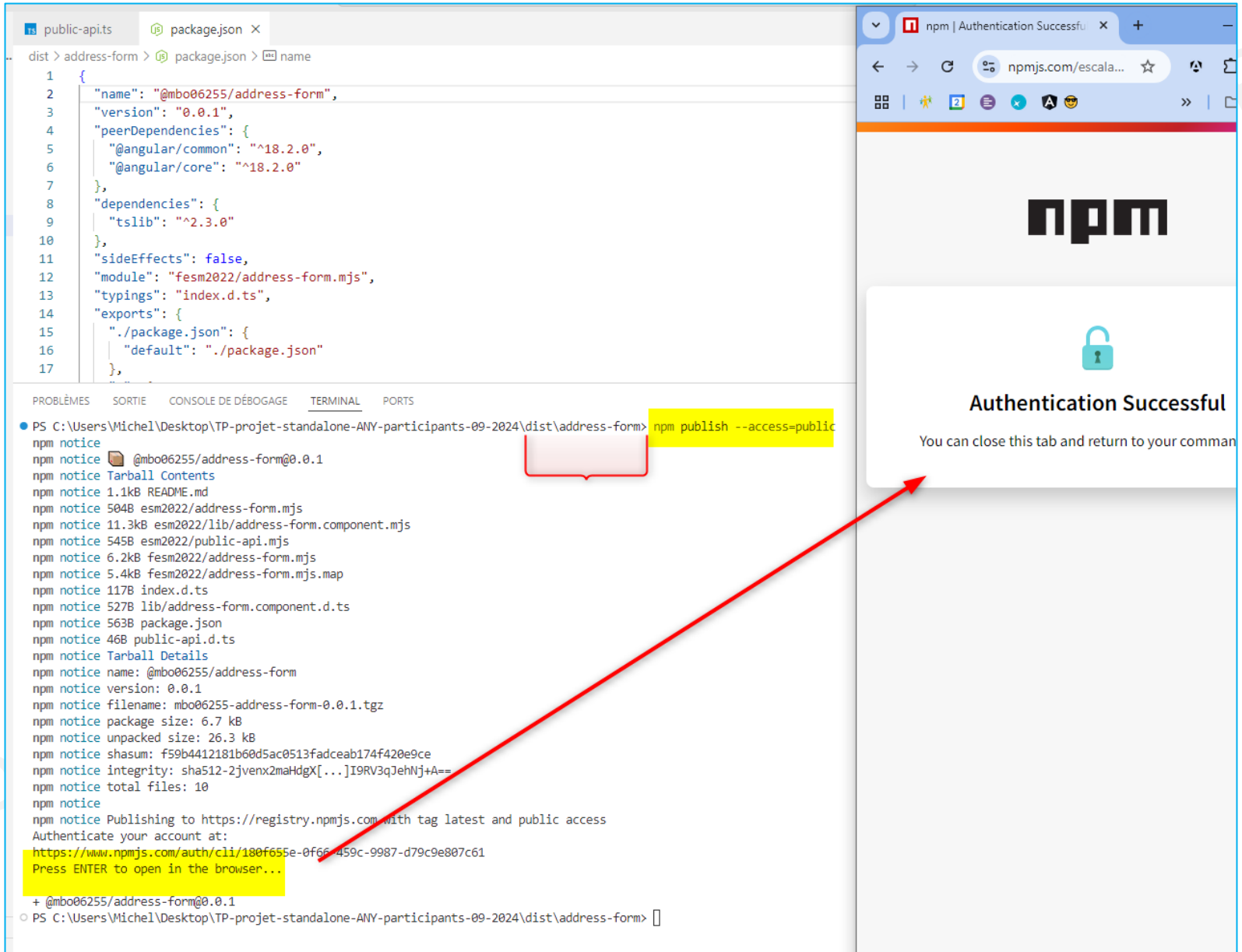


TP2 Compiler le form-group « address » afin de le réexploiter facilement dans ce projet ou d'autres projets.

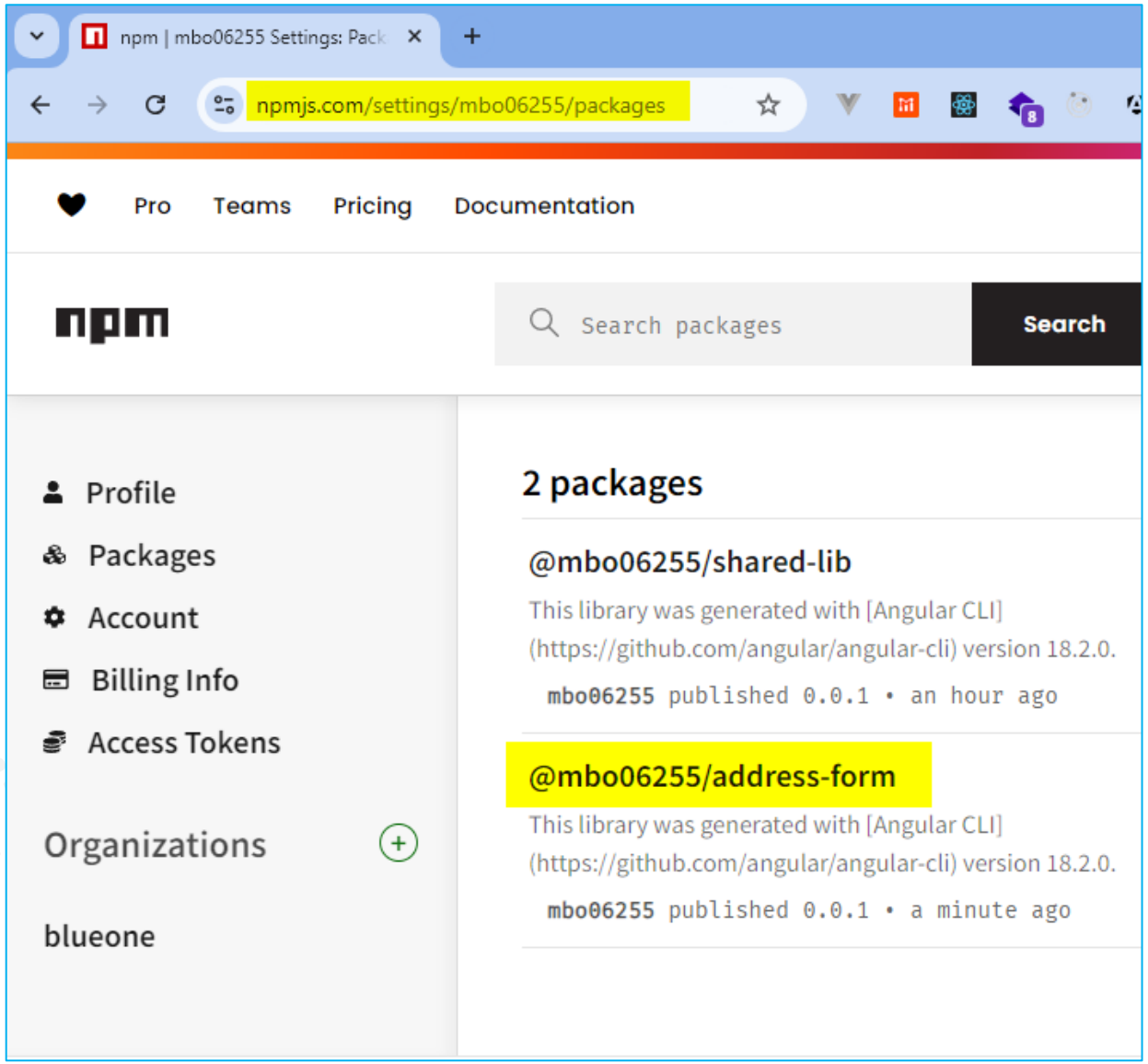


TP2

Compiler le form-group « address » afin de le réexploiter facilement dans ce projet ou d'autres projets.



TP2 Compiler le form-group « address » afin de le réexploiter facilement dans ce projet ou d'autres projets.



TP2

Compiler le form-group « address » afin de le réexploiter facilement dans ce projet ou d'autres projets.
Phase 2 : Déploiement depuis NODE.JS

package.json

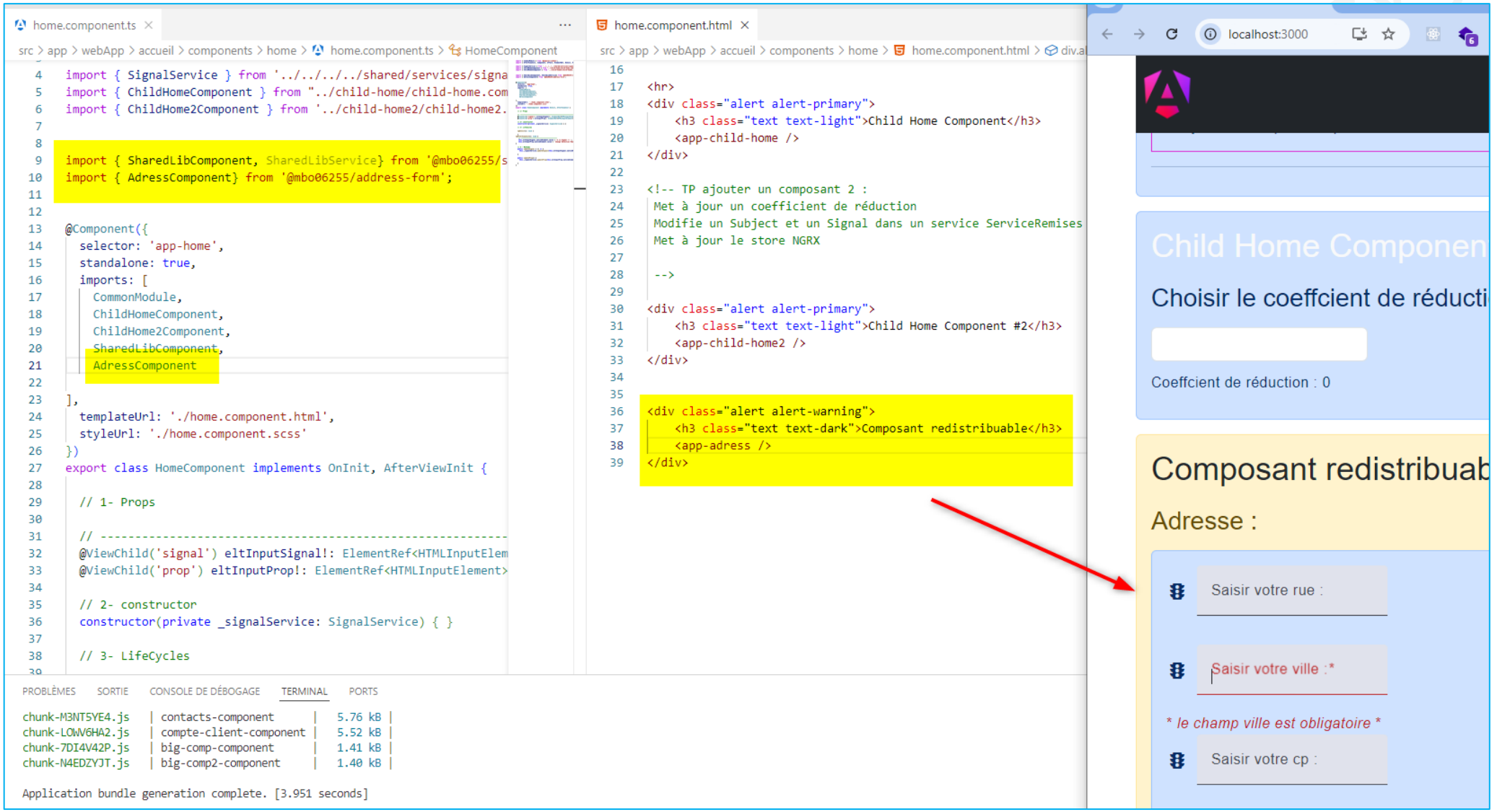
```
package.json > ...
 4    "scripts": {
12    },
13    "private": true,
14    "dependencies": {
15      "@angular/animations": "^18.0.0",
16      "@angular/cdk": "^18.0.0",
17      "@angular/common": "^18.0.0",
18      "@angular/compiler": "^18.0.0",
19      "@angular/core": "^18.0.0",
20      "@angular/forms": "^18.0.0",
21      "@angular/material": "^18.2.6",
22      "@angular/platform-browser": "^18.0.0",
23      "@angular/platform-browser-dynamic": "^18.0.0",
24      "@angular/router": "^18.0.0",
25      "@angular/service-worker": "^18.0.0",
26      "@mbo06255/address-form": "^0.0.1",
27      "@ngrx/effects": "^18.0.2",
28      "@ngrx/store": "^18.0.2",
29      "@ngrx/store-devtools": "^18.0.2",
30      "bootstrap": "^5.3.3",
31      "bootstrap-icons": "^1.11.3",
32      "rxjs": "~7.8.0",
33      "tslib": "^2.3.0",
34      "zone.js": "~0.14.3"
```

PROBLÈMES SORTIE CONSOLE DE DÉBOGAGE TERMINAL PORTS

```
PS C:\Users\Michel\Desktop\TP-projet-standalone-ANY-participants-09-2024\src> npm search  @mbo06255/address
@mbo06255/address-form
This library was generated with [Angular CLI](https://github.com/angular/angular-cli) version 18.2.0.
Version 0.0.1 published 2024-10-04 by mbo06255
Maintainers: mbo06255
https://npm.im/@mbo06255/address-form

PS C:\Users\Michel\Desktop\TP-projet-standalone-ANY-participants-09-2024\src> npm i @mbo06255/address-form

added 1 package, and audited 1064 packages in 3s

171 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\Users\Michel\Desktop\TP-projet-standalone-ANY-participants-09-2024\src>
```

TP2

Compiler le form-group « address » afin de le réexploiter facilement dans ce projet ou d'autres projets.
Phase 2 : Déploiement depuis NODE.JS



TP2

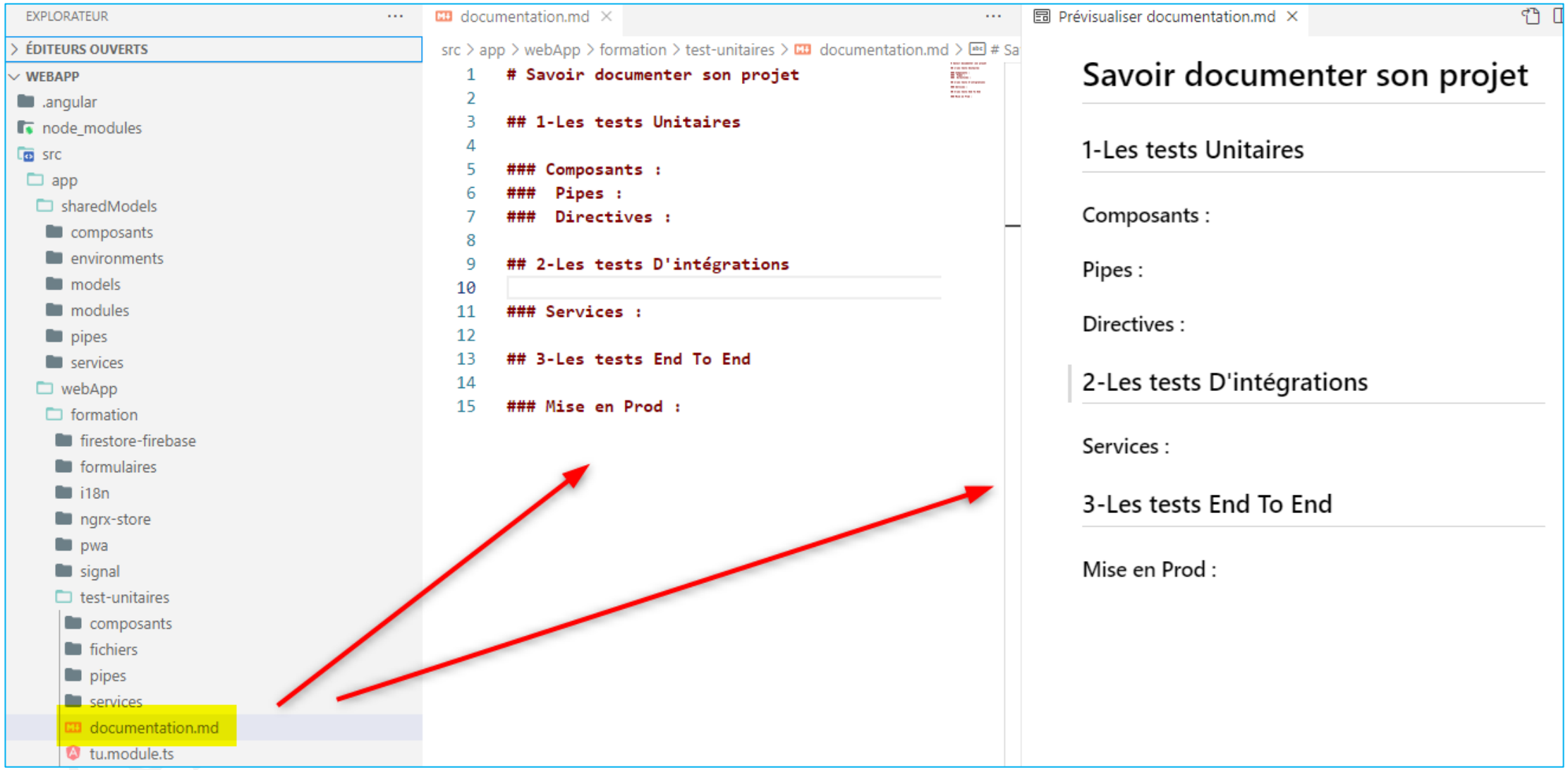
Compiler le form-group « address » afin de le réexploiter facilement dans ce projet ou d'autres projets.
Phase 2 : Déploiement depuis NODE.JS

Les Tests

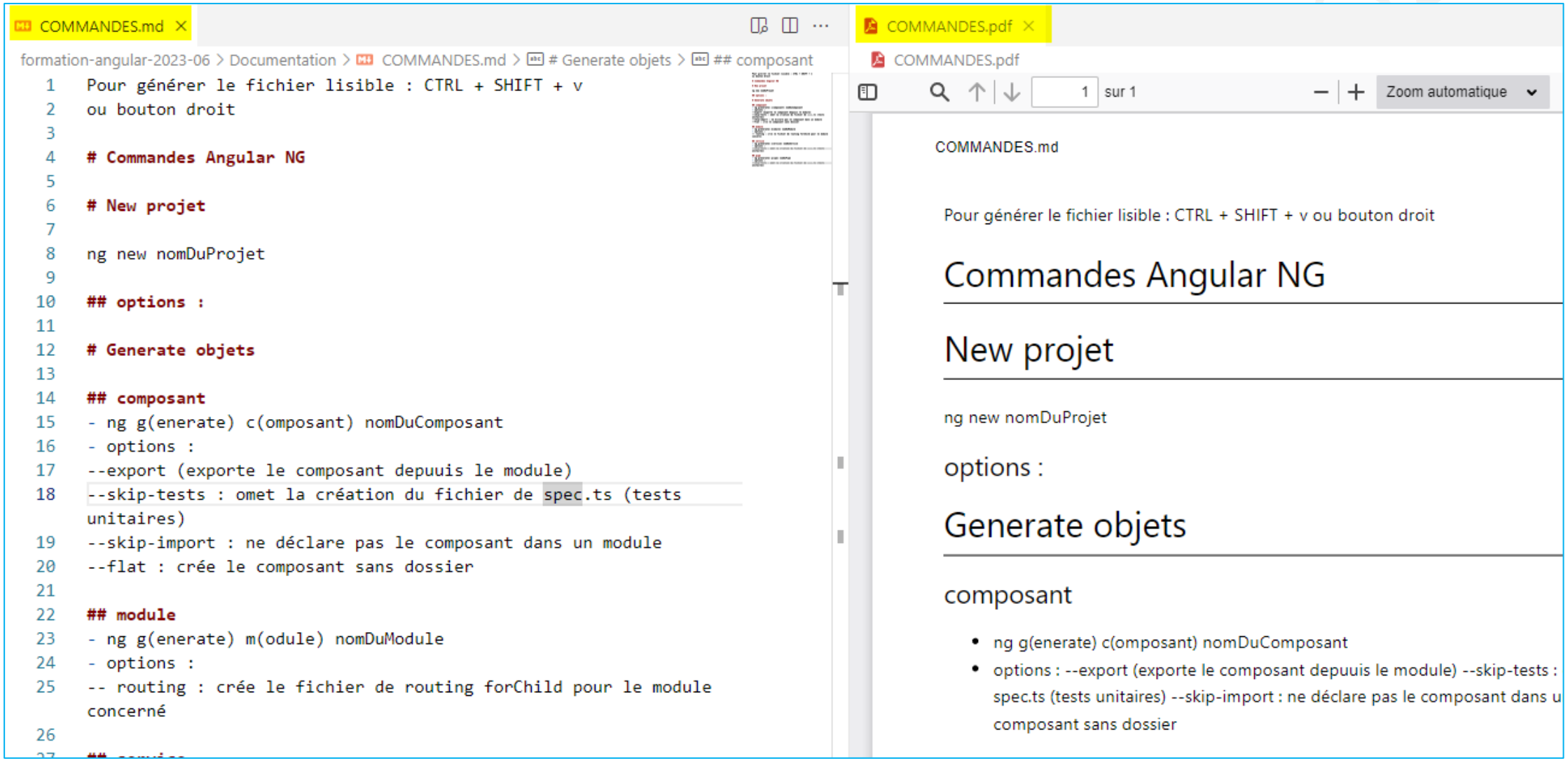
Le Test

Le test (ou le Test Driven Development) permet de :

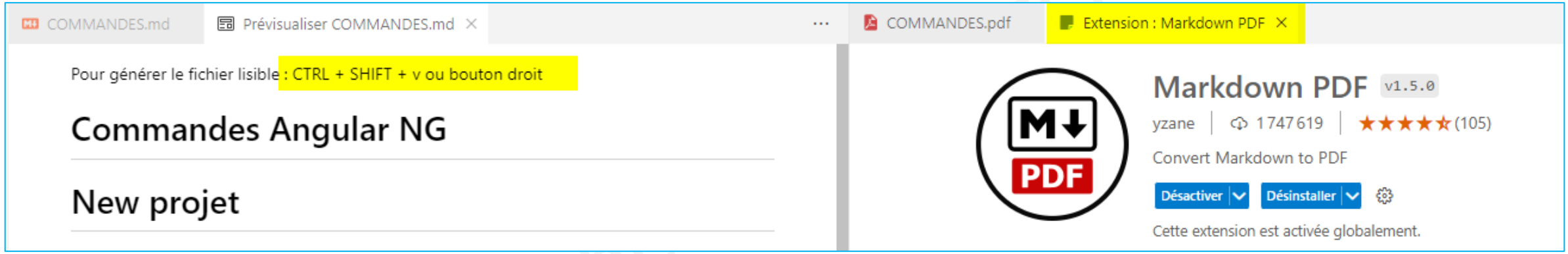
- Documenter le projet au fur et à mesure de sa création
- Faciliter les migrations entre versions majeures
- Simplifier la recherche d'erreurs
- Isoler l'objet de son contexte et de tester son intégration (Mock)


EXPLORATEUR
ÉDITEURS OUVERTS
WEBAPP
.angular
node_modules
src
app
sharedModels
composants
environments
models
modules
pipes
services
webApp
formation
firestore-firebase
formulaires
i18n
ngrx-store
pwa
signal
test-unitaires
composants
fichiers
pipes
services
documentation.md
tu.module.ts
documentation.md
src > app > webApp > formation > test-unitaires > documentation.md > # Sa
1 # Savoir documenter son projet
2
3 ## 1-Les tests Unitaires
4
5 ### Composants :
6 ### Pipes :
7 ### Directives :
8
9 ## 2-Les tests D'intégrations
10
11 ### Services :
12
13 ## 3-Les tests End To End
14
15 ### Mise en Prod :
Prévisualiser documentation.md
Savoir documenter son projet
1-Les tests Unitaires
Composants :
Pipes :
Directives :
2-Les tests D'intégrations
Services :
3-Les tests End To End
Mise en Prod :

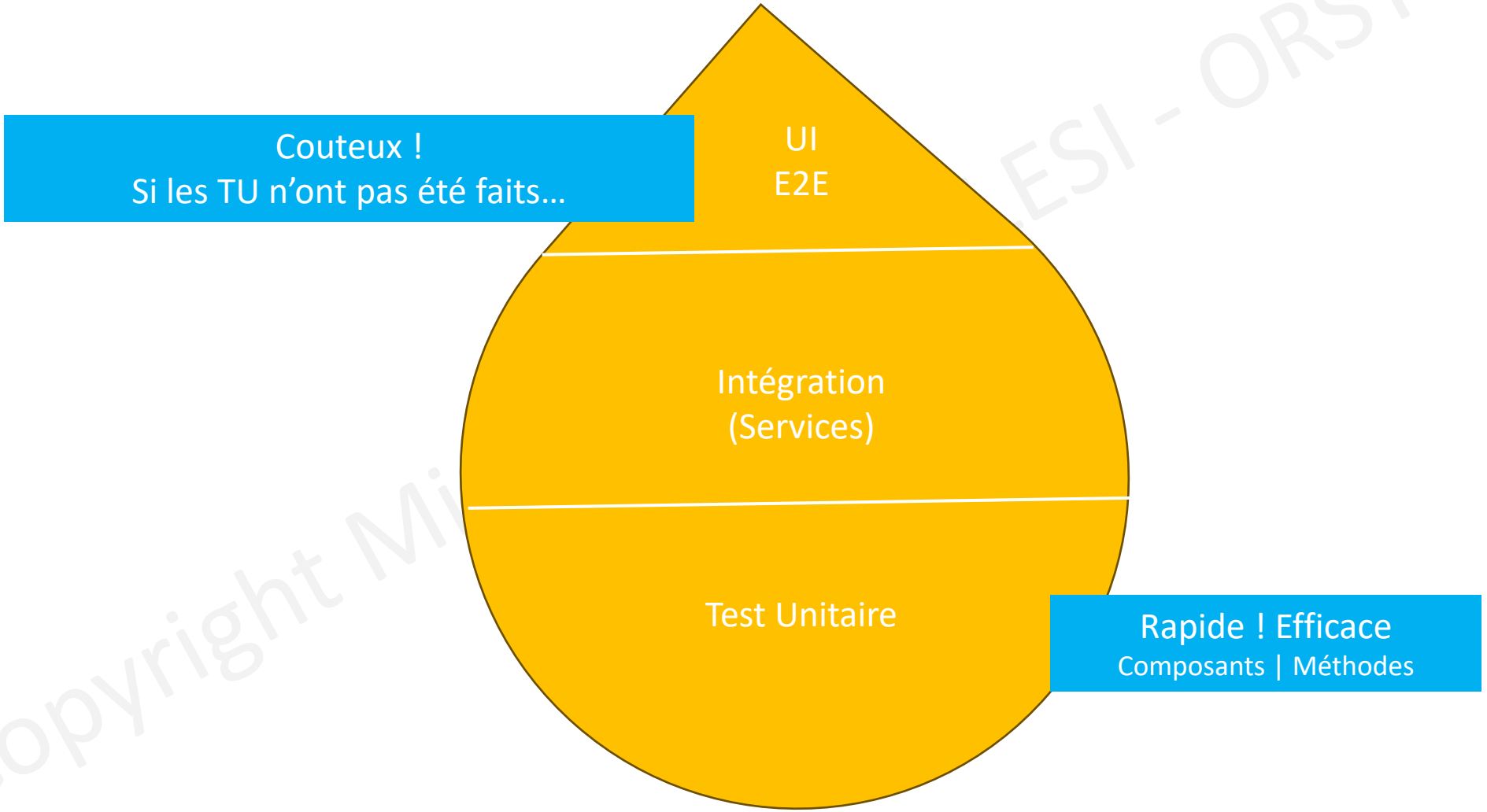
Documenter son projet



Documenter son projet



Le Test et la Documentation



1Er exemple : Apprendre de l'existant

app.component.ts

TP07-tests-unitaires > src > app > app.component.ts > AppComponent > titl

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'TU';
}

```

app.component.html

TP07-tests-unitaires > src > app > app.component.html > h1

```html
<h1>{{title}}</h1>
```

app.component.spec.ts

TP07-tests-unitaires > src > app > app.component.spec.ts > describe('AppComponent') callback

```ts
import { TestBed } from '@angular/core/testing';
import { AppComponent } from './app.component';

// -----------------------------------------------------
// describe permet de créer un Objet de Test Unitaire
// -----------------------------------------------------
describe('AppComponent', () => {
  beforeEach(() => TestBed.configureTestingModule({
    declarations: [AppComponent],
  }));

    // it est une fonction qui crée une spécification
    it('should create the app', () => {
      const fixture = TestBed.createComponent(AppComponent);
      const app = fixture.componentInstance;
      // une spéc (it) attend un retour (expectation)
      expect(app).toBeTruthy(); // matchers Jasmine
    });


    it(`should have as title 'TP07-tests-unitaires'`, () => {
      const fixture = TestBed.createComponent(AppComponent);
      const composant = fixture.componentInstance;
      expect(composant.title).toEqual('TU');
    });


    it('should render title', () => {
      const fixture = TestBed.createComponent(AppComponent);
      fixture.detectChanges();
      const compiled = fixture.nativeElement as HTMLElement;
      expect(compiled.querySelector('h1')?.textContent).toContain('TU');
    });
});

```

2nd exemple : Injection de services

warn.service.ts

TP07-tests-unitaires > src > app > services > warn.service.ts > WarnService

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

@Injectable({
  providedIn: 'root'
})
export class WarnService {
  // ------------------------
  // méthodes
  // ------------------------
  public warnMessage = (msg: string) => {
    console.warn(`Le message est : ${msg}`);
  }
}
```

operations.service.ts

sts-unitaires > src > app > services > operations.service.ts > OperationsService > ajouter

```ts
import { Injectable } from '@angular/core';
import { WarnService } from './warn.service';

@Injectable({
  providedIn: 'root'
})
export class OperationsService {

  constructor(
    private _warn: WarnService
  ) { }

  // ------------------------
  // méthodes
  // ------------------------
  public ajouter = (nb1: number, nb2: number) => {
    this._warn.warnMessage('Ajout OK');
    // this._warn.warnMessage('Ajout OK');
    return nb1 + nb2;
  }

  public soustraire = (nb1: number, nb2: number) =>
    this._warn.warnMessage('Soustraction OK');
    return nb1 - nb2;
  }
}
```

operations.service.spec.ts

TP07-tests-unitaires > src > app > services > operations.service.spec.ts > describe('Test Injection de dépendances') ca

```ts
import { OperationsService } from "./operations.service";
import { WarnService } from "./warn.service";


// 1- Création de l'objet de Test
describe('Test Injection de dépendances', () => {

  // 2- liste des spécifications

  // it (xit - fit)
  it('Ajouter 2 nombres OK ?', () => {

    const operations = new OperationsService(new WarnService);
    const resultat = operations.ajouter(10, 5);
    // 3- expectation => ce que l'on attend
    expect(resultat).withContext('--- Erreur Expectation ---').toBe(15);

  });

  // 4- duplication du premier IT

  // it (xit - fit)
  it('Soustraire 2 nombres OK ?', () => {

    const operations = new OperationsService(new WarnService);
    const resultat = operations.soustraire(10, 5);
    // 3- expectation => ce que l'on attend
    expect(resultat).withContext('--- Erreur Expectation ---').toBe(5);

  });


})
```

2nd exemple : Injection de services

warn.service.ts

TP07-tests-unitaires > src > app > services > warn.service.ts > WarnService > logErreur

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

@Injectable({
  providedIn: 'root'
})
export class WarnService {
  // ------------------------
  // méthodes
  // ------------------------
  public warnMessage = (msg: string) => {
    console.warn(`Le message est : ${msg}`);
  }
  // ---
  public logErreur = (msg: string) => {
    console.error(`L'erreur' est : ${msg}`);
  }
}
```

operations.service.ts

sts-unitaires > src > app > services > operations.service.ts > OperationsService > ajouter

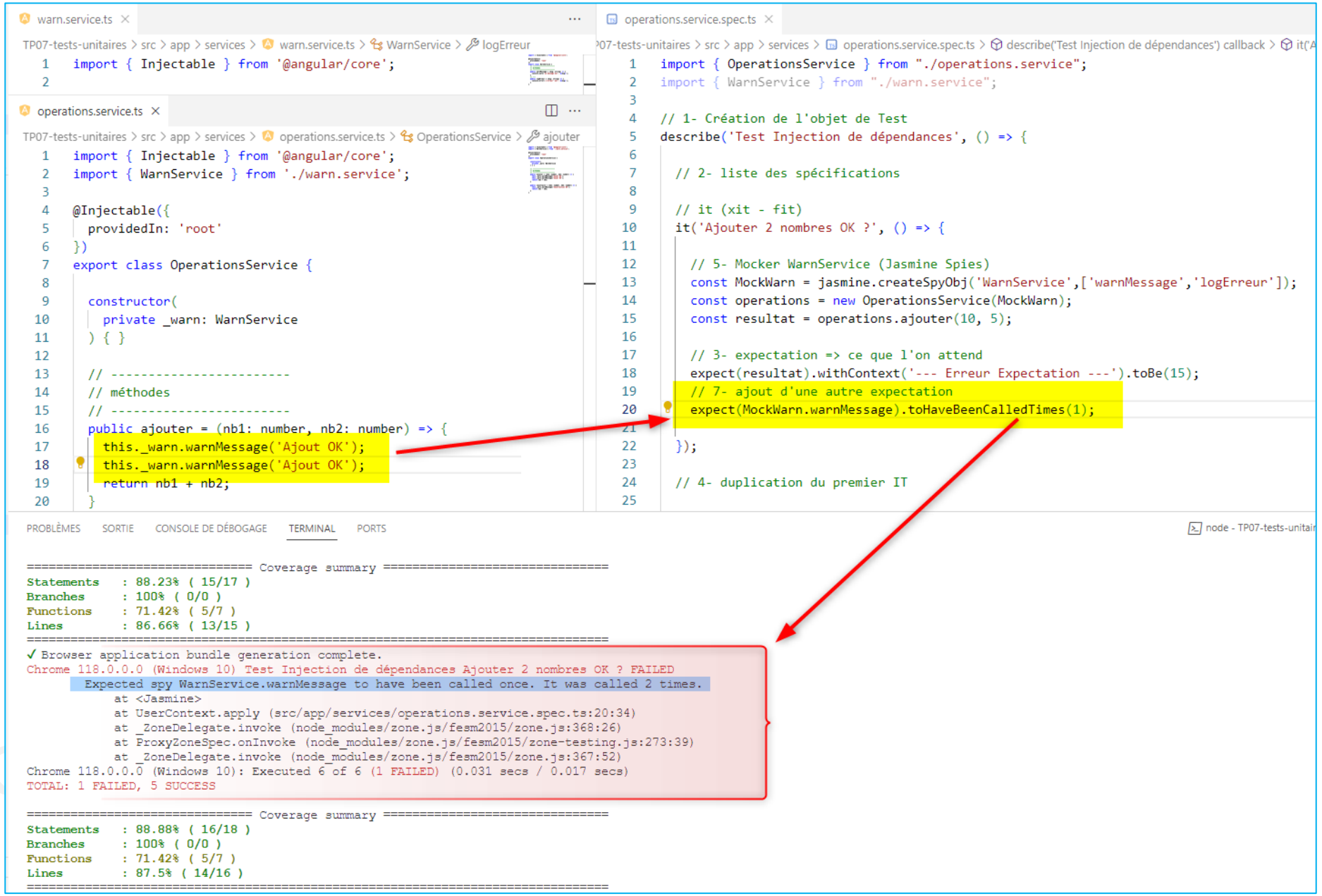
```ts
import { Injectable } from '@angular/core';
import { WarnService } from './warn.service';

@Injectable({
  providedIn: 'root'
})
export class OperationsService {

  constructor(
    private _warn: WarnService
  ) { }

  // ------------------------
  // méthodes
  // ------------------------
  public ajouter = (nb1: number, nb2: number) => {
    this._warn.warnMessage('Ajout OK');
    // this._warn.warnMessage('Ajout OK');
    return nb1 + nb2;
  }

  public soustraire = (nb1: number, nb2: number) =>
```

operations.service.spec.ts

TP07-tests-unitaires > src > app > services > operations.service.spec.ts > ...

```ts
import { OperationsService } from "./operations.service";
import { WarnService } from "./warn.service";

// 1- Création de l'objet de Test
describe('Test Injection de dépendances', () => {

  // 2- liste des spécifications

  // it (xit - fit)
  it('Ajouter 2 nombres OK ?', () => {

    // 5- Mocker WarnService (Jasmine Spies)
    const MockWarn = jasmine.createSpyObj('WarnService',['warnMessage','logErreur']);
    const operations = new OperationsService(MockWarn);
    const resultat = operations.ajouter(10, 5);

    // 3- expectation => ce que l'on attend
    expect(resultat).withContext('--- Erreur Expectation ---').toBe(15);

  });

  // 4- duplication du premier IT

  // it (xit - fit)
  it('Soustraire 2 nombres OK ?', () => {

    // 6- reproduire auusi ici le MockWarn
    const MockWarn = jasmine.createSpyObj('WarnService',['warnMessage','logErreur']);
    const operations = new OperationsService(MockWarn);
    const resultat = operations.soustraire(10, 5);
    // 3- expectation => ce que l'on attend
    expect(resultat).withContext('--- Erreur Expectation ---').toBe(5);

  });


})
```

2nd exemple : Injection de services



warn.service.ts

TP07-tests-unitaires > src > app > services > warn.service.ts > WarnService > logErreur

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

operations.service.ts

TP07-tests-unitaires > src > app > services > operations.service.ts > OperationsService > ajouter

```ts
import { Injectable } from '@angular/core';
import { WarnService } from './warn.service';

@Injectable({
  providedIn: 'root'
})
export class OperationsService {

  constructor(
    private _warn: WarnService
  ) { }

  // ------------------------
  // méthodes
  // ------------------------
  public ajouter = (nb1: number, nb2: number) => {
    this._warn.warnMessage('Ajout OK');
    this._warn.warnMessage('Ajout OK');
    return nb1 + nb2;
  }
```

operations.service.spec.ts

TP07-tests-unitaires > src > app > services > operations.service.spec.ts > describe('Test Injection de dépendances') callback > it('A

```ts
import { OperationsService } from "./operations.service";
import { WarnService } from "./warn.service";

// 1- Création de l'objet de Test
describe('Test Injection de dépendances', () => {

  // 2- liste des spécifications

  // it (xit - fit)
  it('Ajouter 2 nombres OK ?', () => {

    // 5- Mocker WarnService (Jasmine Spies)
    const MockWarn = jasmine.createSpyObj('WarnService',['warnMessage','logErreur']);
    const operations = new OperationsService(MockWarn);
    const resultat = operations.ajouter(10, 5);

    // 3- expectation => ce que l'on attend
    expect(resultat).withContext('--- Erreur Expectation ---').toBe(15);
    // 7- ajout d'une autre expectation
    expect(MockWarn.warnMessage).toHaveBeenCalledTimes(1);

  });

  // 4- duplication du premier IT
```

PROBLÈMES SORTIE CONSOLE DE DÉBOGAGE TERMINAL PORTS

node - TP07-tests-unitair

```
============================== Coverage summary ==============================
Statements   : 88.23% ( 15/17 )
Branches     : 100% ( 0/0 )
Functions    : 71.42% ( 5/7 )
Lines        : 86.66% ( 13/15 )
==============================================================================
✓ Browser application bundle generation complete.
Chrome 118.0.0.0 (Windows 10) Test Injection de dépendances Ajouter 2 nombres OK ? FAILED
        Expected spy WarnService.warnMessage to have been called once. It was called 2 times.
            at <Jasmine>
            at UserContext.apply (src/app/services/operations.service.spec.ts:20:34)
            at _ZoneDelegate.invoke (node_modules/zone.js/fesm2015/zone.js:368:26)
            at ProxyZoneSpec.onInvoke (node_modules/zone.js/fesm2015/zone-testing.js:273:39)
            at _ZoneDelegate.invoke (node_modules/zone.js/fesm2015/zone.js:367:52)
Chrome 118.0.0.0 (Windows 10): Executed 6 of 6 (1 FAILED) (0.031 secs / 0.017 secs)
TOTAL: 1 FAILED, 5 SUCCESS

============================== Coverage summary ==============================
Statements   : 88.88% ( 16/18 )
Branches     : 100% ( 0/0 )
Functions    : 71.42% ( 5/7 )
Lines        : 87.5% ( 14/16 )
==============================================================================
```

2nd exemple : Injection de services

warn.service.ts

TP07-tests-unitaires > src > app > services > warn.service.ts > WarnService > logErreur

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

@Injectable({
  providedIn: 'root'
})
export class WarnService {
  // ------------------------
  // méthodes
  // ------------------------
  public warnMessage = (msg: string) => {
    console.warn(`Le message est : ${msg}`);
  }
  // ---
  public logErreur = (msg: string) => {
    console.error(`L'erreur' est : ${msg}`);
  }
}
```

operations.service.ts

TP07-tests-unitaires > src > app > services > operations.service.ts > OperationsService > ajouter

```ts
import { Injectable } from '@angular/core';
import { WarnService } from './warn.service';

@Injectable({
  providedIn: 'root'
})
export class OperationsService {

  constructor(
    private _warn: WarnService
  ) { }

  // ------------------------
  // méthodes
  // ------------------------
  public ajouter = (nb1: number, nb2: number) => {
    this._warn.warnMessage('Ajout OK');
    // this._warn.warnMessage('Ajout OK');
    return nb1 + nb2;
  }

  public soustraire = (nb1: number, nb2: number) => {
    this._warn.warnMessage('Soustraction OK');
    return nb1 - nb2;
  }
}
```

operations.service.spec.ts

TP07-tests-unitaires > src > app > services > operations.service.spec.ts > describe('Test Injection de dépendances') callback

```ts
import { OperationsService } from "./operations.service";
import { WarnService } from "./warn.service";

// Création de l'objet de Test
describe('Test Injection de dépendances', () => {

  // ----------------------------------------------------
  // Zone de déclarations de variables - const - objets
  // ----------------------------------------------------
  let MockWarn:any;
  let operations:any;

  // ------------------------------------------------------------------------
  // traitement spécifiques communs (beforeEach - AfterEach - beforeAll)
  // ------------------------------------------------------------------------
  beforeEach(()=> {
    MockWarn = jasmine.createSpyObj('WarnService', ['warnMessage', 'logErreur']);
    operations = new OperationsService(MockWarn);
  });

  // ----------------------------
  //   liste des spécifications
  // ----------------------------

  it('Ajouter 2 nombres OK ?', () => {
    const resultat = operations.ajouter(10, 5);
    expect(resultat).withContext('--- Erreur Expectation ---').toBe(15);
    expect(MockWarn.warnMessage).toHaveBeenCalledTimes(1);
  });

  it('Soustraire 2 nombres OK ?', () => {
    const resultat = operations.soustraire(10, 5);
    expect(resultat).withContext('--- Erreur Expectation ---').toBe(5);
  });
})
```

2nd exemple : Injection de services

warn.service.ts

TP07-tests-unitaires > src > app > services > warn.service.ts > WarnService > warnMess

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

@Injectable({
  providedIn: 'root'
})
export class WarnService {
  // ------------------------
  // méthodes
  // ------------------------
  public warnMessage = (msg: string) => {
    console.warn(`Le message est : ${msg}`);
  }
  // ---
  public logErreur = (msg: string) => {
    console.error(`L'erreur' est : ${msg}`);
  }
}
```

operations.service.ts

TP07-tests-unitaires > src > app > services > operations.service.ts > OperationsService >

```ts
  // ------------------------
  // méthodes
  // ------------------------
  public ajouter = (nb1: number, nb2: number) => {
    this._warn.warnMessage('Ajout OK');
    // this._warn.warnMessage('Ajout OK');
    return nb1 + nb2;
  }

  public soustraire = (nb1: number, nb2: number) => {
    this._warn.warnMessage('Soustraction OK');
    return nb1 - nb2;
  }
}
```

operations.service.spec.ts

TP07-tests-unitaires > src > app > services > operations.service.spec.ts > describe('Test Injection de dépendances') callb

```ts
import { TestBed } from "@angular/core/testing";
import { OperationsService } from "./operations.service";
import { WarnService } from "./warn.service";

// Création de l'objet de Test
describe('Test Injection de dépendances', () => {
  // ------------------------------------------------
  // Zone de déclarations de variables - const - objets
  // ------------------------------------------------
  let MockWarn: any;
  let operations: any;
  // ------------------------------------------------------------------------
  // traitement spécifiques communs (beforeEach - AfterEach - beforeAll)
  // ------------------------------------------------------------------------
  beforeEach(() => {
    MockWarn = jasmine.createSpyObj('WarnService', ['warnMessage','logErreur']);
    // ------------------------------------------------------------------------
    // Création d'un Objet complet de test et de simulation : TESTBED
    // Créons une vraie injection de dépendances avec les providers
    // ------------------------------------------------------------------------
    TestBed.configureTestingModule({
      providers: [
        // injection de dépendances
        OperationsService,
        {
          provide: WarnService,
          // on n'utilise pas la méthode de WarnService
          // mais la méthode du Mock ('warnMessage', 'logErreur')
          useValue: MockWarn
        }
      ]
    });
    // ------------------------------------------
    operations = TestBed.inject(OperationsService);
    // ------------------------------------------
  });
  // ----------------------------
  //   liste des spécifications
  // ----------------------------
  it('Ajouter 2 nombres OK ?', () => {
    const resultat = operations.ajouter(10, 5);
    expect(resultat).withContext('--- Erreur Expectation ---').toBe(15);
    expect(MockWarn.warnMessage).toHaveBeenCalledTimes(1);
  });
```

TP : Rédiger un TU

Rédiger le test unitaire du pipe see-more.pipe.ts
Le pipe seeMorePipe permet grâce à la méthode substring de javascript de n'afficher qu'une partie du texte.

see-more.pipe.ts

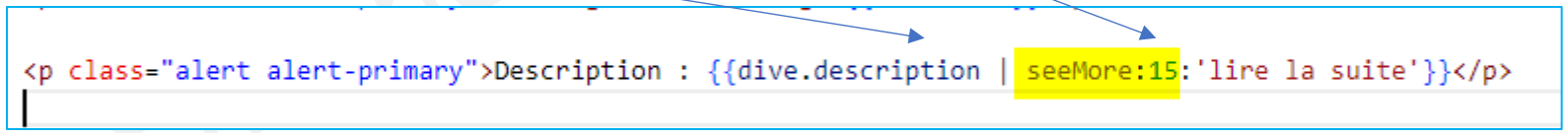
src > app > shared > pipes > see-more.pipe.ts > ...

```ts
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
    name: 'seeMore',
    standalone:true
})
export class SeeMorePipe implements PipeTransform {

    transform(datas: string, nbCar: number, texte: string) {

        return `${datas.substring(0, nbCar)} ${texte}`;
    }
}
```

TP : Rédiger un TU

Rédiger le test unitaire du pipe see-more.pipe.ts
Le pipe seeMorePipe permet grâce à la méthode substring de javascript de n'afficher qu'une partie du texte.
En le codant, les dév n'ont pas forcément penser à plusieurs cas :

- Le nombre de caractères est absent ?
- La position initiale n'est pas fournie ?
- Le texte n'est pas fourni par le service ?

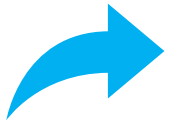


Rédiger le test unitaire permettant de gérer ces cas de figure et par conséquence modifier le fichier see-more.pipe.ts

L’InternationalisatioN I18N

L’InternationalisatioN I18N

Comment traduire dans un projet Angular ?

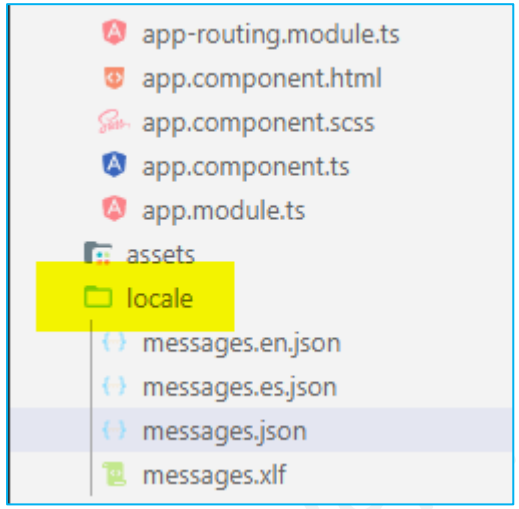


Commande d’installation et d’intégration de package : ng add @angular/localize

L'InternationalisatioN I18N

L'InternationalisatioN I18N

Commande : Commande : ng extract-i18n --output-path src/locale --format="json"



messages.en.json | messages.json

formation-any-2023-09 > TP02-i18n > src > locale > messages.json > ...

```
{
  "locale": "fr",
  "translations": {
    "div-vue": "{$START_PARAGRAPH}L'internationalisation I18N{$CLOSE_PARAGRAPH}{$S
    "4044202032648885552": "Bonjour Angular !",
    "slogan": "Angular est le meilleur Framework !",
    "4266238317959690326": "assets/images/drapeaux/fr/drapeau_fr.jpg",
    "4425774926678752758": "Mon Drapeau",
    "3194024123575565444": "www.angular.io",
    "8703060702766654522": "Doc Angular IO",
    "4435388665562277072": "Formation Angular Développement Avancé"
  }
}
```

L'InternationalisatioN I18N

Editer Angular.json

messages.en.json | angular.json 1

formation-any-2023-09 > TP02-i18n > angular.json > ...

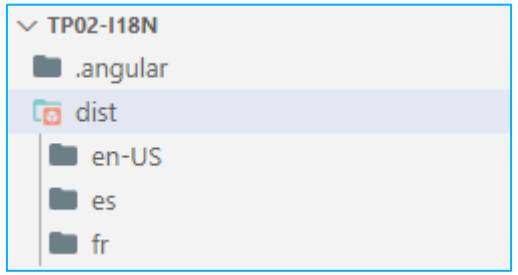
```
{
    "$schema": "./node_modules/@angular/cli/lib/config/schema.json",
    "version": 1,
    "newProjectRoot": "projects",
    "projects": {
        "webApp": {
            "i18n": {
                "sourceLocale": "fr",
                "locales": {
                    "en-US": "src/locale/messages.en.json",
```

messages.en.json | angular.json 1

formation-any-2023-09 > TP02-i18n > angular.json > ...

```
                "node_modules/bootstrap-icons/font/bootstra
                "src/styles.scss"
            ],
            "scripts": [
                "node_modules/bootstrap/dist/js/bootstrap.m
            ]
        },
        "configurations": {
            "en": {
                "localize": [
                    "en-US"
                ]
            },
```

ng build --localize



L'InternationalisatioN I18N

TP : Rajouter la traduction en espagnol

Hola Angular !
Angular esta mejor !
assets/images/drapeaux/es/drapeau_es.jpg
Mi bandera

TP :
- Changer le dossier de compilation en build
- Utiliser une cible ES2017
- désactiver ou activer le mode Strict

L’InternationalisatioN I18N

Comment traduire non pas la vue mais les données qui proviennent du TS ?

```
constructor() {
  // this.title=$localize `Formation Angular Développement Avancé`;
  this.title=$localize `:@@id:Formation Angular Développement Avancé`;
}
```

L'InternationalisatioN I18N

Option : afficher la traduction en mode serve !

schema.json | angular.json | package.json | messages.es.json

angular.json > $schema

```
                "sourceMap": true,
                "namedChunks": true
              }
            },
            "defaultConfiguration": "production"
          },
          "serve": {
            "builder": "@angular-devkit/build-angular:dev-server",
            "configurations": {
              "production": {
                "browserTarget": "webApp:build:production"
              },
              "development": {
                "browserTarget": "webApp:build:development"
              },
              "page_en": {
                "browserTarget": "webApp:build:en"
              },
              "page_es": {
                "browserTarget": "webApp:build:es"
              }
```

schema.json | angular.json | package.json | messages.es.json

package.json > { } scripts

```
{
  "name": "web-app",
  "version": "1.0.0",
  ▷ Déboguer
  "scripts": {
    "ng": "ng",
    "start": "ng serve --open --port=3000",
    "start-en": "ng serve --open --port=3000 --configuration=page_en",
    "start-es": "ng serve --open --port=3000 --configuration=page_es",
    "build": "ng build",
    "build-localize": "ng build --localize",
    "watch": "ng build --watch --configuration development  --stats-json",
    "test": "ng test",
    "json-server": "json-server -p 3001 src/assets/json/films.json ",
    "json-server-post": "json-server -p 3002 src/assets/json/bdd.json",
    "analyze": "webpack-bundle-analyzer dist/stats.json"
  },
```

Le Server Side Rendering « SSR »

SSR ou CSR

Le « **S**erver **S**ide **R**endering » existe depuis les années 1990 !
La page est rendue depuis le back et est affichée dans le navigateur du client.

Début 2010, les premiers « Webpack » et les « **S**ingle **P**age **A**pplication » font leur arrivée dans le monde du développement Web.

On parle alors de **C**lient **S**ide **R**endering, puisque le projet final contient seulement une page index.html qui fait elle même appel à des « chunk files javascript » (*des fichiers JavaScripts buildés*).

SSR ou CSR

Pourquoi revient-on depuis peu au SSR ?

- Il y aura de meilleures performance en SEO, bien que les moteurs dont Google garantit que les pages CSR seront correctement indexées, cela nécessite plus de temps mais les moteurs peuvent le faire !

- Le temps de chargement du contenu est un peu plus rapide, car on doit attendre que le contenu total de l'applis soit chargé (NB: le lazy loading solutionne cette problématique …)

- React, Vue et Angular l'implémentent facilement !

SSR ou CSR

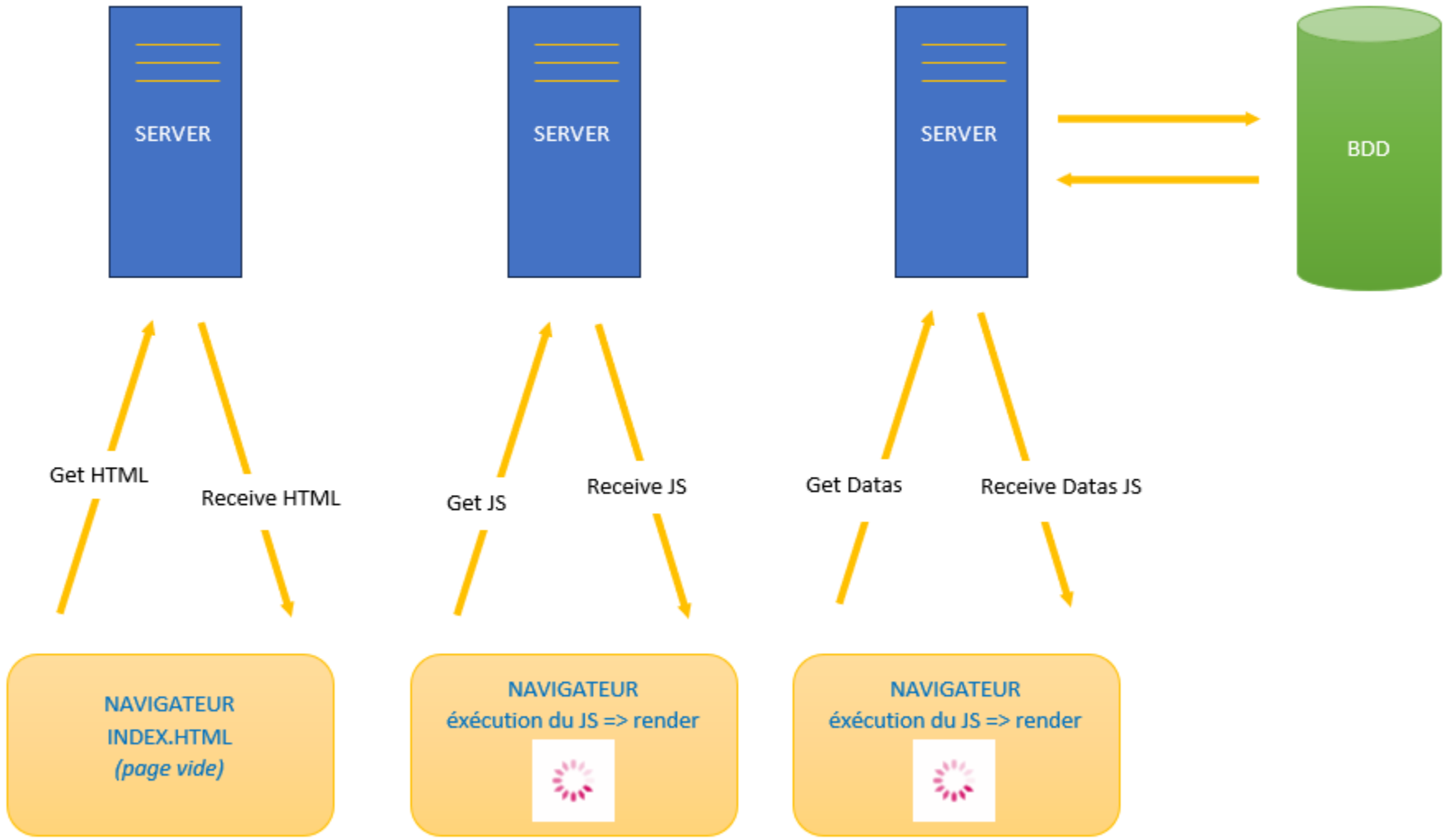
Pourquoi revient-on depuis peu au SSR ?

- Il y aura de meilleures performance en SEO, bien que les moteurs dont Google garantit que les pages CSR seront correctement indexées, cela nécessite plus de temps mais les moteurs peuvent le faire !
- Le temps de chargement du contenu est un peu plus rapide, car on doit attendre que le contenu total de l'applis soit chargé (NB: le lazy loading solutionne cette problématique …)
- React, Vue et Angular l'implémentent facilement !

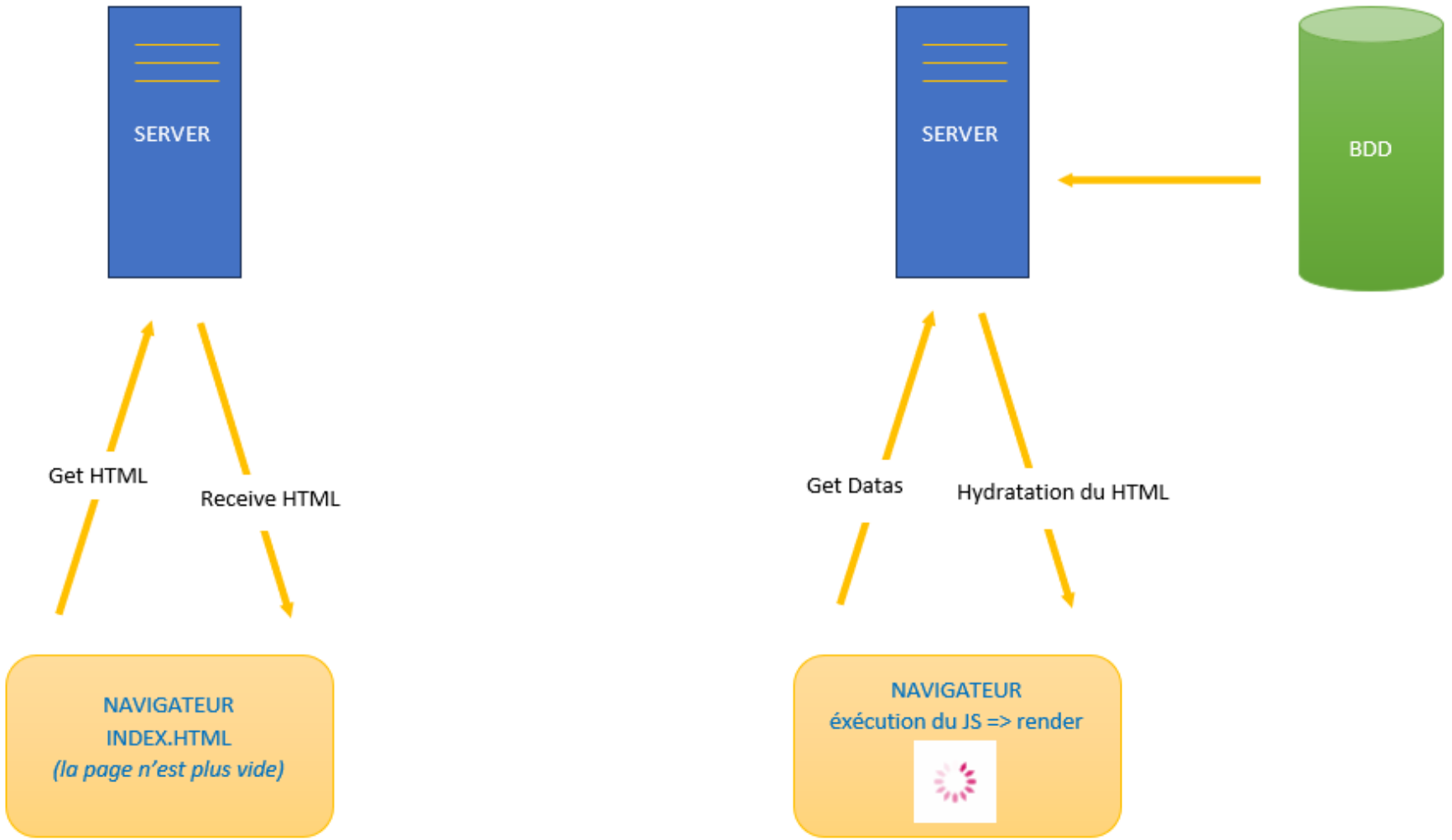
Inconvénients :

- La charge « back » est évidement plus couteuse
- Il est nécessaire d'implémenter Node.js car ce n'est plus un simple hébergement web statique !

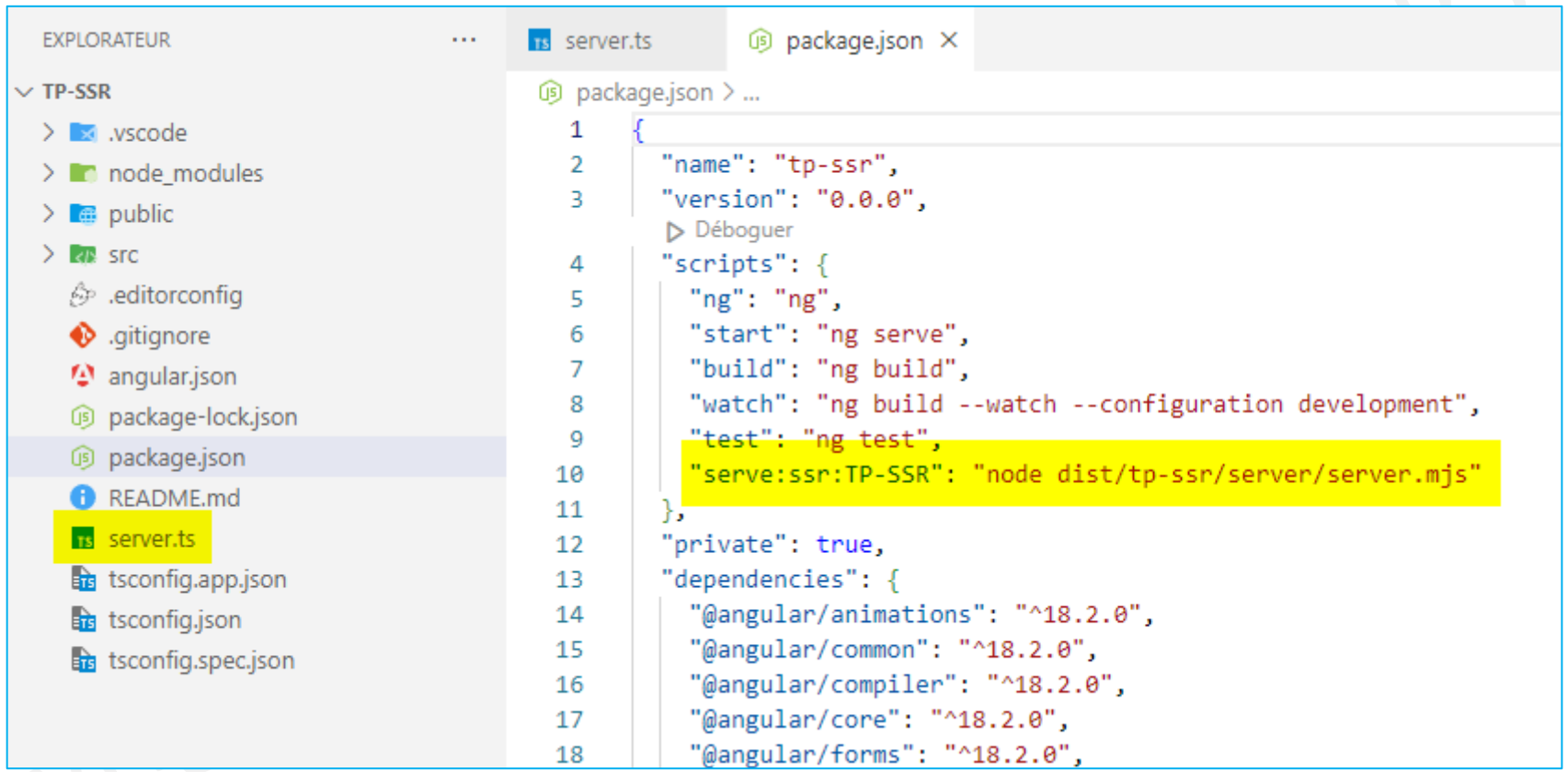
CSR : Client-Side Rendering | Single Page Application : Le rendu se fait côté client

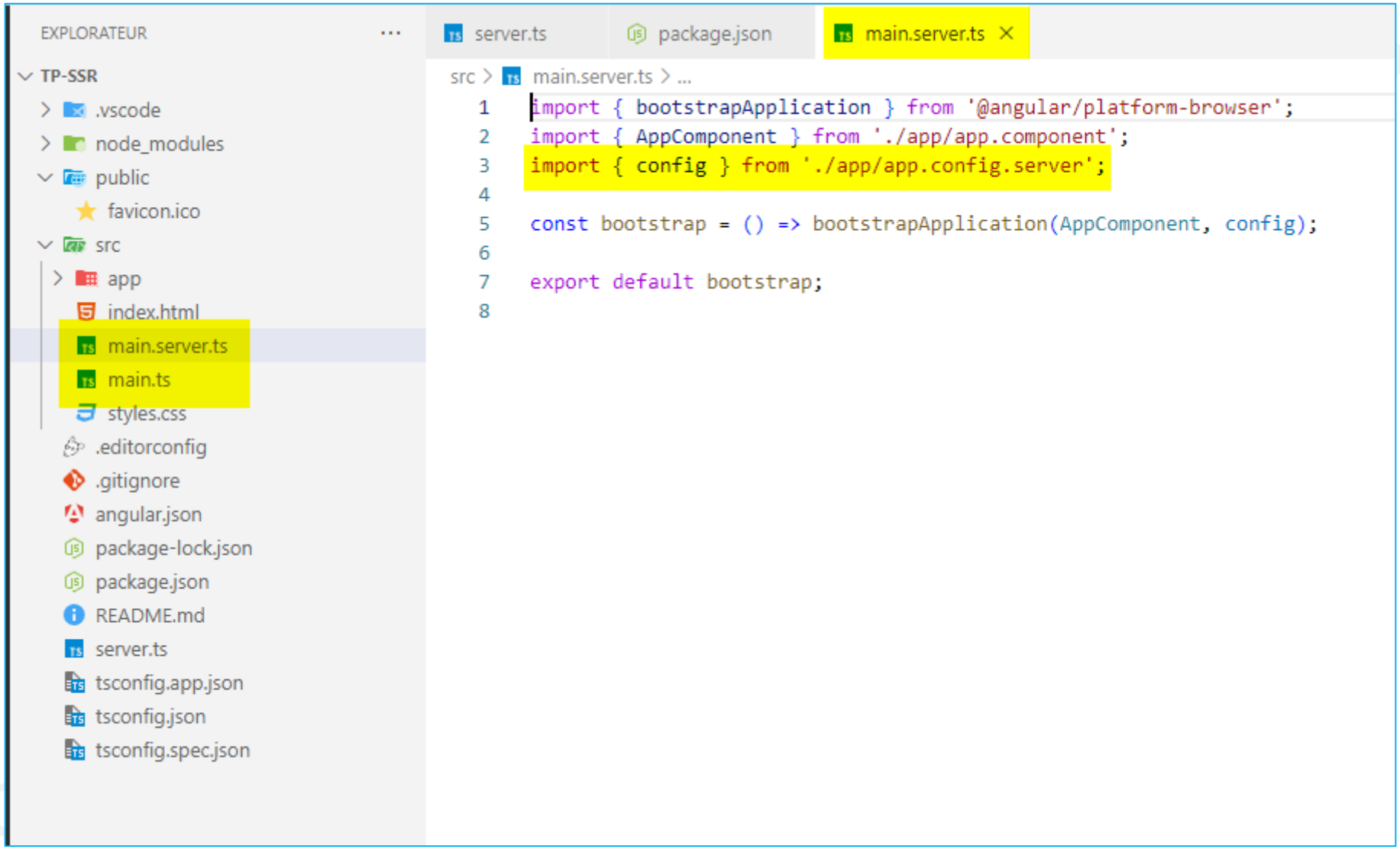


SSR : Server-Side Rendering



Mise en pratique : https://angular.dev/guide/ssr




EXPLORATEUR
TP-SSR
.vscode
node_modules
public
favicon.ico
src
app
index.html
main.server.ts
main.ts
styles.css
.editorconfig
.gitignore
angular.json
package-lock.json
package.json
README.md
server.ts
tsconfig.app.json
tsconfig.json
tsconfig.spec.json
server.ts
package.json
main.server.ts
src > main.server.ts > ...
1 import { bootstrapApplication } from '@angular/platform-browser';
2 import { AppComponent } from './app/app.component';
3 import { config } from './app/app.config.server';
4
5 const bootstrap = () => bootstrapApplication(AppComponent, config);
6
7 export default bootstrap;
8

```
  "watch": "ng build --watch --configuration development",
  "test": "ng test",
  "ssr": "node dist/tp-ssr/server/server.mjs"
},
```

PROBLÈMES SORTIE CONSOLE DE DÉBOGAGE TERMINAL PORTS

```
PS C:\Users\Michel\Desktop\TP-SSR> npm run build

> tp-ssr@0.0.0 build
> ng build

Browser bundles
Initial chunk files     | Names               |  Raw size | Estimated transfer size
main-YFAVBIXK.js        | main                | 219.77 kB |               59.76 kB
polyfills-FFHMD2TL.js   | polyfills           |  34.52 kB |               11.28 kB
styles-5INURTSO.css     | styles              |   0 bytes |                0 bytes

                        | Initial total       | 254.29 kB |               71.05 kB


Server bundles
Initial chunk files     | Names               |  Raw size
server.mjs              | server              |   1.11 MB |
chunk-WHPWUB66.mjs      | -                   | 528.79 kB |
polyfills.server.mjs    | polyfills.server    | 268.60 kB |
chunk-YFYEKWGK.mjs      | -                   |  19.16 kB |
chunk-5XUXGTUW.mjs      | -                   |   2.55 kB |
render-utils.server.mjs | render-utils.server |   1.46 kB |
main.server.mjs         | main.server         | 149 bytes |

Lazy chunk files        | Names               |  Raw size
chunk-VCVCHJZV.mjs      | xhr2                |  12.07 kB |

Prerendered 1 static route.
Application bundle generation complete. [13.800 seconds]

Output location: C:\Users\Michel\Desktop\TP-SSR\dist\tp-ssr

PS C:\Users\Michel\Desktop\TP-SSR> npm run ssr

> tp-ssr@0.0.0 ssr
> node dist/tp-ssr/server/server.mjs

Node Express server listening on http://localhost:4000
```

Annexes

Rappels – Révisions Ecmascript*

* Si besoin

Ecmascript 2015+

Ecmascript, la normalisation du langage Javascript a révolutionné Javascript en 2015 (ES6) en le dotant de fonctionnalités et spécificités dignes d'un « langage de haut niveau ».
Cette transformation totale et globale devenait nécessaire au vu des besoins des développements front web et mobile. *Angular utilise Ecmascript*.

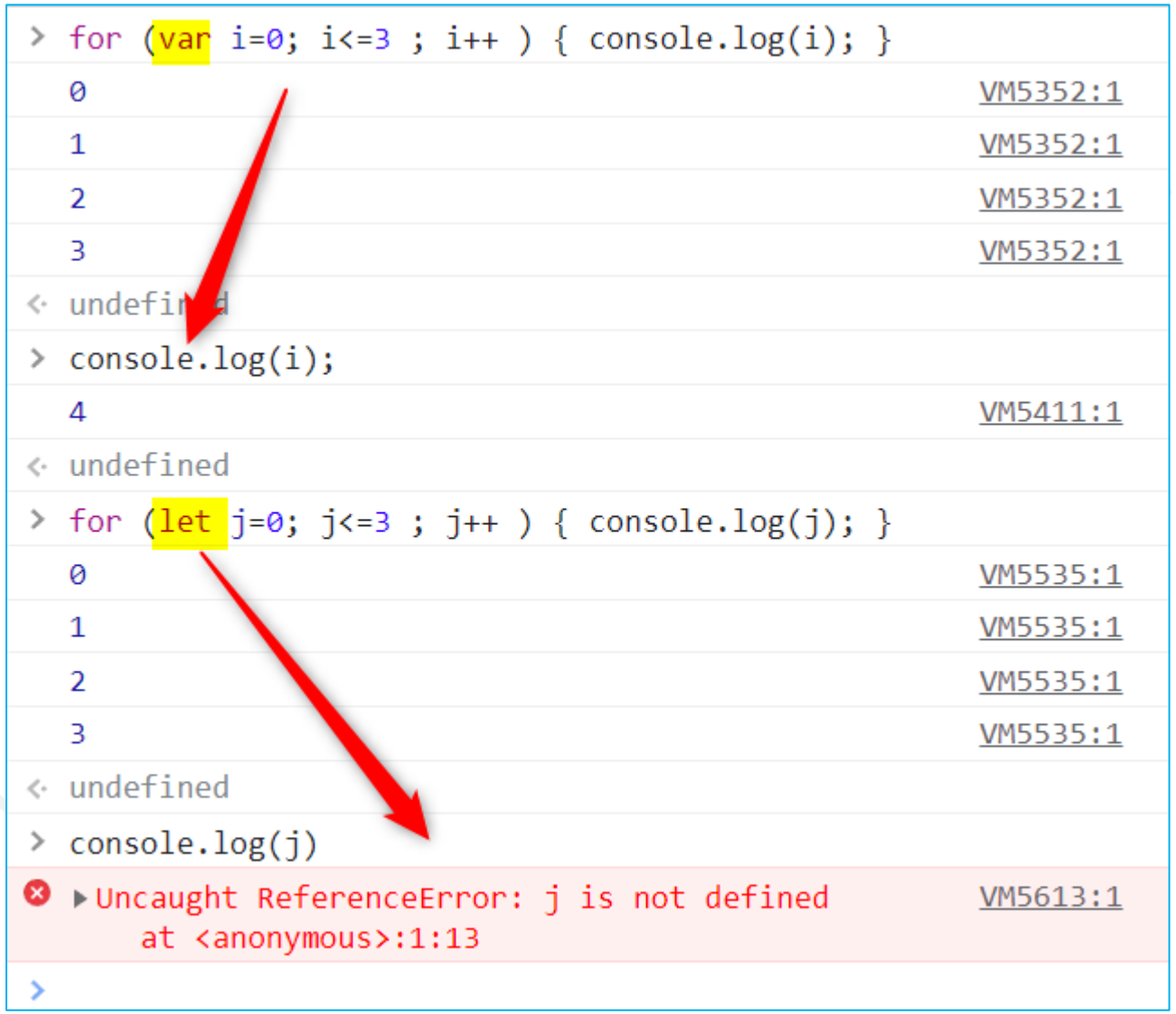
Les améliorations les plus importantes du langage concernent :

1. La définition des variables et leur portée : VAR, LET et CONST
2. La possibilité de créer des modules de code exportable et importable : IMPORT, EXPORT

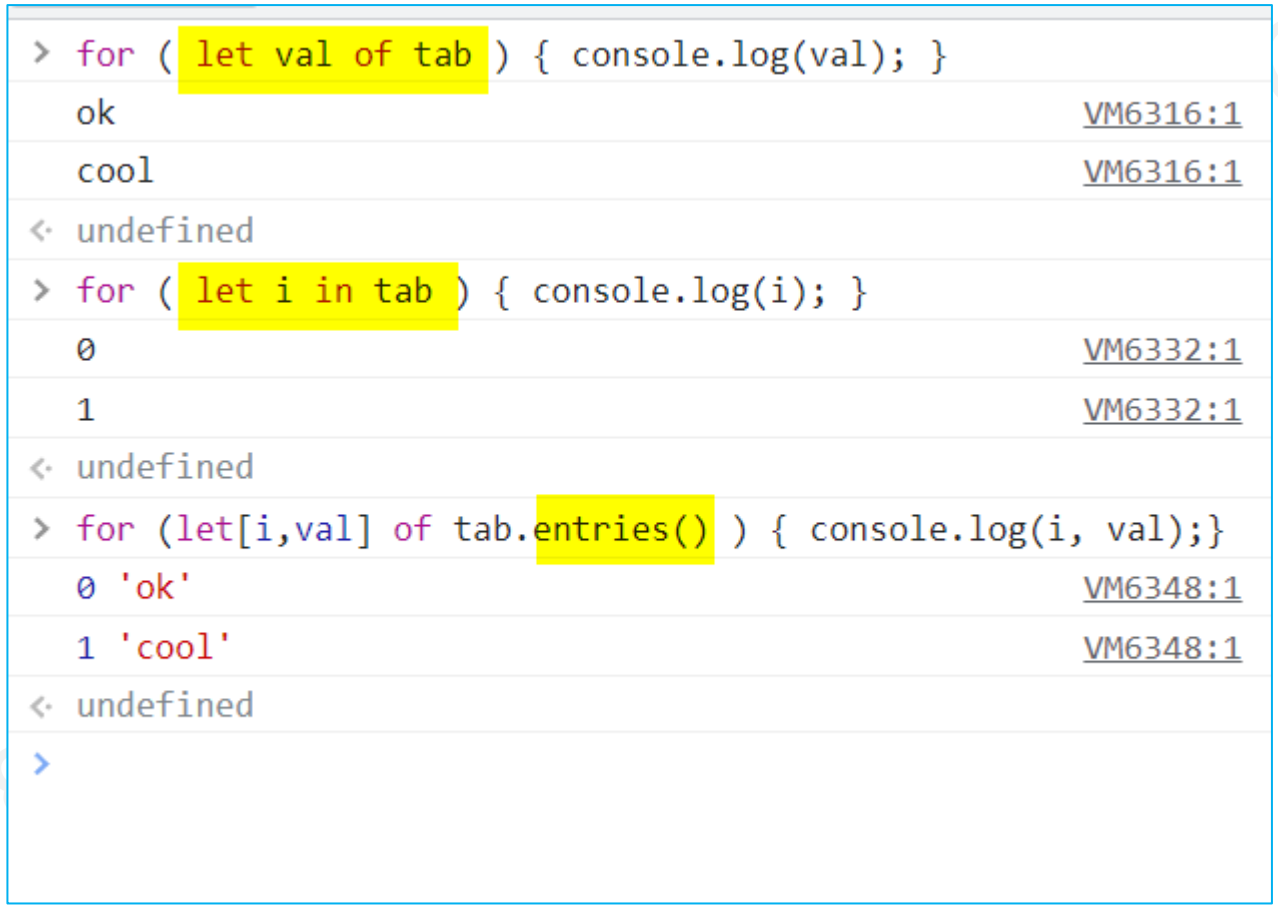
3. L'écriture simplifiée des FAT ARROWS
4. La string interpolation des variables avec ${maVariable} et la concatenation avec les back stits
5. Les itérateurs et les boucles FOR OF, FOR IN et FOR OF ENTRIES
6. Les spread operators pour déstructurer les tableaux
7. La notion de promesses afin de gérer les événements et les procédures asynchrones
8. Les objets et les CLASS*

```
<script src="js/index.js" type="module" defer></script>
```

Ecmascript 2015+



Ecmascript 2015+



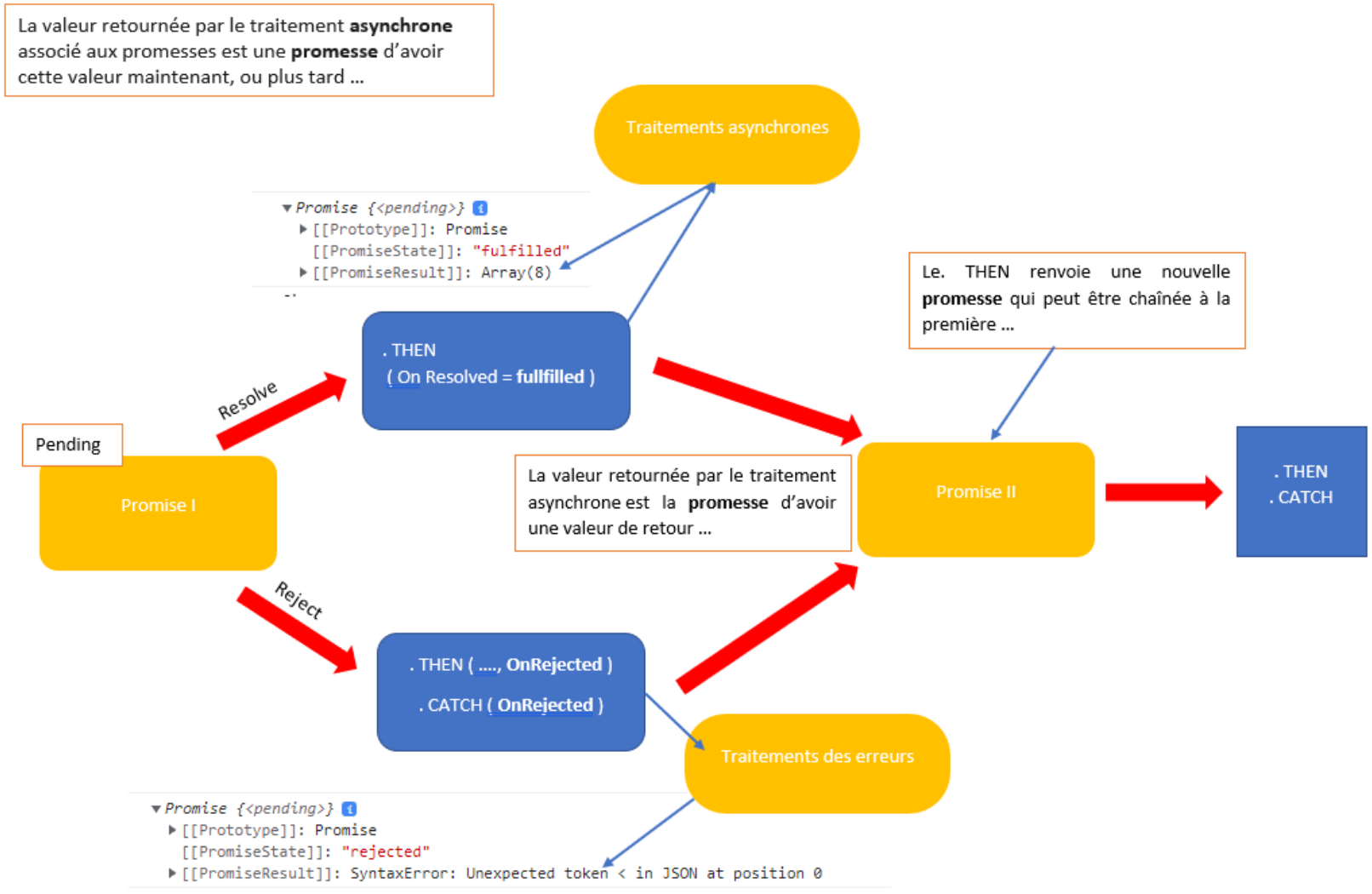
Ecmascript 2015+

```
1   // nouveautés (fonctions)
2   function direBonjour(nom = "SkyWalker") {
3       // passage d'arguments par défaut
4       // console.log("Bonjour " + nom);
5
6       // string interpolation des variables
7       console.log(`Bonjour ${nom}`);
8       document.querySelector("#resultat").innerHTML = `Bonjour ${nom}`;
9   }
10
11  const direBonjour2 = (prenom, nom) => {
12      console.log(`Bonjour ${prenom} ${nom}`);
13      document.querySelector("#resultat").innerHTML = `Bonjour ${prenom} ${nom}`;
14  }
15
16  direBonjour(); // javascript s'écrit en camelCase majuscule à chaque mot sauf le 1er
17  direBonjour2("Dark", "Vador");
18
```

Ecmascript 2015+

```
58  // ------------------------------------------------------------
59  let personne = {
60      prenom: "Luke",
61      nom: "Skywalker",
62      // tableaux
63      langages: ["JS", "React", "NG", "Vue"],
64      // méthodes
65      nomComplet() {
66          return `${this.prenom} ${this.nom}`;
67      }
68  }
69  console.log(personne.nomComplet());
70
71  // ---- Nouveautés class (PascalCase = maj à chaque mot )
72  class Personne {
73
74      // Constructor
75      constructor(nom, age) {
76          this._nom = nom;
77          this._age = age;
78      }
79      // Méthodes
80      infosPersonne() {
81          return `Bonjour je m'appelle ${this._nom} et j'ai ${this._age} ans.`;
82      }
83  }
84
85  let p1 = new Personne("Emilie",  35);
86  console.log(p1);
```

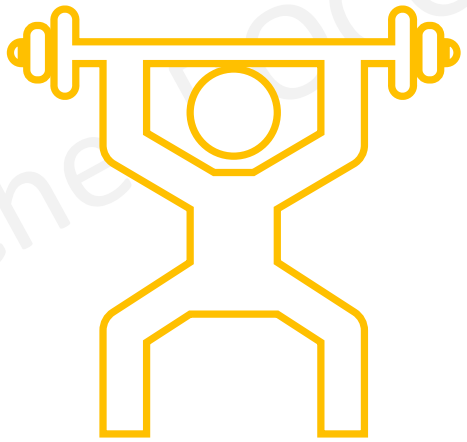
Ecmascript 2015+



Ecmascript 2015+

```
export const fnFilm = () => {
    // ES6 : fonction fetch
    // simplication d'écriture d'Ajax
    // amélioration du process JS ( Promises )

    //  1-  Request Query (requête Infos):
    // const _url = 'https://test.webjs.fr/films.json';
    const _url = 'https://dev.webjs.fr/films.json';
    // const _url='http://127.0.0.1:3000/json/films.json'; // en local avec un serveur json

    //  2-  Request Init Query
    //  définir la méthode (get ou post) ... ou patch ou put ou delete)
    const _method = 'get';
    // définir nos entêtes HTTP
    const _headers = new Headers();

    _headers.append('Content-Type', 'text/json'); //type mime (ex:image/png)
    // _headers.append('Access-Control-Allow-Origin', 'cors');

    //  3- Ecriture du FETCH
    // fetch renvoie une promesse
    fetch(
        _url,
        {
            method: _method,
            headers: _headers
        }
    ).then(
        // si on arrive à avoir une réponse du serveur
        // état onFullFilled
        (responseHTTP) => {
            // le then récupère une réponse 200 ok 404
            console.warn('On Fullfilled du 1er Then');
            console.log(responseHTTP);
            console.log(responseHTTP.body);
```

Le Detection Change Mode (value based) Zone.JS ?



Le Detection Change Mode Angular

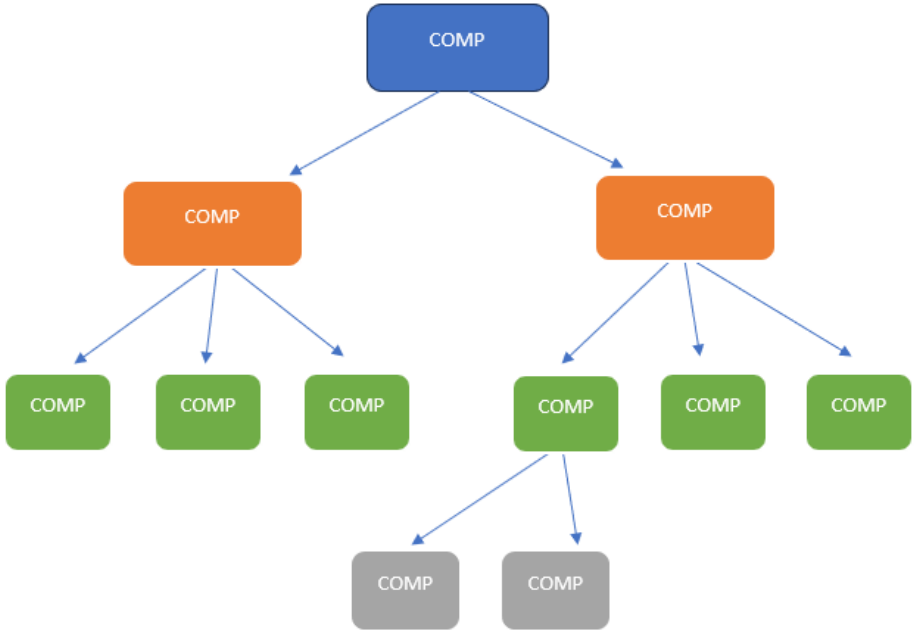
Lorsque des propriétés d'entrée (Input) ou des événements d'entrée sont déclenchés, Angular met à jour l'arbre des composants pour détecter d'éventuels changements à apporter aux affichages des Vues.

Modifier le comportement par défaut du « Change Detection Mode » d'Angular peut être utile lorsque des composants « rendent » (affichent) des données qui ne changent pas souvent.

Le nombre de cycles de change detection sera réduit et les performances de l'application en seront augmentées.

Le Detection Change Mode Angular

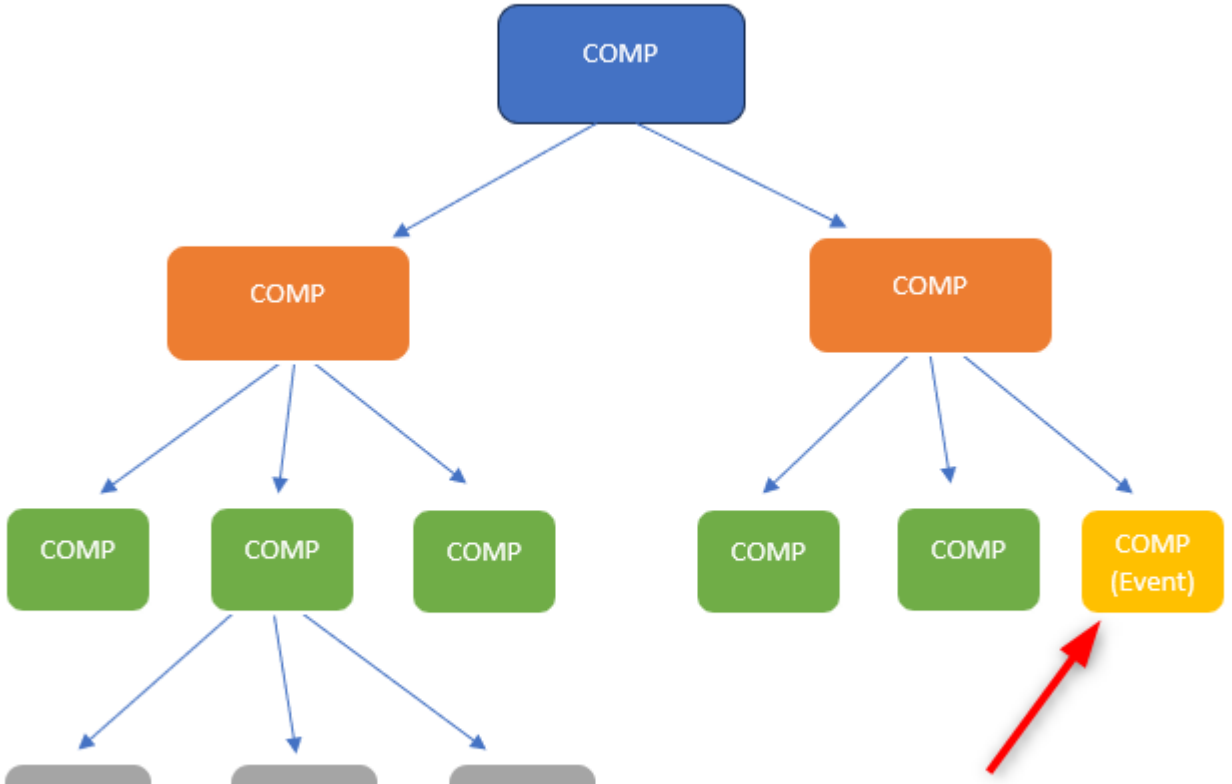
Comment Angular met à jour l'affichage des composants imbriqués par des Inputs, dans une stratégie composants parents-enfants



Le Detection Change Mode par défaut

```
@Component({
  selector: 'app-liste-des-films',
  templateUrl: './liste-des-films.component.html',
  styleUrls: ['./liste-des-films.component.scss'],
  changeDetection: ChangeDetectionStrategy.Default,
})
```

2-Angular parcourt tout l'arbre des composants pour afficher un changement de valeur sur les composants



1-Une interaction survient sous la forme d'un Event sur un composant enfant

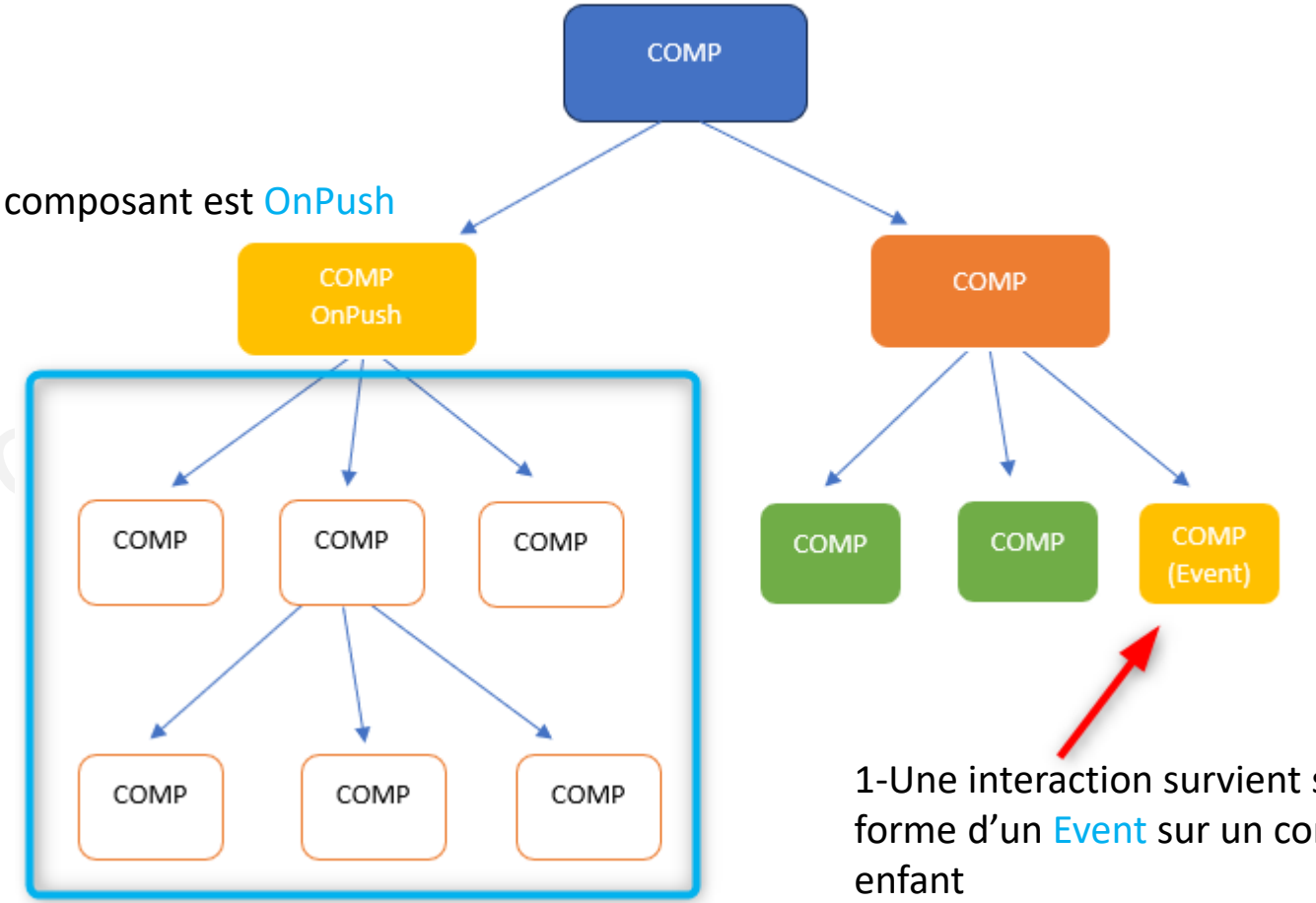
Le Detection Change Mode onPush

Angular limite le nombre de vérifications sur les enfants

Si la référence de l'objet (valeur…) passée en Input ne change pas, le composant enfant ne doit pas être modifié.

Angular arrête son parcours…

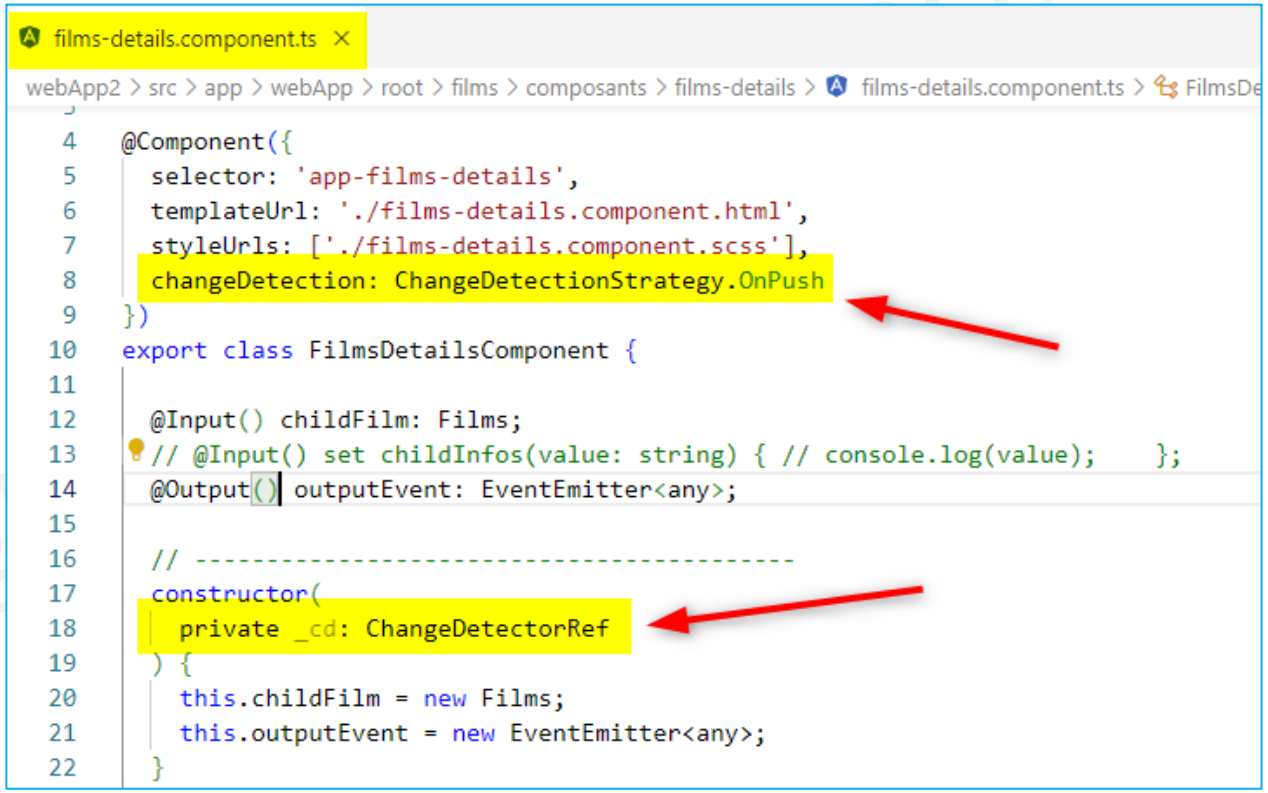
Le composant est OnPush



1-Une interaction survient sous la forme d'un Event sur un composant enfant

La class ChangeDetetctorRef

On peut forcer la mise à jour de l'affichage de la Vue d'un composant de manière explicite, sans attendre (ou dépendre) du déclenchement des valeurs d'entrées du composant.



La class ChangeDetetctorRef

« markForCheck » indique que le composant doit être vérifié lors du prochain parcours de l'arbre des composants, même si la stratégie de mise à jour de « change detection » est OnPush

```
// ----------------------------------------
public updateDataFromBack = () => {
    // obtenir les datas du back pour ur
    // this.childFilm = this._serviceUp
    this._cd.markForCheck();
}
```

```
// ----------------------------------------
constructor(
  private _cd: ChangeDetectorRef
) {
  this.childFilm = new Films;
  this.outputEvent = new EventEmitter<any>;
}
// ----------------------------------------
// ngAfterViewChecked() {console.log('After View Checked (Child)'); }

// ngOnChanges(changes: SimpleChanges) {
//   if (changes) {
//     console.log('Changes : ', changes);
//     // console.log('Previous Valeur : ' , changes['childInfos'].previousVal
//     // console.log('Current Valeur : ' , changes['childInfos'].currentValue
//   }
// }
// ----------------------------------------
public get detectionChange() {
  console.log('View Checked !', this.childFilm);
  return true;
}
// ----------------------------------------
public updateDataFromBack = () => {
    // obtenir les datas du back pour une MAJ du composant en tâche de fond
    // this.childFilm = this._serviceUpdate.getCheckDatasFromBack();
    this._cd.
}
// ----------
public choixFil
```

checkNoChanges
detach
detectChanges
markForCheck
reattach

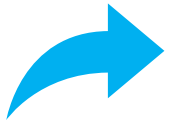
La class ChangeDetectorRef

```
  templateUrl: './films-details.component.html',
  styleUrls: ['./films-details.component.scss'],
  changeDetection: ChangeDetectionStrategy.OnPush
})
export class FilmsDetailsComponent {

  @Input() childFilm: Films;
  // @Input() set childInfos(value: string) { // console.log(value);    };
  @Output() outputEvent: EventEmitter<any>;

  // ----------------------------------------
  constructor(
    private _cd: ChangeDetectorRef
  ) {
    this.childFilm = new Films;
    this.outputEvent = new EventEmitter<any>;
  }
  // ----------------------------------------
  // ngAfterViewChecked() {console.log('After View Checked (Child)'); }

  // ngOnChanges(changes: SimpleChanges) {
  //   if (changes) {
  //     console.log('Changes : ', changes);
  //     // console.log('Previous Valeur : ' , changes['childInfos'].previousValue);
  //     // console.log('Current Valeur : ' , changes['childInfos'].currentValue);
  //   }
  // }
  // ----------------------------------------
  public get detectionChange() {
    console.log('View Checked !', this.childFilm);
    return true;
  }
  // ----------------------------------------
  public updateDataFromBack = () => {
      // obtenir les datas du back pour une MAJ du composant en tâche de fond
      // this.childFilm = this._serviceUpdate.getCheckDatasFromBack();
      this._cd.markForCheck();
  }
```

PWA

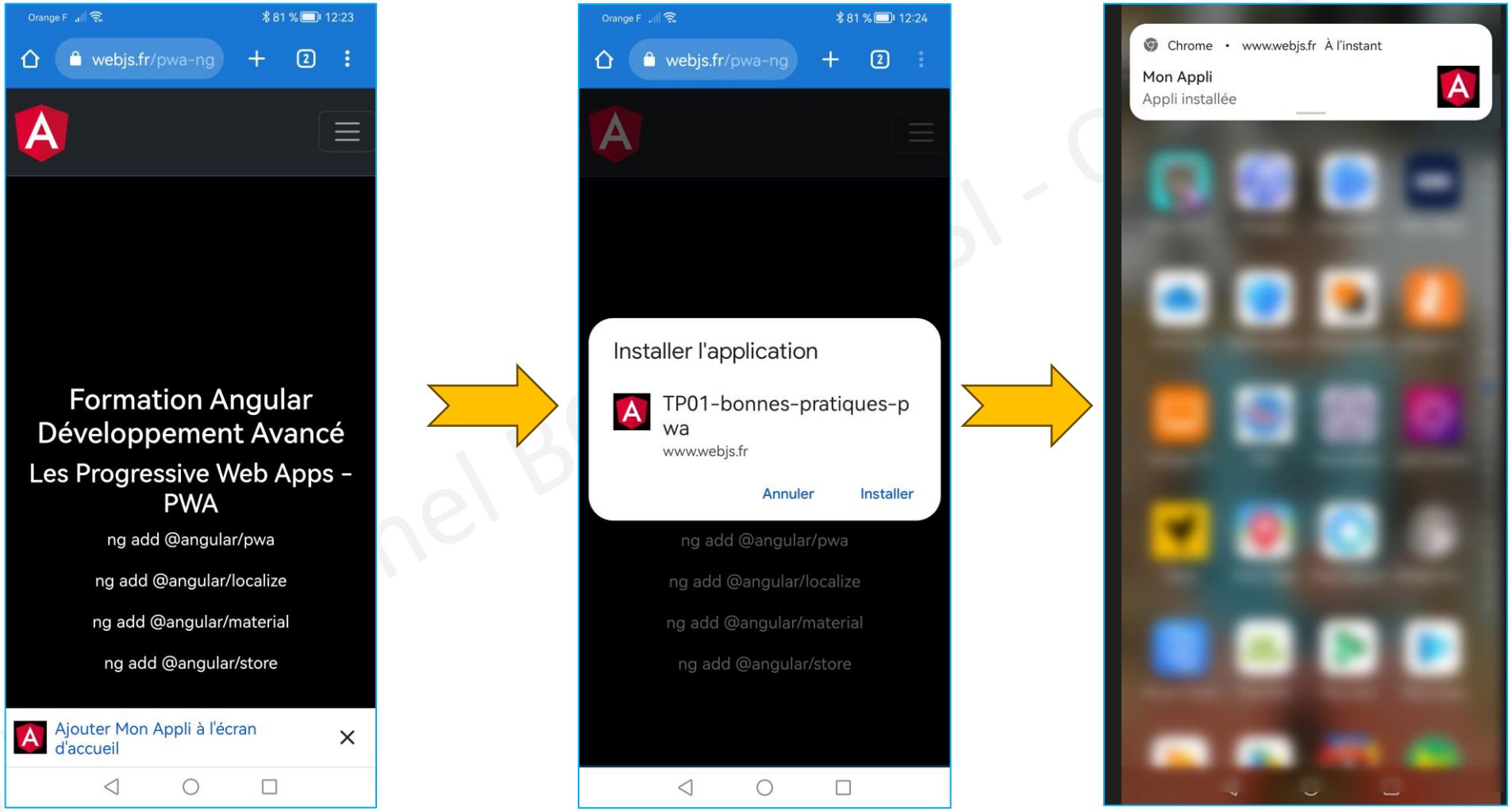
PWA – les Progressive Web Apps ?

Une Application Web ou une Application Mobile



Mise en situation :
consulter cette appli depuis un smartphone :
https://webjs.fr/pwa-ng/
https://www.webjs.fr/pwa-vanilla/

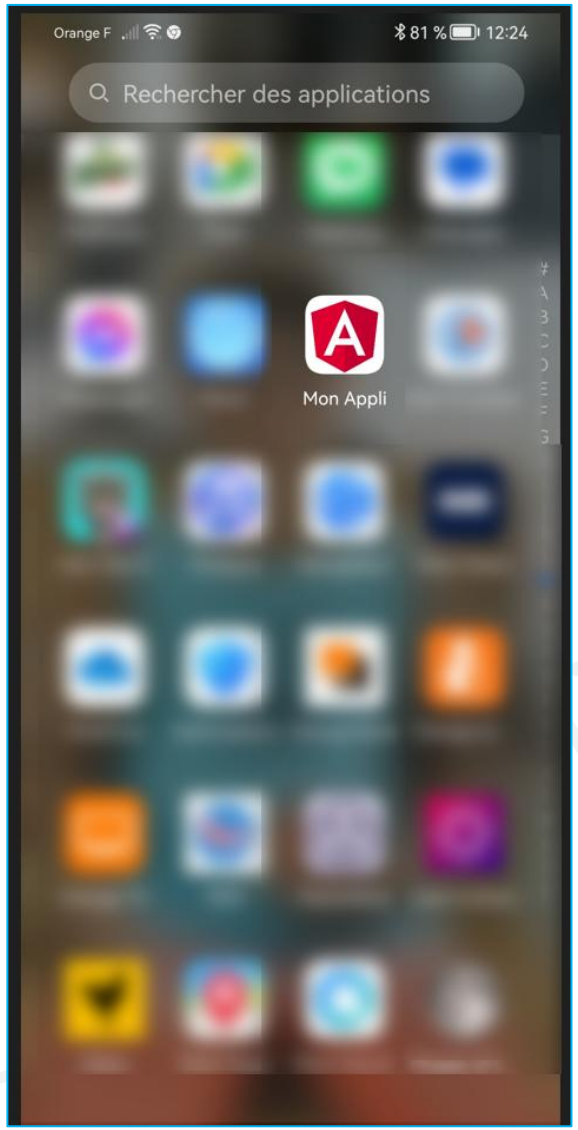
PWA – (le fichier web-manifest)



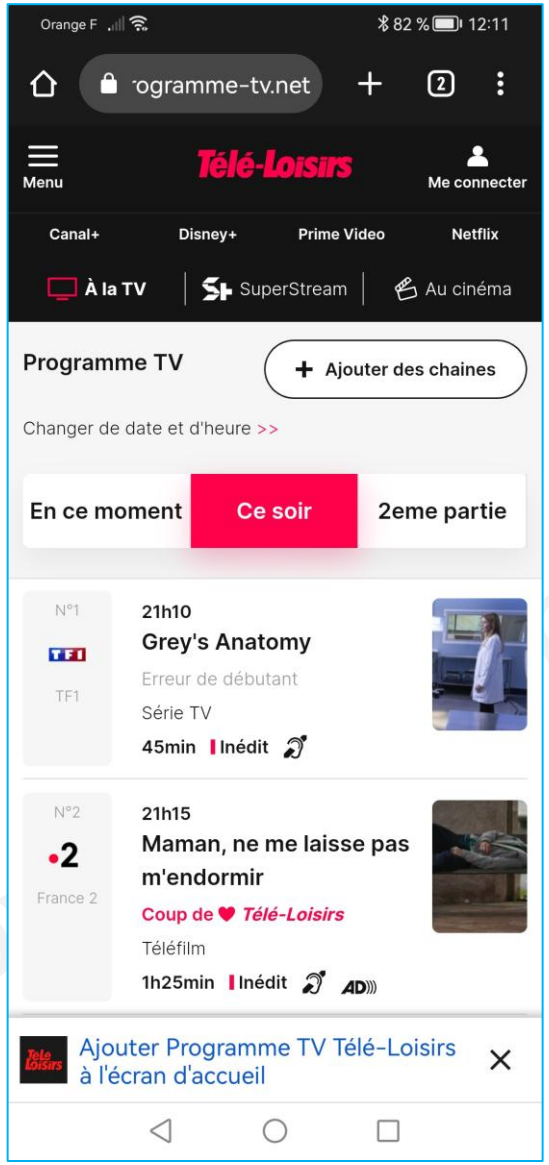
Références :

- https://developer.mozilla.org/fr/docs/Web/Progressive_web_apps/Guides/Making_PWAs_installable
- https://developer.mozilla.org/fr/docs/Web/Manifest

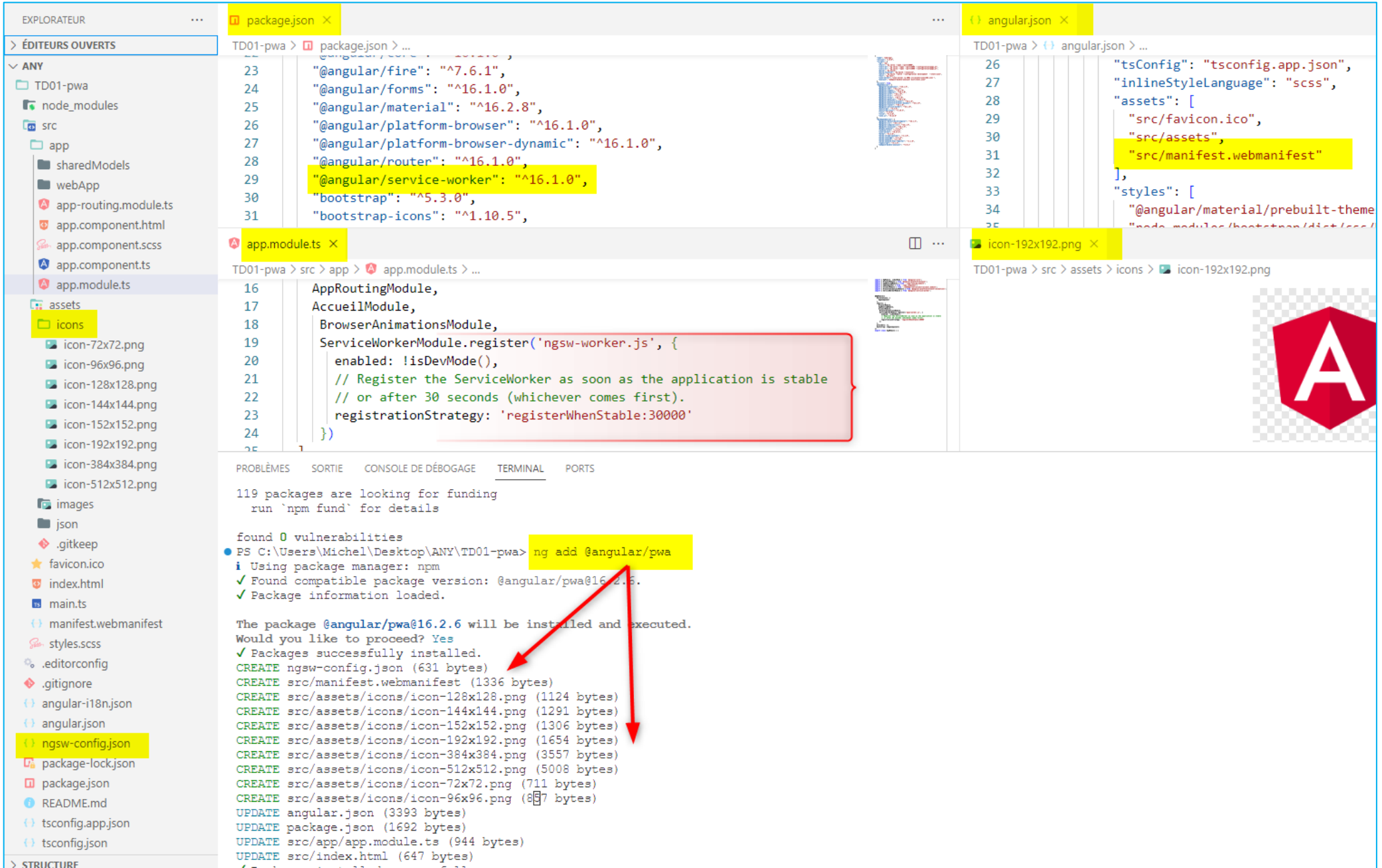
PWA – les Progressive Web Apps



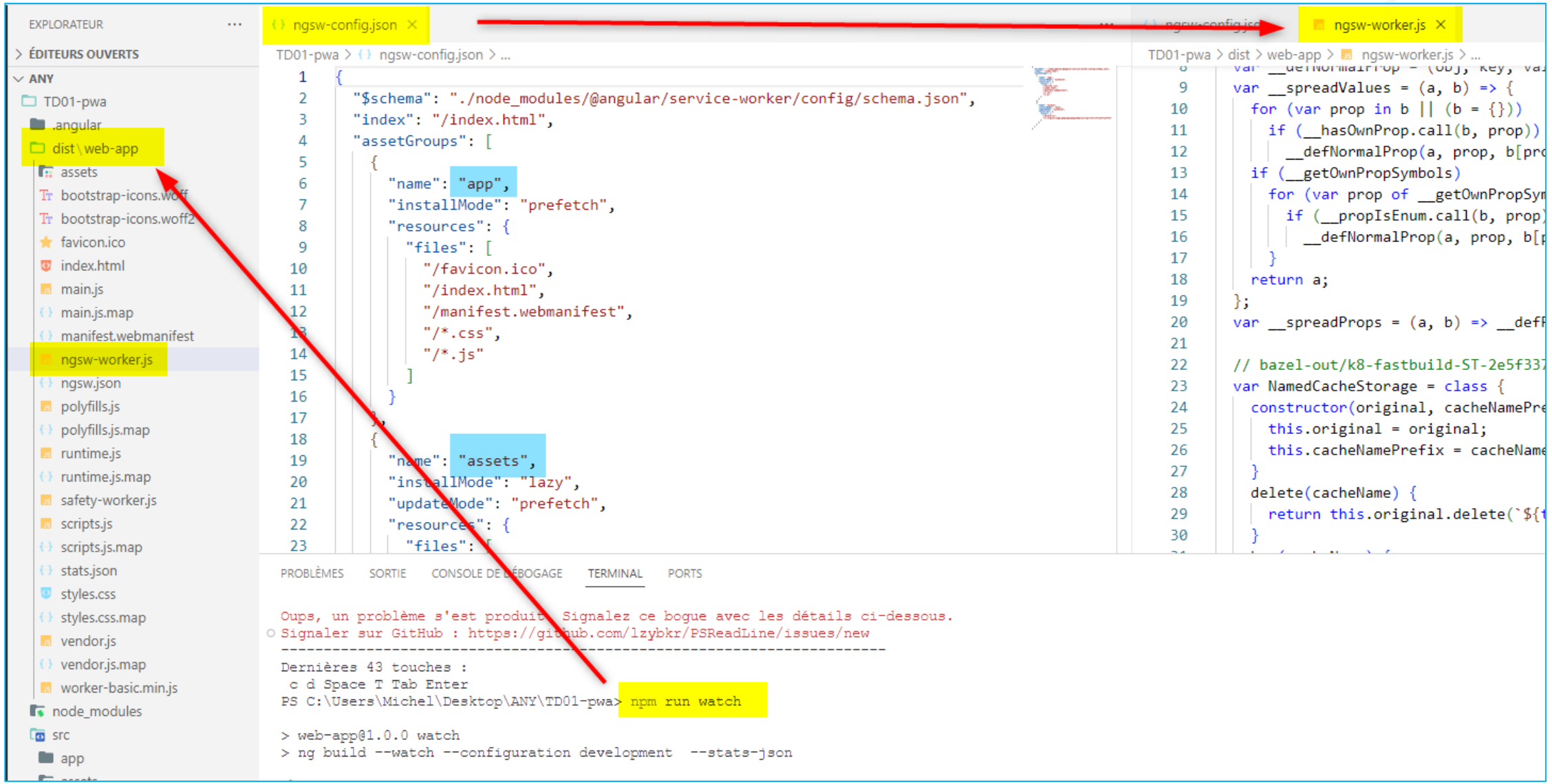
PWA – les Progressive Web Apps



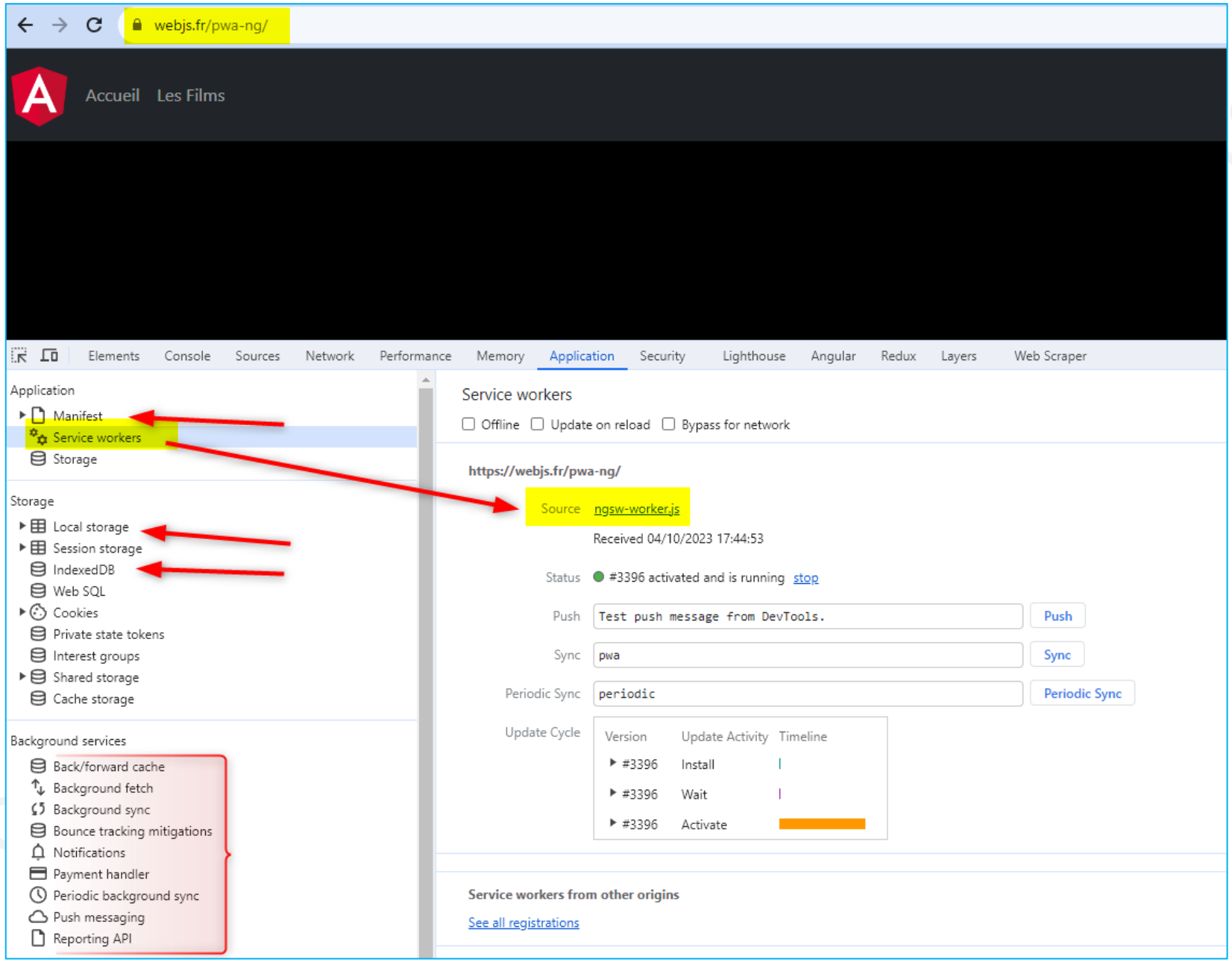
Commande d'installation et d'intégration de package : **ng add @angular/pwa**



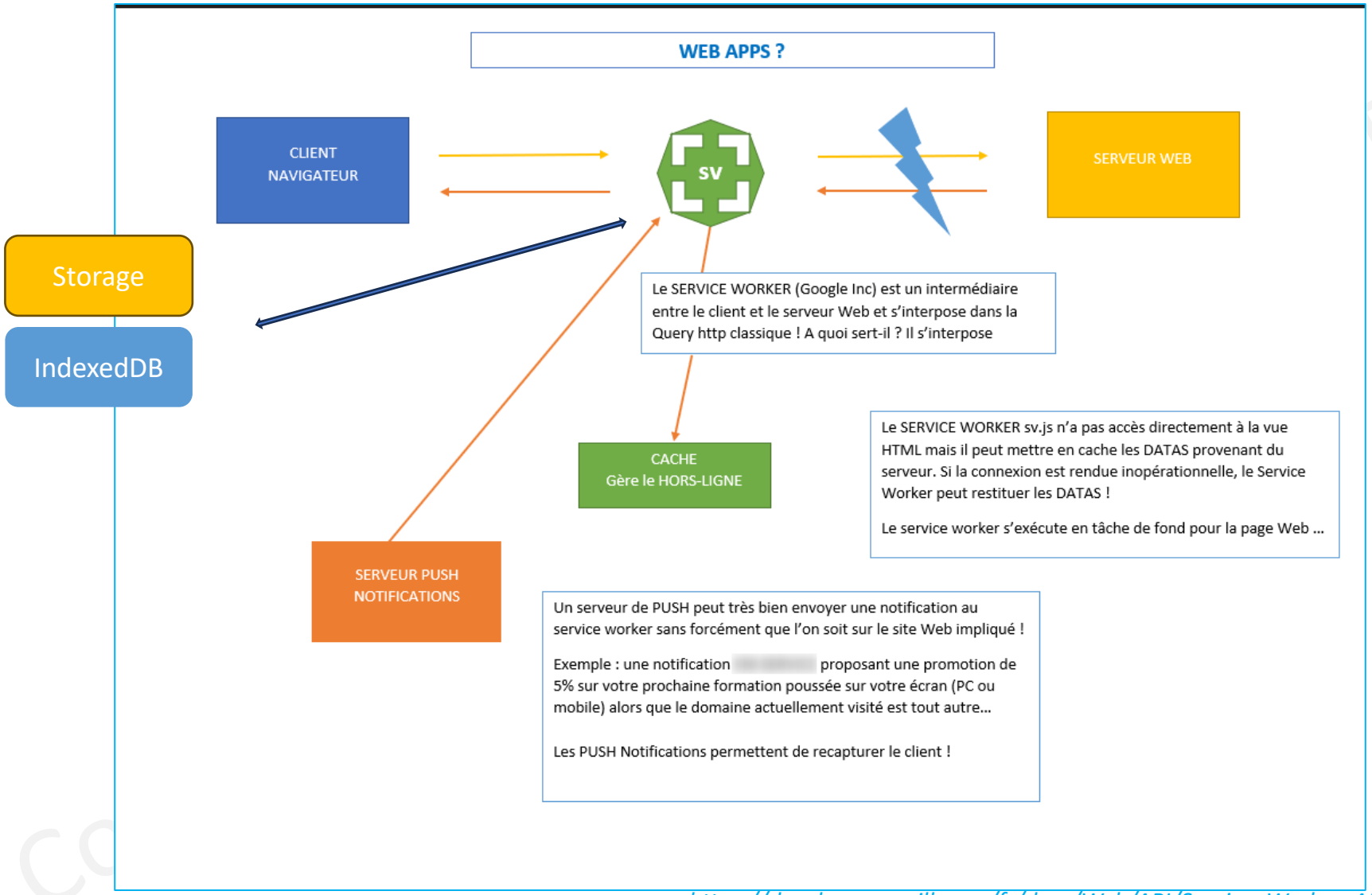
npm run build – npm run watch



PWA – les Progressive Web Apps



PWA – le Service Worker



https://developer.mozilla.org/fr/docs/Web/API/Service_Worker_API
https://angular.io/guide/service-worker-config

PWA – le Storage

```
const addToStorage = (objet) => {

    console.log(JSON.stringify(objet)); // output {'':''}
     const nbDocs = localStorage.length;
     localStorage.setItem(nbDocs, JSON.stringify(objet));
```

```
const showNotesFromStorage = () => {
    for (let i=0; i < localStorage.length ; i++) {
        console.log(localStorage.key(i) ,localStorage.getItem(i) );
        const noteStorage = localStorage.getItem(i);

    }
// For OF
 // https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Global_Objects/Object/entries

    for (const [key, value] of Object.entries(localStorage)) {
        console.log(key,value);
        addDom(JSON.parse(value));
    }
};
```

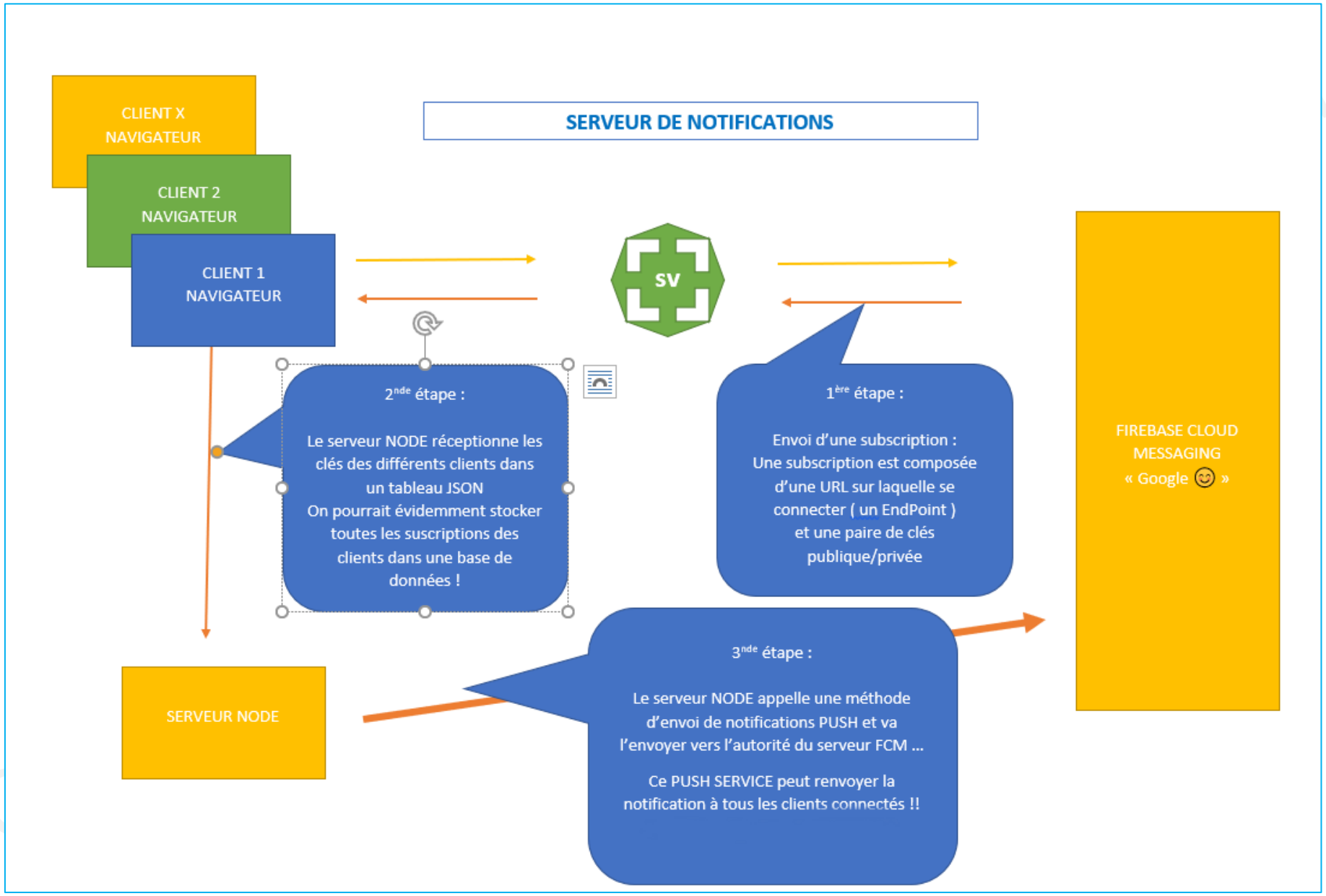
```
const delNotes = () => {
    localStorage.clear();
    eltUlListe.innerHTML='';
};
```

PWA – Background-Periodic-Sync

```
export const sync = () => {

    console.clear();
    console.log('---SYNC---');

    //demander les permissions
    navigator.permissions.query(
        {
            name:'background-sync'
        }
    ).then(
        (e) => {
            const statusQueryPerm = e.state;
            console.warn(statusQueryPerm);
            if (statusQueryPerm==='granted') {
                console.log(`---> Background -sync est autorisé`);
            }
        });
// ------------------------------------------------
navigator.serviceWorker.ready
.then(
    (registration) => {
        registration.sync.register('tag-maj-cache');
    }
);
```

```
export const periodicSync = () => {

    console.clear();
    console.log('---Periodic SYNC---');

    //demander les permissions
    navigator.permissions.query(
        {
            name:'periodic-background-sync'
        }
    ).then(
        (e) => {
            const statusQueryPerm = e.state;
            console.warn(statusQueryPerm);
            if (statusQueryPerm==='granted') {
                console.log(`---> Periodic Background sync est autorisé`);
            }
        });
// ------------------------------------------------
navigator.serviceWorker.ready
.then(
    async registration => {
        await registration.periodicSync.register(
            'tag-periodic-sync',
            {
                // minInterval: 24 * 60 * 60 * 1000
                // 24 heures => daiy news
                //  exprimé en ms
                minInterval : 5
            }
            );
    }
);
}
```

PWA – les Progressive Web Apps

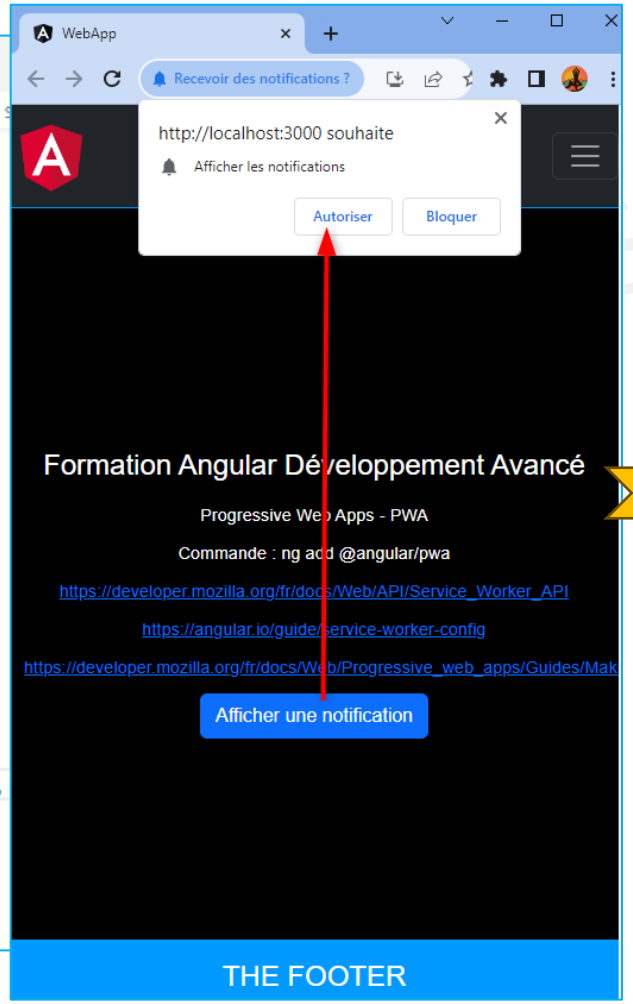


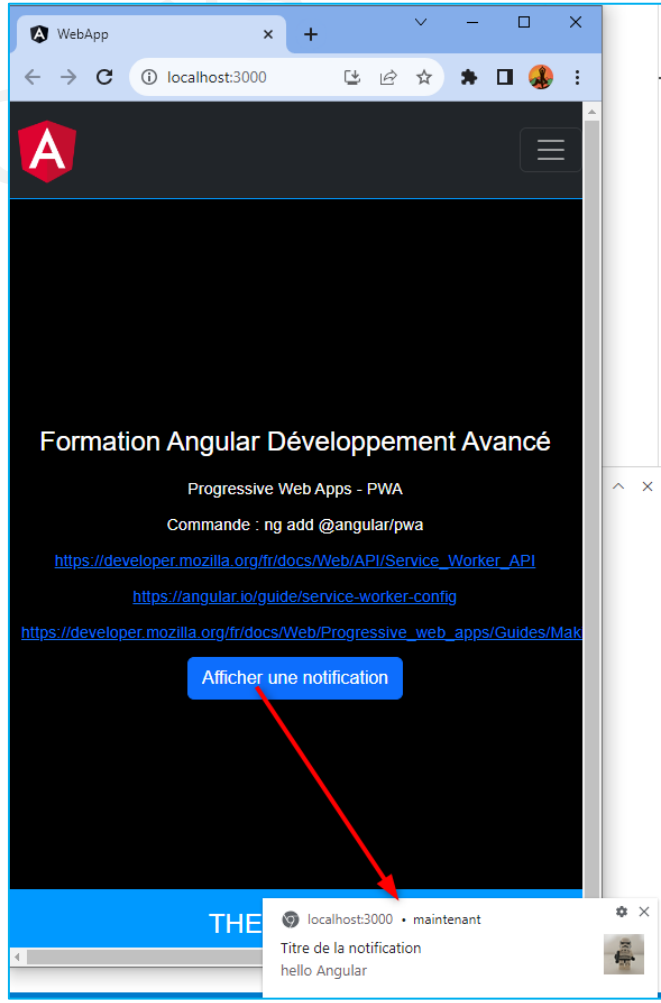
PWA – Notification

body.component.ts

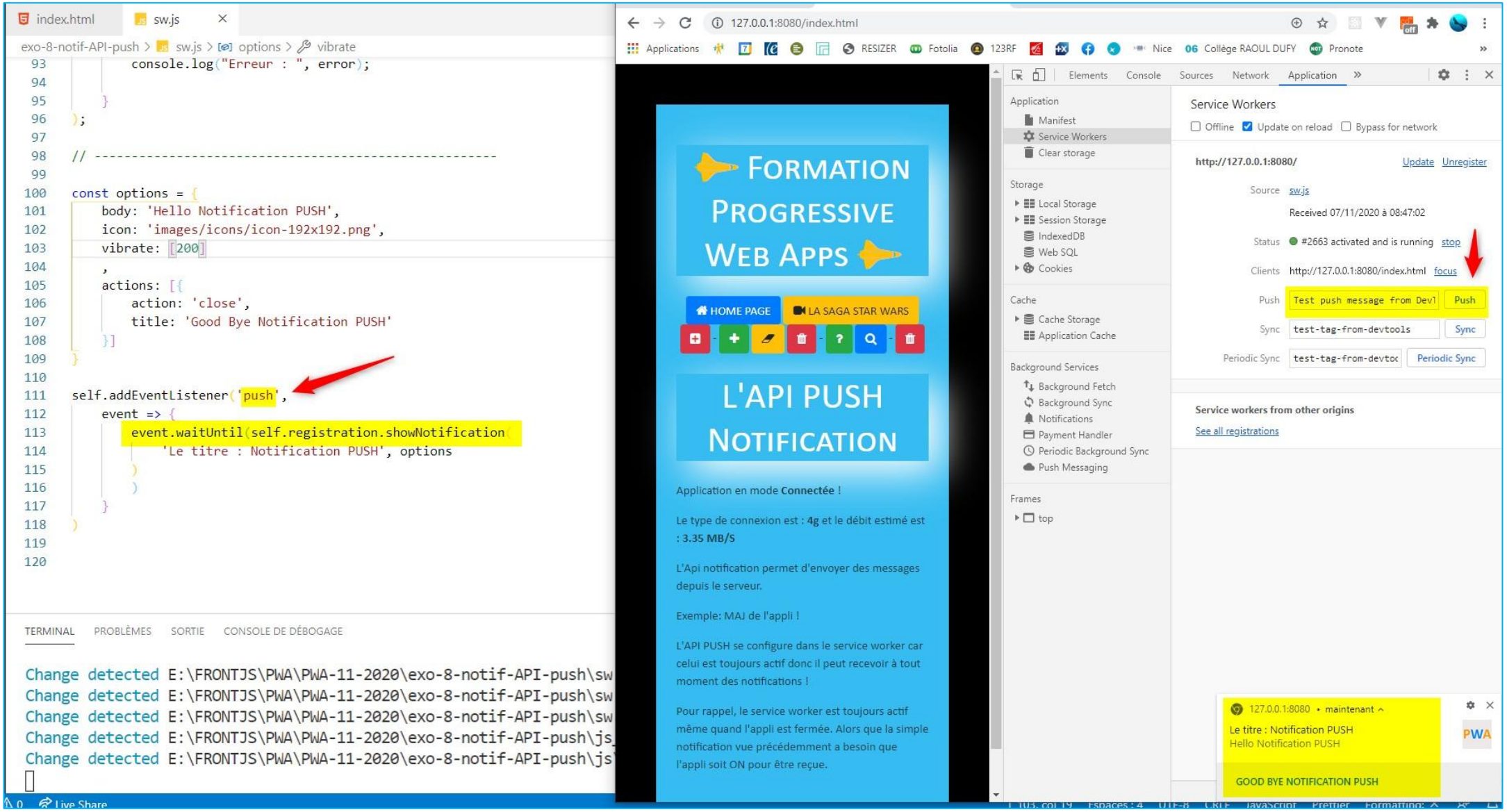
TD01-pwa > src > app > webApp > root > accueil > body > body.component.ts > BodyComponent

```
19
20  // ---------- notif simple --------------------
21  public showNotification = () => {
22
23    if ('Notification' in window) {
24      console.log('Notifications autorisées');
25
26      Notification.requestPermission(
27        (e) => {
28          console.log(e);
29          if (e==='granted') {
30
31            const options={
32              body:'hello Angular',
33              lang: 'fr-FR',
34              icon: 'assets/images/avatars/avatar1.jpg',
35              vibrate: [200, 100, 200]
36            };
37
38            const notif = new Notification('Titre de la notification',
              options);
39          }
40        }
41      );
```

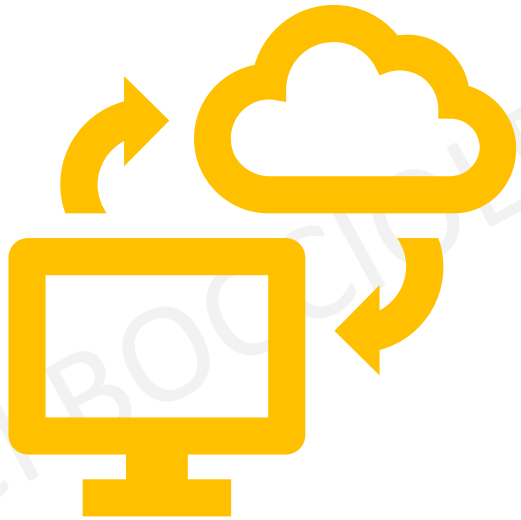




PWA – Notification

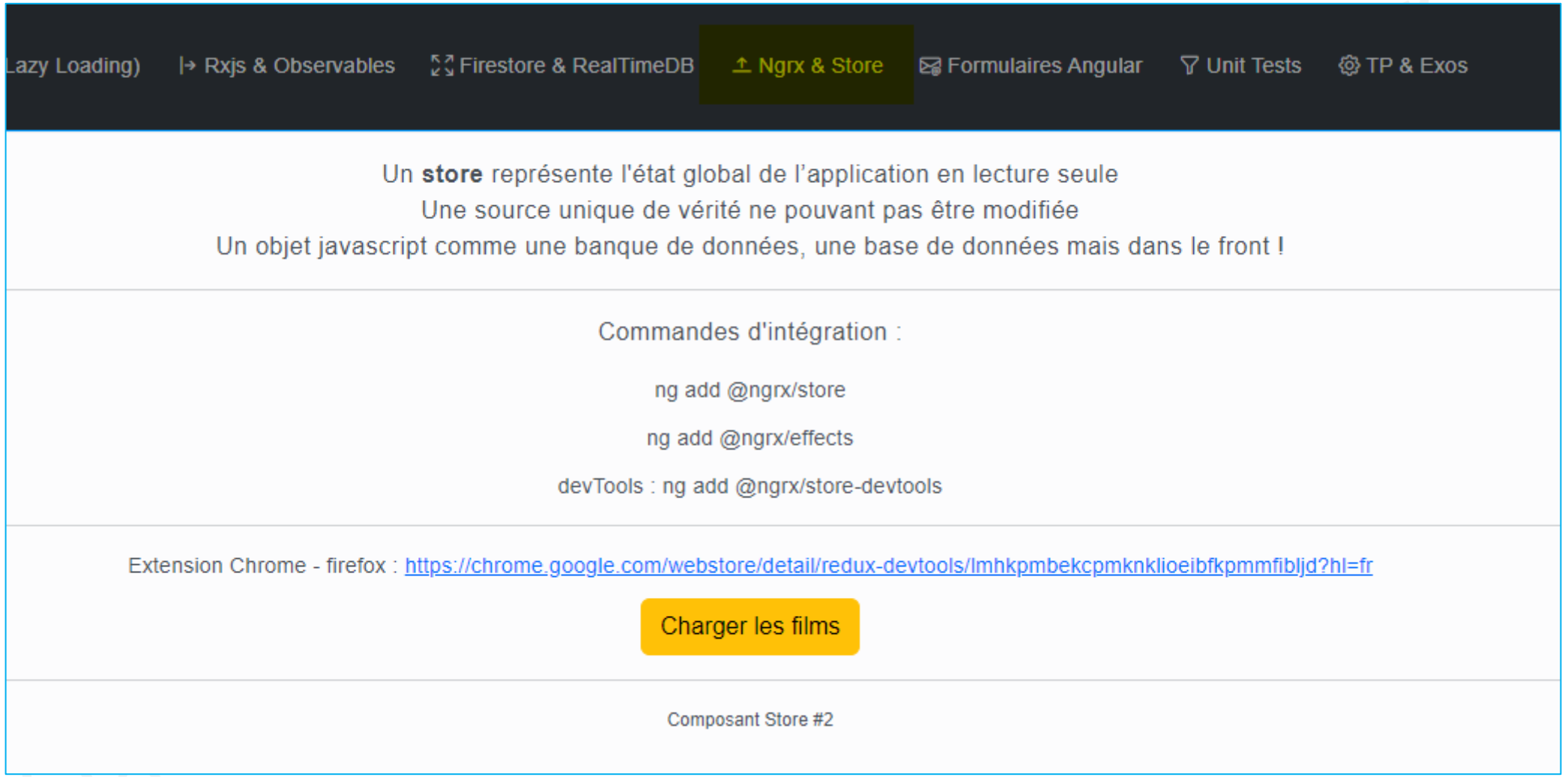


NGRX – La notion de STORE



aNGular – Redux - rXjs

NGRX


Lazy Loading)
Rxjs & Observables
Firestore & RealTimeDB
Ngrx & Store
Formulaires Angular
Unit Tests
TP & Exos
Un **store** représente l'état global de l'application en lecture seule
Une source unique de vérité ne pouvant pas être modifiée
Un objet javascript comme une banque de données, une base de données mais dans le front !
Commandes d'intégration :
ng add @ngrx/store
ng add @ngrx/effects
devTools : ng add @ngrx/store-devtools
Extension Chrome - firefox : https://chrome.google.com/webstore/detail/redux-devtools/lmhkpmbekcpmknklioeibfkpmmfibljd?hl=fr
Charger les films
Composant Store #2

Installation

package.json

```
"@angular/animations": "^16.1.0",
"@angular/cdk": "^16.2.8",
"@angular/common": "^16.1.0",
"@angular/compiler": "^16.1.0",
"@angular/core": "^16.1.0",
"@angular/fire": "^7.6.1",
"@angular/forms": "^16.1.0",
"@angular/material": "^16.2.8",
"@angular/platform-browser": "^16.1.0",
"@angular/platform-browser-dynamic": "^16.1.0",
"@angular/router": "^16.1.0",
"@ngrx/effects": "^16.3.0",
"@ngrx/store": "^16.3.0",
"@ngrx/store-devtools": "^16.3.0",
"bootstrap": "^5.3.0",
"bootstrap-icons": "^1.10.5",
"rxfire": "^6.0.3",
"rxjs": "~7.8.0",
"tslib": "^2.3.0",
"zone.js": "~0.13.0"
},
"devDependencies": {
"@angular-devkit/build-angular": "^16.1.3",
"@angular/cli": "~16.1.3",
"@angular/compiler-cli": "^16.1.0",
"@angular/localize": "^16.1.3",
"@types/jasmine": "~4.3.0",
"jasmine-core": "~4.6.0",
```

app.module.ts

```
import { StoreDevtoolsModule } from '@ngrx/store-devtools';

// import { rootReducer } from './webApp/formation/ngrx-store/reducers/roo
// import { metaReducersX } from './webApp/formation/ngrx-store/reducers/m
// import { appEffects } from './webApp/formation/ngrx-store/effects/effec

@NgModule({
  declarations: [AppComponent],
  imports: [BrowserModule, AppRoutingModule, AccueilModule, BrowserAnimati
    // NGRx - Store
    StoreModule.forRoot(
      {
        // rootReducer: formate le store (modifie)
        // nom à notre state (root): reducer principal
        // root: rootReducer | STATE_NAME: rootReducer
      },
      { // metaReducer
        // metaReducers: metaReducersX
      }
    ),
    EffectsModule.forRoot([
      // appEffects
    ]),
    StoreDevtoolsModule.instrument({ maxAge: 25, logOnly: !isDevMode() })
  ],
  providers: [],
  bootstrap: [AppComponent]
})
```

PROBLÈMES SORTIE CONSOLE DE DÉBOGAGE TERMINAL PORTS

```
✓ Found compatible package version: @ngrx/effects@16.3.0.
✓ Package information loaded.

The package @ngrx/effects@16.3.0 will be installed and executed.
Would you like to proceed? Yes
✓ Packages successfully installed.
UPDATE src/app/app.module.ts (750 bytes)
UPDATE package.json (1735 bytes)
✓ Packages installed successfully.
PS C:\Users\Michel\Desktop\ANY\webApp> ng add @ngrx/store-devtools
i Using package manager: npm
✓ Found compatible package version: @ngrx/store-devtools@16.3.0.
✓ Package information loaded.
```

NGRX pourquoi ?

Lorsqu'il y a beaucoup d'interactions entre composants et composants-enfants,
Lorsque le projet devient très conséquent en termes de développement,
Lorsque le projet grandit et se construit au fur et à mesure …

Le projet est de plus en plus difficile à maintenir.
On peut rencontrer des effets de bords non désirés.
La structure « complexe » du projet rend de plus en plus difficile les opérations d'amélioration et d'enrichissement.

BEST NOUVELLE EDITION

Formation : Angular, maîtriser le Framework Front-End de Google

concepts de développement

4,5 / 5

Angular est le framework javascript de référence de Google. Il utilise tous les standards du Web. Il offre des performances accrues avec une conception modulaire adaptée à la mobilité ainsi qu'une amélioration de la productivité de vos équipes de développement. Angular bénéficie immédiatement d'un écosystème riche et d'une communauté toujours plus grande.

Objectifs pédagogiques

À l'issue de la formation, le participant sera en mesure de :

- Organiser, modulariser et tester ses développements JavaScript

> Formations > Technologies numériques > Technologies Web > Développement Front-End > Formation Angular, développement avancé

BEST

Formation : Angular, développement avancé

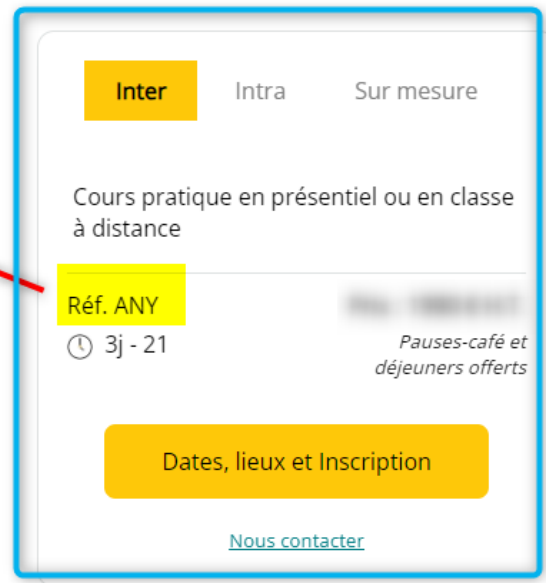
4,3 / 5

Vous découvrirez en profondeur les bonnes pratiques de développement des applications Angular avec les dernières version du framework Angular et le moteur de rendu optimisé Ivy. Vous apprendrez à maitriser le FormBuilder pour des formulaires réactifs ainsi que la génération de tests unitaires.

Objectifs pédagogiques

À l'issue de la formation, le participant sera en mesure de :

- Savoir utiliser les décorateurs Angular
- Architecturer les applications web complexes
- Intégrer les outils de documentation et les tests unitaires
- Développer et intégrer des librairies de composants

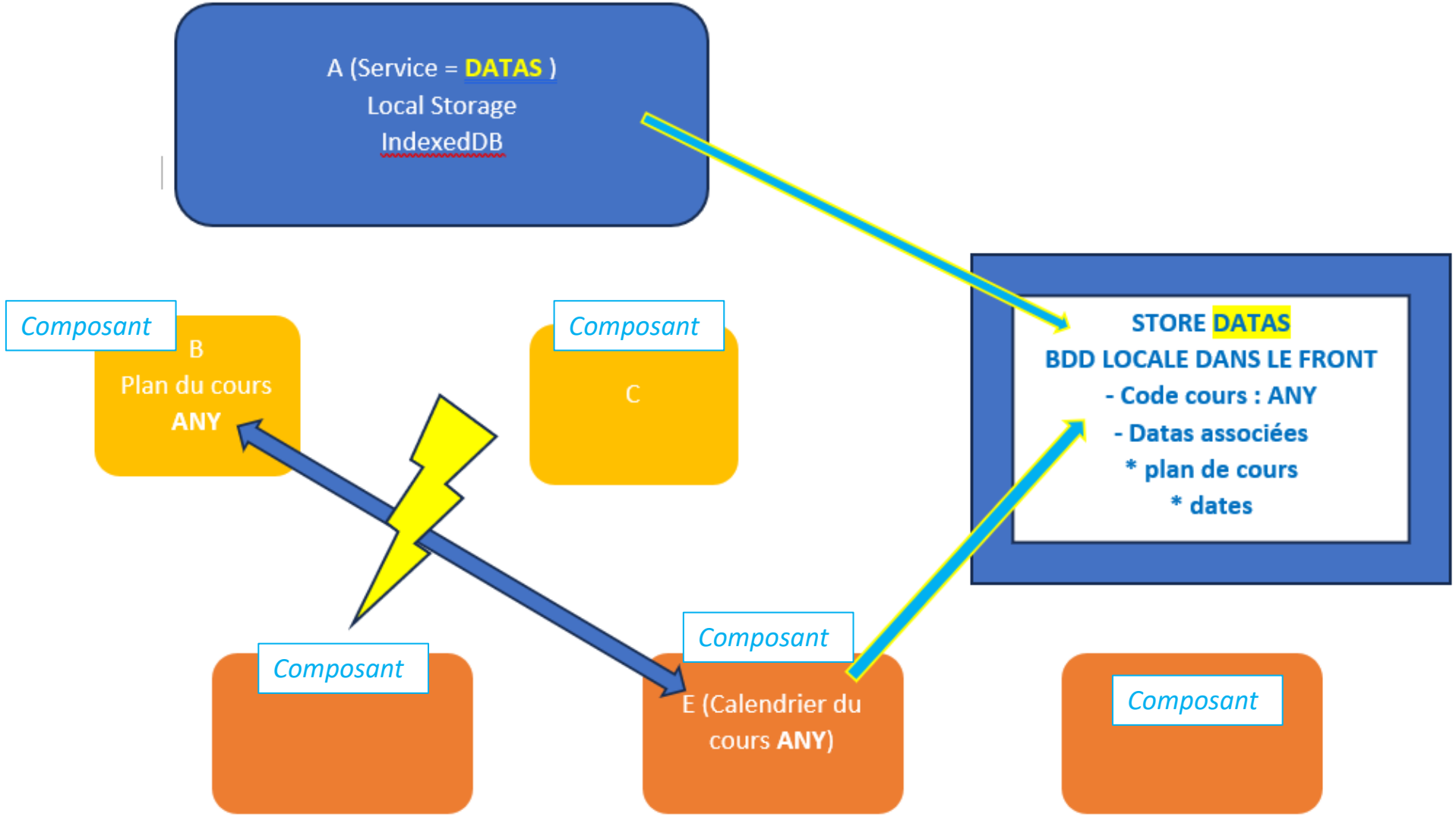


NGRX la solution

Au lieu de traverser tout l'héritage des Input et des Output des composants parents-enfants, le composant peut s'inscrire auprès du Store et dispatcher une Action.
C'est-à-dire demander quelque chose ou demander à faire quelque chose.

L'accès est direct sans passer par la structure parfois « trop rigide » des web-components imbriqués.

NGRX

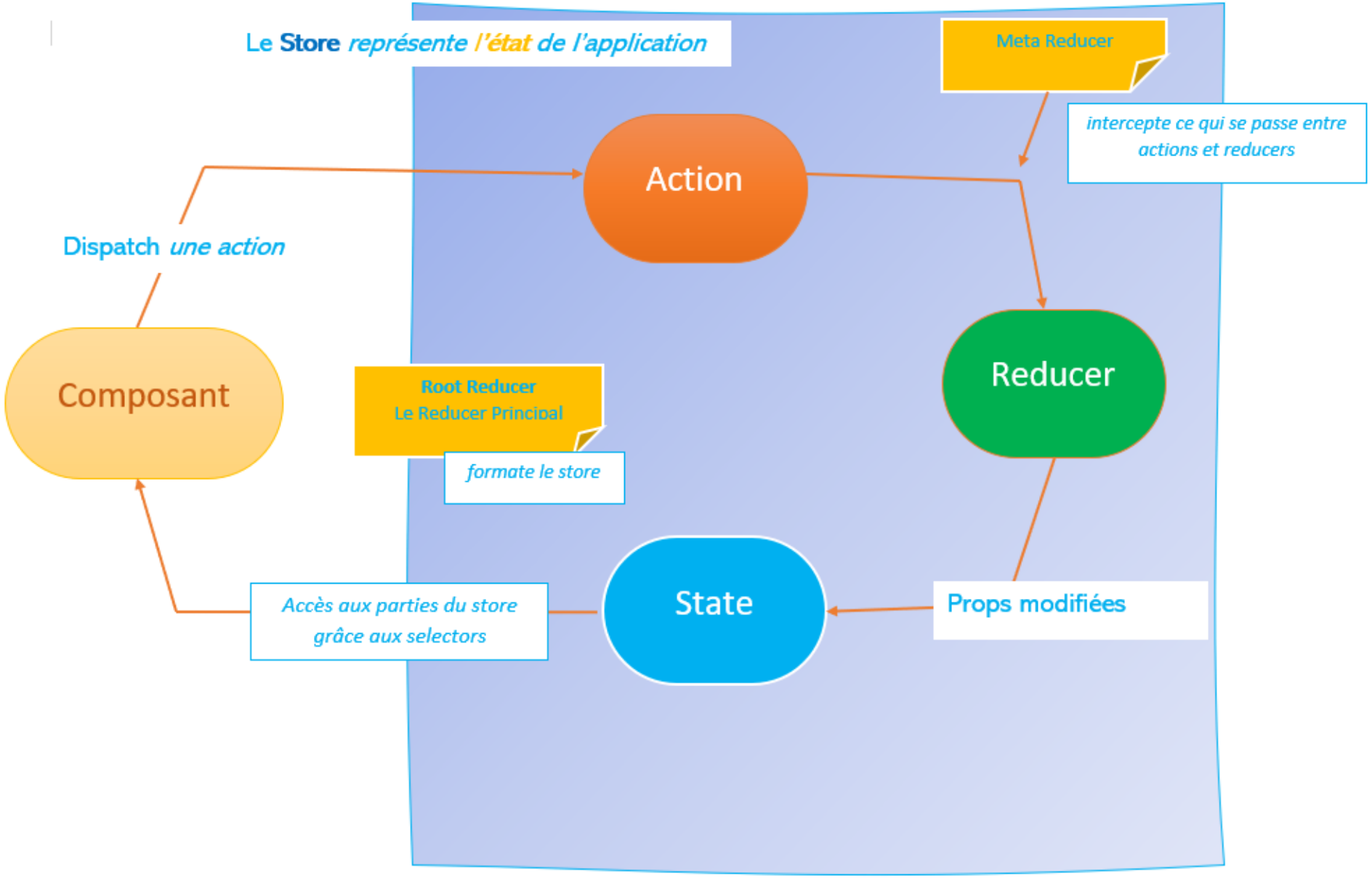


1 Composant s'inscrit auprès du STORE !
Dispatcher des actions (demande à faire qq chose avec les datas enregistrées dans le store)

NGRX Principe de fonctionnement

1- Le composant dispatche une action
2- Celle-ci doit être listée dans le store
3- Un reducer réceptionne l'action « dispatchée » et se charge de son éxécution
4- Le reducer modifie le state de l'application
5- Le composant dispatchant l'action demande à accéder à une partie du store grâce aux selectors
6- Un nouveau rendu HTML est proposé à tous les composants ayant accès à ce selector !

NGRX

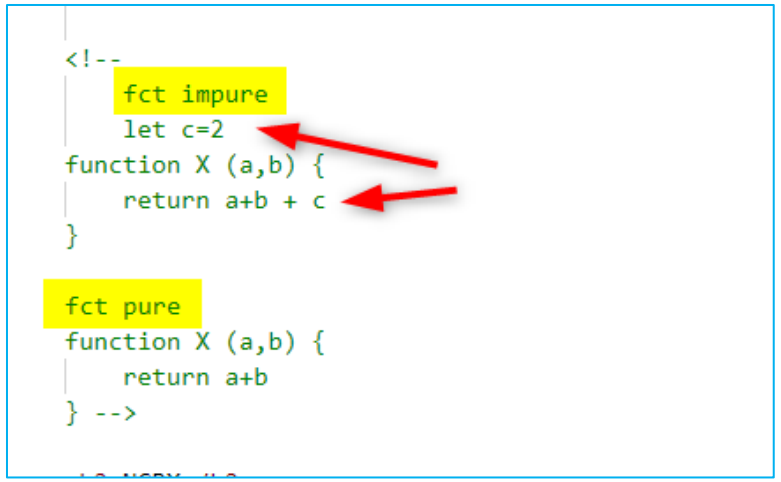

Le Store représente l'état de l'application
Meta Reducer
intercepte ce qui se passe entre actions et reducers
Action
Dispatch une action
Composant
Root Reducer
Le Reducer Principal
formate le store
Reducer
Accès aux parties du store grâce aux selectors
State
Props modifiées

NGRX SideEffect - BackEnd

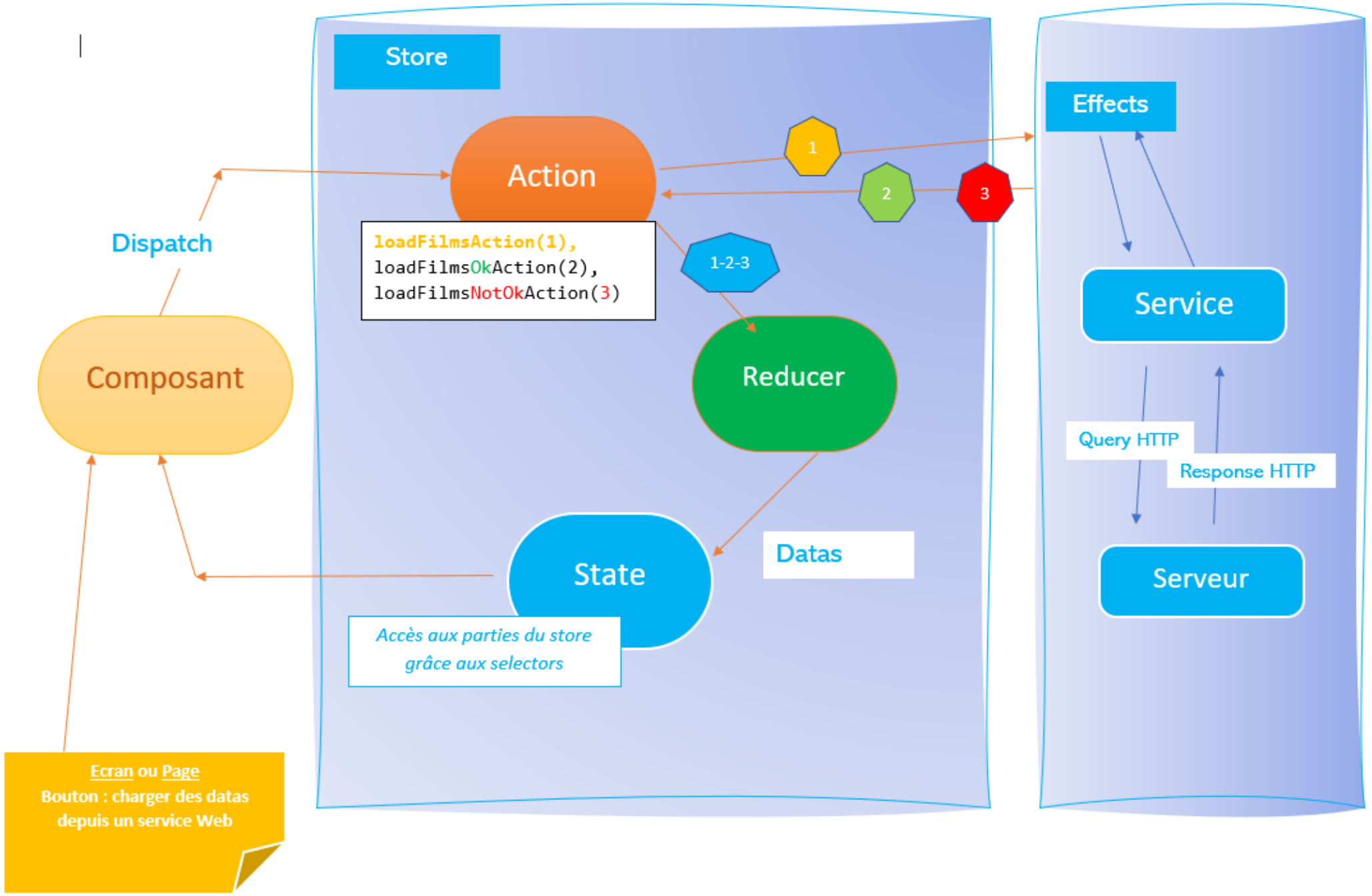
Les sideEffects permettent de connecter le back via des services web à notre application Angular-STORE.

Le schéma peut « paraître » complexe au 1er abord mais il est logique dans la conception de la « fonction pure » du Reducer !

Pour rappel, une fonction pure ne peut retourner que les paramètres reçus en appel (en entrée)



NGRX


Store
Action
loadFilmsAction(1),
loadFilmsOkAction(2),
loadFilmsNotOkAction(3)
1
2
3
1-2-3
Reducer
State
Accès aux parties du store
grâce aux selectors
Datas
Dispatch
Composant
Ecran ou Page
Bouton : charger des datas
depuis un service Web
Effects
Service
Query HTTP
Response HTTP
Serveur

Root Reducer

root-reducers.ts

Formation-ANY-07-2023 > TP07-NGRX > src > app > webApp > formation > ngrx-store > reducers > root-reducers.ts

```ts
import { createReducer, on } from "@ngrx/store";
import { changenameAction, initAction, loadFilmsAction, loadFilmsNotOkActi
import { Films } from "src/app/sharedModels/models/class/films";

// const INITIAL_STATE = {}; // à l'init => en prod
// ------------------- OPTIMISATION ------------------
// export const STATE_NAME = 'appli';
export interface rootState {
    appName: string;
    actor: {
        name: string;
        isJedi: boolean;
    }
    film?: Films;
    films?: Films[],
    loaded?: boolean
}
//-----------------------------------------------------
const INITIAL_STATE: rootState = {
    appName: 'Formation Angular NGRX',
    actor: {
        name: '',
        isJedi: false
    }
};
export const rootReducer = createReducer(
    // state initial
    INITIAL_STATE,
    // ons => 'on' sur une action
    // définir les actions invoquées
    on(
        initAction,
        // tâche ou le job que va faire le reducer ...
        // sur les props du store représenté par un state
        (state) => {
            // le reducer est capable de prendre qu'une partie du store(st
            return {
                ...state, // ... spred operators ES2015 (déstructure notre
                actor: {
                    ...state.actor, // copie de la valeur de actor
                    isJedi: true // param modifié
                }
            }
        }
    ),
```

root-reducers.ts

TP07-NGRX > src > app > webApp > formation > ngrx-store > reducers > root-reducers.ts

```ts
    // autre on
    on(
        changenameAction,
        (state, props) => {
            return {
                ...state,
                actor: {
                    ...state.actor,
                    name: props.paramNameActionCOMP
                }
            }
        }
    ),
    // autre on
    on(loadFilmsAction,
        (state) => {
            return {
                ...state,
                loaded: false
            }
        }
    ),
    // autre on
    on(loadFilmsOkAction,
        (state, props) => {
            return {
                ...state,
                films:props.films,
                loaded: true
            }
        }
    ),
    // autre on
    on(loadFilmsNotOkAction,
        (state, props) => {
            return {
                ...state,
                loaded: true,
                KO:true,
                erreurDesc:props.err
            }
        }
    )
);
```

Meta Reducer

root-reducers.ts | selectors.ts | actions.ts | effects.ts | meta-reducers.ts

Formation-ANY-07-2023 > TP07-NGRX > src > app > webApp > formation > ngrx-store > reducers > meta-reducers.ts > log > <fun

```ts
// affiche en console les différentes étapes
// qui se déroulent dans le store entre les actions et les reducers…

import { ActionReducer, MetaReducer } from "@ngrx/store"

// actionReducer représente la combinaison de l'action et du reducer
// nous donne le résultat de ce qui se passe entre l'action et le reducer

const log = (paramReducer:ActionReducer<any>) => {

    return(state:any, action:any) => {

        const currentState = paramReducer(state, action);

        console.groupCollapsed(action.type);

            console.log('Etat précédent : ', state);
            console.warn('Action : ', action);
            console.log('Etat suivant : ', currentState);

        console.groupEnd();

        return currentState;
    }
}
// ------------ export le metaReducer ---
export const metaReducersX: MetaReducer[] = [log];
```

selectors

root-reducers.ts | selectors.ts

tion-ANY-07-2023 > TP07-NGRX > src > app > webApp > formation > ngrx-store > selectors > sel

```ts
// on va créer des parties du store qu'on veut select
// pour le smettre à dispo des composants

import { createSelector } from "@ngrx/store";

const selectRootState =  (state:any) => {
    return state.root;
}

export const selectorGetActor =  createSelector(
    selectRootState,
    (state:any) => state.actor
)

export const selectorGetFilms = createSelector(
    // on prend une partie du state
    selectRootState, // a partir de quoi on va filtrer ce que l
    (state:any) => state.films
);
export const selectorGetLoaded = createSelector(
    selectRootState,
    (state:any) => state.loaded
);
export const selectorGetKO = createSelector(
    selectRootState,
    (state: any) => state.KO
  );
```

selectors

root-reducers.ts | selectors.ts

tion-ANY-07-2023 > TP07-NGRX > src > app > webApp > formation > ngrx-store > selectors > sel

```ts
// on va créer des parties du store qu'on veut select
// pour le smettre à dispo des composants

import { createSelector } from "@ngrx/store";

const selectRootState =  (state:any) => {
    return state.root;
}

export const selectorGetActor =  createSelector(
    selectRootState,
    (state:any) => state.actor
)

export const selectorGetFilms = createSelector(
    // on prend une partie du state
    selectRootState, // a partir de quoi on va filtrer ce que l
    (state:any) => state.films
);
export const selectorGetLoaded = createSelector(
    selectRootState,
    (state:any) => state.loaded
);
export const selectorGetKO = createSelector(
    selectRootState,
    (state: any) => state.KO
  );
```

actions

root-reducers.ts | selectors.ts | actions.ts

Formation-ANY-07-2023 > TP07-NGRX > src > app > webApp > formation > ngrx-store > actions > act

```ts
import { createAction, props } from "@ngrx/store";
import { Films } from "src/app/sharedModels/models/class/films";

export const initAction =  createAction(
    // nom de l'action
    '[ROOT] Init Action'
);

export const changenameAction =  createAction(
    // nom de l'action
    '[ROOT] Change Name Actor Action',
    // props à faire ajouter par le reducer dans le state
    props<{paramNameActionCOMP:string}>()
);

export const loadFilmsAction =  createAction(
    // nom de l'action
    '[FILMS] Ask Load Films Action'
);

export const loadFilmsOkAction = createAction(
    '[FILMS DATAS] Load films OK',
    props<{films:Films[]}>()
);

export const loadFilmsNotOkAction = createAction(
    '[FILMS DATAS] Load films Not OK',
    props<{err:any}>()
);
```

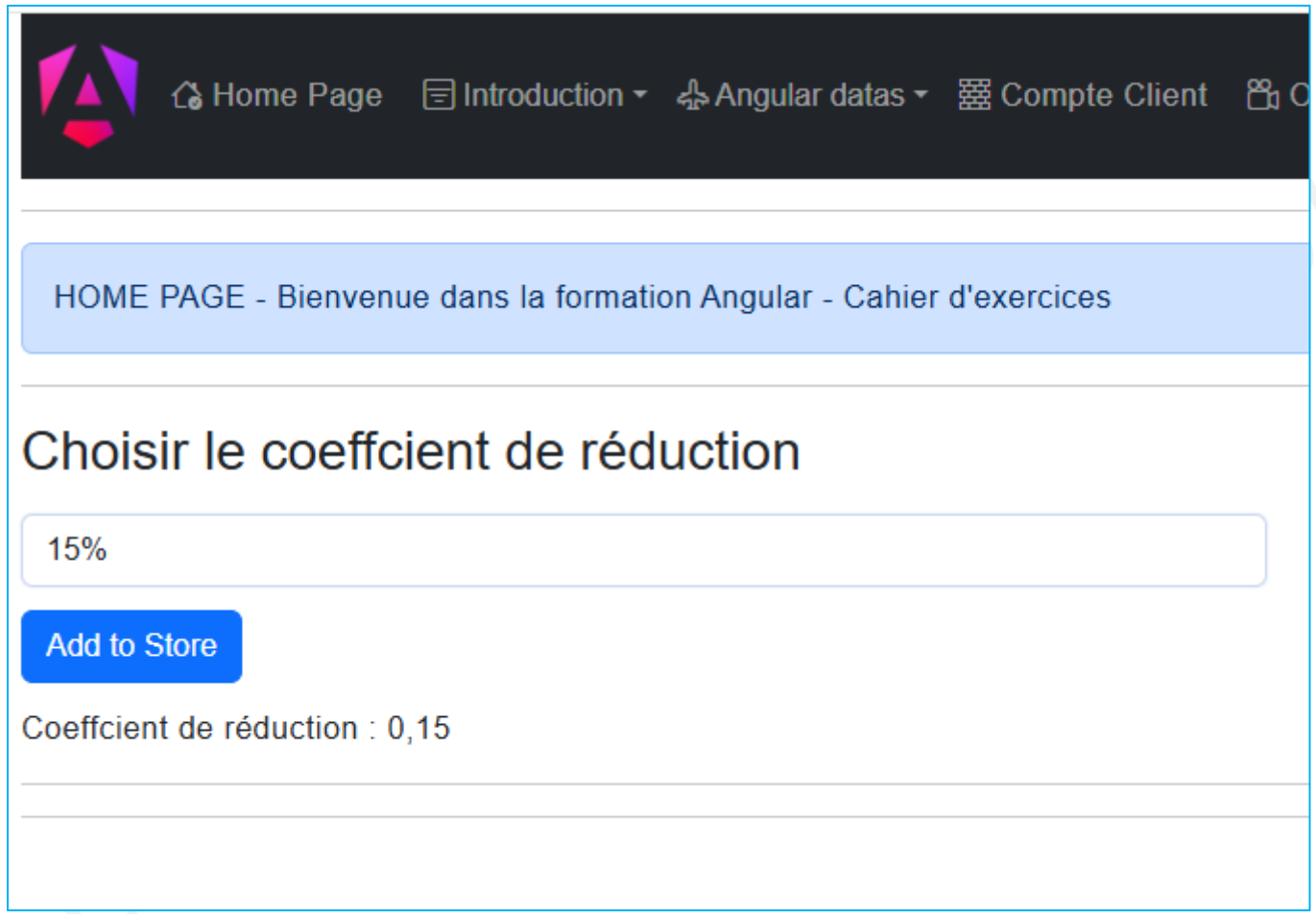
effects

root-reducers.ts | selectors.ts | actions.ts | effects.ts

Formation-ANY-07-2023 > TP07-NGRX > src > app > webApp > formation > ngrx-store > effects > effects.ts > appEffects > loadFilms$

```
import * as compActions from '../actions/actions';
import { Films } from "src/app/sharedModels/models/class/films";

@Injectable()

export class appEffects {


    constructor(private _serviceEffects: NgrxEffectsService, private _actions: Actions) { }

    // création d'un Observable

    loadFilms$: Observable<any> = createEffect(
        () => {
            // détection des actions invoquées
            return this._actions.pipe(
                tap(
                    (action) => {
                        console.log('******** Actions : ', action);
                    }
                ),
                ofType(
                    // ofType peut filtrer la liste de ce qu'on lui passe
                    compActions.loadFilmsAction
                ),
                mergeMap(
                    action => this._serviceEffects.getFilms$()
                    // on récupère les datas : films
                    .pipe(
                        map(
                            (films:Films[]) => {
                                return compActions.loadFilmsOkAction({films})
                                // c'est l'effect qui passe des datas à l'action
                            }
                        ),
                        catchError(
                            (err) => {
                                // console.log(err);
                                return of(compActions.loadFilmsNotOkAction({err}));

                            }
                        )
                    )
                )
            )
        }
```

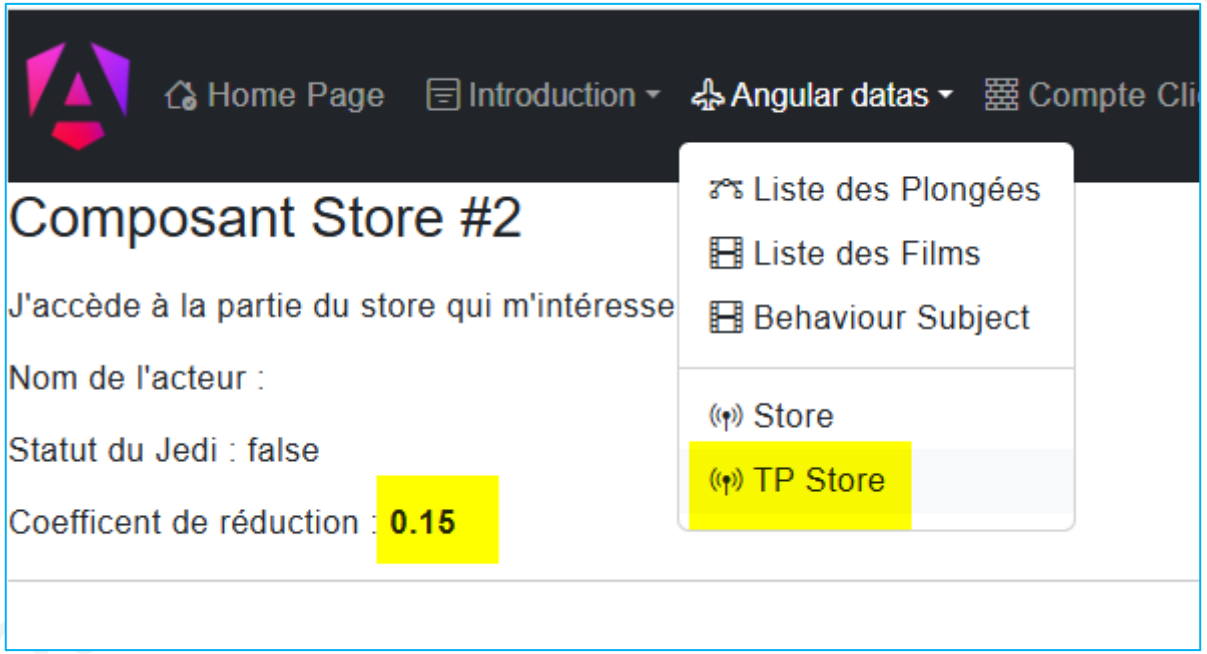
NGRX TP



Ajouter sur la home-page un composant qui va permettre de valider un coefficient de réduction dans le store.

Il faudra ensuite le réexploiter et l'afficher , soit dans la même vue dans un autre composant soit dans le composant TP-Store.

NGRX TP



NGRX TP

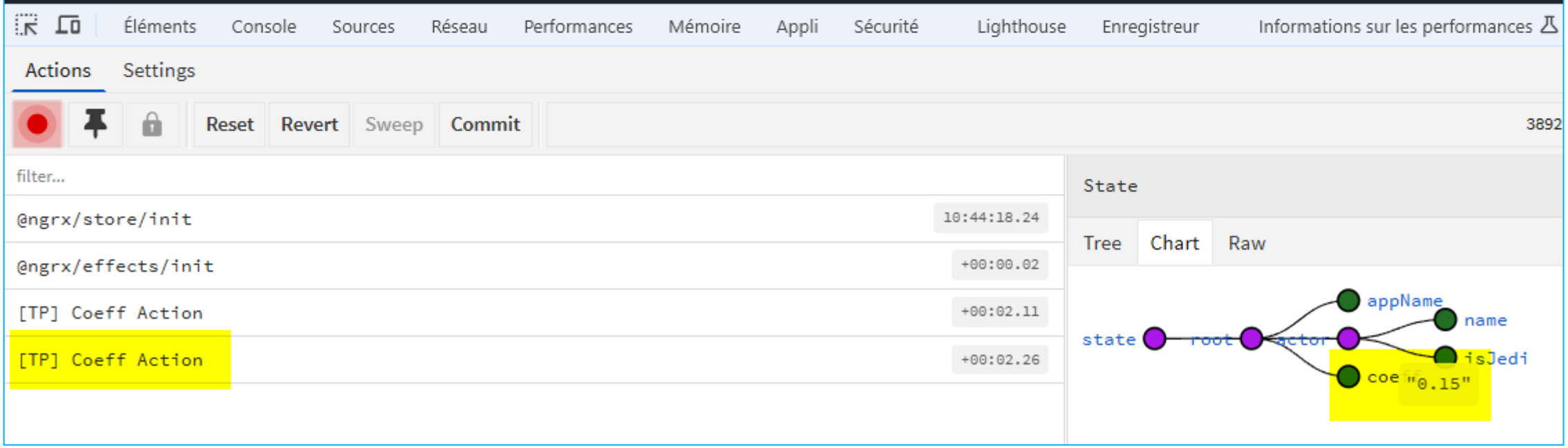
Choisir le coeffcient de réduction

15%

Add to Store

Coeffcient de réduction : 0,15

THE FOOTER - Formation Angular - *Michel Bocciolesi - Orsys Formation*



NGRX TP

```
<div>

    <h5>Choisir le coeffcient de réduction</h5>
    <select class="form-control" style="width:50%" [(ngModel)]="coeff" (change)="addToStore()">

        <option value="">Choisir le coefficient de réduction</option>
        <option value="0.1">10%</option>
        <option value="0.15">15%</option>
        <option value="0.2">20%</option>

    </select>
    <p></p>

    <button class="btn btn-primary" (click)="addToStore()">Add to Store</button>
    <p></p>
    <p>Coeffcient de réduction : {{ coeff   | number:'1.0-2'  }}</p>
</div>
```

NGRX TP

```
//-----------------------------------
  public coeff: number = 0;
_store = inject(Store);

public addToStore() {
  this._store.dispatch(
    coeffAction({
      coeffStore: this.coeff
    })
  );
}
//-----------------------------------
```

```
// TP

export const coeffAction =  createAction(
    // nom de l'action
    '[TP] Coeff Action',
    // props à faire ajouter par le reducer dans le state
    props<{coeffStore:number}>()
);
```

```
  //   TP

export const selectorGetCoeff =  createSelector(
    selectRootState,
    (state:any) => state.coeff
)
```

```
    ,
    // TP COEFF
    // un autre ON sur une autre action
    on(
        coeffAction,
        (state, props) => {
            return {
                ...state,
                coeff: props.coeffStore

            }
        }
    )
)
```

NGRX TP

comp-store2.component.html

src > app > webApp > ngrx-store > components > comp-store2 > comp-store2.component.html > h5

Go to component

```
1  <h5>Composant Store #2</h5>
2  <p>J'accède à la partie du store qui m'intéresse ...</p>
3  <p>Nom de l'acteur : {{(actor$ | async).name}} </p>
4  <p>Statut du Jedi : {{(actor$ | async).isJedi}}</p>
5  <p>Coefficent de réduction : <strong> {{coeff$ | async  }}</strong></p>
6
```

comp-store2.component.ts

src > app > webApp > ngrx-store > components > comp-store2 > comp-store2.component

```
14  export class CompStore2Component {
16    public actor$:Observable<any> = new Observable()
17    public coeff$: Observable<number> = new Observable();
18    constructor(private _store: Store) { }
19
20    ngOnInit() {
21  // ----------Avec les selectors :-) ---------
22  this.actor$ = this._store
23  .pipe(
24    select(selectorGetActor)
25  );
26  this.coeff$ = this._store.pipe(
27    select(selectorGetCoeff)
28  )
```

Formation ANGULAR

A bientôt.