ORSYS formation

Formation ANGULAR

Animé par Michel BOCCIOLESI

Table des matières

Framework Orienté ou Progressif

Framework ou librairies ?

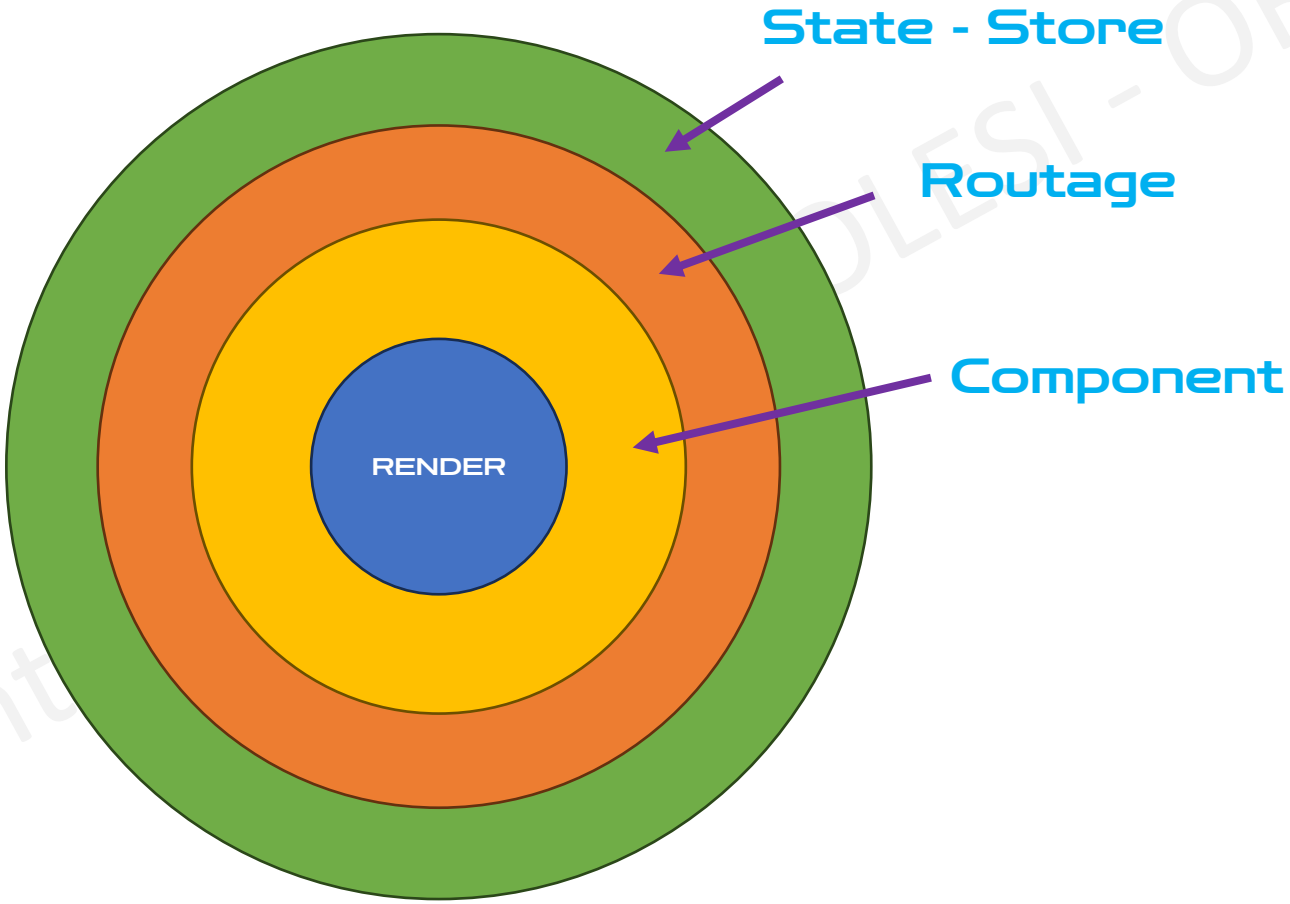
Dans le monde du développement Front, il existe plusieurs catégories de frameworks :

- 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

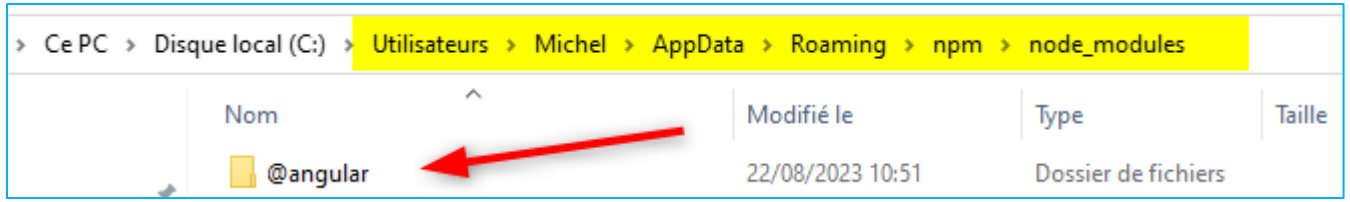
Installation et configuration

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

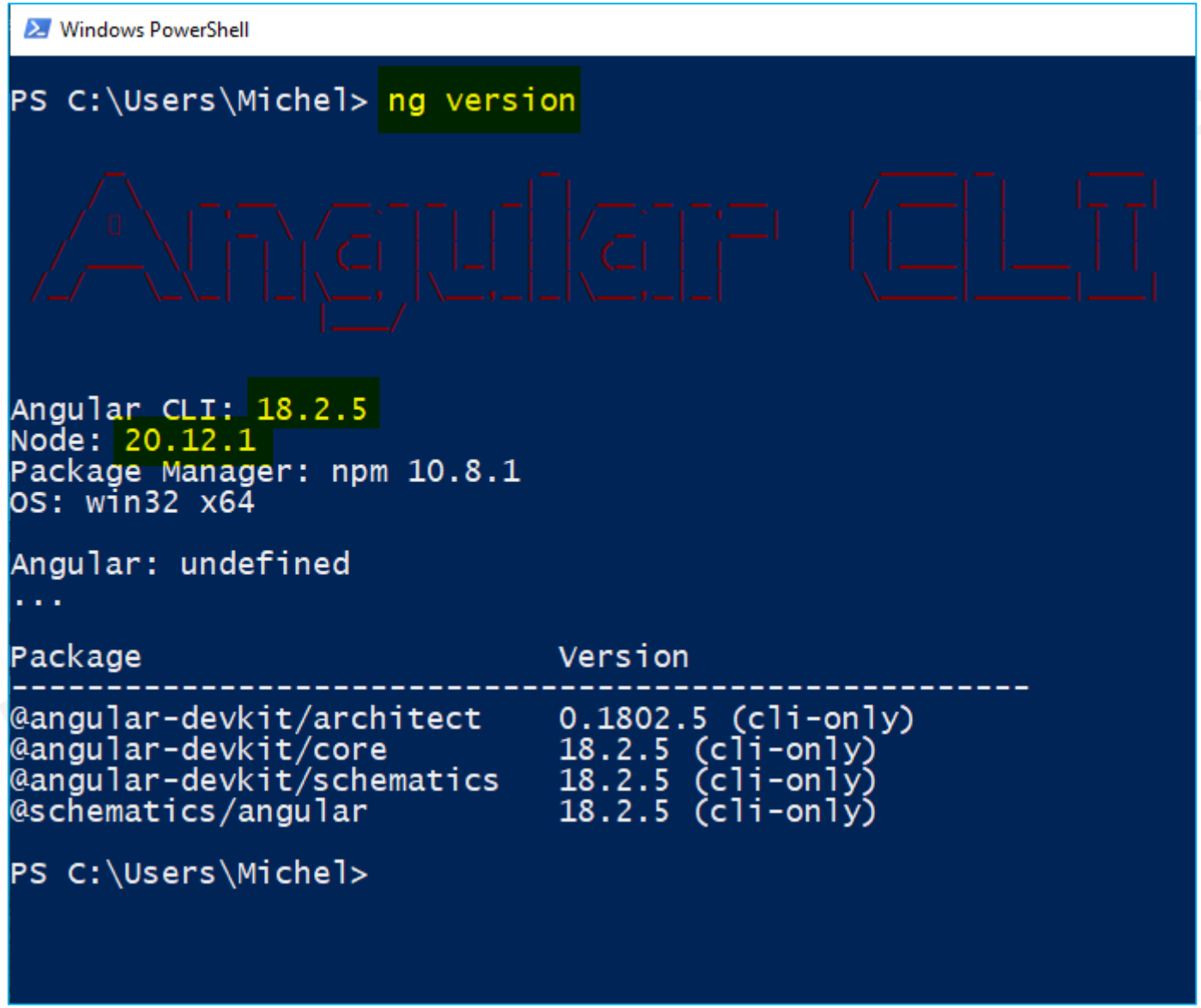
Tous ces frameworks utilisent NODEJS (2009), un serveur javascript développé par Google.

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>
```

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 ?

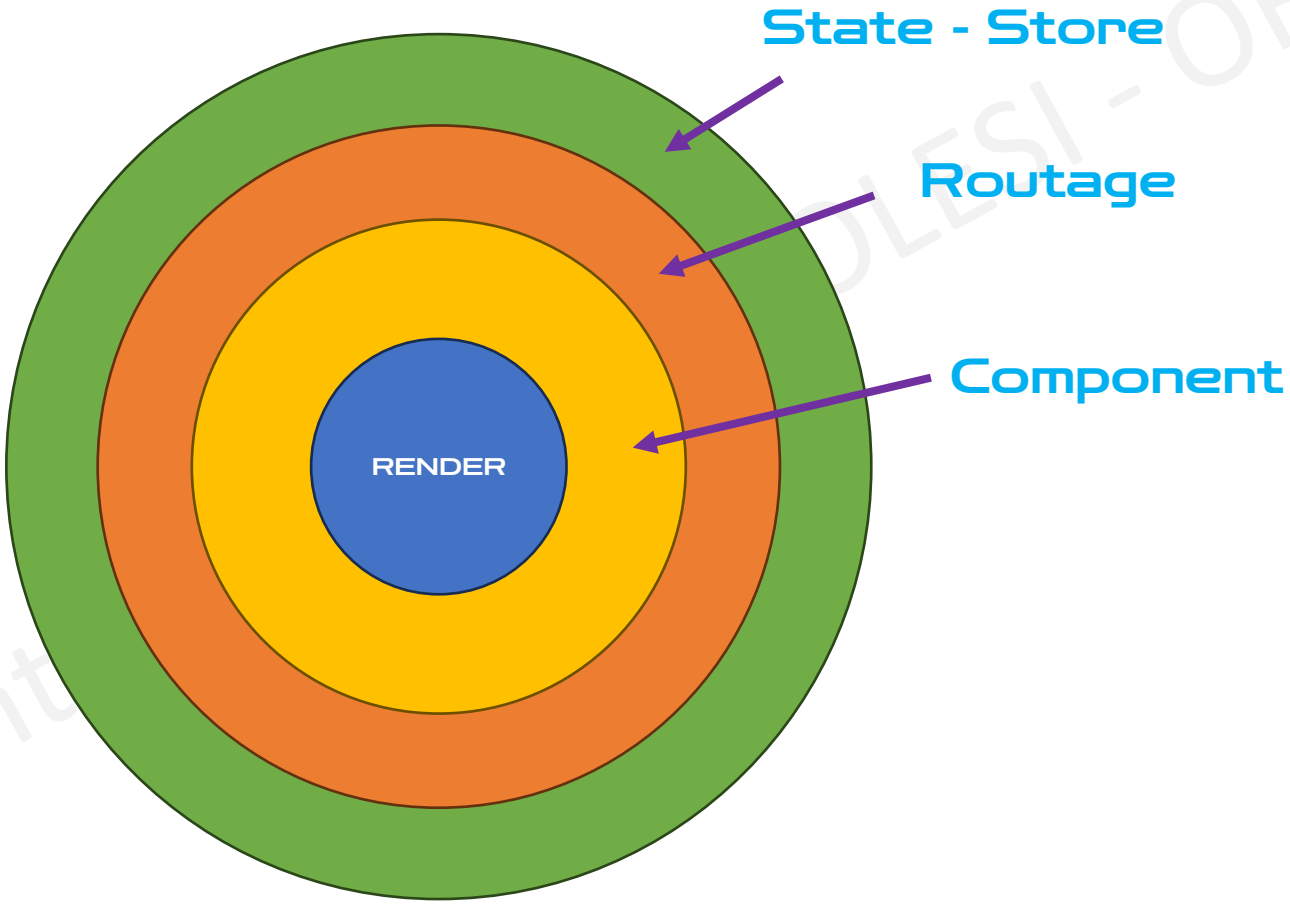
Dans le monde du développement Front, il existe plusieurs catégories de frameworks :

- 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

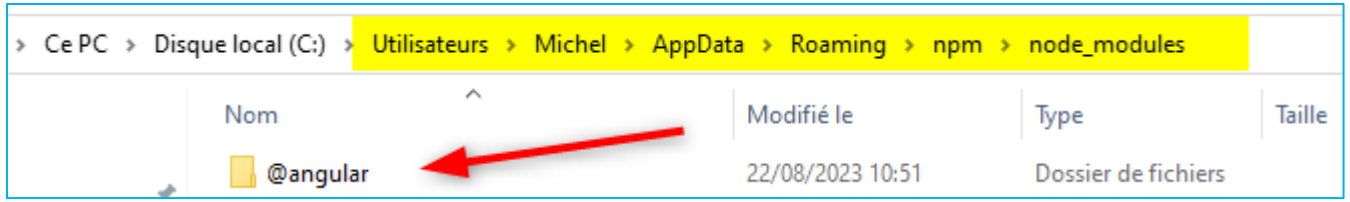
Installation et configuration

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

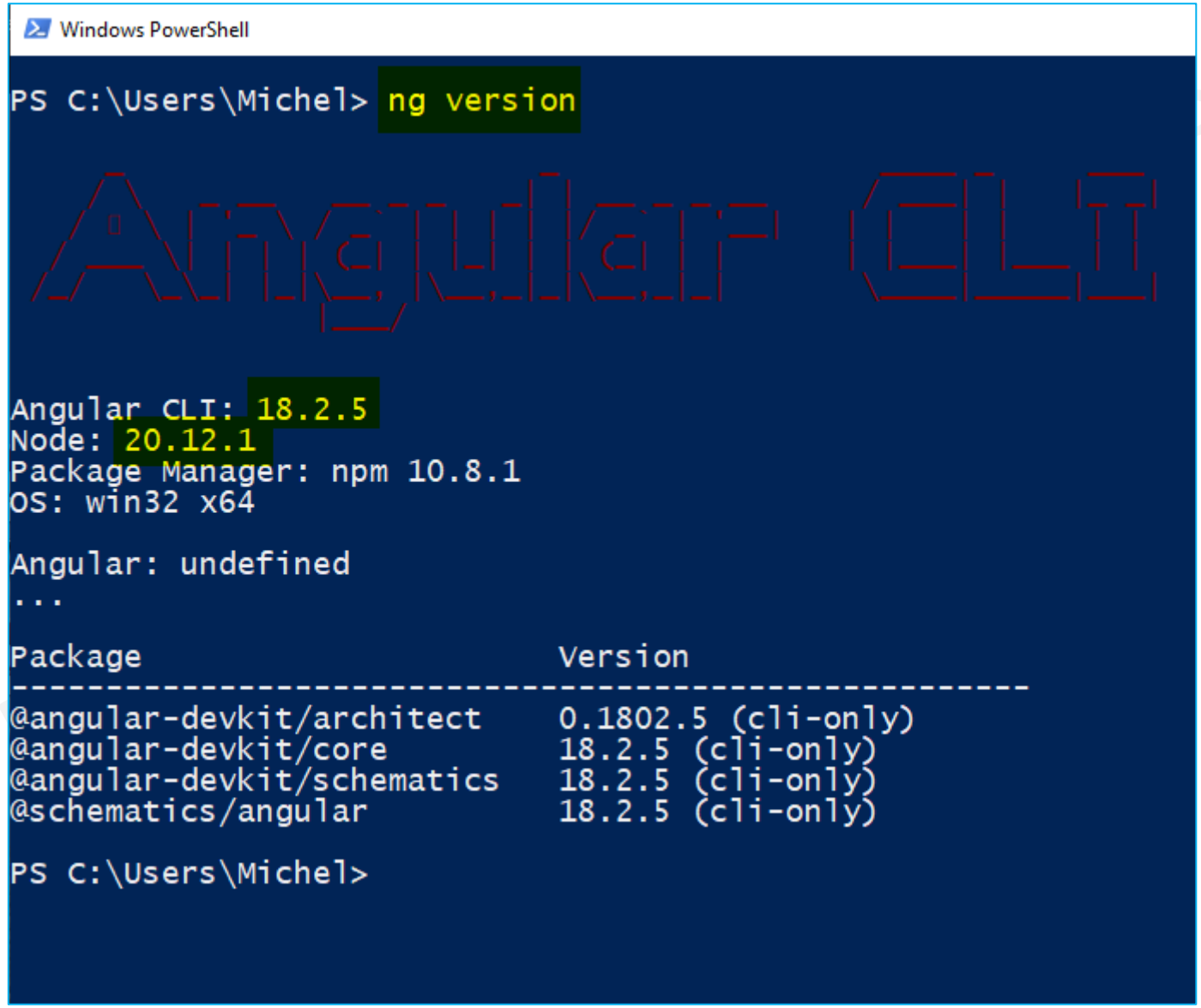
Tous ces frameworks utilisent NODEJS (2009), un serveur javascript développé par Google.

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
```

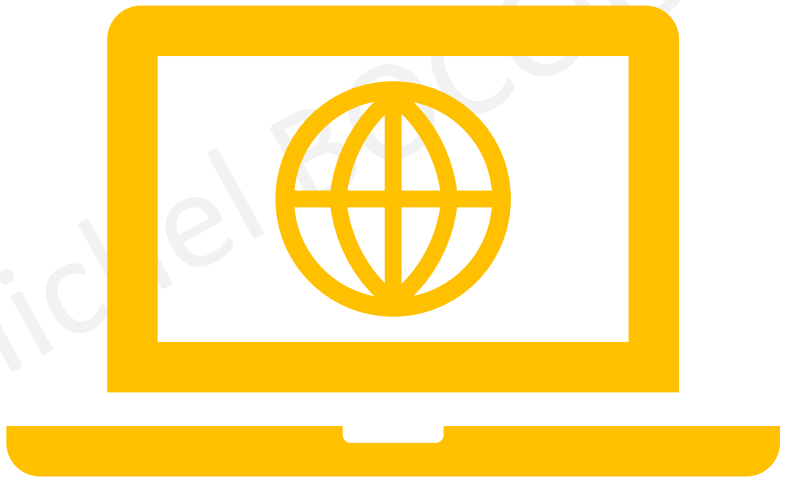
```
PS C:\Users\Michel> ng new projet-standalone-ng-18
? Which stylesheet format would you like to use? CSS                [ https://developer.mozilla.org/docs/Web/CSS
? Do you want to enable Server-Side Rendering (SSR) and Static Site Generation (SSG/Prerendering)? no
```

```
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>
```

1ER PROJET ANGULAR

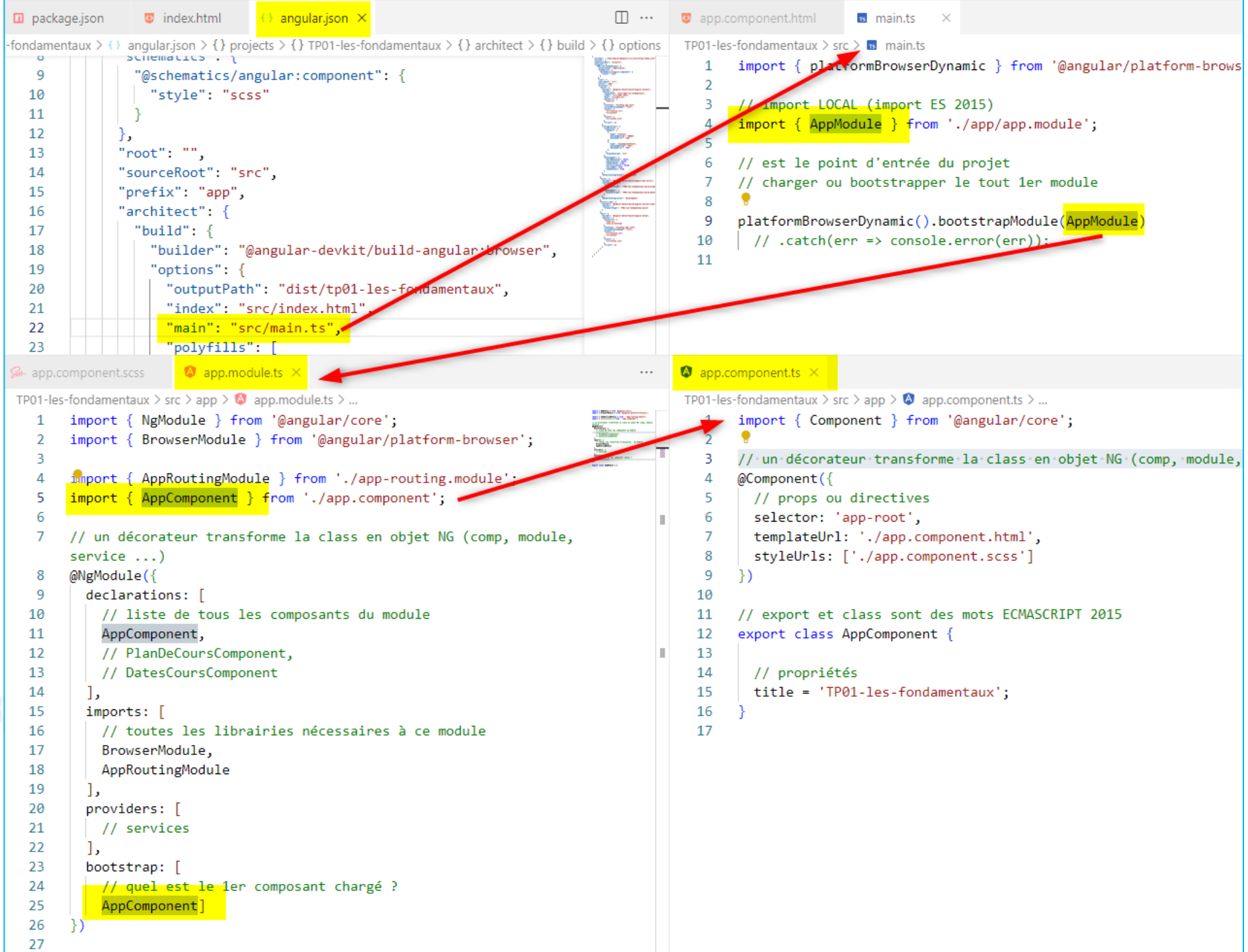


Mode Modules *

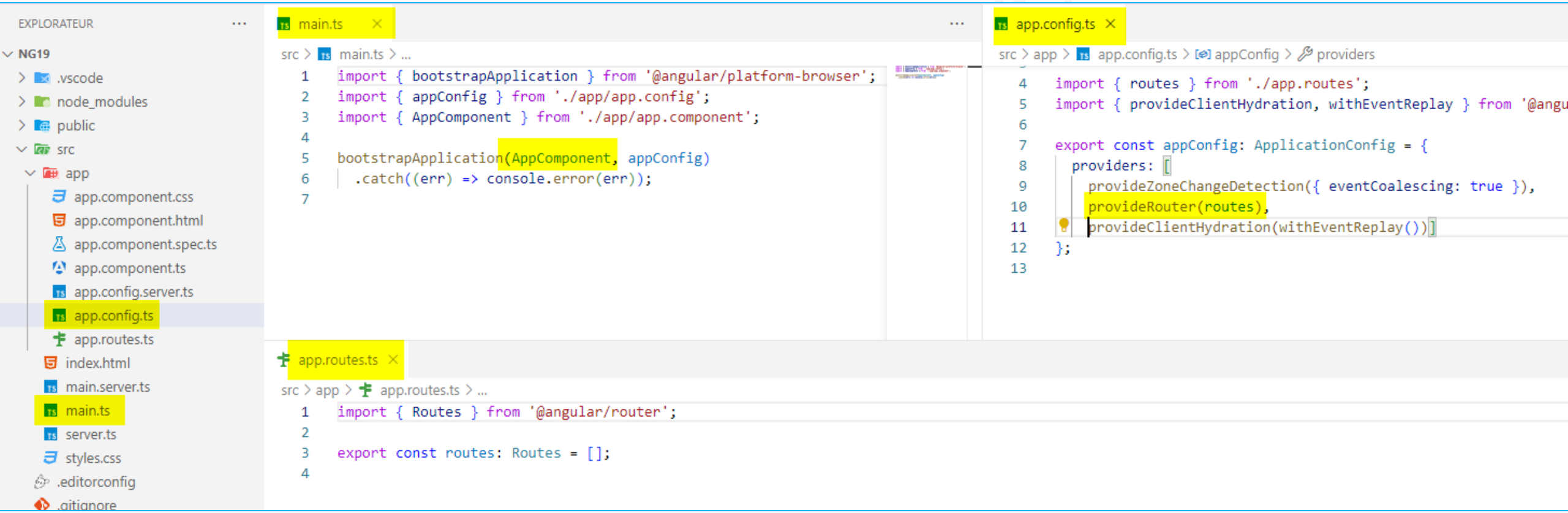
** Compréhension des anciens projets Angular*

** Migration des projets déjà développés et créés jusqu'à Angular 17*

Prise en main
de l'architecture
d'un projet Angular **avec Modules**



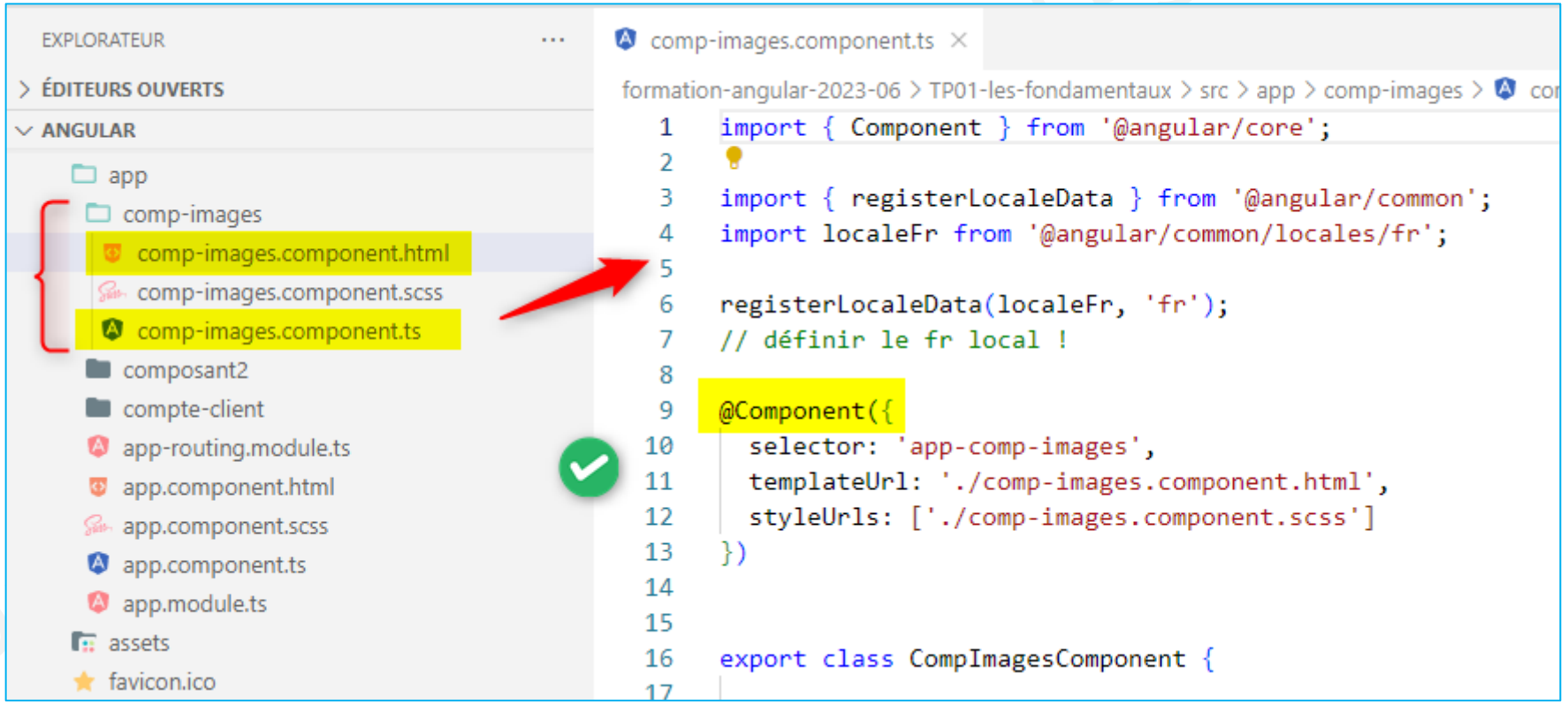
Prise en main
de l'architecture
d'un projet Angular en **standalone components**



Intérêts du Framework Angular

Les web-components (composants web)

1 composant est constitué de 3 fichiers : TS, la vue HTML et le fichier de style CSS*



Intérêts du Framework Angular

Les web-components (composants web)

le composant web permet de travailler une partie de la page globale index.html

tant au niveau des données (le TS) que de la vue HTML+CSS*

app.component.ts

rmation-angular-2023-06 > TP01-les-fondamentaux > src > app > app.component.ts > ...

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

// un décorateur : son rôle : transformer la
classe en ... (composant)

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.scss']
})

// une classe (ES2015) exportée
export class AppComponent {

  // 1- déf des propriétés
  public title: string = 'Angular';
  public version: number;

  // 2- constructeur de classe
  constructor() {
    this.version = 16;
  }
}
```

app.component.html

formation-angular-2023-06 > TP01-les-fondamentaux > src > app > app.component.html > ...

```html

<!-- string interpolation de variables {{ }} -->
<!-- Binding : le TS est lié à la vue HTML(template) -->
<!-- Single way Binding: TS => HTML -->
<h1>Formation {{title}} {{version}}</h1>

    <!-- composant enfant de AppComponent -->
    <app-composant2></app-composant2>

    <hr>

    <app-comp-images></app-comp-images>

    <hr>
    <h2>Espace Client ...</h2>
    <app-landing-page></app-landing-page>
```

Mode Modules *
Compréhension des anciens projets Angular

Composants et modules

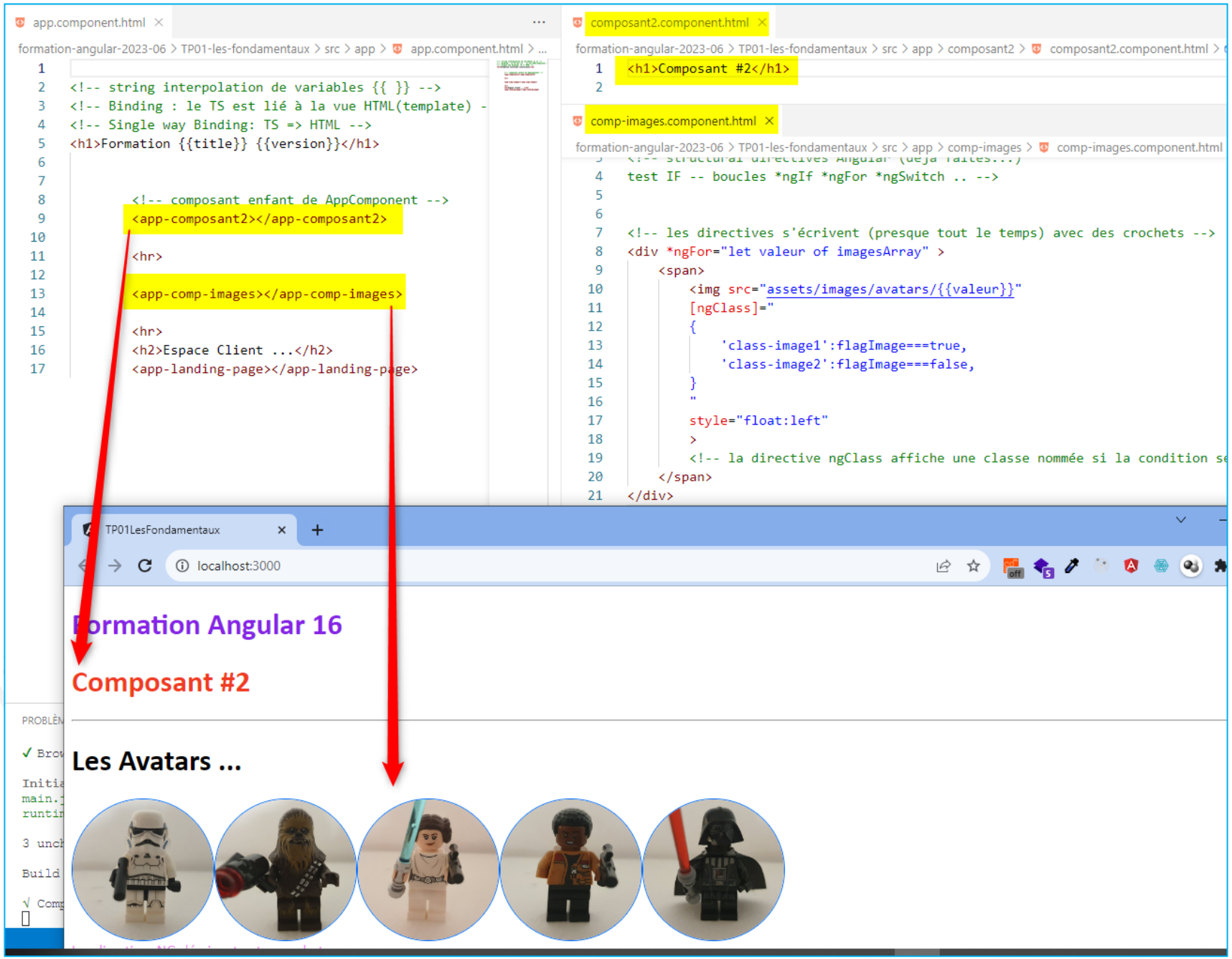
Création de composants : composant2 – comp-images

- Le sélecteur de composant est placé dans le template HTML d'un composant « parent »
- Le composant appelé par le sélecteur devient le composant enfant.

Un module « structurel » ou « fonctionnel » déclare ses composants et les exporte afin que leurs contenus soient exploitables dans les vues de composants d'autres modules. (import-export)

Ressources disponibles :
https://dev.webjs.fr/1-Angular/ressources-formation/

Composants
parents et enfants



Directives
Binding
Pipes

```
1  <h1>{{title | titlecase}}</h1>
2
3  <!-- structural directives Angular (déjà faites...)
4  test IF -- boucles *ngIf *ngFor *ngSwitch .. -->
5
6
7  <!-- les directives s'écrivent (presque tout le temps) avec des crochets -->
8  <div *ngFor="let valeur of imagesArray" >
9      <span>
10         <img src="assets/images/avatars/{{valeur}}"
11         [ngClass]="
12         {
13             'class-image1':flagImage===true,
14             'class-image2':flagImage===false,
15         }
16         "
17         style="float:left"
18         >
19         <!-- la directive ngClass affiche une classe nommée si la condition se réalise -->
20     </span>
21 </div>
22
23 <!-- *ngIf : affiche (ou pas) ce qui est contenu dans l'élément en fonction d'un paramètre-->
24 <div *ngIf="flag===true">
25     <p>{{formation}}</p>
26 </div>
27
28 <!-- *ngSwitchCase ==> gère plusieurs tests (conditions) -->
29 <div [ngSwitch]="codeCours">
30
31     <p *ngSwitchCase="'ng'">Formation Angular</p>
32     <p *ngSwitchCase="'react'">Formation React</p>
33     <p *ngSwitchDefault>Formation non trouvée.</p>
34
35 </div>
36
37 <p>{{maDate | date:'dd/M/Y H:m:s'}}</p>
38 <!-- <p>{{total | number :'1.0-2':'fr'}}</p> -->
39 <p>{{total | currency :'EUR':'symbol':'1.0-2':'fr'}}</p>
```

Mode Modules *
Compréhension des anciens projets Angular

ent.html

lar-2023-06 > TP01-les-fondamentaux > src > app > app.component.html > ...

```
 string interpolation de variables {{ }} -->
 Binding : le TS est lié à la vue HTML(template) -
 Single way Binding: TS => HTML -->
<h1>Formation {{title}} {{version}}</h1>


    <!-- composant enfant de AppComponent -->
    <app-composant2></app-composant2>

    <hr>

    <app-comp-images></app-comp-images>

    <hr>
    <h2>Espace Client ...</h2>
    <app-landing-page></app-landing-page>
```

app.module.ts

formation-angular-2023-06 > TP01-les-fondamentaux > src > app > app.module.ts > ...

```
import { BrowserModule } from '@angular/platform-browser';

// ----------
// IMPORT LOCAL (ressources fichiers)
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { FormsModule } from '@angular/forms';
import { Composant2Component } from './composant2/composant2.component';
import { CompImagesComponent } from './comp-images/comp-images.component';
import { HeaderComponent } from './compte-client/composants/header/header
import { CompteClientModule } from './compte-client/compte-client.module';

// un décorateur , son rôle : transformer la classe en ... (module)
@NgModule({
  // objet de description
```

compte-client.module.ts

formation-angular-2023-06 > TP01-les-fondamentaux > src > app > compte-client > compte-client.module.ts

```
    FooterComponent,
    LandingPageComponent
  ],
  imports: [
    CommonModule
  ],
  exports:[
    HeaderComponent,
    BodyComponent,
    FooterComponent,
    LandingPageComponent
  ]
})
export class CompteClientModule { }
```

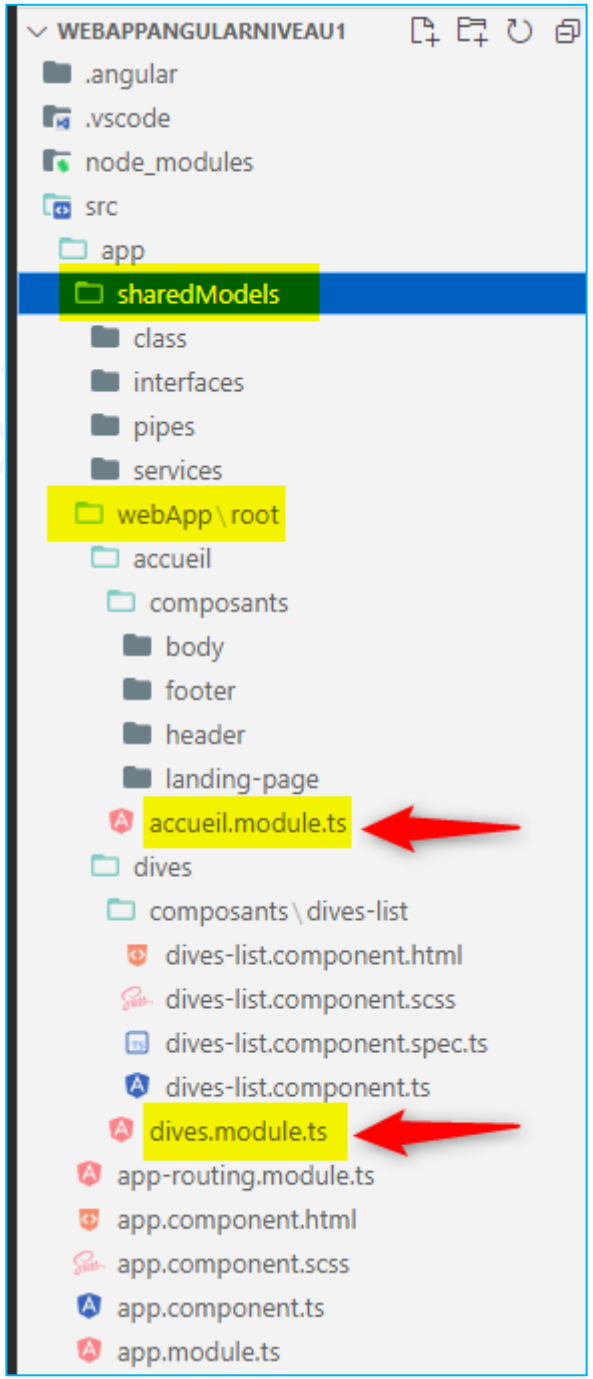
Export des composants depuis le module « compte-client »
Import du module « compte-client » dans « app-module »

Mise en place du TP fil rouge

- Création d'un nouveau projet – avec le routage et SCSS
- Installation des dépendances Bootstrap et bootstrap-icons
  https://www.npmjs.com/package/bootstrap
  https://www.npmjs.com/package/bootstrap-icons
- Installation de la « dev » dépendance json-server
  npm install -D json-server

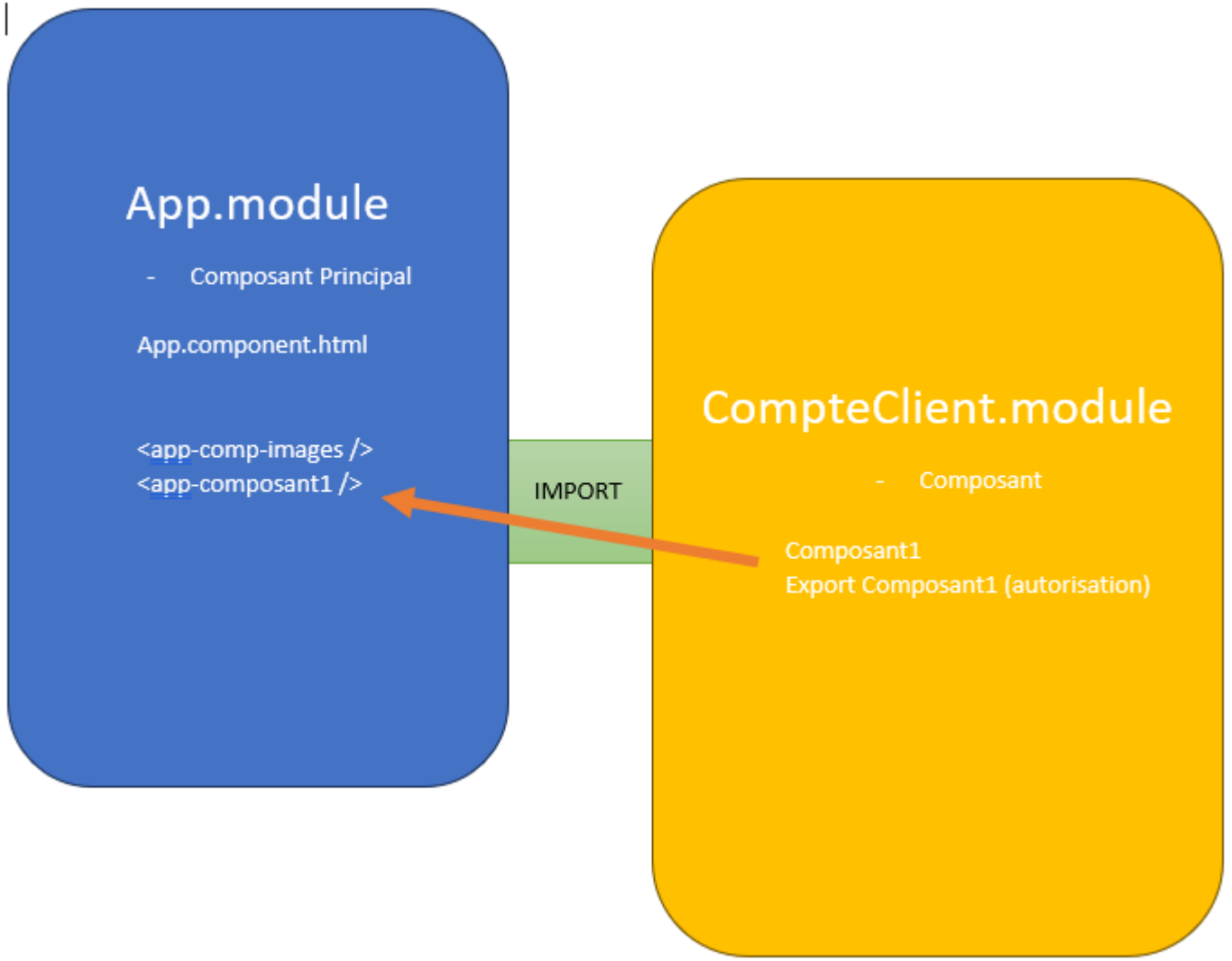
```
9     "test": "ng test",
10    "json-server":"json-server --watch src/assets/json/bdd.json -p 3001"
```

- Bonnes pratiques « structure et architecture » de projet
- injection de dépendances …



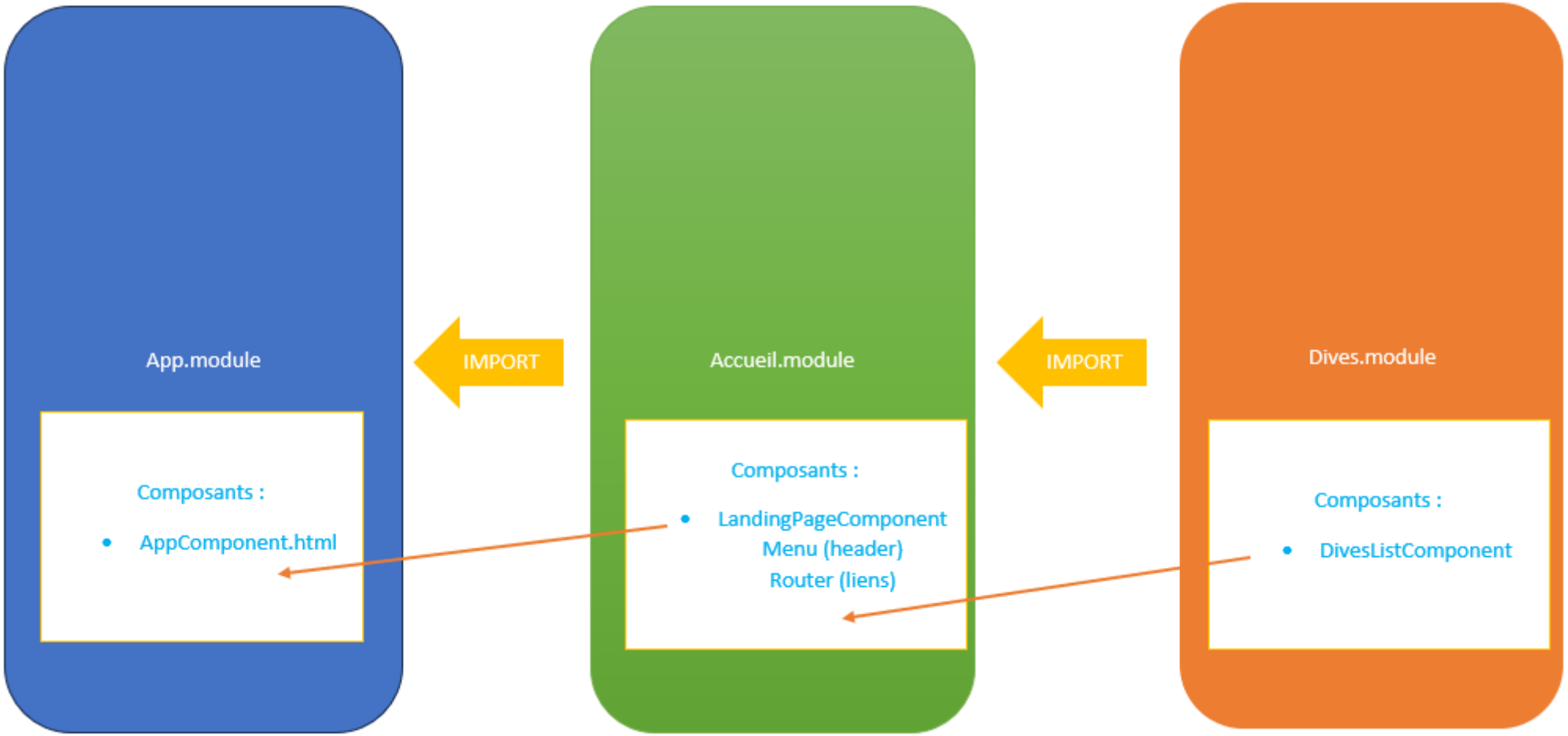
Mode Modules *
Compréhension des anciens projets Angular

Communication entre modules



Mode Modules *
Compréhension des anciens projets Angular

Communication entre modules



Mise en place du TP fil rouge

```
PS C:\Users\Michel\Desktop\formation-angular-sparks-10-2023\TP03-services-HTTP> npm run json-server

> tp02-bonnes-pratiques-injection-dependances@0.0.0 json-server
> json-server --watch src/assets/json/bdd.json -p 3001


  \{^_^}/ hi!

  Loading src/assets/json/bdd.json
  Done

  Resources
  http://localhost:3001/dives
  http://localhost:3001/produits

  Home
  http://localhost:3001

  Type s + enter at any time to create a snapshot of the database
  Watching...
```

Mise en place du TP fil rouge

package.json U | header.component.html U

TP02-bonnes-pratiques > package.json > { } dependencies

```
{
  "name": "tp02-bonnes-pratiques",
  "version": "0.0.0",
  ▷ Déboguer
  "scripts": {
    "ng": "ng",
    "start": "ng serve --open --port=3000",
    "build": "ng build",
    "watch": "ng build --watch --configuration developmer
    "test": "ng test"
  },
  "private": true,
  "dependencies": {
    "@angular/animations": "^16.2.0",
    "@angular/common": "^16.2.0",
    "@angular/compiler": "^16.2.0",
    "@angular/core": "^16.2.0",
    "@angular/forms": "^16.2.0",
    "@angular/platform-browser": "^16.2.0",
    "@angular/platform-browser-dynamic": "^16.2.0",
    "@angular/router": "^16.2.0",
    "bootstrap": "^5.3.2",
    "bootstrap-icons": "^1.11.1",
    "rxjs": "~7.8.0",
    "tslib": "^2.3.0",
```

angular.json U

TP02-bonnes-pratiques > angular.json > { } projects > { } TP02-bonnes-pratiques > { } architect > { } build >

```
    "architect": {
      "build": {
        "builder": "@angular-devkit/build-angular:browser",
        "options": {
          "outputPath": "dist/tp02-bonnes-pratiques",
          "index": "src/index.html",
          "main": "src/main.ts",
          "polyfills": [
            "zone.js"
          ],
          "tsConfig": "tsconfig.app.json",
          "inlineStyleLanguage": "scss",
          "assets": [
            "src/favicon.ico",
            "src/assets"
          ],
          "styles": [
            "node_modules/bootstrap/dist/css/bootstrap.min.css",
            "node_modules/bootstrap-icons/font/bootstrap-icons.css",
            "src/styles.scss"
          ],
          "scripts": [
            "node_modules/bootstrap/dist/js/bootstrap.min.js"
          ]
        },
```

TP : single et double way binding

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

Codiumate: Options | Test this class
@Component({
  selector: 'app-home-page',
  templateUrl: './home-page.component.html',
  styleUrl: './home-page.component.scss'
})
export class HomePageComponent {
  // propriétés

  public codeCours:string='VUE';

  public result:string='ok';

  // méthodes
  Codiumate: Options | Test this method
  public methodClic(){
    this.result='Vous avez cliqué !';
  }

}
```

```html
Go to component
<div class="alert alert-primary">
    <h3>HOME PAGE</h3>

    <!-- single way Binding : HTML => TS -->
    <button class="btn btn-warning" (click)="methodClic
    ()">
        Cliquez pour tester le single way binding...
    </button>

    <p>{{result}}</p>

    <!-- EXO :
     *ngIf *ngSwitchCase
     @if() Control flow NG>17+

    si le codeCours est égal NG => Vous avez choisi
    Formation Angular
    Sinon ..... REACT => .... React
    Sinon ..... VUE => ..... Vue
    Autres => Choisir une formation
    -->


</div>
```

TP :

home-page.component.ts

app > webApp > root > accueil > components > home-page > home-pag

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

Codiumate: Options | Test this class
@Component({
  selector: 'app-home-page',
  templateUrl: './home-page.component.html',
  styleUrl: './home-page.component.scss'
})
export class HomePageComponent {
  // propriétés

  public codeCours:string='VUE';

  public result:string='';

  // méthodes
  Codiumate: Options | Test this method
  public methodClic(){
    this.result='Vous avez cliqué !';
  }

}
```

home-page.component.html

TP02 > src > app > webApp > root > accueil > components > home-page > home-page.component.html > div.ale

Go to component

```html
<div class="alert alert-primary">
    <h3>HOME PAGE</h3>

    <!-- single way Binding : HTML => TS -->

    <button class="btn btn-primary" (click)="methodClic()">
        Cliquez pour tester le single way binding...
    </button>
    <p>{{result}}</p>

    <p></p>

    <select class="form-control" [(ngModel)]="codeCours">
        <option value="">Choisir pour tester le double way binding</option>
        <option value="NG">Angular ...</option>
        <option value="REACT">React ...</option>
        <option value="VUE">Vue ...</option>
    </select>

    <p></p>

    <div class="alert alert-warning">

        @if (codeCours==="NG") {
        <span>Vous avez choisi Angular</span>
        }
        @else if (codeCours==="REACT") {
        <span>Vous avez choisi React</span>
        }
        @else if (codeCours==="VUE") {
        <span>Vous avez choisi Vue</span>
        }
        @else {
        <span>Choisissez une formation</span>
        }
```

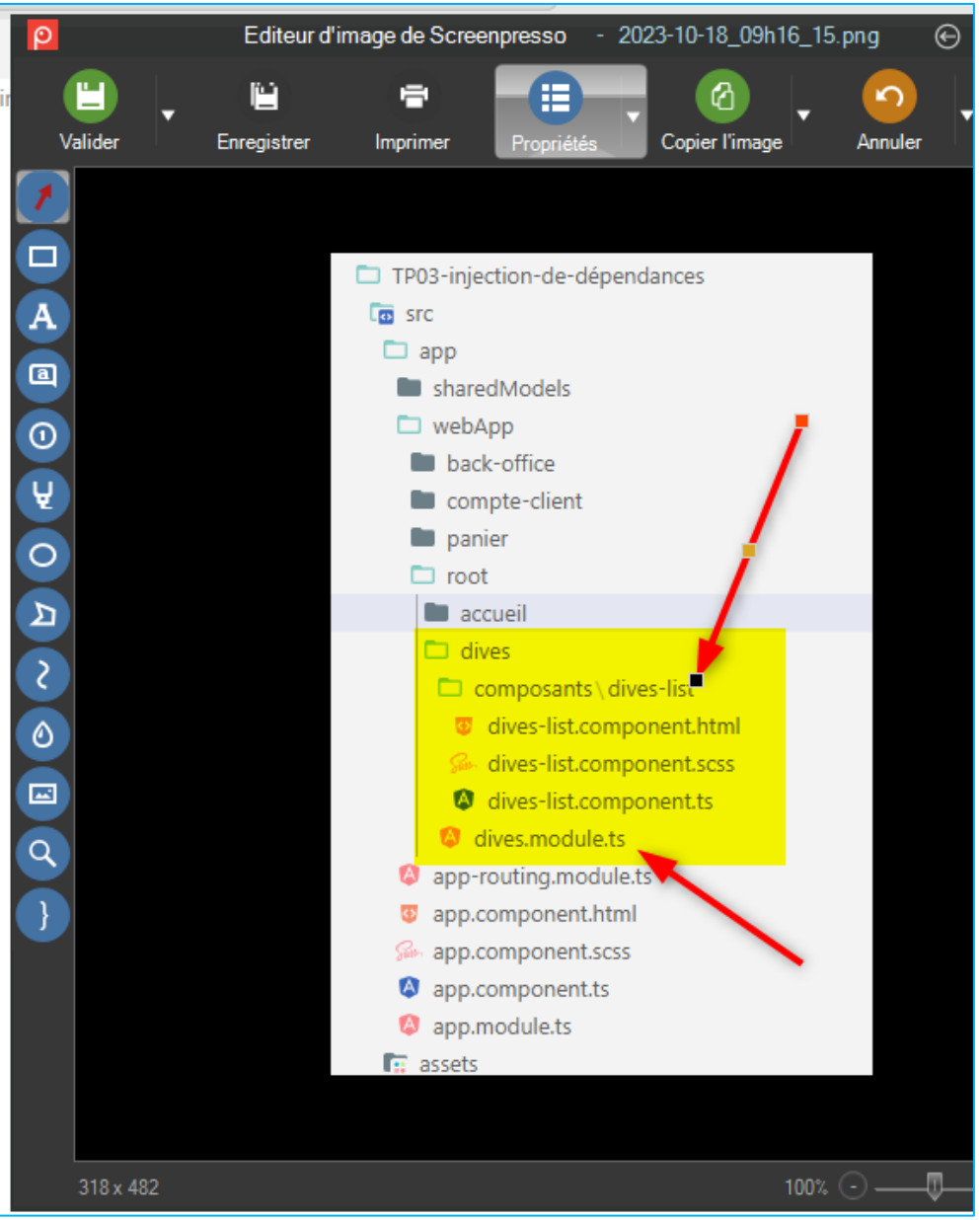
Arborescence de dives

landing-page.component.html

TP02-bonnes-pratiques > src > app > webApp > root > accueil > composants > landing-page > landi

```
1   <div clas="container">
2       <!-- peu de html => juste pour formater l'affichage global -->
3       <!-- les sélecteurs -->
4
5       <app-header></app-header>
6
7
8       <!-- <app-body></app-body> -->
9       <app-dives-list></app-dives-list>
10      <!-- dives-list works  -->
11
12
13      <app-footer></app-footer>
14
15  </div>
```

Mode Modules *
Compréhension des anciens projets Angular



Mode Modules *
Compréhension des anciens projets Angular

Arborescence de dives

webApp > root > accueil > composants > landing-page > landing-page.component.html

```
<!-- les sélecteurs -->

<app-header></app-header>

<!-- <app-body></app-body> -->
<app-dives-list></app-dives-list>
<!-- dives-list works  -->

<app-footer></app-footer>

</div>
```

accueil.module.ts

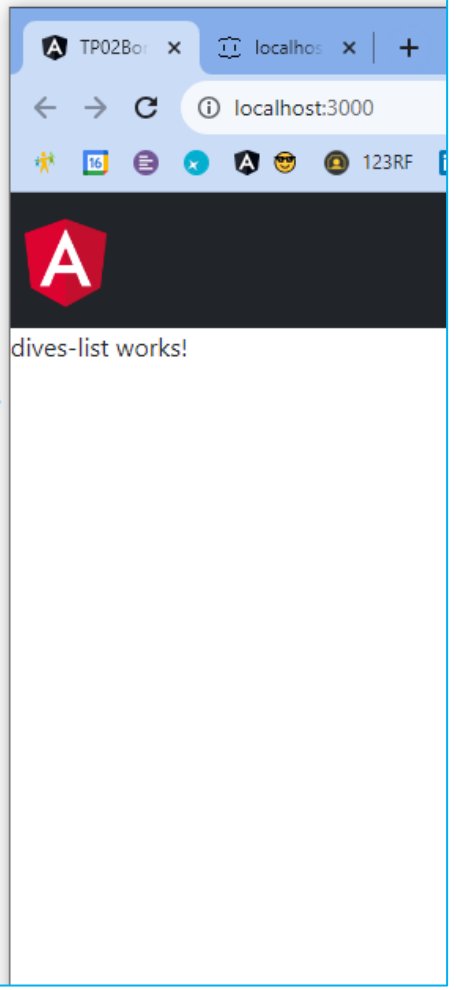
TP02-bonnes-pratiques > src > app > webApp > root > accueil > accueil.module.ts > AccueilModule

```
import { FooterComponent } from './composants/footer/footer.component';
import { DivesModule } from '../dives/dives.module';


@NgModule({
  declarations: [
    LandingPageComponent,
    HeaderComponent,
    BodyComponent,
    FooterComponent
  ],
  imports: [
    CommonModule,
    // import de nos modules
    DivesModule
```

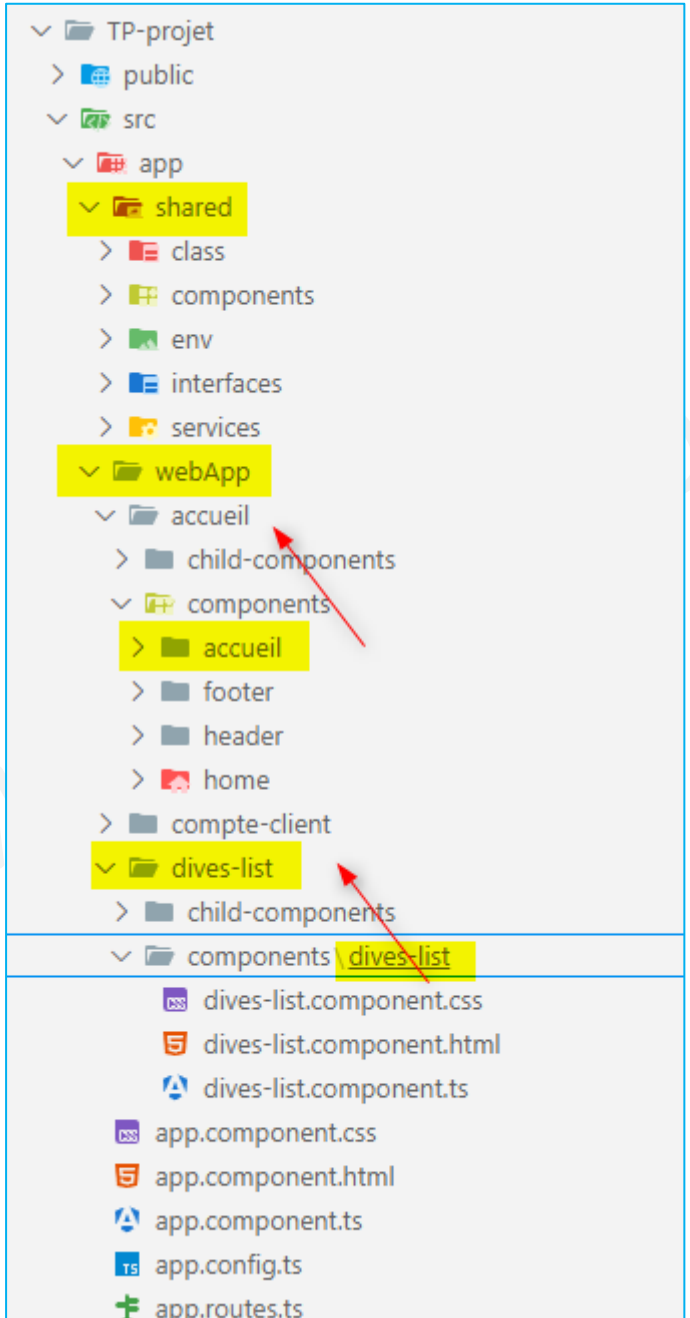
dives.module.ts

TP02-bonnes-pratiques > src > app > webApp > root > dives > dives.module.ts

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { DivesListComponent } from './composants/dives-lis

@NgModule({
  declarations: [
    DivesListComponent
  ],
  imports: [
    CommonModule
  ],
  exports:[
    DivesListComponent
  ]
})
export class DivesModule
```



Arborescence en Mode Standalone 😊



app.module.ts

TP03-injection-de-dépendances > src > app > app.module.ts

```ts
import { NgModule } from '@angular/c
import { BrowserModule } from '@angu

import { AppRoutingModule } from './
import { AppComponent } from './app.
import { AccueilModule } from './web

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    AccueilModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }

```

landing-page.component.html

TP03-injection-de-dépendances > src > app > webApp > root > accueil > composants > landing-page > landing-page.component.html > app-dives-list

```html
<app-header></app-header>
<!-- <app-body></app-body> -->
<app-dives-list></app-dives-list>
<app-footer></app-footer>
```

accueil.module.ts

TP03-injection-de-dépendances > src > app > webApp > root > accueil > accueil.module.ts

```ts
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { LandingPageComponent } from './composants/l
import { HeaderComponent } from './composants/header
import { FooterComponent } from './composants/footer
import { BodyComponent } from './composants/body/body
import { DivesModule } from '../dives/dives.module';

@NgModule({
  declarations: [
    LandingPageComponent,
    HeaderComponent,
    FooterComponent,
    BodyComponent
  ],
  imports: [
    CommonModule,
    DivesModule
  ],
  exports: [
    LandingPageComponent,
```

dives.module.ts

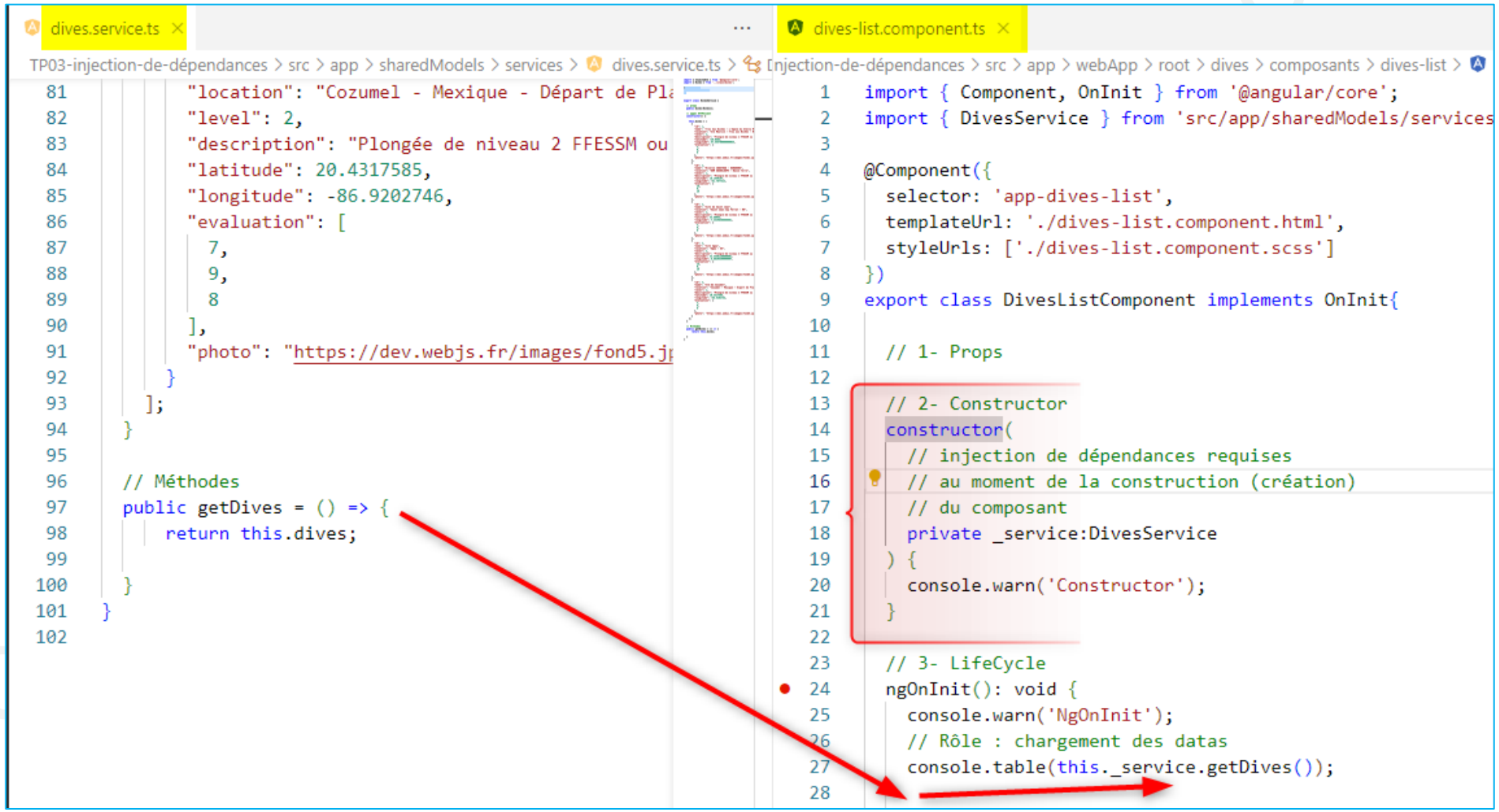
TP03-injection-de-dépendances > src > app > webApp > root

```ts
import { NgModule } from '@angular/co
import { CommonModule } from '@angula
import { DivesListComponent } from '.


@NgModule({
  declarations: [
    DivesListComponent
  ],
  imports: [
    CommonModule
  ],
  exports: [
    DivesListComponent
  ]
})
export class DivesModule { }

```

Mode Modules *
Compréhension des anciens projets Angular

L'injection de dépendances Angular

Au moment de la création du composant (constructor)
Le composant reçoit un service ou une « fonctionnalité » requise



L'injection de dépendances

1er exemple : le service

dives-list.component.ts

bApp > root > dives > composants > dives-list > dives-list.component.ts > DivesListComponent

```ts
// @ViewChild('btnPlus') eltBtn!:ElementRef;
@ViewChildren('btnPlus') eltBtnCollection!: QueryList

// 2- const
// l'injection de dépendances permet de lier un servi

constructor(
  private _service: DivesService,
  private _title: Title,
  private _meta: Meta
) {
  console.warn('Constructor');
  this.dives = [];
  this._title.setTitle('Liste des Plongées SEO ....')
  this._meta.addTags([
    { name: 'description', content: 'Texte de descrip
    { name: 'author', content: 'MB CREATION WEB' }
  ]);
}

// ----------------------------------------------------
// 3- lifeCycle
// ----------------------------------------------------
ngOnInit(): void {
  // Chargement des datas
  console.warn('NgOnInit');
  this._service.getDives()

    // .subscribe(
    //   // next seulement ...
    //   (datas:Dives[]) => {
    //     this.dives = datas;
    //   }
    // );

    .subscribe({
      next: // Call Back OK
        (datas: Dives[]) => {
          this.dives = datas;
        },
      error:
        e => console.log(e)
      ,
      complete:
        () => console.log('Observer Complete')
    }
```

dives.service.ts

TP03-services-http > src > app > sharedModels > services > dives.service.ts > DivesService > getDives

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

@Injectable({
  providedIn: 'root'
})

export class DivesService {

  // 1- props
  private dives: Dives[];

  // 2- const
  constructor(private _http: HttpClient) {
    this.dives = [];
  }

  // 3- Méthodes
  public getDives = (): Observable<Dives[]> => {

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

    return this._http.get<Dives[]>(API_URL).pipe(
      // le pipe permet d'enchainer les opérateurs RXJS
      // 1 opérateur RXJS est une fonction qui modifie
      // ou transforme la séquence(stream ou flux) de valeurs émises par l'observable

      tap(
        // opérateur de débug => console
        (response: any) => {
          console.log('------ Réponse depuis le service : ', response);
        }
      ),
      map(
        (datas:Dives[]) => {
          return datas.filter(
            (data:Dives) => {
              // return data.id===5
              return data.id >=1
            }
          )
        }
      )
```

Life Cycle du composant
constructor & ngOnInit

ng g(enerate) s(ervice) dives

dives-list.component.ts 3 | dives.service.ts

TP02-bonnes-pratiques-injection-dependances > src > app > sharedModel

```
1  import { Injectable } from '@angular/core';
2  import { Dives } from '../class/dives';
3
4  @Injectable({
5    providedIn: 'root'
6  })
7
8  export class DivesService {
9
10   // 1- props
11   private dives: Dives[];
12
13   // 2- const
14   constructor() {
15     this.dives = [
16       {
17         "id": 1,
18         "name": "Trou Aux Biches - L'Epave du
19         "location": "Ile Maurice - Trou aux Bi
```

```
constructor(
  private _service: DivesService,
  private _title: Title,
  private _meta: Meta
) {
  console.warn('Constructor');
  this.dives = [];
  this._title.setTitle('Liste des Plongées SEO ....');
  this._meta.addTags([
    { name: 'description', content: 'Texte de description ...' },
    { name: 'author', content: 'MB CREATION WEB' }
  ]);

}

// ---------------------------------------------------------
// 3- lifeCycle
// ---------------------------------------------------------
ngOnInit(): void {
  // Chargement des datas
  console.warn('NgOnInit');
  // console.table(this._service.getDives());
  // -- on passe les datas du service à la prop dives du composant
  this.dives = this._service.getDives();
}
```

TP : renseigner les valeurs manquantes

dives-list.component.ts 9+

dives > composants > dives-list > dives-list.component.ts > DivesListComponent > ngOnInit

```ts
import { Component, OnInit } from '@angular/core';
import { Dives } from 'src/app/sharedModels/class/dives
import { DivesService } from 'src/app/sharedModels/serv

@Component({
  selector: 'app-dives-list',
  templateUrl: './dives-list.component.html',
  styleUrls: ['./dives-list.component.scss']
})
export class DivesListComponent implements OnInit{

  // 1- Props
  ......................

  // 2- Constructor
  constructor(
    // injection de dépendances requises
    // au moment de la construction (création)
    // du composant
    private _service:DivesService
  ) {
    console.warn('Constructor');
    ....................
  }

  // 3- LifeCycle
  ngOnInit(): void {
    console.warn('NgOnInit');
    // Rôle : chargement des datas
    console.table(this._service.getDives());
    ....................

```

dives-list.component.html

root > dives > composants > dives-list > dives-list.component.html > div.row > div.col-md-3 > div.card > div.card-boc

```html
<h1 class="alert alert-primary">Liste des Plongées</h1>

<div class="row">

    <div class="col-md-3" *ngFor="..................">

        <div class="card">
            <!-- récupérer chaque image -->
            <img src="..............." class="card-img-top">
            <div class="card-body">
                <h3 class="card-title">Nom : ..............</h3>
                <p class="card-text">Description : ..................</p>

                <!-- cibler un élément du dom en template référence # -->
                <button class="btn btn-primary" #btnPlus>En Savoir Plus</button>

            </div>
        </div>
    </div>
</div>
```

Life Cycle du composant
AfterViewInit – AfterViewChecked - Ondestroy

```
// ----------------------------------------
//      cycles de vie
// ----------------------------------------
ngOnInit():void {
  // exploitation des datas
  console.warn('OnInit');
  console.table(this._service.getDives());
  this.dives=this._service.getDives();
  console.log('dans le init : ', this.eltBtn);
  // output => undefined

}
```

```
// ----------------------
ngAfterViewInit(): void {
  console.warn('ngAfterViewInit');
  // quand le DOM (document object model) est chargé
  // toute la vue HTML et tous ces éléments sont prêts
  console.log('dans le After View Init : ', this.eltBtn);
  // output => ElementRef {nativeElement: button.btn.btn-primary}
  let texte='AfterViewInit';
  // this.eltBtn.nativeElement.innerHTML = `Plus d'infos ${texte}`;

  console.log(this.eltCollectionBtn);

  this.eltCollectionBtn.forEach(
    (elt:ElementRef) => {
        elt.nativeElement.innerHTML=`Plus d'infos`;
        // elt.nativeElement.className='btn btn-warning';
        elt.nativeElement.classList.replace('btn-primary', 'btn-success');
    }
  );
```

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 stocké 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.

RXJS – Les Observables
Niveau 1

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 le gestion des requêtes asynchrones.

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)

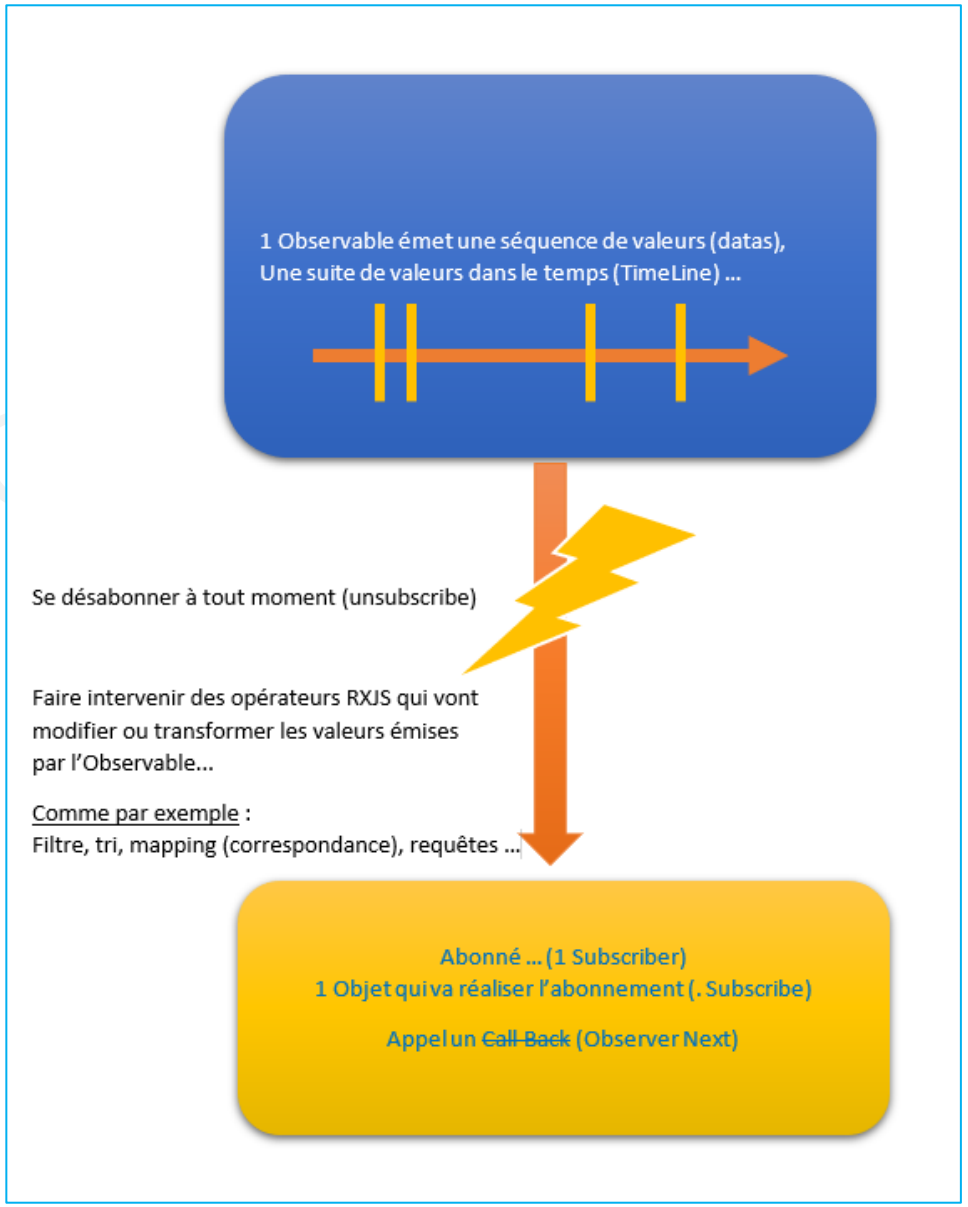
Opérateurs : des fonctions tap, map, merge, filter qui vont pouvoir modifier/transformer ce flux de valeurs émises par l'observable

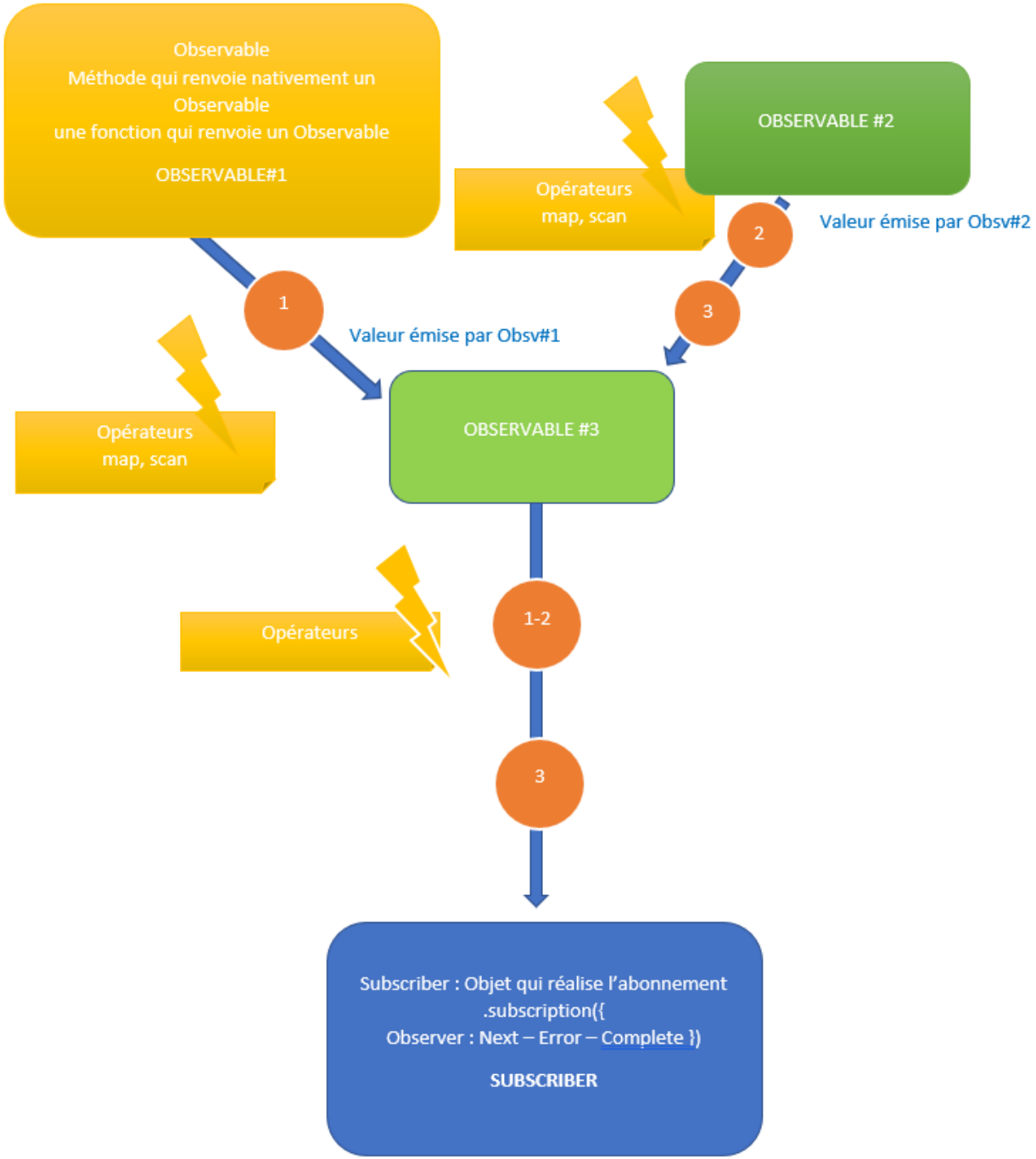
Dans le subscribe => Les observers (fcts de call back) qui se définissent dans le subscribe : Next - Error – Complete

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 !





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

Routage

Routing : Concepts de base

Un path (URL ou fragment d'URL, l'URI) est associé à un composant dans le fichier « app-routing.module.ts »

Le template du composant est chargée dans le router-outlet, lorsque ce path est invoqué.

Routing : Concepts de base

app-routing.module.ts | header.component.html

TP04-routage > src > app > app-routing.module.ts > routes

```ts
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';

// import des composants
import { DivesListComponent } from './webApp/root/dives/composants/dives-list/
import { BodyComponent } from './webApp/root/accueil/composants/body/body.comp

const routes: Routes = [

{path:'', component:BodyComponent},
{ path:'liste-des-plongees', component:DivesListComponent}
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }

```

header.component.html

...vbar-expand-lg.bg-dark.navbar-dark.sticky-top > div#menu.collapse.navbar-col

```html
<button class="navbar-toggler" type="button"
    <span class="navbar-toggler-icon"></span>
</button>

<div class="collapse navbar-collapse" id="men
    <ul class="navbar-nav">
        <li class="nav-item">
            <a class="nav-link"><i class="bi
        </li>
        <li class="nav-item">
            <a class="nav-link"

            routerLink="liste-des-plongees"

            ></i>
        </li>
        <li class="nav-item">
            <a class="nav-link"><i class="bi
        </li>
    </ul>
</div>
</nav>
```

landing-page.component.html

TP04-routage > src > app > webApp > root > accueil > composants > landing-page > landing-page.component.html > div.alert.ale

```html
<!-- <div class="container"> -->
    <app-header></app-header>

        <div class="alert alert-warning"
            <!-- comparé à un composant angular -->
            <router-outlet></router-outlet>
        </div>



    <app-footer></app-footer>
```

accueil.module.ts

TP04-routage > src > app > webApp > root > accueil > accueil.module.ts

```ts

@NgModule({
  declarations: [
    LandingPageComponent,
    HeaderComponent,
    FooterComponent,
    BodyComponent
  ],
  imports: [
    CommonModule,
    RouterModule,
```

Routing : Concepts de base

app-routing.module.ts

TP04-routage > src > app > app-routing.module.ts > routes

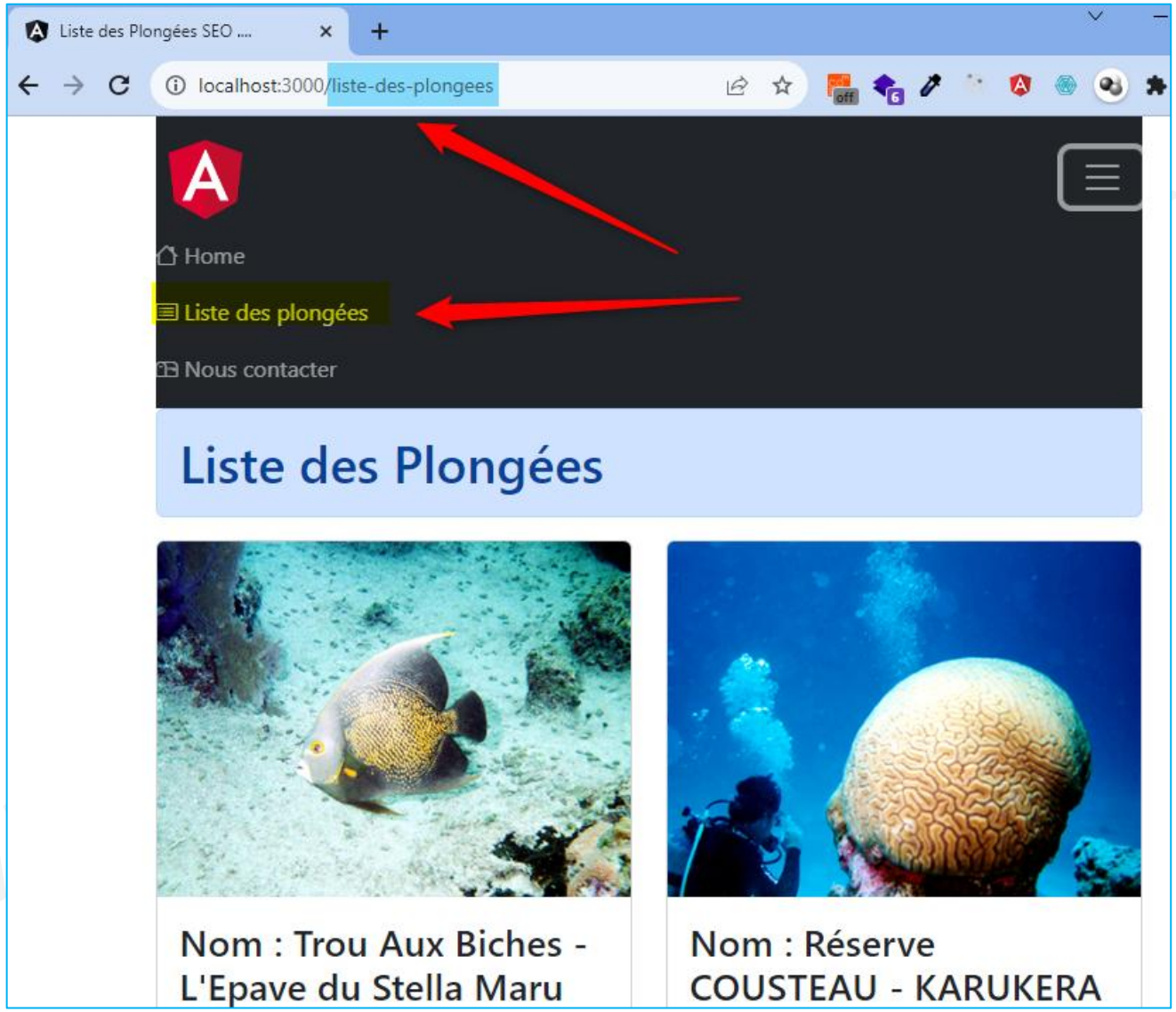
```ts
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { BodyComponent } from './webApp/root/accueil/composants/body/body.component';
import { DivesListComponent } from './webApp/root/dives/composants/dives-list/dives-list.component';
import { Page404Component } from './sharedModels/composants/page404.component';
import { ContactsComponent } from './webApp/root/form/composants/contacts/contacts.component';
import { DivesDetailComponent } from './webApp/root/dives/composants/dives-detail/dives-detail.component';

const routes: Routes = [
  { path: '', component: BodyComponent },
  // domain.tld/ (page de démarrage)
  // { path:'accueil', component:BodyComponent}, // domain.tld/accueil
  { path: 'accueil', redirectTo: '', pathMatch: 'full' },
  // { path:'nouvelle-url', component:DivesListComponent},
  // { path:'ancienne-url-seo-déjà-référencée', redirectTo:'nouvelle-url', pathMatch:'full'},
  { path: 'contactez-nous', component: ContactsComponent },
  { path: 'liste-des-plongees', component: DivesListComponent },
  { path: 'liste-des-plongees/:id', component: DivesDetailComponent },

  //  Gestion des erreurs : toujours en dernier !!
  // { path:'**', redirectTo:'', pathMatch:'full'}
  { path: '**', component: Page404Component }
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

header.component.html

TP04-routage > src > app > webApp > root > accueil > composants > header > header.component.html > nav.nav

```html
<nav class="navbar navbar-expand-lg bg-dark navbar-dark sticky-top">
    <img width="70" alt="Angular Logo" src="./assets/images/angular.png">

    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" dat
        <span class="navbar-toggler-icon"></span>
    </button>

    <div class="collapse navbar-collapse" id="menu">
        <ul class="navbar-nav">
            <li class="nav-item">
                <a class="nav-link"
                routerLink="accueil"
                ><i class="bi bi-house"></i> Home</a>
            </li>

            <li class="nav-item">
                <!-- <a class="nav-link"
                [routerLink]="['liste-des-plongees']"
                ><i class="bi bi-card-list"></i> Liste des plongées</a> -->
                <a class="nav-link"
                routerLink="liste-des-plongees"
                ><i class="bi bi-card-list"></i> Liste des plongées</a>
            </li>

            <li class="nav-item">
                <a class="nav-link"
                routerLink="contactez-nous"
                ><i class="bi bi-mailbox"></i> Nous contacter</a>
            </li>
        </ul>
    </div>
</nav>
```

Routing : Concepts de base



Routing : Concepts de base

landing-page.component.html

src > app > webApp > root > accueil > composants > landing-page > landing-page.component.html

```
<div style="min-height:100vh;" class="alert alert-warning"
    <app-header></app-header>
    <!-- <app-body></app-body>     -->
    <!-- <app-dives-list></app-dives-list> -->

    <!-- router-outlet primary -->
    <div class="alert alert-primary">

        <router-outlet></router-outlet>
    </div>

    <!-- router-outlet autres -->
    <div class="alert alert-success">

        <router-outlet name="outlet-2"></router-outlet>
    </div>



</div>
<app-footer></app-footer>
```

app-routing.module.ts

src > app > app-routing.module.ts > routes > canActivate

```
domain.tld/accueil
// { path:'accueil' , redirectTo:'',
pathMatch:'full'},
// { path:'nouvelle-url',
component:DivesListComponent},
// { path:'ancienne-url-seo-déjà-référencée',
redirectTo:'nouvelle-url', pathMatch:'full'},
{ path: 'liste-des-plongees', component:
DivesListComponent },
{ path: 'liste-des-plongees/:id', component:
DiveDetailComponent },

{ path: 'routing-angular', component:
CompRoutingOutletComponent, outlet: 'outlet-2',
canActivate: [RouteGuardService]  },

// 1ère manière de faire : charger un composant depuis
un module qui a son propre sytème de routage forChild
// {
//   path: 'mon-compte-client',
//   component: ClientLandingPageComponent,
//   children: clientRoutes
// },

// 2nde manière oprimisée : lazy loading avec ES2015
(promesses-import)
// {
//   path: 'mon-compte-client',
//   component: ClientLandingPageComponent,
//   loadChildren:
//     ():any => import('./webApp/compte-client/
compte-client.module').then(
```

header.component.html

src > app > webApp > root > accueil > composants > header > header.component.html

```
                Liste des plongées
            </a>
        </li>

        <li class="nav-item">
            <a class="nav-link" routerLink="mon-compt
        </li>

        <li class="nav-item">
            <a class="nav-link" [routerLink]="[{
            outlets : {
                'outlet-2':'routing-angular'
            }
        }]"

        [skipLocationChange]="true"><i class="bi bi-m
        </li>

        <li class="nav-item">
            <a class="nav-link"><i class="bi bi-mailb
        </li>
    </ul>
</div>
</nav>
```

Routing : Concepts avancés – QueryParams - Router


```
{ path: 'liste-des-plongees/:id', component: DivesDetailComponent },
```

Routing : Concepts avancés – QueryParams - Router

EXPLORATEUR

ÉDITEURS OUVERTS

GROUPE 1

dives-list.component.html TP04-rout...

GROUPE 2

dives-detail.component.ts TP04-routa...

FORMATION-ANGULAR-2023-06

- accueil.module.ts
- dives
  - composants
    - dives-detail
      - dives-detail.component.html
      - dives-detail.component.scss
      - dives-detail.component.ts
    - dives-list
      - dives-list.component.html
      - dives-list.component.scss
      - dives-list.component.ts
  - dives.module.ts
- form
- app-routing.module.ts
- app.component.html
- app.component.scss
- app.component.ts
- app.module.ts
- assets
- favicon.ico
- index.html
- main.ts
- styles.scss
- .editorconfig
- .gitignore

dives-list.component.html

TP04-routage > src > app > webApp > root > dives > composants > dives-list > dives-list.component.html > h1.alert.alert-prima

```html
<h1 class="alert alert-primary">Liste des Plongées</h1>

<div class="row">

    <div class="col-md-6" *ngFor="let dive of dives">

        <div class="card">
            <!-- récupérer chaque image -->
            <img src="{{dive.photo}}" class="card-img-top">
            <div class="card-body">
                <h3 class="card-title">Nom : {{dive.name}} </h3>
                <p class="card-text">Description : {{dive.description}}</p>

                <!-- <a
                [routerLink]="[dive.id]"
                [queryParams]="[dive.name]"
                > -->
                <!-- on interprète un des params de l'objet -->
                <a
                [routerLink]="[dive.id]"
                [queryParams]="dive"
                [skipLocationChange]="true"

                >
                <!--  on passe tout l'objet -->
                    <button class="btn btn-primary" #btnPlus>En Savoir Plus<
                </a>
            </div>
        </div>
    </div>
</div>
```

dives-detail.component.ts

TP04-routage > src > app > webApp > root > dives > composants > dives-detail > dives-detail.compone

```ts
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute, Router } from '@angular/router';
import { Dives } from 'src/app/sharedModels/class/dives';

@Component({
  selector: 'app-dives-detail',
  templateUrl: './dives-detail.component.html',
  styleUrls: ['./dives-detail.component.scss']
})
export class DivesDetailComponent implements OnInit {
  // 1- props
  public title: string;
  public dive:Dives;

  // 2- const
  constructor(
    private _routeActive: ActivatedRoute,
    private _router:Router
  ) {
    this.title = '';
    this.dive =  new Dives();
  }

  // 3- lifeCycle
  ngOnInit(): void {

    const id = this._routeActive.snapshot.params['id'];
    console.log(id);
    this.title = `Plongée N° ${id}`;
    // re-consommer le service ?
    // ré-exploiter les datas stockées en storage ou en indexedDB
    // Evidemment utiliser les fonctions de routage d'Angular ...

```

Routing : Concepts avancés – QueryParams - Router

dives-list.component.html

```
<h1 class="alert alert-primary">Liste des Plongées</h1>

<div class="row">

    <div class="col-md-6" *ngFor="let dive of dives">

        <div class="card">
            <!-- récupérer chaque image -->
            <img src="{{dive.photo}}" class="card-img-top">
            <div class="card-body">
                <h3 class="card-title">Nom : {{dive.name}} </h3>
                <p class="card-text">Description : {{dive.description}}</

                <!-- [routerLink]="[dive.id]" -->
                <!-- Ecriture simplifiée du routerLink => routerLink="{{di
                <!-- routerLink="details/{{dive.id}}"  -->
                <!-- en chemin relatif -->
                 <button

                routerLink="/liste-des-plongees/details/{{dive.id}}"
                [queryParams]="dive"
                [skipLocationChange]="true"

                class="btn btn-primary" #btnPlus>En Savoir Plus</button>

            </div>
```

dives-details.component.ts

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

@Component({
  selector: 'app-dives-details',
  templateUrl: './dives-details.component.html',
  styleUrls: ['./dives-details.component.scss']
})
export class DivesDetailsComponent implements OnInit {

  // -------------------------
  constructor(
    private _routeActive:ActivatedRoute
  )
  { }
  // -------------------------
  ngOnInit() {
    const id=this._routeActive.snapshot.params['param'];
    console.clear;
    console.log('ID récupéré :' , id);
    // TP : passer l'ID à la Vue
    // ---------
    this._routeActive.queryParams
    .subscribe(
      (params:any) => {
        console.log('QueryParams : ', params);
        // TP : passer l'objet récupéréré à la vue
      }
    )
```

dives-details.component.html

```
<h1>Détail de la plongée</
h1>
<!-- TP afficher l'objet
passé par routage -->
```

app-routing.module.ts

```
const routes: Routes = [

  { path: '', component: BodyComponent },   // domain.tld/ (page de démar
  { path: 'liste-des-plongees', component: DivesListComponent },

  // { path:'accueil', component:BodyComponent}, // domain.tld/accueil

  // { path:'nouvelle-url', component:DivesListComponent},
  // { path:'ancienne-url-seo-déjà-référencée', redirectTo:'nouvelle-url'

  { path: 'liste-des-plongees/details/:param', component: DivesDetailsCom

  { path:'**', component:Page404Component}
```

bdd.json

```
{
    "dives": [
        {
            "id": 1,
            "name": "Trou Aux Biches - L'Epave du Stella Maru",
            "location": "Ile Maurice - Trou aux Biches - Nord/Ouest",
            "level": 2,
            "description": "Plongée de niveau 2 FFESSM ou PADI au arge de
            "latitude": -20.0363,
            "longitude": 57.544700000000034,
            "evaluation": [
```

dives.module.ts 1

```
import { NgModule } from '@
import { CommonModule } fro
import { DivesListComponent
import { HttpClientModule }
import { FormsModule } from
import { RouterModule } fro
import { DivesDetailsCompon


@NgModule({
  declarations: [
```

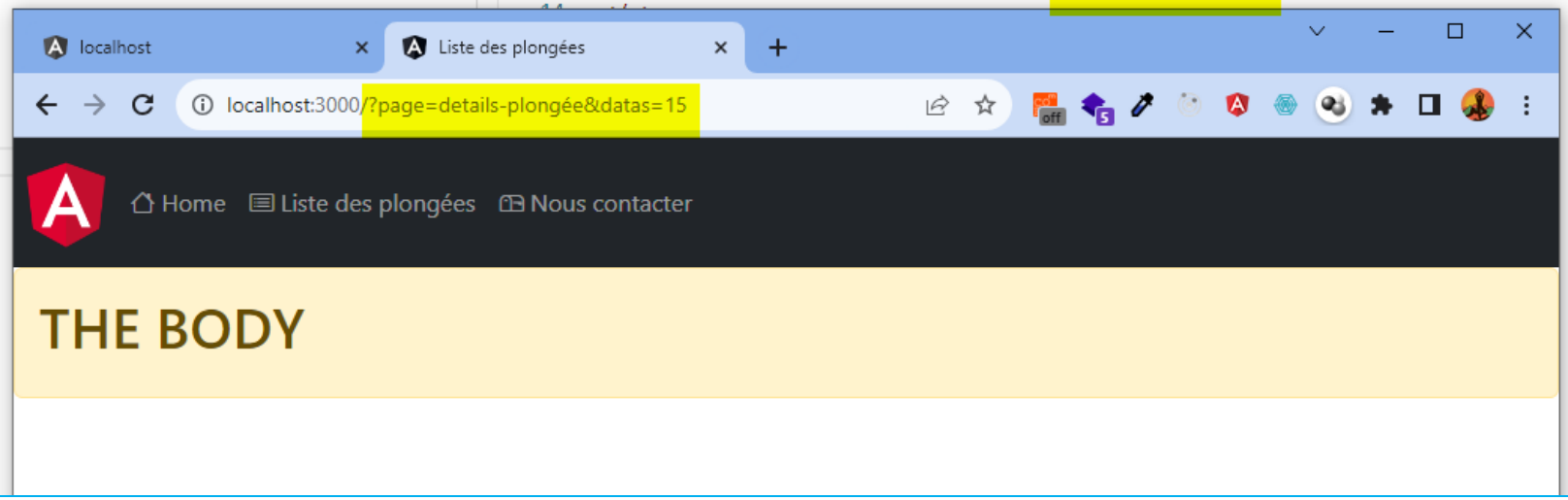
Routing : Concepts avancés – QueryParams - Router

dives-details.component.ts

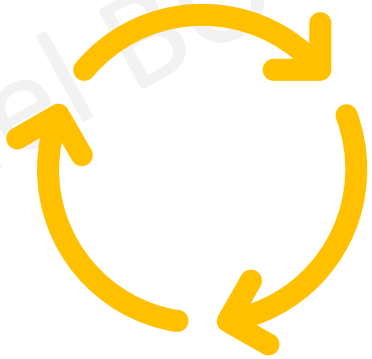
```
ngOnInit() {
  const id = this._routeActive.snapshot.params['param'];
  console.clear;
  console.log('ID récupéré :', id);
  // TP : passer l'ID à la Vue
  // ---------
  this._routeActive.queryParams
    .subscribe(
      (params: any) => {
        console.log('QueryParams : ', params);
        // TP : passer l'objet récupéréré à la vue
        this.dive = params;
      }
    )
}
// -- Méthodes
public goBack = () => {
  this._router.navigateByUrl('liste-des-plongees');
}

public goHome = () => {
  this._router.navigate(
    [''],
    {
      queryParams: {
        page: 'details-plongée',
        datas: 15
      }
    }
  )
```

dives-details.component.html

```
<h1>Détail de la plongée N° {{dive.id}}</h1>
<!-- TP afficher l'objet passé par routage -->

<!-- Single way binding -->
<!-- TS => HTML : [directive] et {{string interpolation}} -->
<!-- HTML => TS : (event du DOM) par ex : (click)-->
<p>
    <button class="btn btn-warning" (click)="goBack()">Go Back</button>

    <button class="btn btn-danger" routerLink="/liste-des-plongees">RouterLink</button>
</p>
<p>
    <button class="btn btn-success" (click)="goHome()">Go Home</button>
```

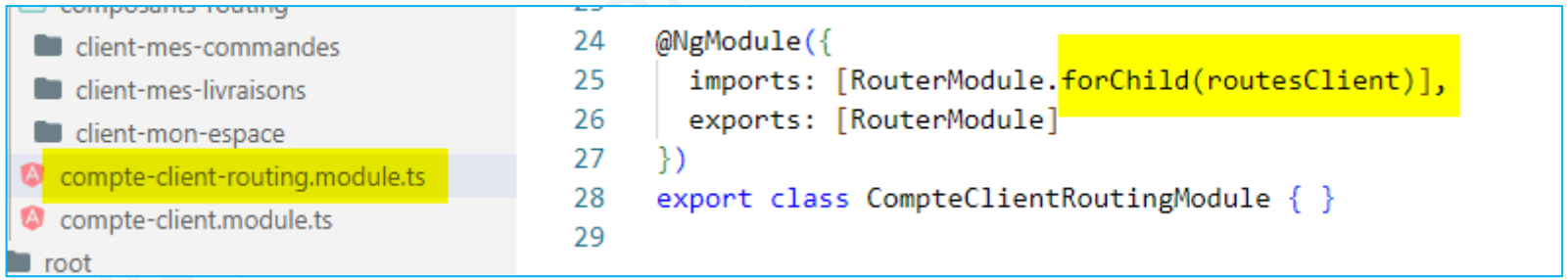


Routage Avancé
Lazy Loading
Optimisation du build



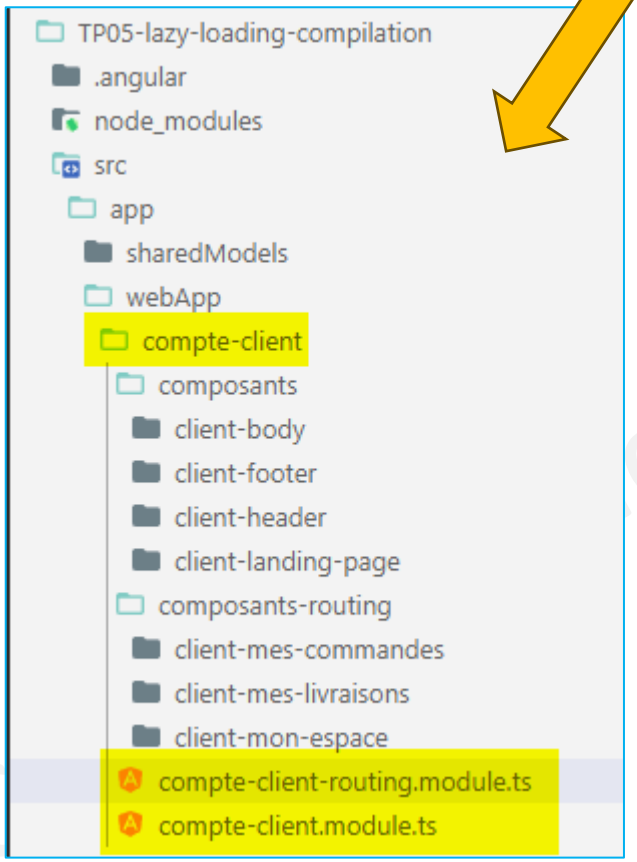
Routing : Concepts avancés – Le Lazy Loading

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

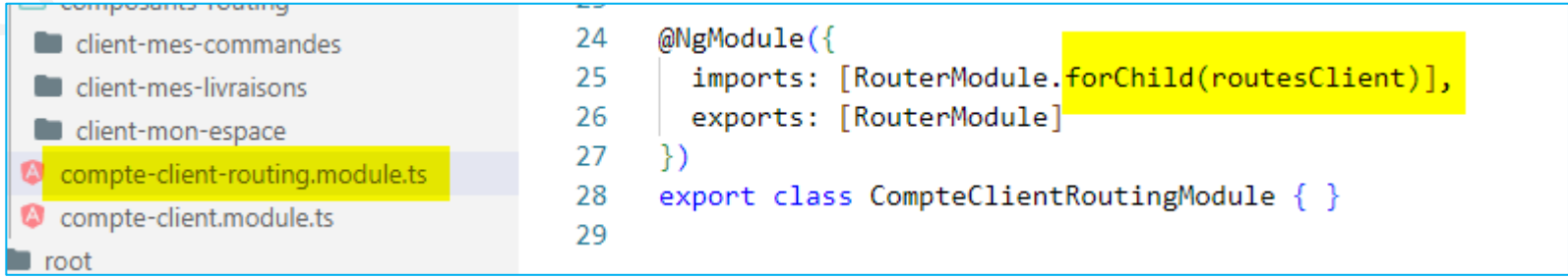


Routing : Concepts avancés – Le Lazy Loading

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



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

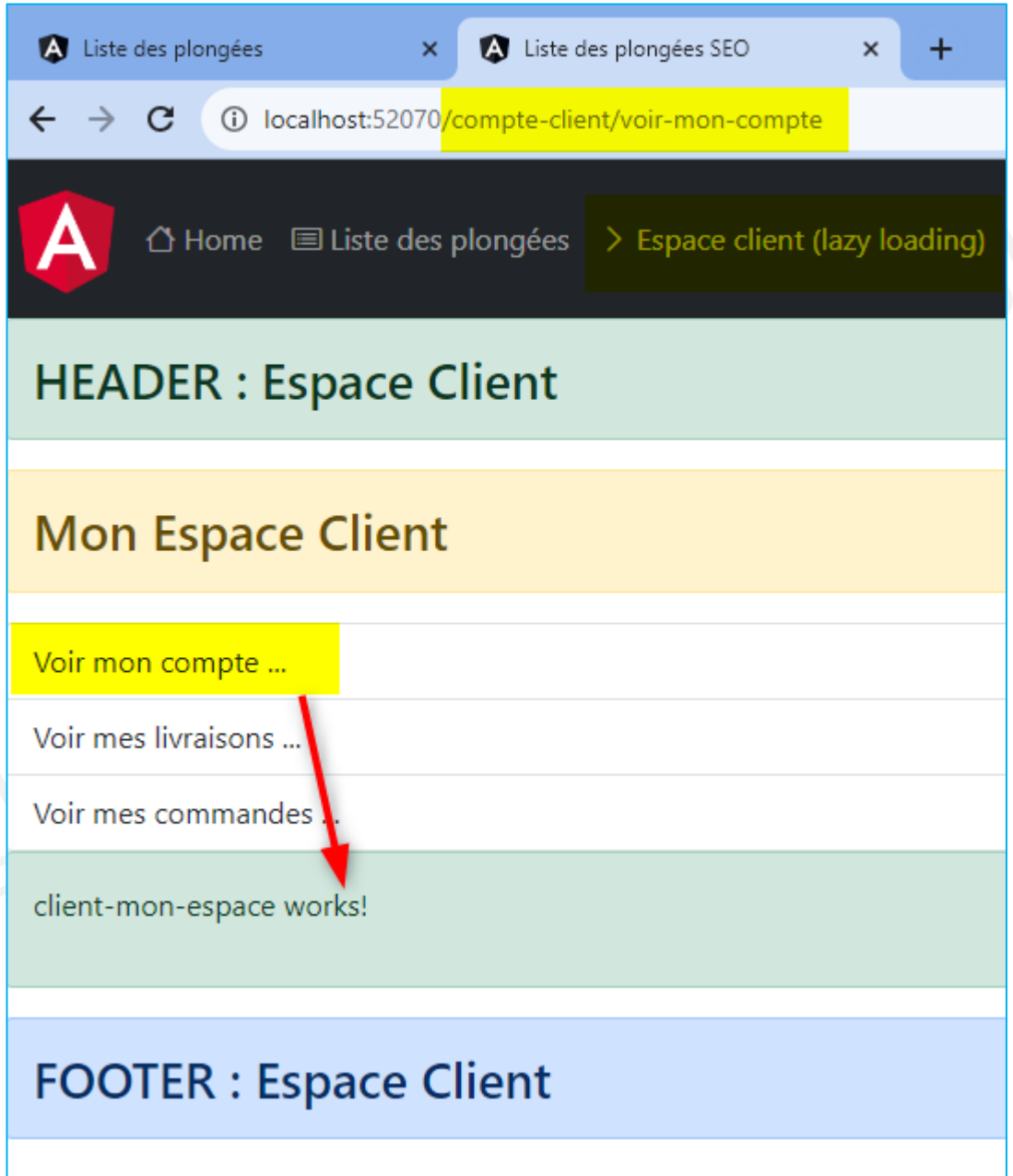


Routing : Concepts avancés – Le Lazy Loading

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.

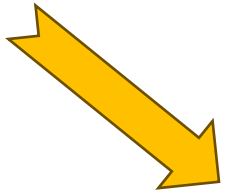
```
PS C:\Angular...ation-angular-2023-06\TP05-lazy-loading-compilation> npm run build

> 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



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
    }

},
```

Mode standalone
1 seul fichier de routage

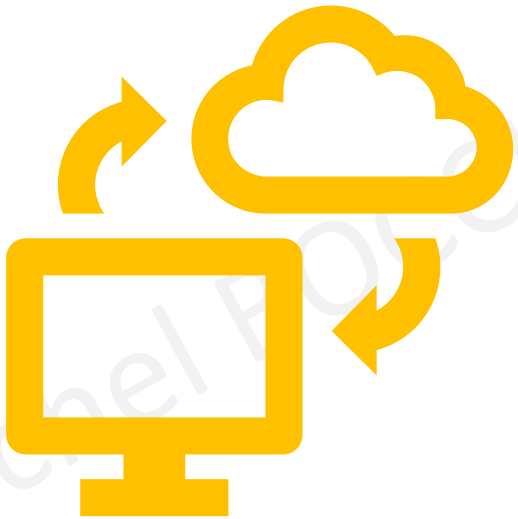
Le Lazy loading
se gère encore plus facilement 😊

app.routes.ts

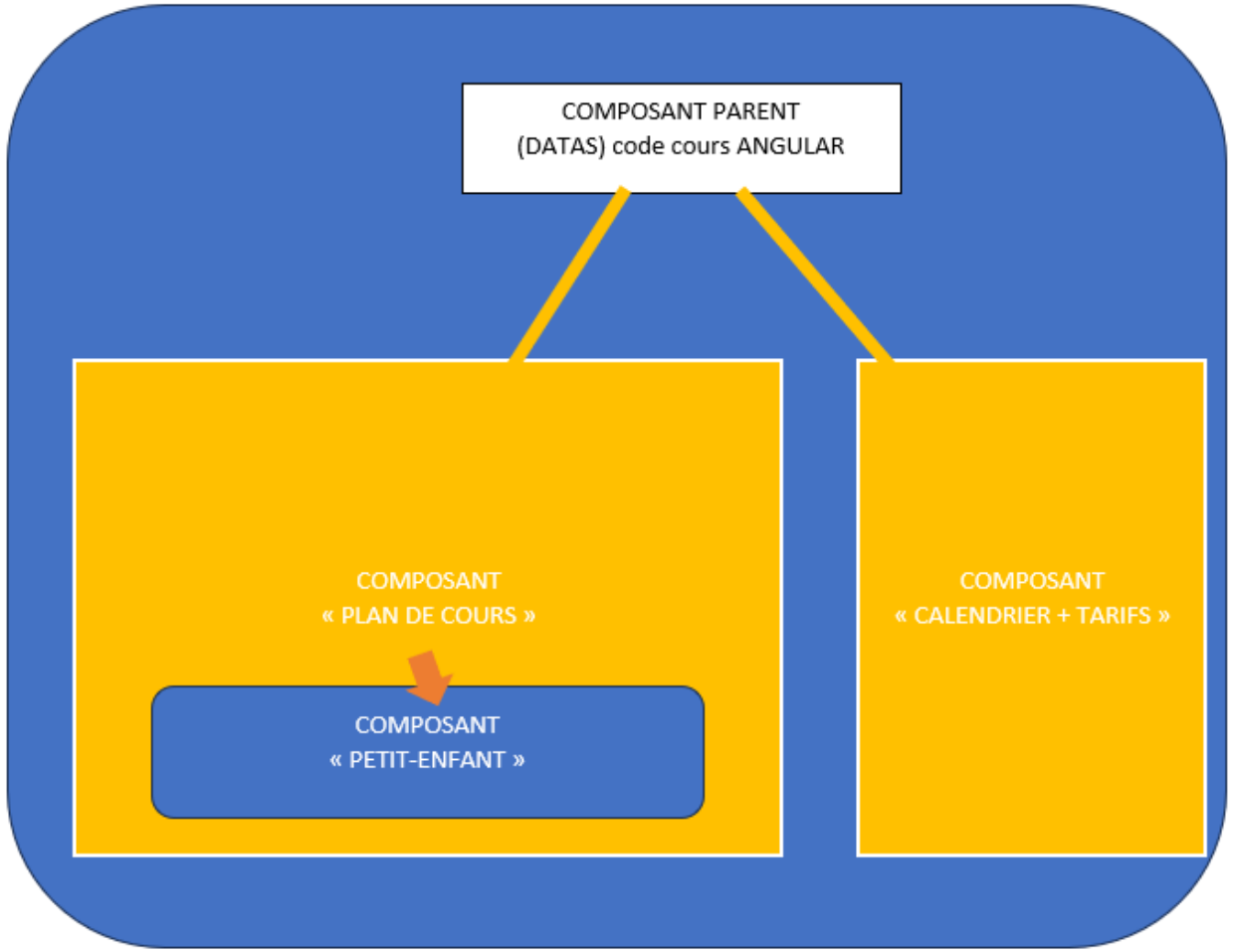
TP05-communication-composants > src > app > app.routes.ts > ...

```ts
import { Routes } from '@angular/router';
import { HomeComponent } from './webApp/accueil/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/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 { DivesDetailComponent } from './webApp/dives-list/components/dives-detail/dives-detail.component';
import { TpComponsantsComponent } from './webApp/tp-composants/components/tp-componsants/tp-componsants.component';


export const routes: Routes = [
    { path: '', component: HomeComponent, title: 'Angular 18 - StandAlone Components' },
    { path: 'liste-des-plongees', component: DivesListComponent, title: 'Liste des plongees' },
    { path:'liste-des-plongees/:id', component:DivesDetailComponent},
    { path: 'les-signals-angular-17', component: SignalsComponent, title: 'Les signals Angulars 17 - trop Cool' },
    { path: 'tp-composants', component: TpComponsantsComponent, title: '😎' },

    // 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.EntryCompteClientComponent
                ),
        children: [
            { path: 'mes-commandes', component: CommandesComponent },
            { path: 'mes-livraisons', component: LivraisonsComponent },
        ]
    },
    // TP ajouter le composant ContactsComponent en lazy Loading
    // { path: 'nous-contactez', component: ContactsComponent, title:'Wow Génial les formulaires NG 🚕' },

    // {path:'**', redirectTo:'',title:'Page Introuvable !'}
    { path: '**', component: Page404Component, title: 'Page Introuvable !' }
];

```

Input - Output

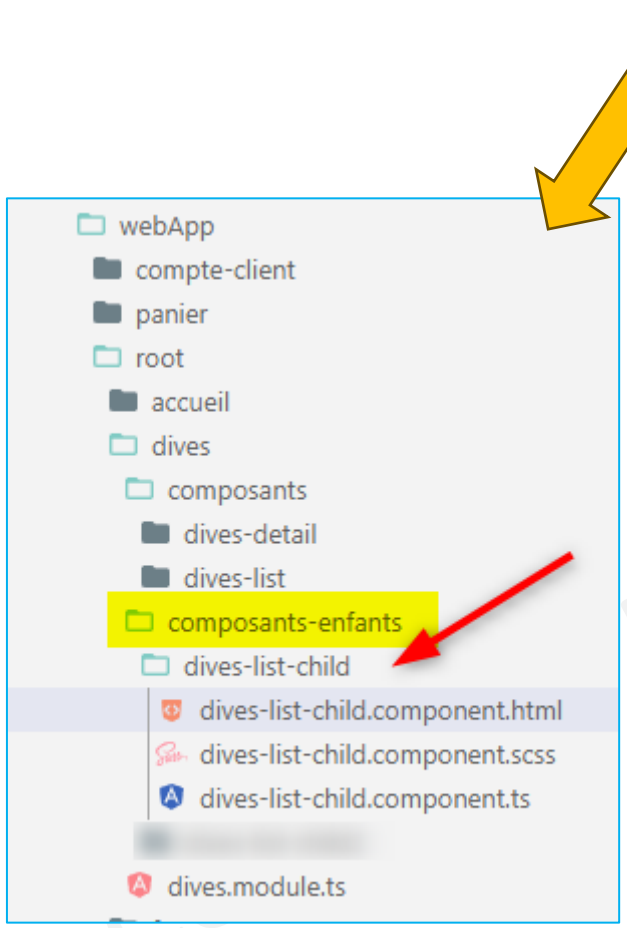


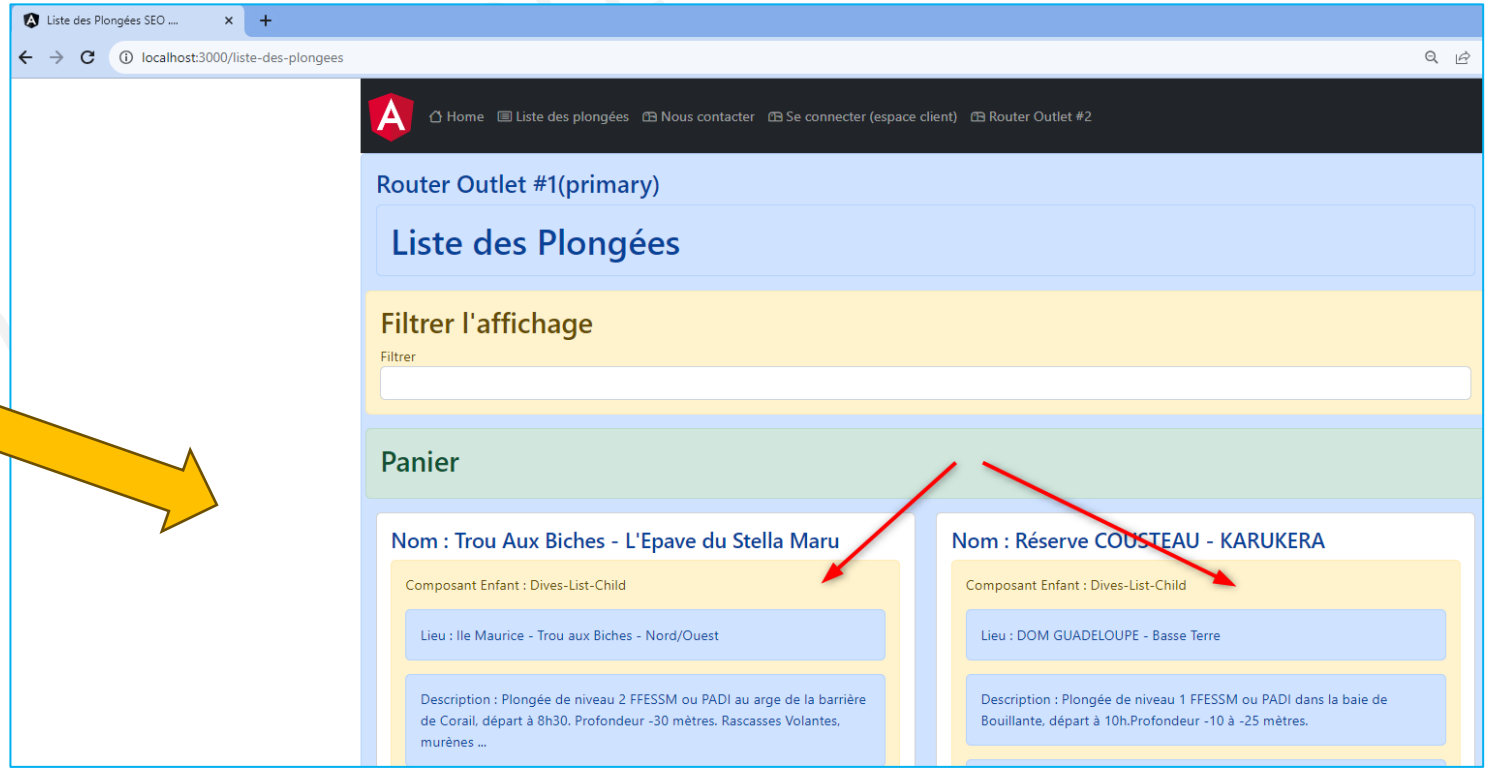
Communication : composants parents et enfants



Communication : composants parents et enfants

TP : déployer une architecture de composants dans la partie « dives » comme illustré ci-dessous





Communication :
composants parents et enfants

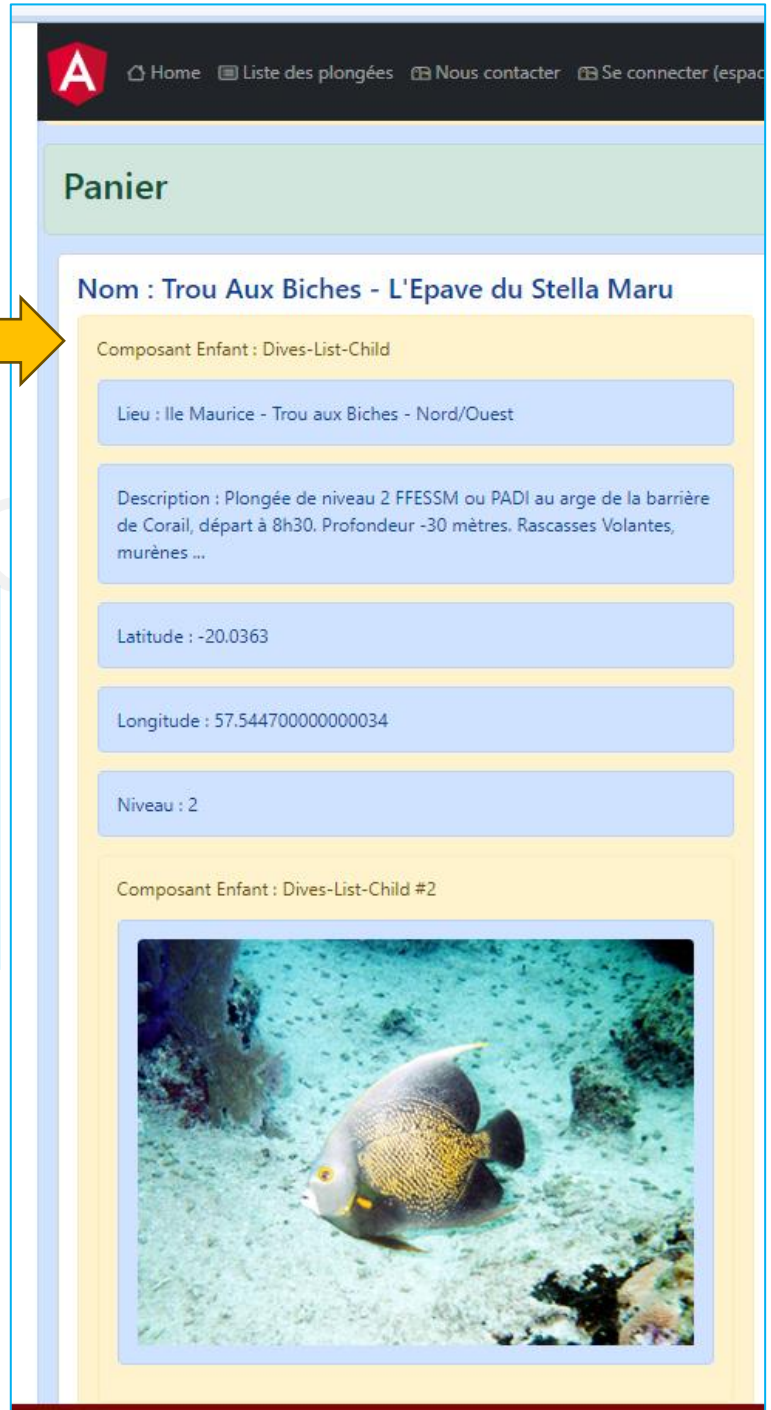
TP : Chaque composant enfant généré par la boucle *ngFor est représenté dans une classe alert-warning.

Pour faire passer une donnés de la vue HTML du composant parent vers le TS du composant enfant, nous utilisons les @Input() qui sont toujours définis dans le composant enfant.

Cet input est utilisé
par le parent comme une directive

```
<!-- Composant Enfant -->
<div class="alert alert-warning">

    <p>Composant Enfant : Dives-List-Child #2</p>
    <app-dives-list-child2
    [inputDive]="inputDive"

    ></app-dives-list-child2>

</div>
<!-- Composant Enfant -->
```

```
9   export class DivesListChildComponent {
10
11  // les @input et les @output permettent d'établir la communication entre comp parents et enfant(s)
12  // ils se définissent toujours dans l'enfant !
13
14  // 1- props
15  // décorateur   nom_du_décorateur:type
16  // le nom du décorateur (devient) une directive dans le parent
17  @Input() inputDive!:Dives;
18  @Input() inputInfos!:string;
19  // ----------------------
20  @Output() outputEventDive:EventEmitter<any>;
21
```



Communication : composants parents et enfants

dives-list.component.ts

```ts
// 4- méthodes
public diveSelected = (e: any) => {
  console.log('depuis le parent $event => ',e);

  let isChecked=e.paramIsChecked;
  let diveSelect=e.paramDive;

  console.log(`Plongée : ${diveSelect.name} - Etat : ${isChecked}`

  if (isChecked) {
    this.panier.push(diveSelect);
    console.table(this.panier);
  } else {
    let keyPanier = this.panier.indexOf(diveSelect);
    if (keyPanier >= 0) {
      this.panier.splice(keyPanier, 1);
      console.table(this.panier);
    }
  }

}
```

dives-list-child.component.ts

```ts
export class DivesListChildComponent {

  // les @input et les @output permettent d'établir la co
  // ils se définissent toujours dans l'enfant !

  // 1- props
  // décorateur   nom_du_décorateur:type
  // le nom du décorateur (devient) une directive dans le
  @Input() inputDive!:Dives;
  @Input() inputInfos!:string;
  // ----------------------
  @Output() outputEventDive:EventEmitter<any>;

  // 2- const
  constructor(){
    this.outputEventDive =  new EventEmitter();
  }

  // 3- méthodes
  public choixDive = (e:any) => {
    console.clear();
    console.log(e.target.checked);
```

dives-list-child2.component.ts

```ts
import { Component, Input } from '@angular/core';
import { Dives } from 'src/app/sharedModels/class

@Component({
  selector: 'app-dives-list-child2',
  templateUrl: './dives-list-child2.component.html
  styleUrls: ['./dives-list-child2.component.scss
})
export class DivesListChild2Component {

  @Input() inputDive!:Dives;

}
```

dives-list.component.html

```html
<div class="card-body">
    <h4 class="card-title">Nom : {{dive.name}} </h

    <!-- Composant Enfant -->
    <div class="alert alert-warning">
        <p>Composant Enfant : Dives-List-Child</p>
        <app-dives-list-child
        [inputDive]="dive"
        [inputInfos]="infos"
        (outputEventDive)="diveSelected($event)"
        ></app-dives-list-child>

    </div>
    <!-- Composant Enfant -->
```

dives-list-child.component.html

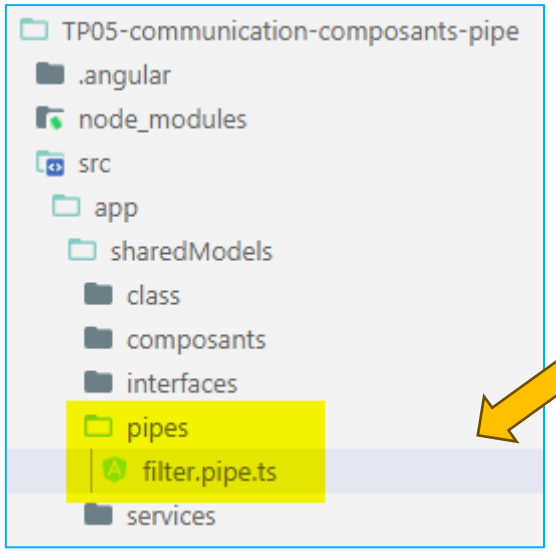
```html
    <p>Composant Enfant : Dives-List-Child #2</p>
    <app-dives-list-child2
    [inputDive]="inputDive"

    ></app-dives-list-child2>

</div>
<!-- Composant Enfant -->


<p class="alert alert-primary">
    <input type="checkbox" (change)="choixDive($event)">
</p>
<!-- $event est l'objet global evenement -->
```

dives-list-child2.component.html

```html
<p class="alert alert-primary">
    <img src="{{inputDive.photo}}" class="card-img

</p>
```

Pipes Angular

Les PIPES Angular



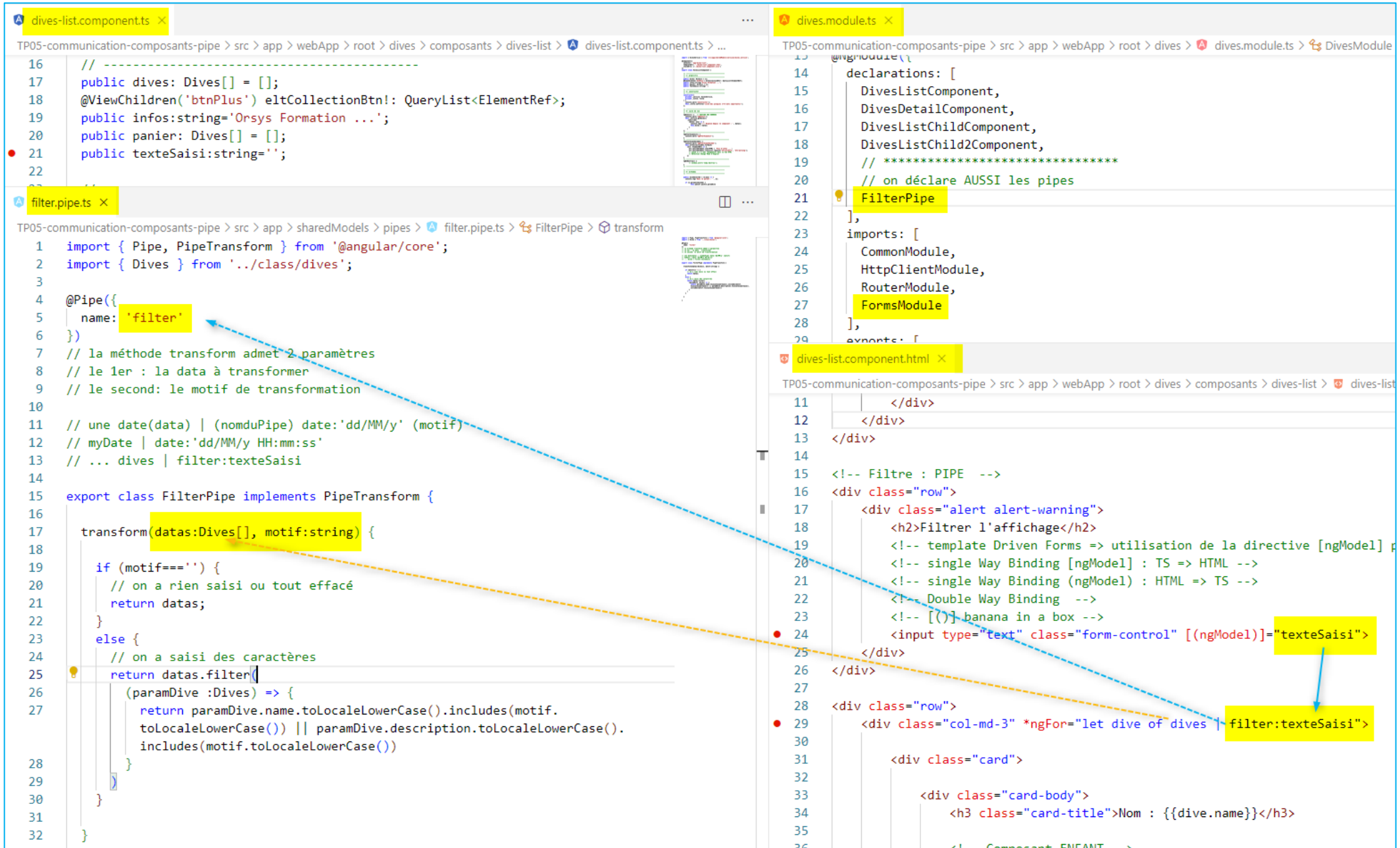
Créer ses propres Pipes avec la commande :

ng g(enerate) p(ipe)

Ce pipe va permettre de filtrer l'affichage des datas en fonction des caractères saisis.

Exemple : Agay

Les PIPES Angular



dives-list.component.ts

TP05-communication-composants-pipe > src > app > webApp > root > dives > composants > dives-list > dives-list.component.ts > ...

```
16    // ----------------------------------------
17    public dives: Dives[] = [];
18    @ViewChildren('btnPlus') eltCollectionBtn!: QueryList<ElementRef>;
19    public infos:string='Orsys Formation ...';
20    public panier: Dives[] = [];
21    public texteSaisi:string='';
22
```

filter.pipe.ts

TP05-communication-composants-pipe > src > app > sharedModels > pipes > filter.pipe.ts > FilterPipe > transform

```
1   import { Pipe, PipeTransform } from '@angular/core';
2   import { Dives } from '../class/dives';
3
4   @Pipe({
5     name: 'filter'
6   })
7   // la méthode transform admet 2 paramètres
8   // le 1er : la data à transformer
9   // le second: le motif de transformation
10
11  // une date(data) | (nomduPipe) date:'dd/MM/y' (motif)
12  // myDate | date:'dd/MM/y HH:mm:ss'
13  // ... dives | filter:texteSaisi
14
15  export class FilterPipe implements PipeTransform {
16
17    transform(datas:Dives[], motif:string) {
18
19      if (motif==='') {
20        // on a rien saisi ou tout effacé
21        return datas;
22      }
23      else {
24        // on a saisi des caractères
25        return datas.filter(
26          (paramDive :Dives) => {
27            return paramDive.name.toLocaleLowerCase().includes(motif.
              toLocaleLowerCase()) || paramDive.description.toLocaleLowerCase().
              includes(motif.toLocaleLowerCase())
28          }
29        )
30      }
31
32    }
```

dives.module.ts

TP05-communication-composants-pipe > src > app > webApp > root > dives > dives.module.ts > DivesModule

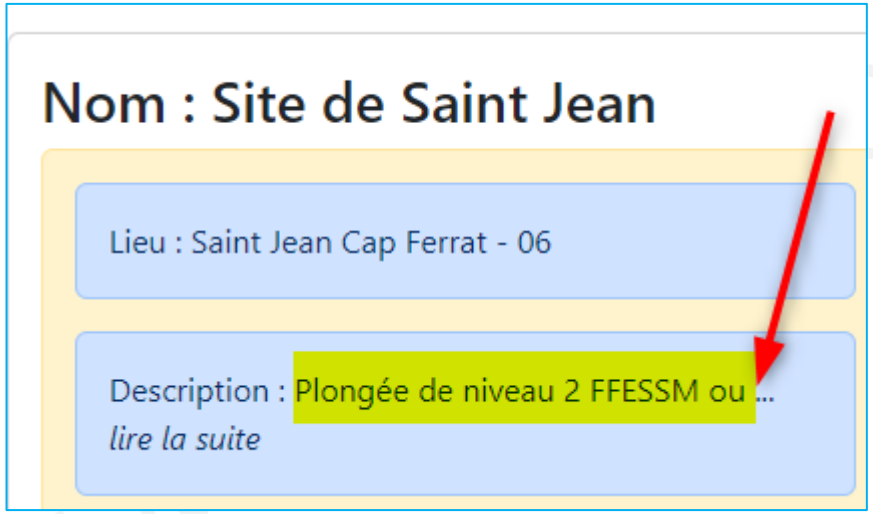
```
14    declarations: [
15      DivesListComponent,
16      DivesDetailComponent,
17      DivesListChildComponent,
18      DivesListChild2Component,
19      // ********************************
20      // on déclare AUSSI les pipes
21      FilterPipe
22    ],
23    imports: [
24      CommonModule,
25      HttpClientModule,
26      RouterModule,
27      FormsModule
28    ],
```

dives-list.component.html

TP05-communication-composants-pipe > src > app > webApp > root > dives > composants > dives-list > dives-list

```
11          </div>
12        </div>
13    </div>
14
15    <!-- Filtre : PIPE  -->
16    <div class="row">
17        <div class="alert alert-warning">
18            <h2>Filtrer l'affichage</h2>
19            <!-- template Driven Forms => utilisation de la directive [ngModel] p
20            <!-- single Way Binding [ngModel] : TS => HTML -->
21            <!-- single Way Binding (ngModel) : HTML => TS -->
22            <!-- Double Way Binding  -->
23            <!-- [()] banana in a box -->
24            <input type="text" class="form-control" [(ngModel)]="texteSaisi">
25        </div>
26    </div>
27
28    <div class="row">
29        <div class="col-md-3" *ngFor="let dive of dives | filter:texteSaisi">
30
31            <div class="card">
32
33                <div class="card-body">
34                    <h3 class="card-title">Nom : {{dive.name}}</h3>
35
```

TP : Créer un pipe qui affiche seulement les X premiers caractères de la description avec la méthode javascript substring()



TP : Créer un pipe qui affiche seulement les x premiers caractères (20 par ex) et ajoute un petit texte (ex : ... lire la suite) de la description avec la méthode javascript substring()

dives.module.ts

TP05-communication-composants-pipe > src > app > webApp > root > dives > dives.module.ts > DivesModule

```ts
import { VoirPlusPipe } from 'src/app/sharedModels/pipes/voir-plus.pipe';


@NgModule({
  declarations: [
    DivesListComponent,
    DivesDetailComponent,
    DivesListChildComponent,
    DivesListChild2Component,
    // ********************************
    // on déclare AUSSI les pipes
    FilterPipe,
    VoirPlusPipe
  ],
  imports: [
    CommonModule,
    HttpClientModule,
```

voir-plus.pipe.ts

TP05-communication-composants-pipe > src > app > sharedModels > pipes > voir-plus.pipe.ts > VoirPlusPipe > transform

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

@Pipe({
  name: 'voirPlus'
})
export class VoirPlusPipe implements PipeTransform {

  transform(datas:string, nbCars:number) {
    return `${datas.substring(0, nbCars)} ...`;
  }

}
```

dives-list-child.component.html

...munication-composants-pipe > src > app > webApp > root > dives > composants-enfant > dives-list-child > dives-list-child.component.html > p.al

```html
<p class="alert alert-primary">Lieu : {{inputDive.location}}</p>


<p class="alert alert-primary">Description : {{inputDive.description | voirPlus:30 }} <em>lire la suite</em></p>


<p class="alert alert-primary">Latitude : {{inputDive.latitude}}</p>
<p class="alert alert-primary">Longitude : {{inputDive.longitude}}</p>
<p class="alert alert-primary">Niveau : {{inputDive.level}}</p>

<p class="alert alert-danger">{{inputInfos}}</p>

<!-- appel enfant -->
<app-dives-list-child2
[inputDive2]="inputDive"


></app-dives-list-child2>

<p class="alert alert-primary">
    <input type="checkbox" (change)="choixDive($event)">
    <!-- $event = EVENT GLOBAL -->
</p>
```

PROBLÈMES SORTIE CONSOLE DE DÉBOGAGE TERMINAL PORTS

powershell - pipes

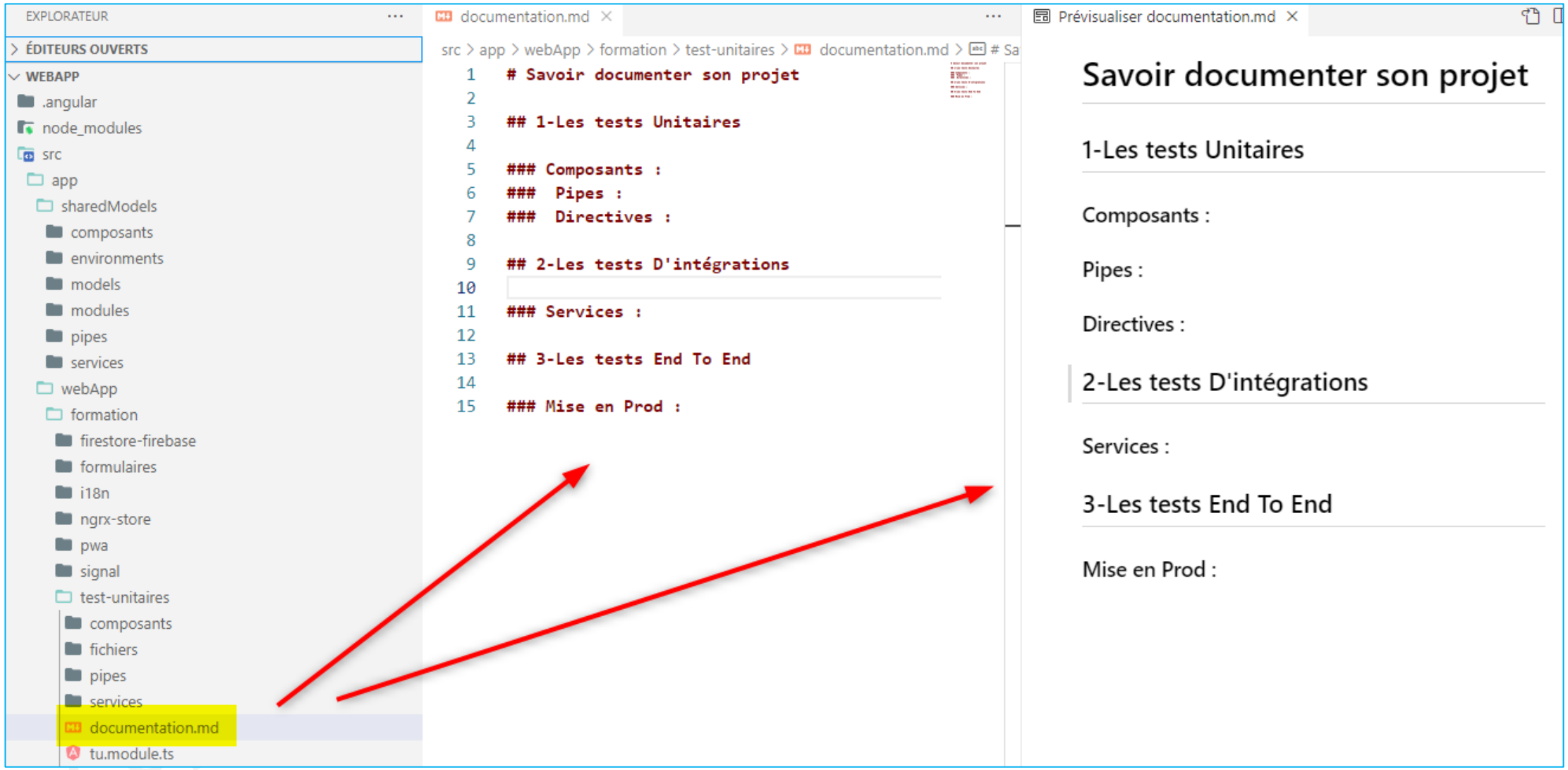
```
PS C:\Users\Michel\Desktop\Formation-angular-alteca-10-2023\TP05-communication-composants-pipe\src\app\sharedModels\pipes> ng g p voir-plus --skip-tests --skip-import
```

Les Tests Unitaires

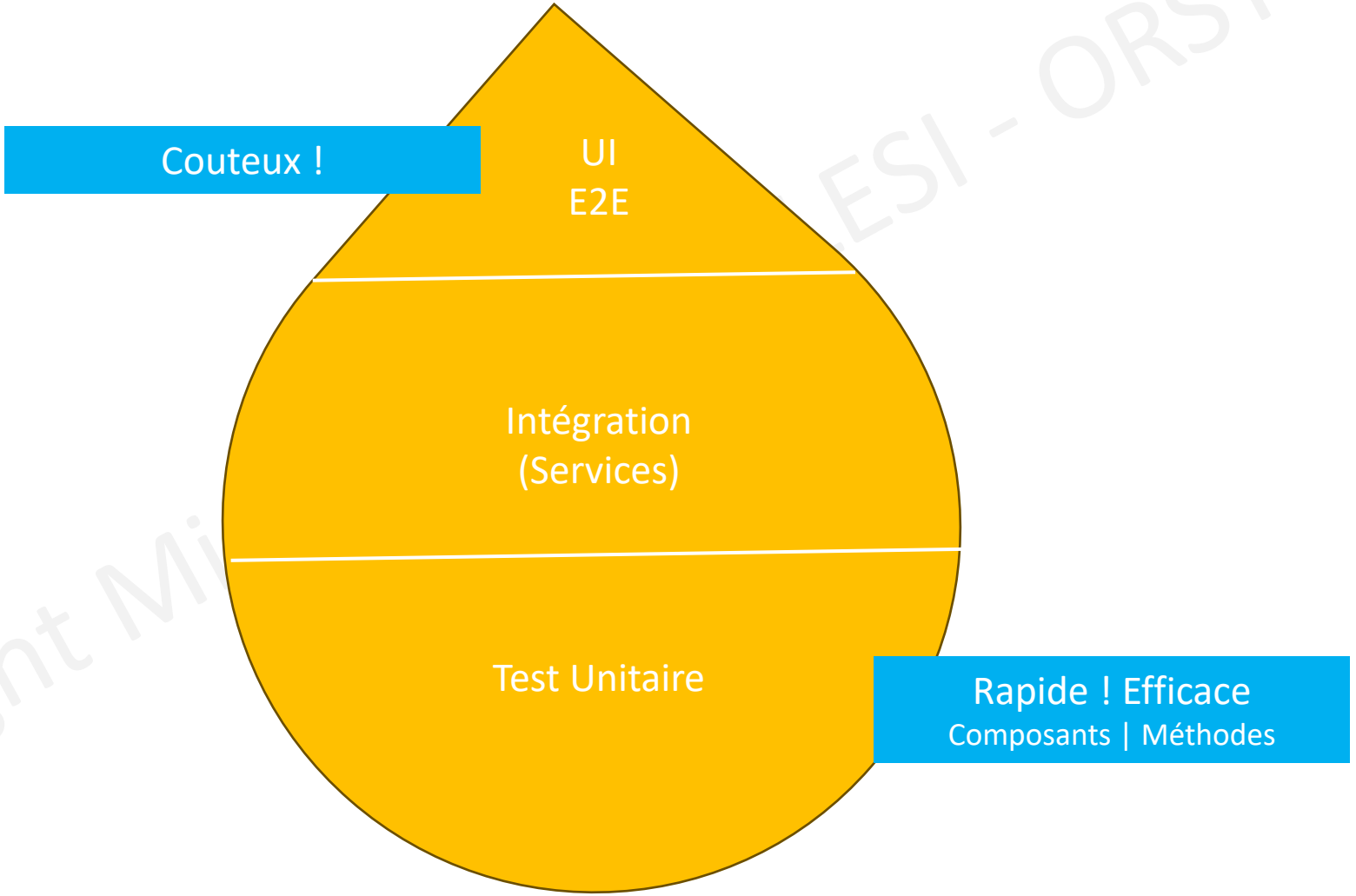
Le Test

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 :

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

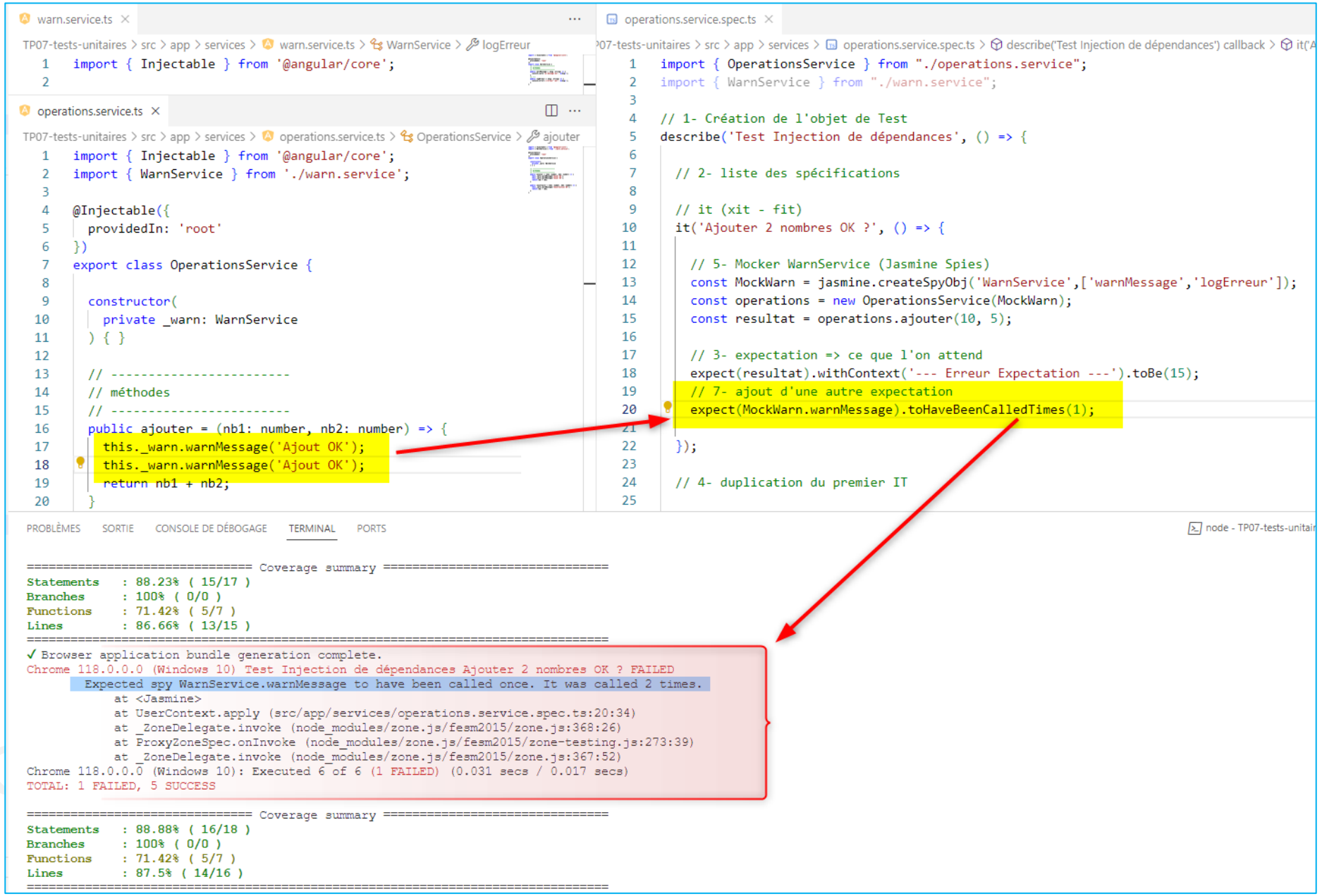
```ts
import { Injectable } from '@angular/core';
import { WarnService } from './warn.service';

@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);
  });
```

Les Formulaires

Les formulaires Angular (avec Matérial Design)

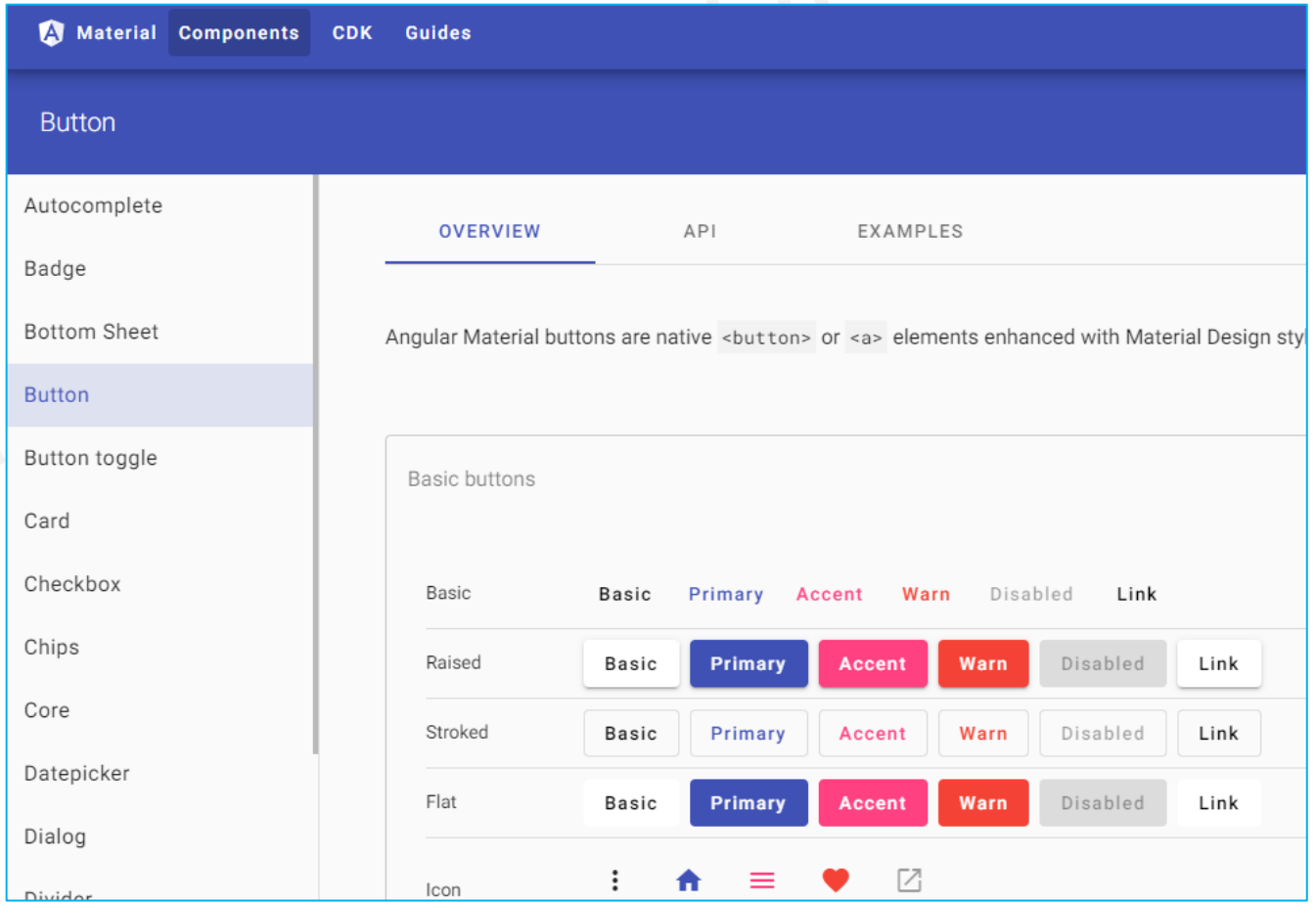
- https://material.angular.io/components/categories
- https://fonts.google.com/icons

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)

Angular est un framework « **orienté** de ce fait beaucoup de packages peuvent s'installer grâce à la commande « **ng add** »

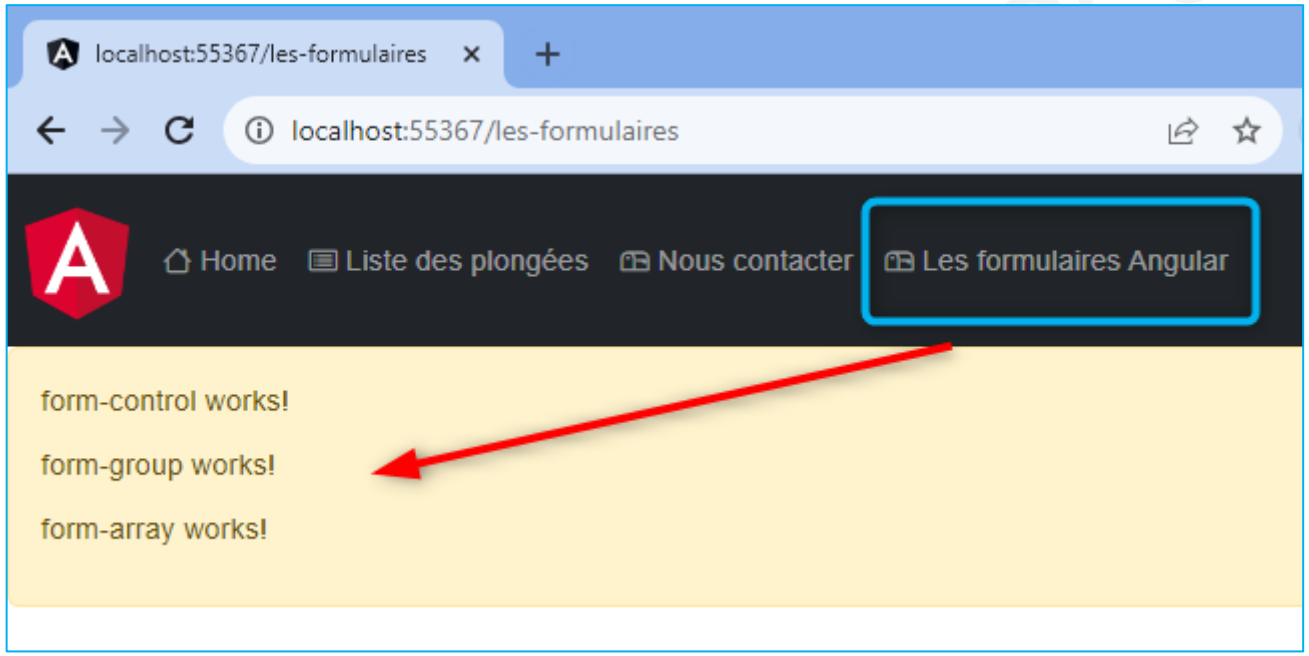
ng add @angular/material

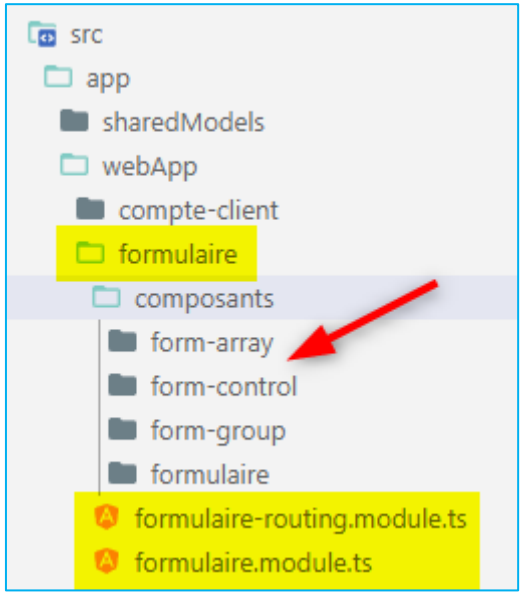
ng add @angular/localize (I18N : internationalisation)
ng add @angular/pwa (Progressive web app)
Ng add @ngrx/store (Le store Angular)

Les formulaires Angular (avec Matérial Design)

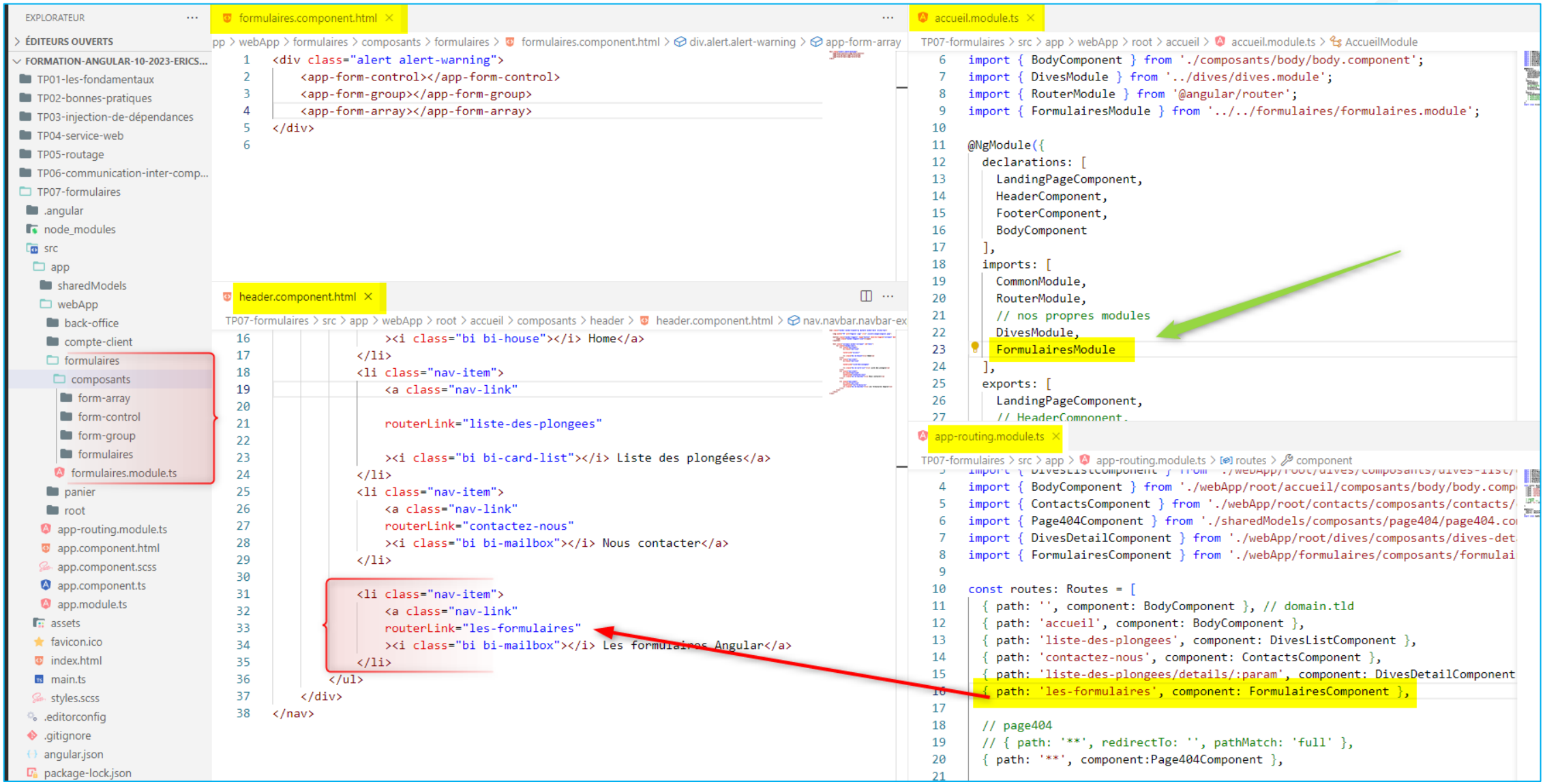
https://dev.webjs.fr/1-Angular/ressources-formation/material.module.ts
https://dev.webjs.fr/1-Angular/ressources-formation/material-all.module.ts

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

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)
      ]
    );
  }

}

```

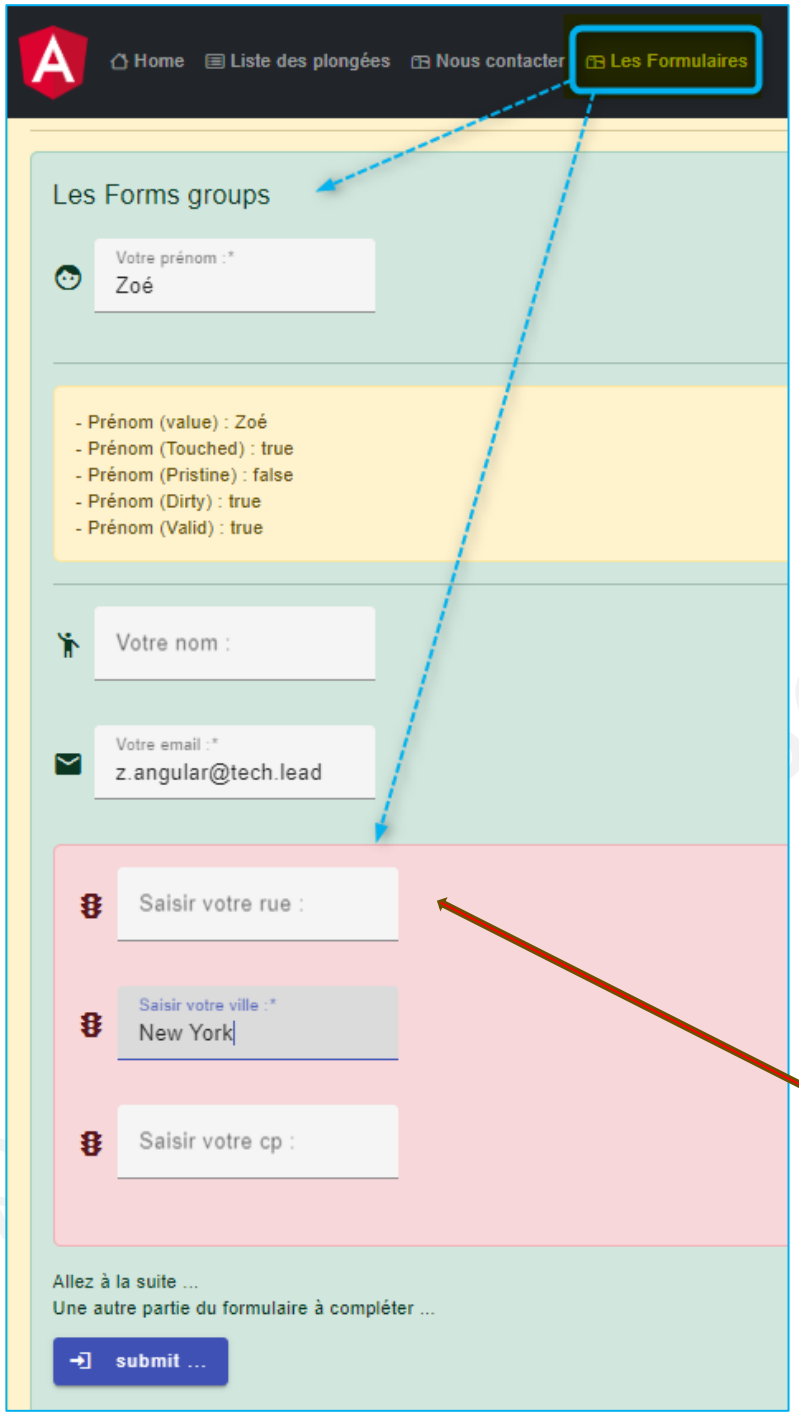
form-control.component.html

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

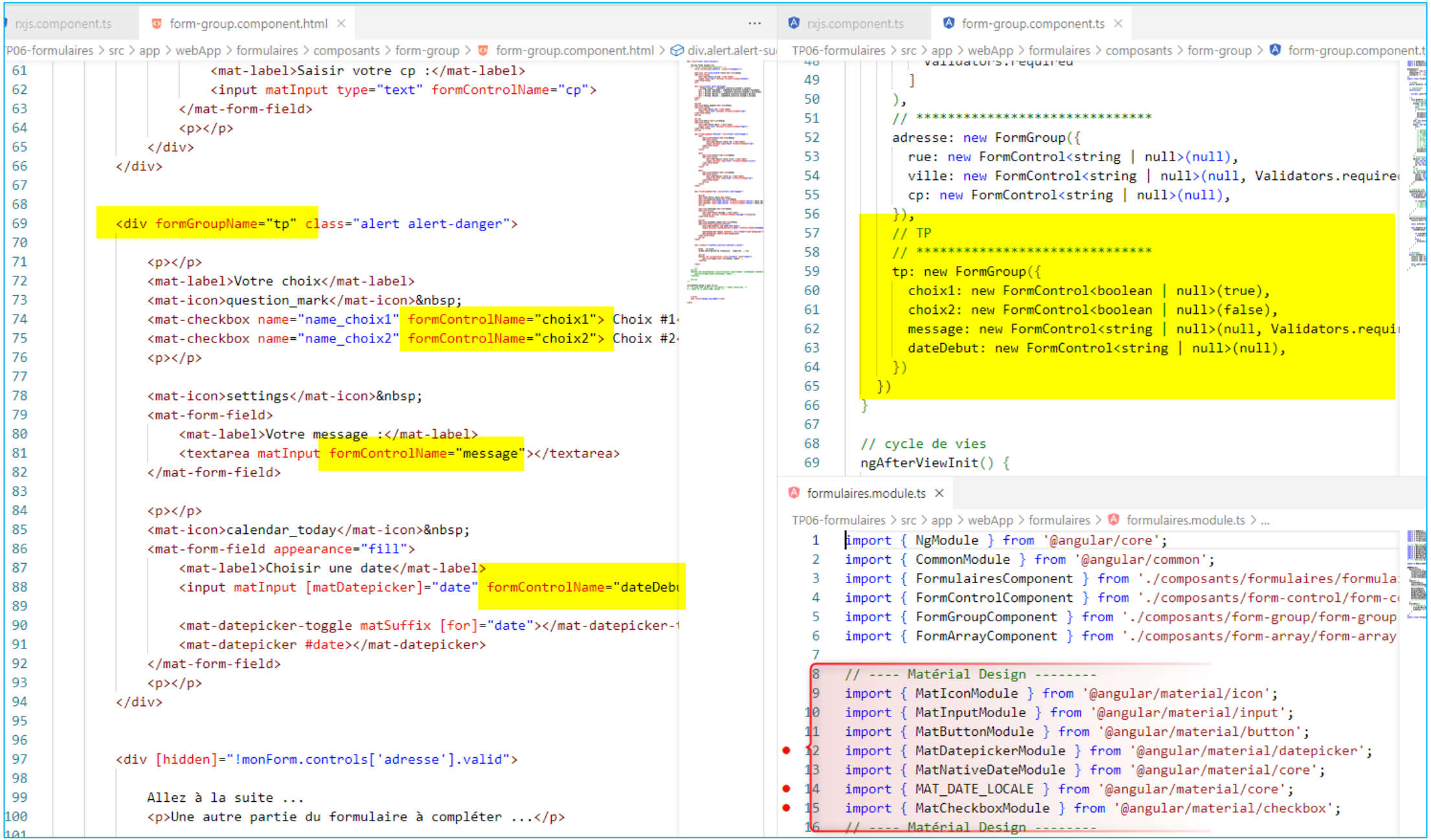
3- Créer le service « post.service.ts » qui poste les datas sur un json-server

```
"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 > ...

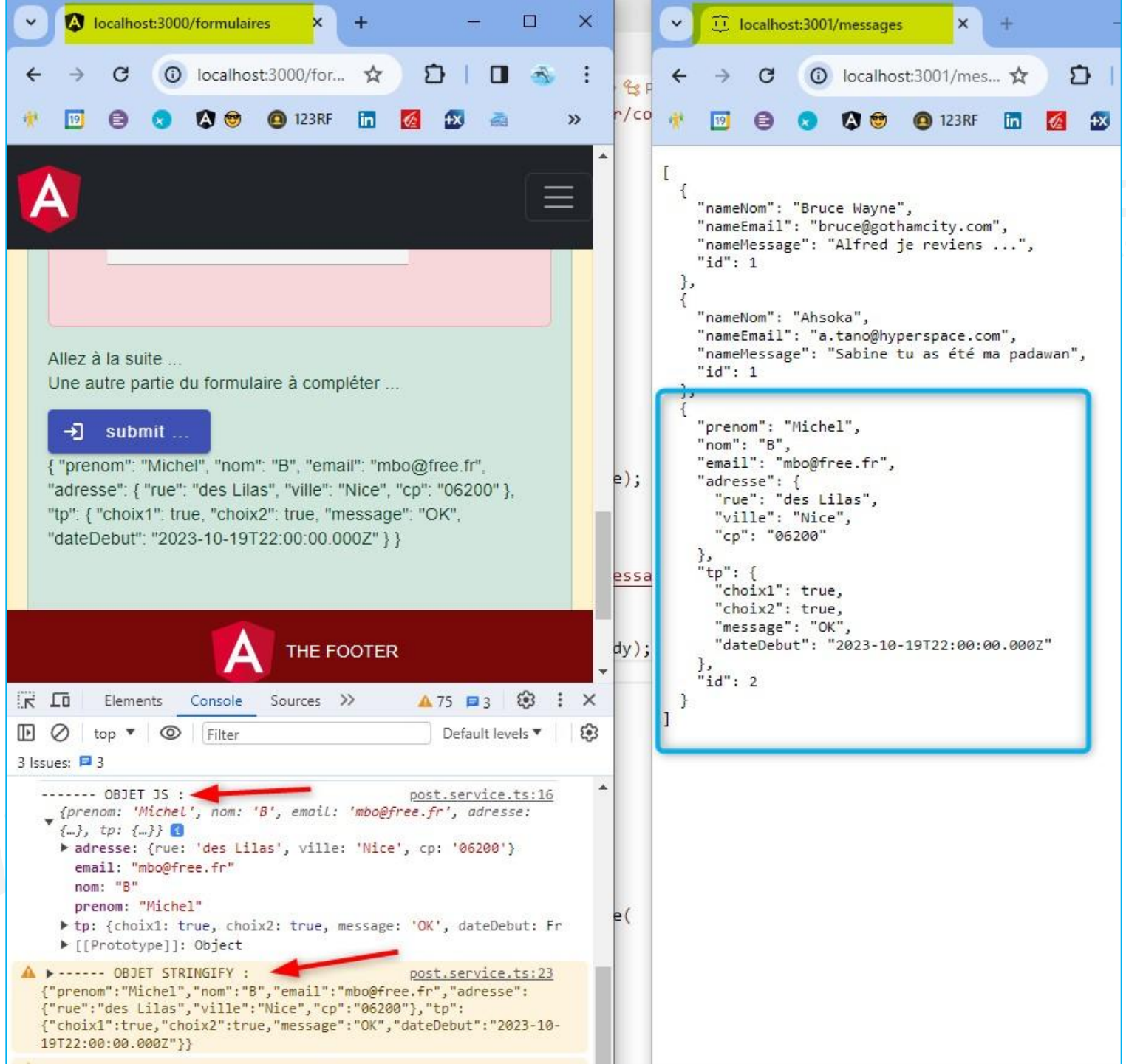
```
1   import { NgModule } from '@angular/core';
2   import { CommonModule } from '@angular/common';
3   import { FormulairesComponent } from './composants/formulaires/formula
4   import { FormControlComponent } from './composants/form-control/form-c
5   import { FormGroupComponent } from './composants/form-group/form-group
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

```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>
```

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

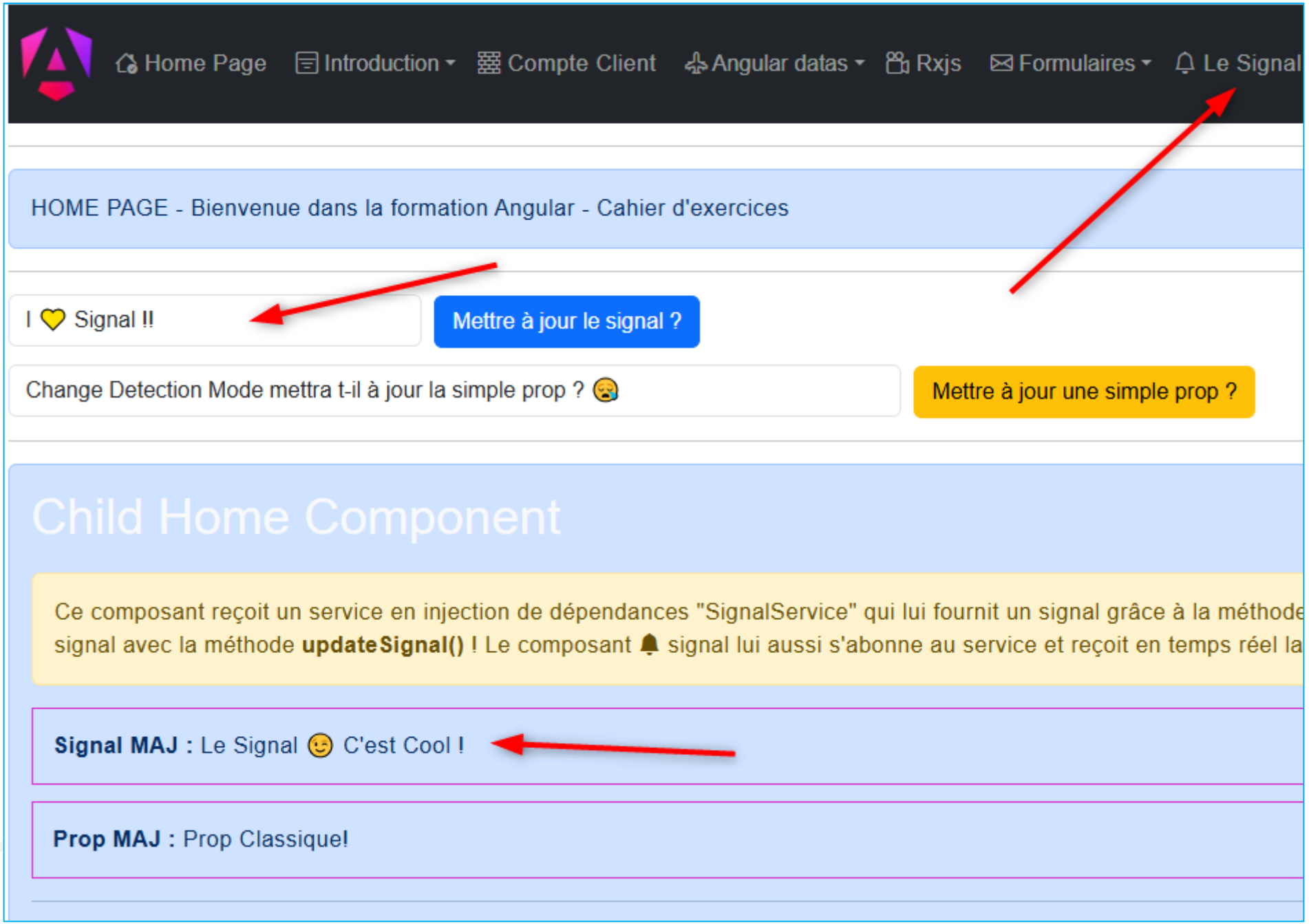
1- Le signal permet de réduire le nombre de mise à jour effectuées lors du "Change Detection Mode" d'Angular !
La réactivité est performante ! Plus besoin de Zone.js ! Zone va ou est en train d'être abandonné ! ✨🎈
Interopérabilité avec RXJS : méthodes toSignal() et toObservable()

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()
  }
}
```

Rxjs – Observables
Approfondissement
Niveau 2

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 le gestion des requêtes asynchrones.

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)

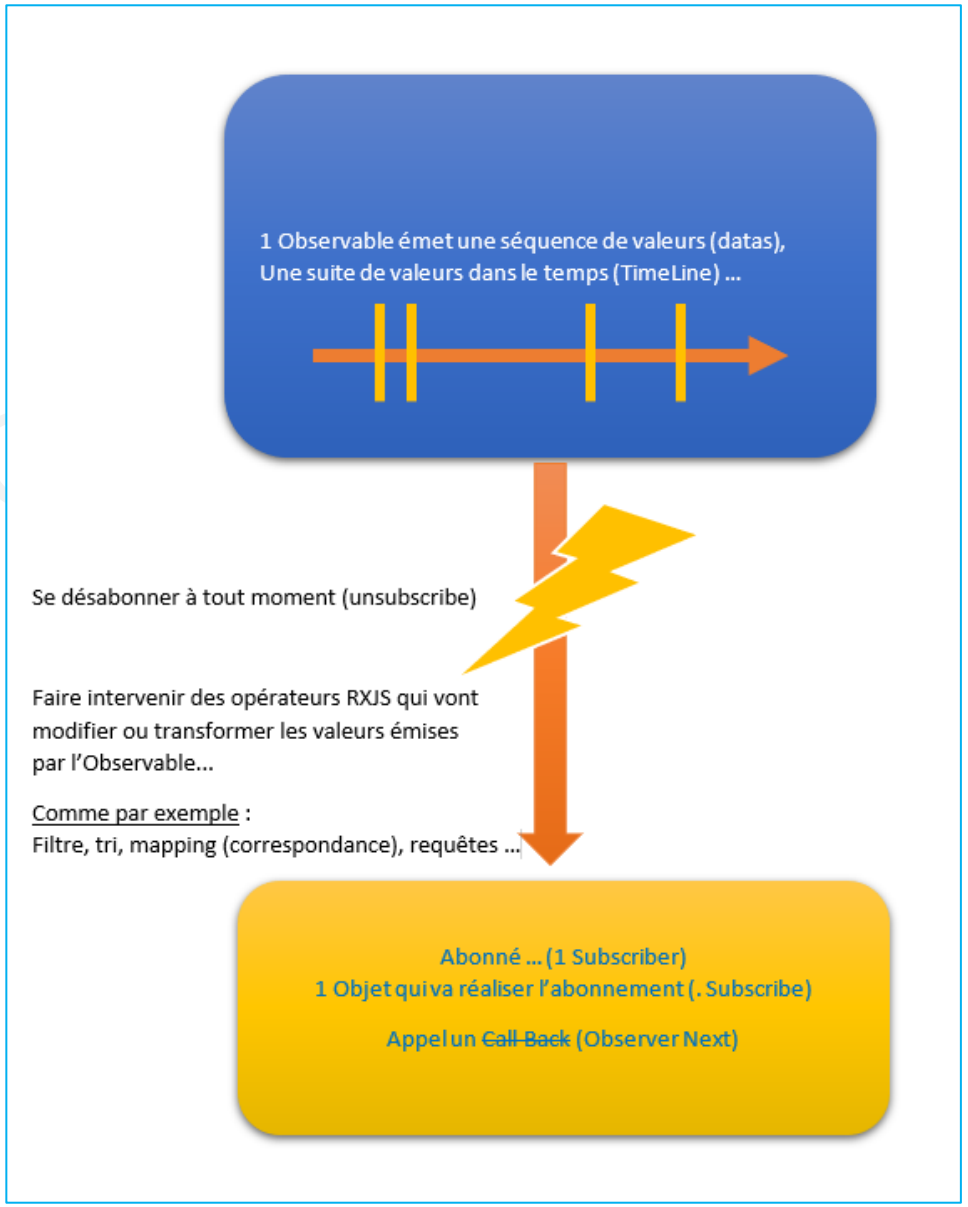
Opérateurs : des fonctions tap, map, merge, filter qui vont pouvoir modifier/transformer ce flux de valeurs émises par l'observable

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 !

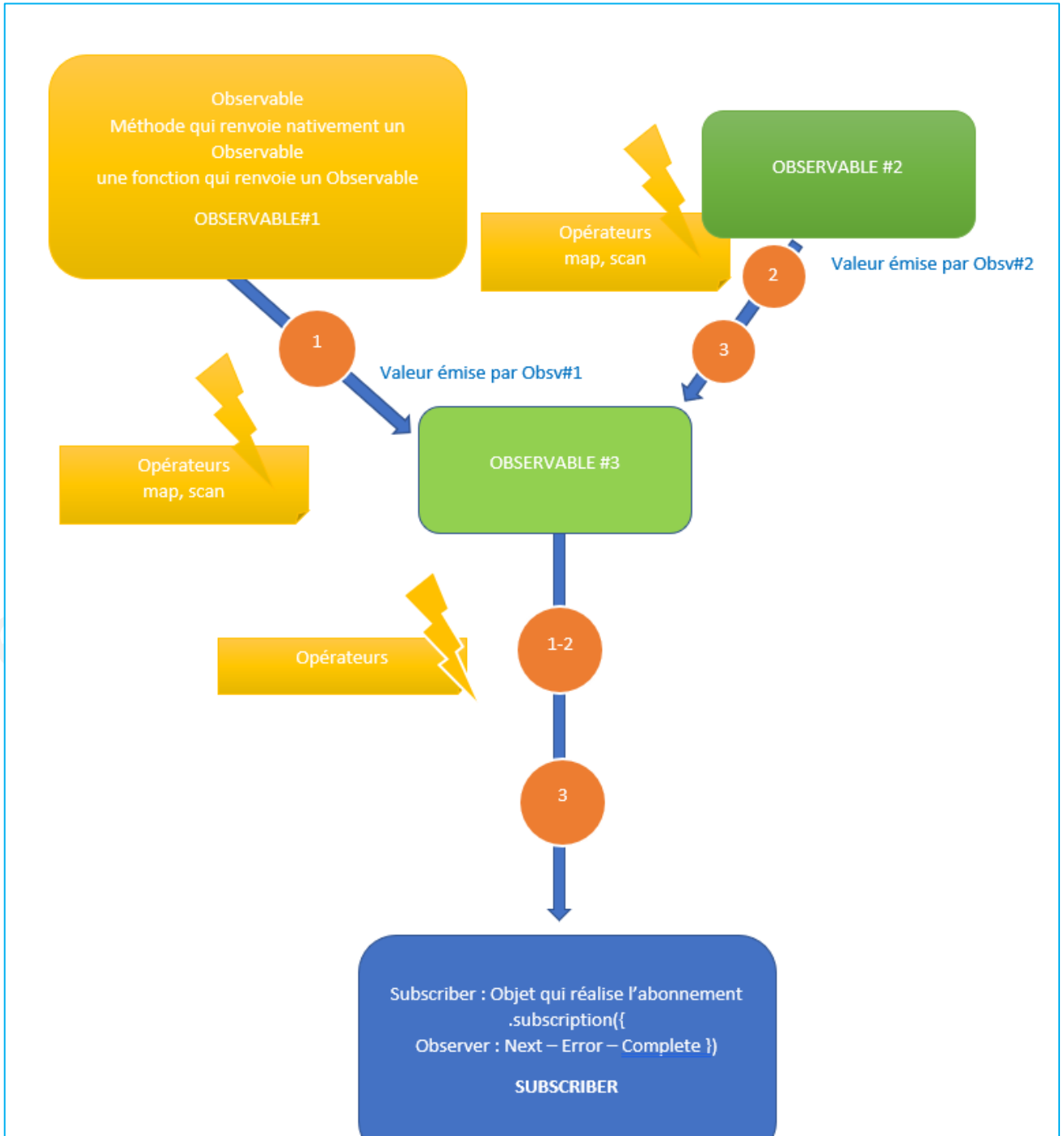
Les séquences de valeurs émises sont facilement annulables !

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

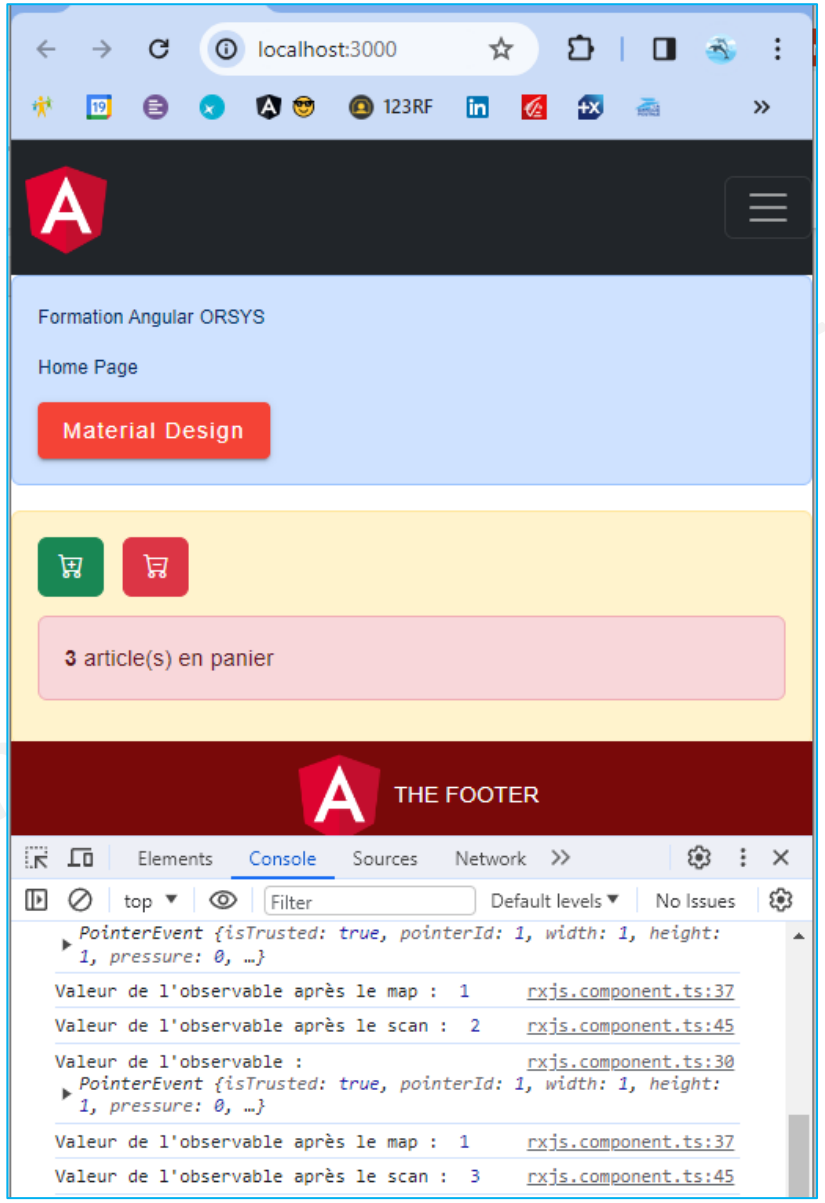


RXJS et les OBSERVABLES

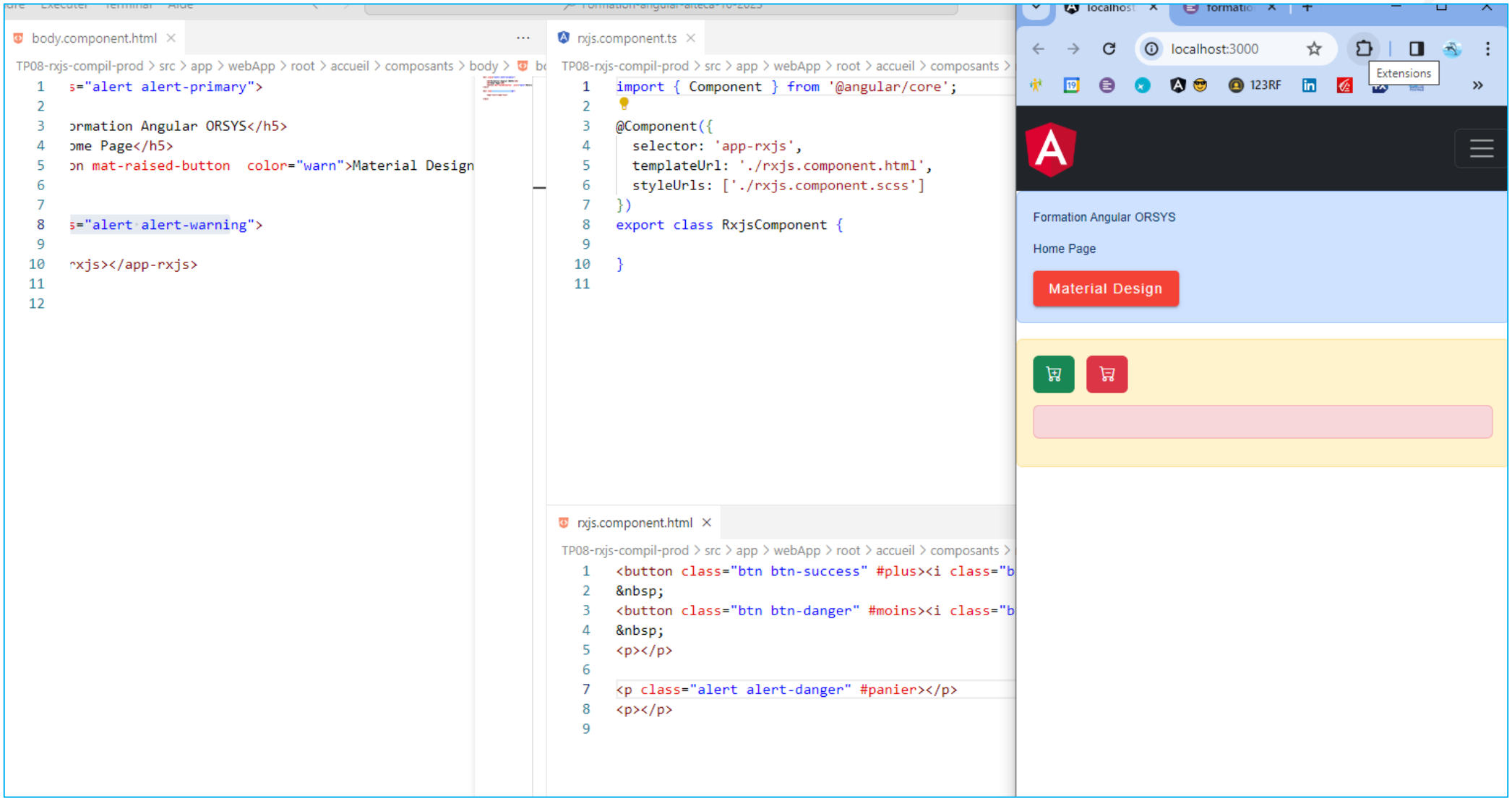
On peut facilement les recomposer
ou les recombiner pour former
une nouvelle séquence de valeurs !



RXJS et les OBSERVABLES



RXJS et les OBSERVABLES



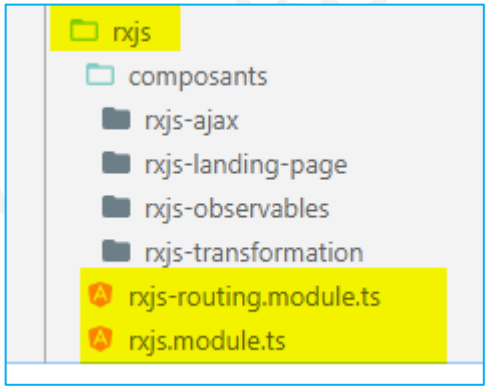
```
// ----------------------------------------
// création de l'observable #1 : PLUS
// ----------------------------------------
const plus$ = fromEvent(this.eltPlus.nativeElement, 'click')
  .pipe(
    startWith(0),
    tap(
      (valObs) => console.log(`Valeur de l'observable : `, valObs)
      // 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';
      }
    }
  );
```

RXJS et les OBSERVABLES

TP : mise en place d'une architecture de projet avec le module « formulaires » comme illustré ci-dessous.



```
// 4- Methodes
public createObservable1: any = () => {

  const eltUl = <HTMLElement>document.querySelector('#formations');
  const formationsArray: string[] = ['NG15', 'REACT 17', 'PWA', 'XML', 'JSON',

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

  // abonnement à cet observable
  this.subArrayFormation = formation$.subscribe(
    // code OK success
    // la notion d'observer : ext - error - complete
    {
      next:
        (formation: string) => {
          console.log(formation);
          const eltLi = <HTMLElement>document.createElement('li');
          eltLi.innerHTML = formation;
          eltLi.className = 'list-group-item';
          eltUl.appendChild(eltLi);
        }
      ,
      error:
        (e) => {
          console.log(e);
        }
      ,
      complete:
        () => console.warn('Complete')
    }
  );
}
```

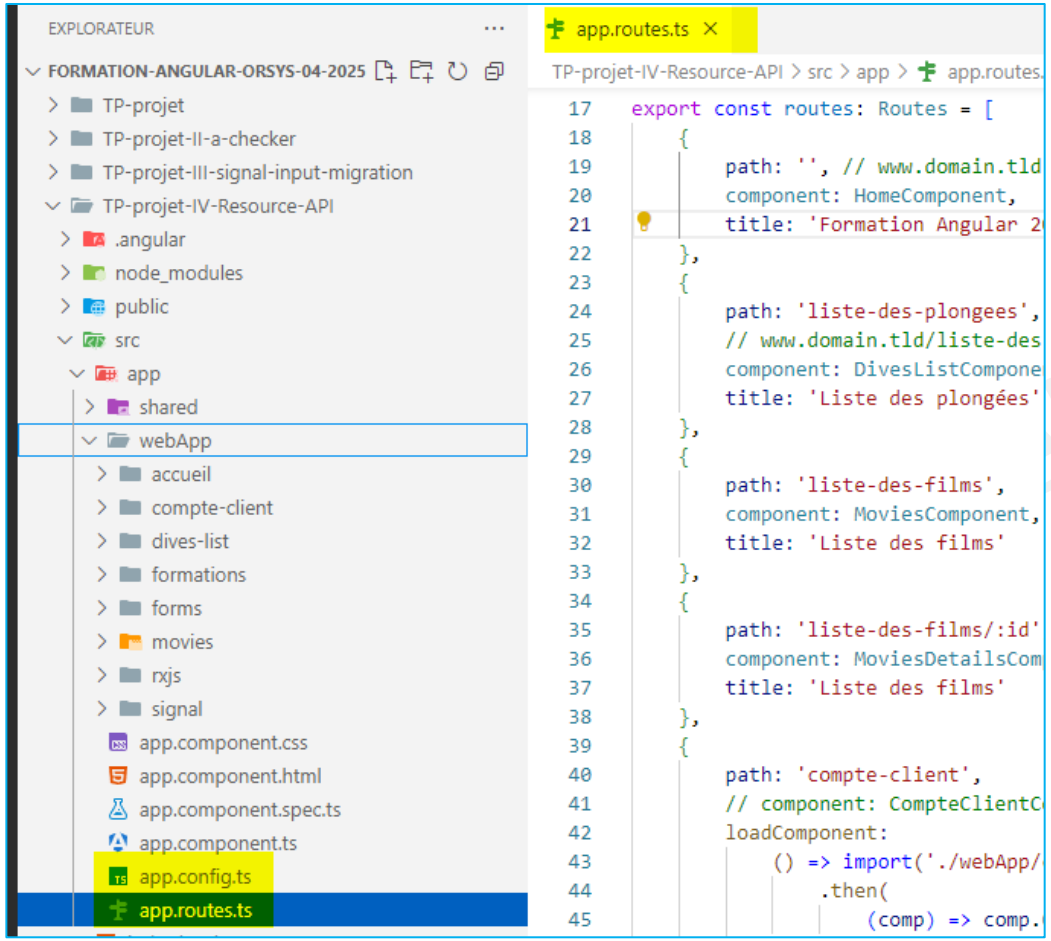
```
<button class="btn btn-warning" (click)="createObservable1()">Créer un Observable</button>
<ul id="formations" class="list-group"></ul>

<p></p>


<button class="btn btn-success" (click)="createObservable2()">Créer un Observable</button>
<!-- template référence Angular # -->
<p></p>
<ul #formations2 class="list-group"></ul>
```

```ts
// ---------------------------------------------
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),
      filter(
        (formation: Formation) => {
          return formation.duree === 4;
        }
      ),
      tap(
        // observer NEXT
        (formation: Formation) => {
          console.warn(formation);
          const eltLi = <HTMLElement>document.createElement('li');
          eltLi.innerHTML = `${formation.cours} : ${formation.duree} jours.`;
          eltLi.className = 'list-group-item';
          // nativeElement est obligatoire ici car on récupère l'élément par Vie
          this.eltUl2.nativeElement.appendChild(eltLi);
        }
      ),
      catchError(
        // capturer les erreurs de l'observable
        (e: any): any => {
          console.log(e);

        }
      )
    )
    .subscribe(
    // // observer NEXT
    // (formation: Formation) => {
    //   console.log(formation);
    //   const eltLi = <HTMLElement>document.createElement('li');
    //   eltLi.innerHTML = `${formation.cours} : ${formation.duree} jours.`;
    //   eltLi.className = 'list-group-item';
    //   // nativeElement est obligatoire ici car on récupère l'élément par View
    //   this.eltUl2.nativeElement.appendChild(eltLi);
    // }
  )
    .unsubscribe();
```

ANNEXES

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
```

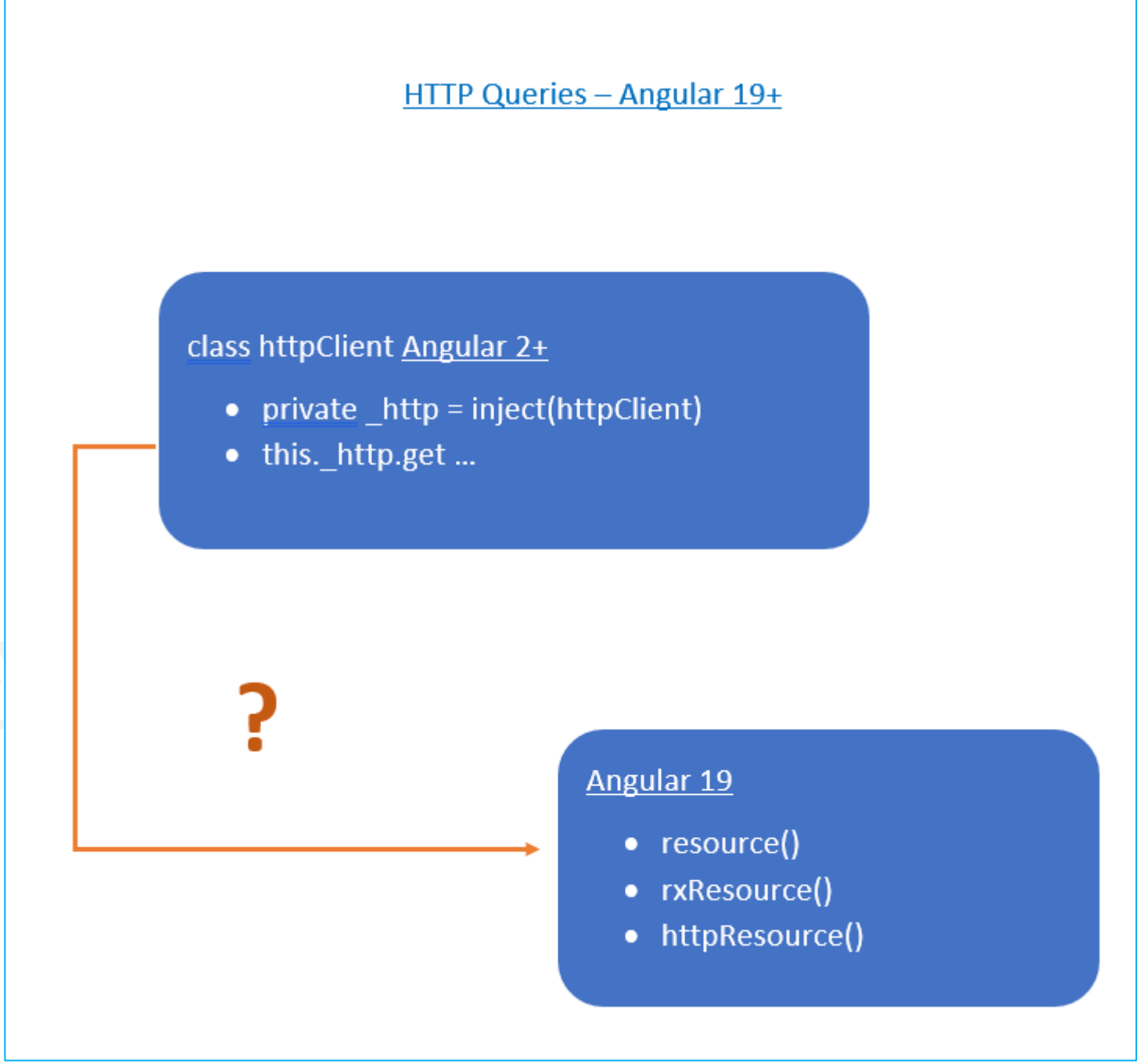
- New Refactoring -> Input & Output -> Signal Use

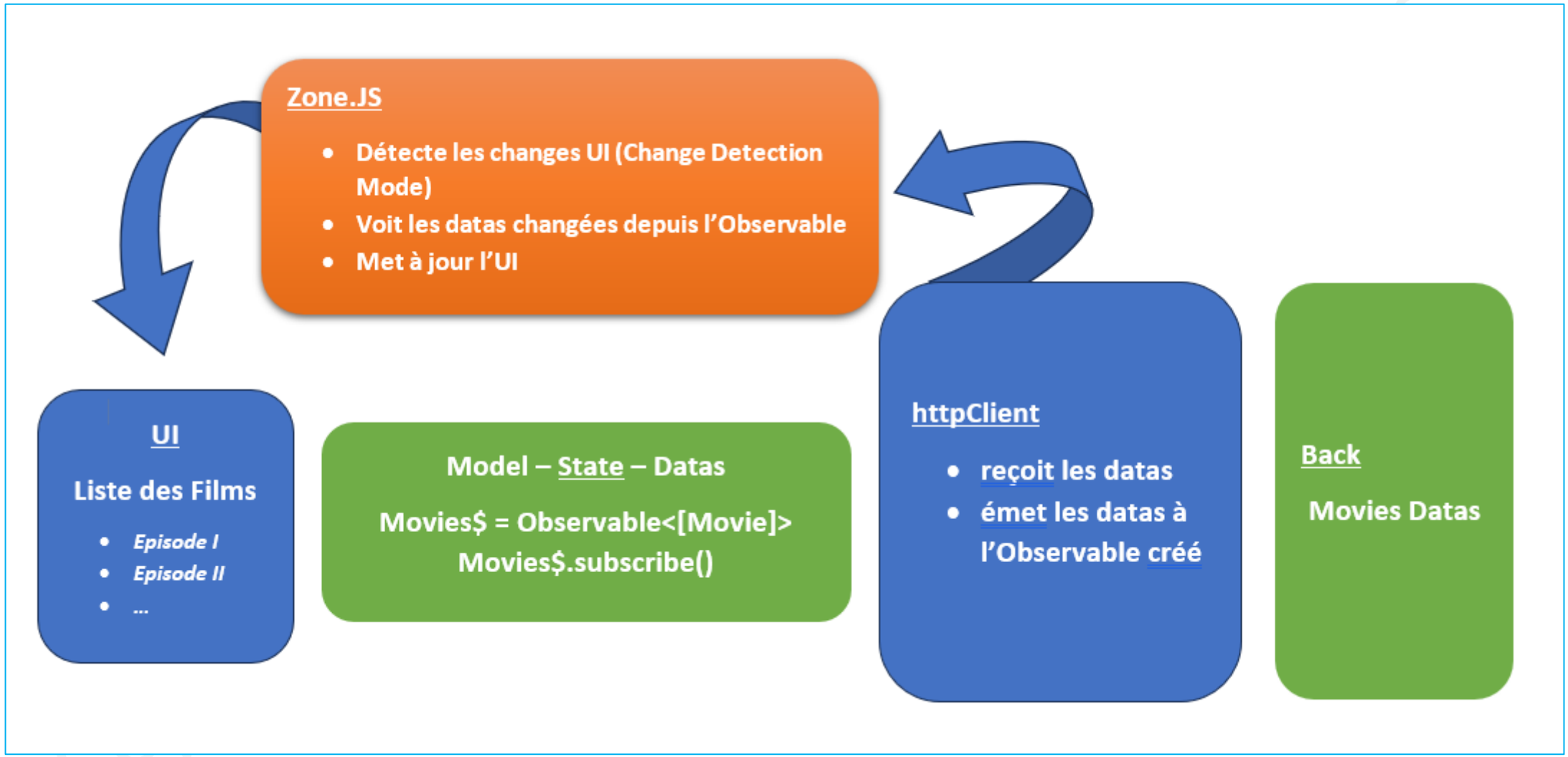
```
ng g @angular/core:signal-input-migration
```

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

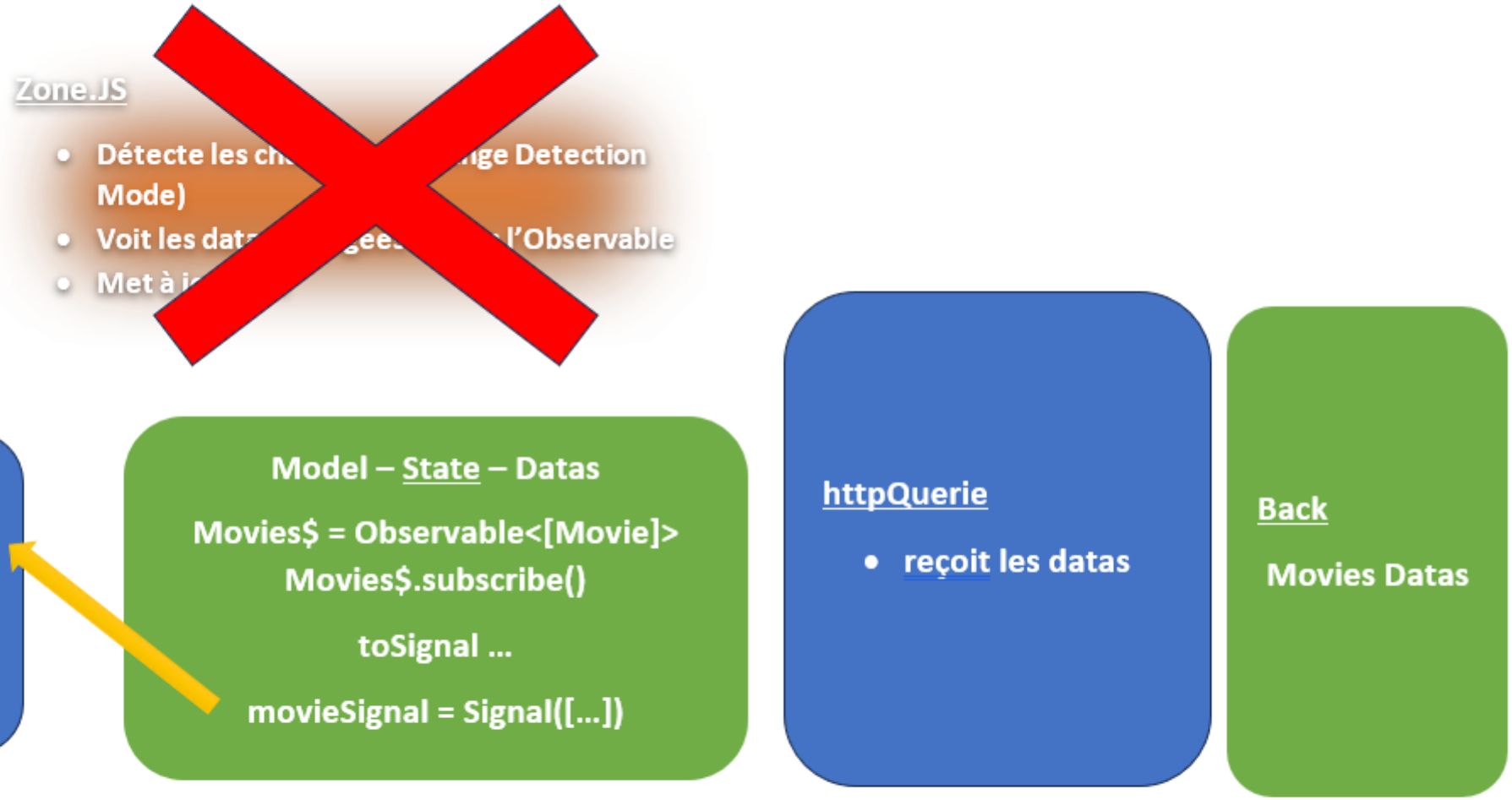
Angular 19 – API Resource

API Resource Pourquoi ?



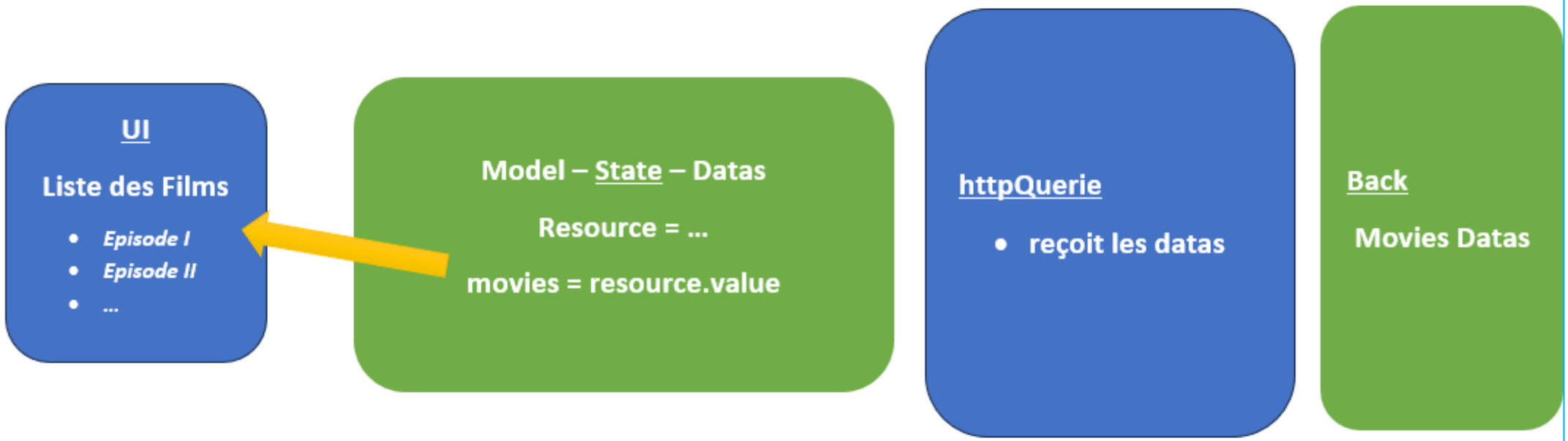

Zone.JS
Détecte les changes UI (Change Detection Mode)
Voit les datas changées depuis l'Observable
Met à jour l'UI
UI
Liste des Films
Episode I
Episode II
...
Model – State – Datas
Movies$ = Observable<[Movie]>
Movies$.subscribe()
httpClient
reçoit les datas
émet les datas à l'Observable créé
Back
Movies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

SIGNAL & RESOURCE API



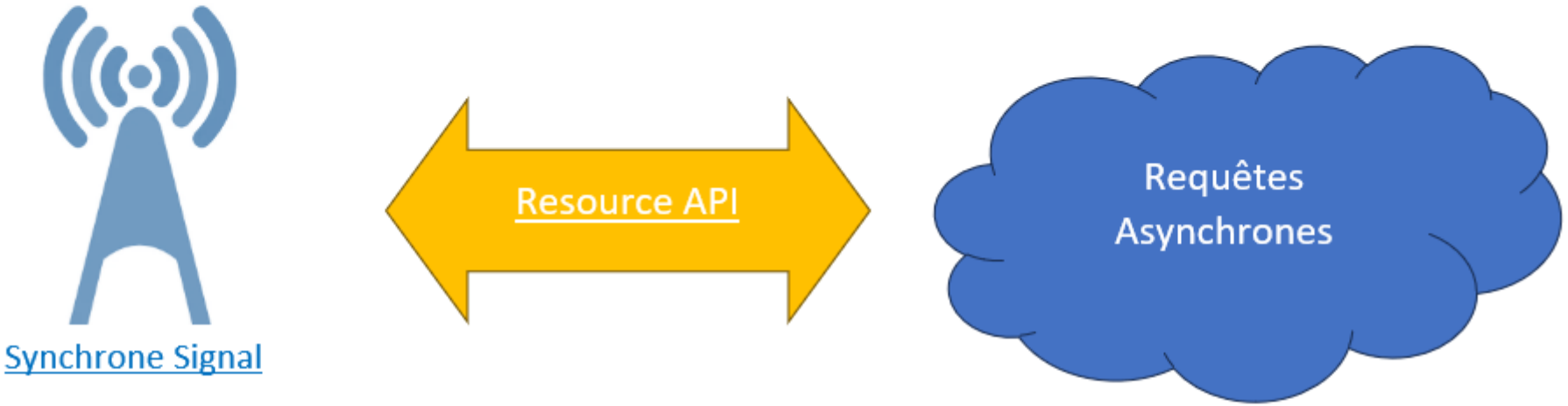
Le signal (SYNCHRONE) n'utilise plus Zone.JS et peut mettre à jour directement l'UI. Mais Angular 19+ a introduit une solution encore plus pratique (Resource API), qui est une sorte de pont entre la requête asynchrone http et le signal qui est synchrone ! Plus besoin de Subscription, les datas peuvent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

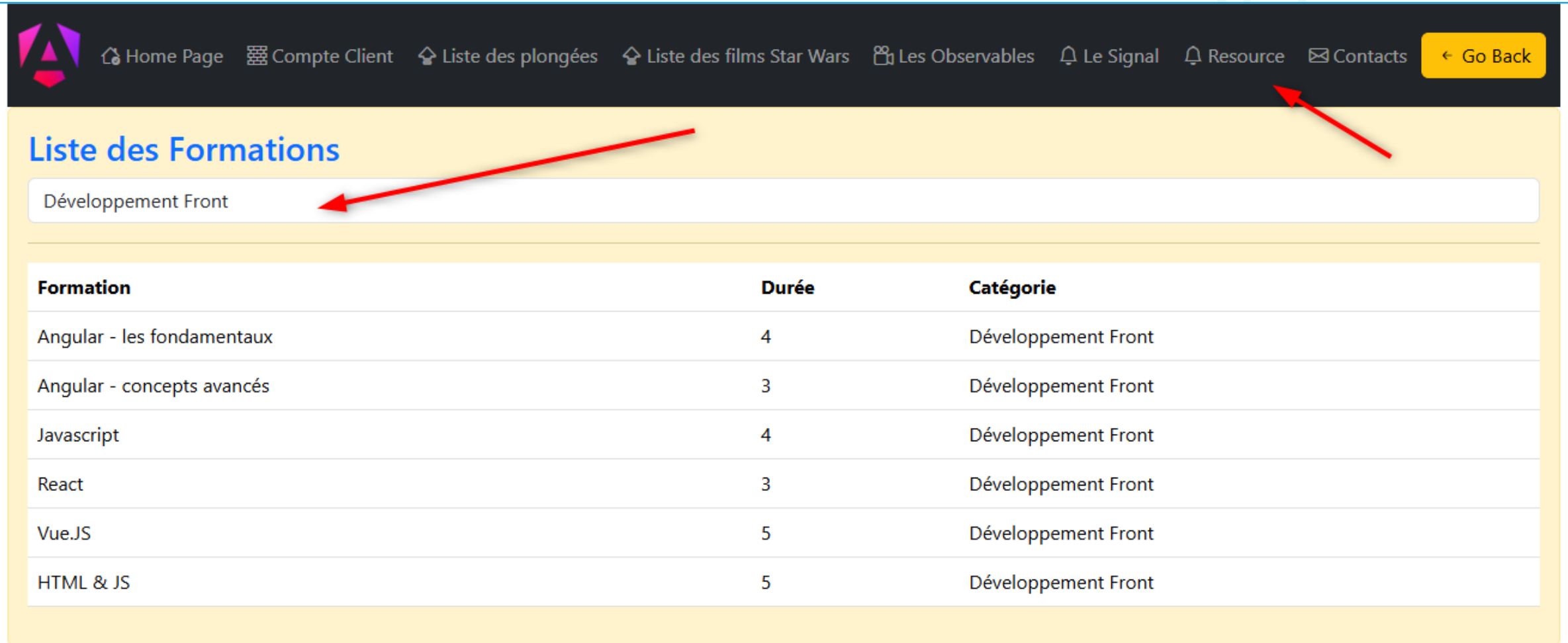
Resource API



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

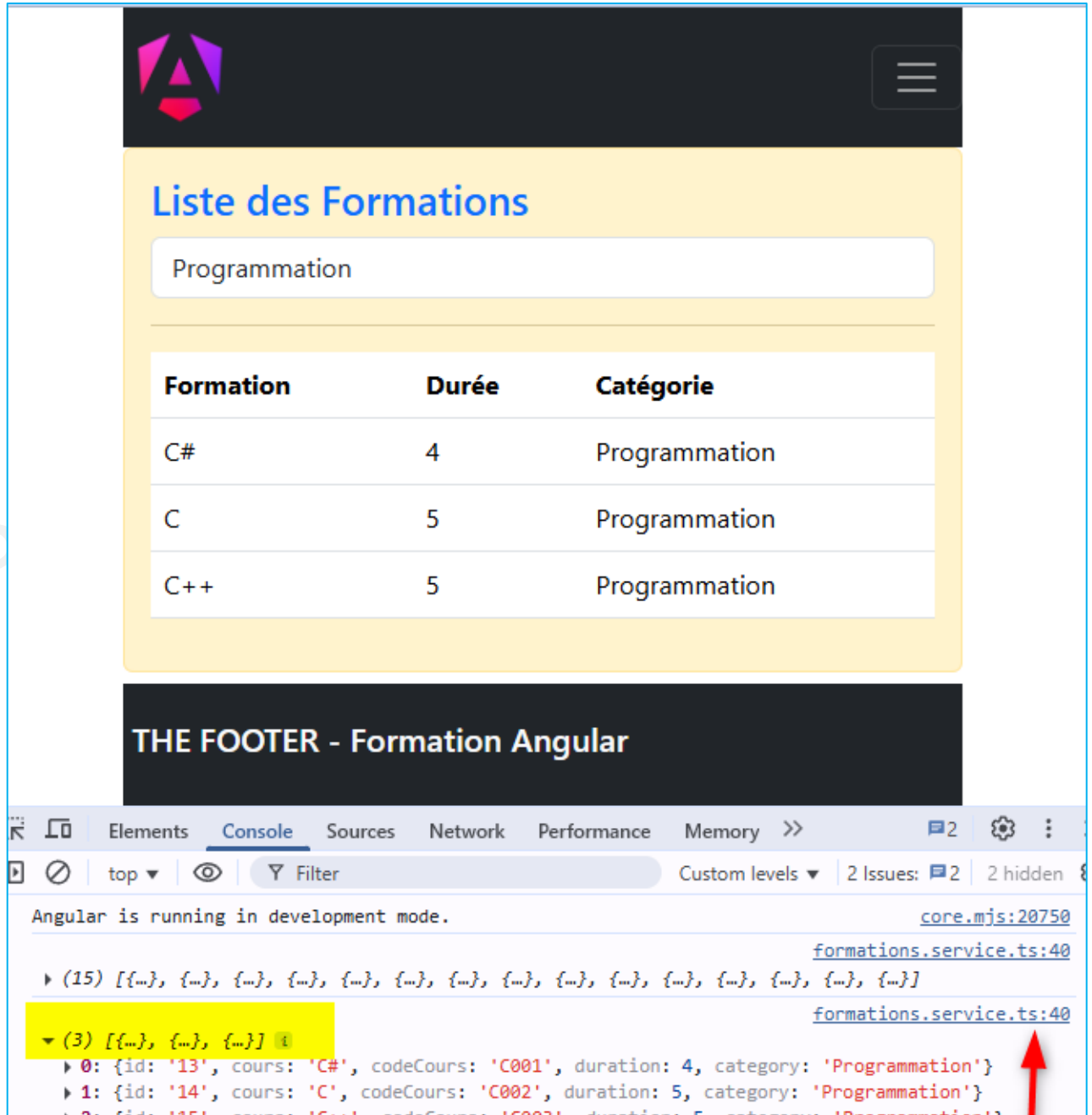
```
<div class="container alert alert-warning">

    <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

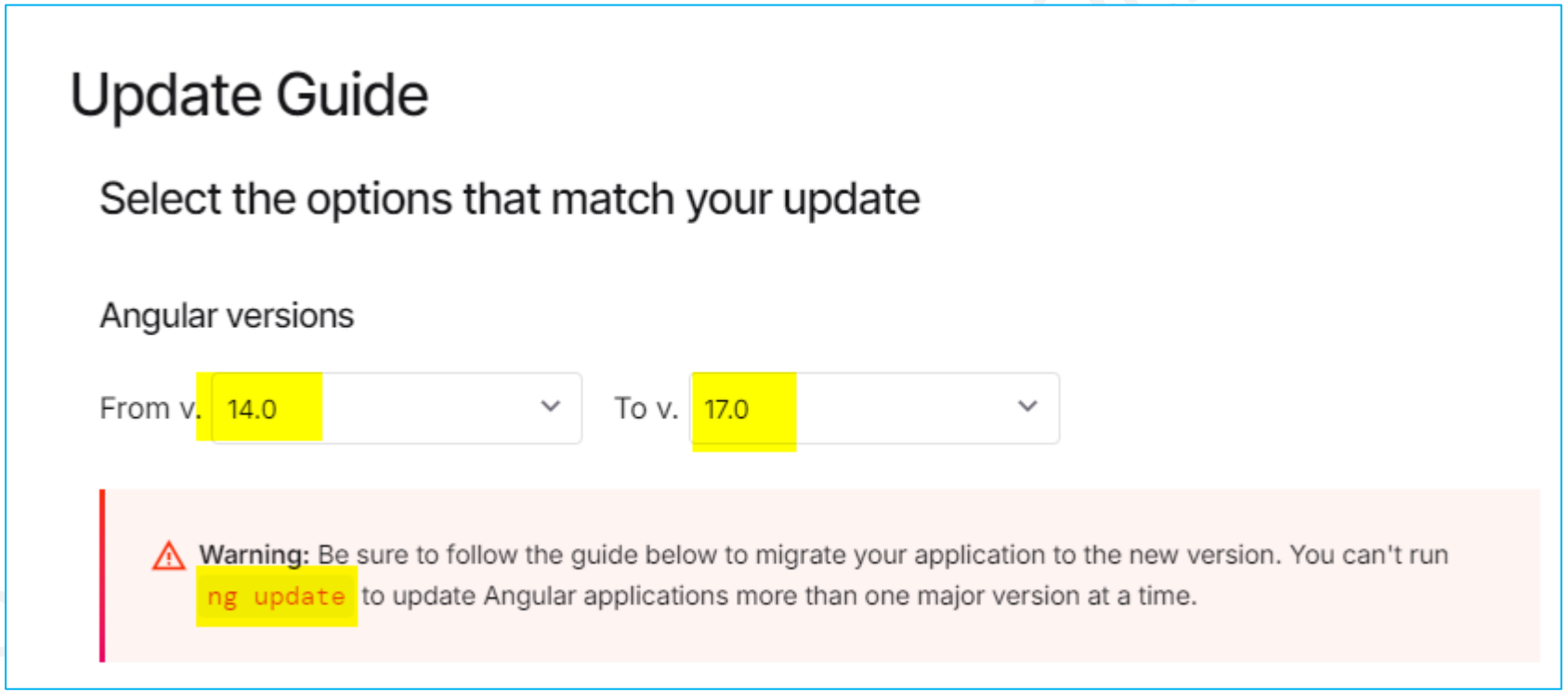
TP20 > src > app > app.ts > App > constructor > afterEveryRender() callback

```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 *

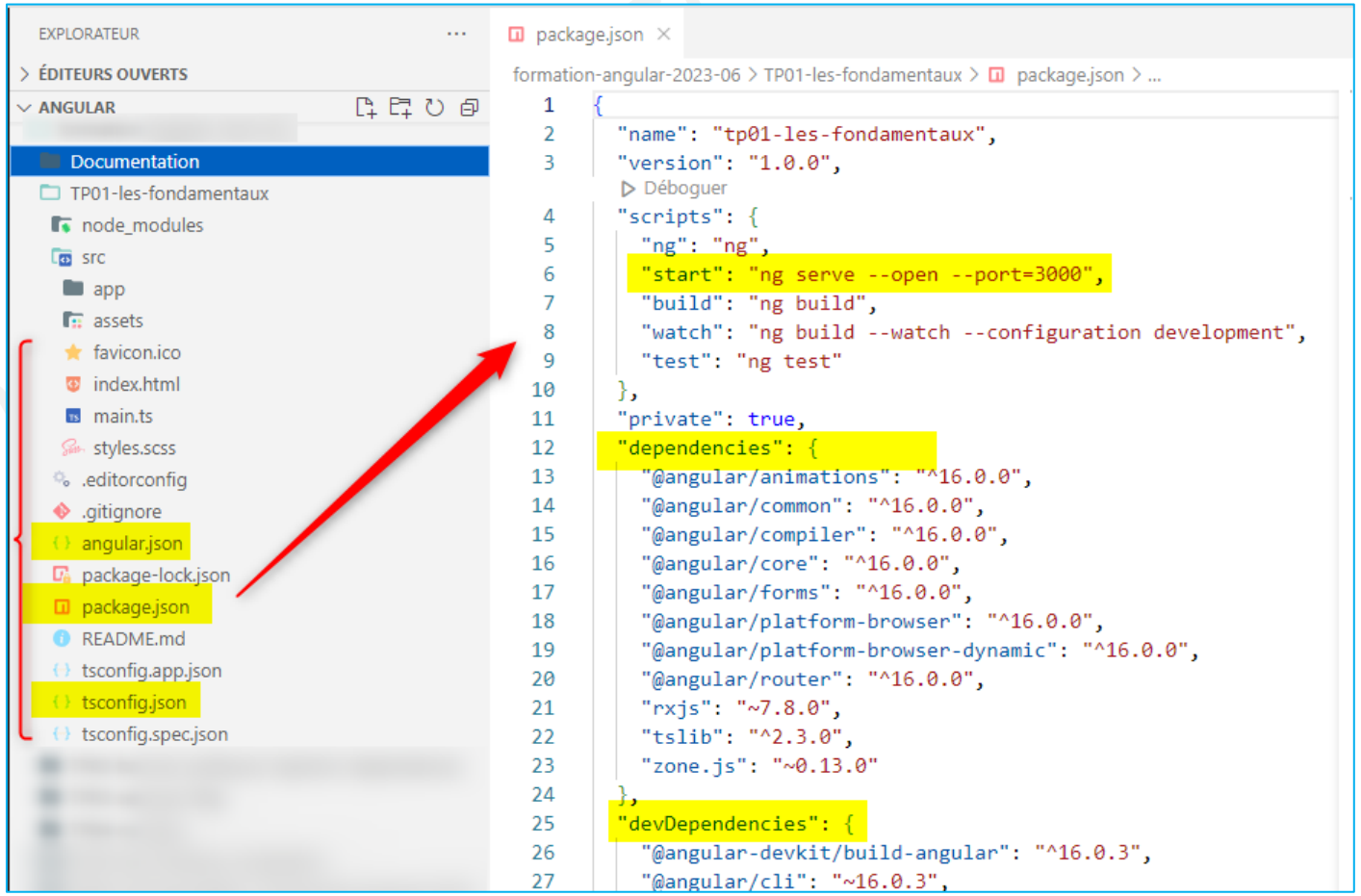
- node_modules * contient toutes les librairies (dépendances) nécessaires au projet webpack.

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
```

Rappels – Révisions Ecmascript

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*.

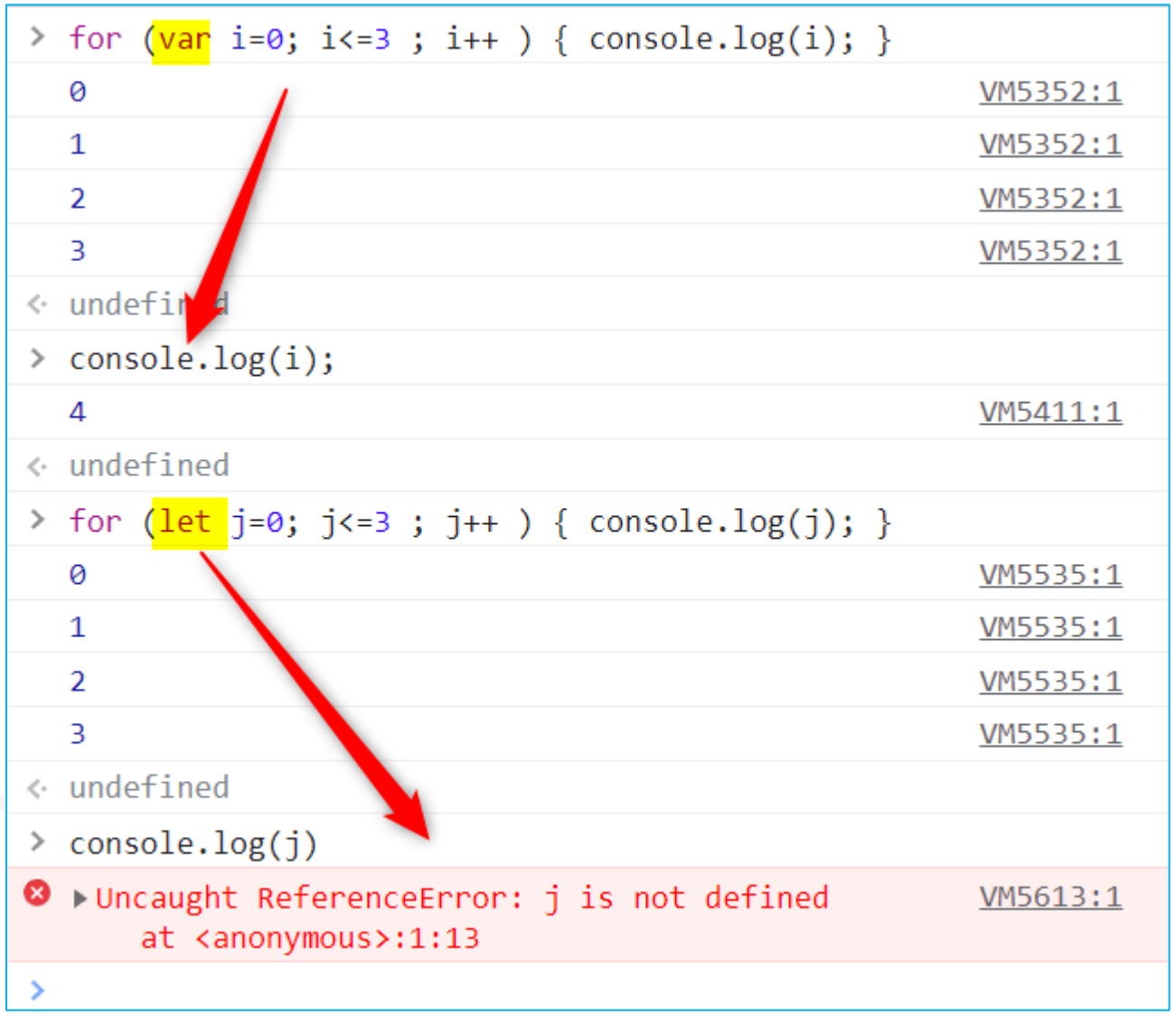
Les améliorations les plus importantes du langage concernent :

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

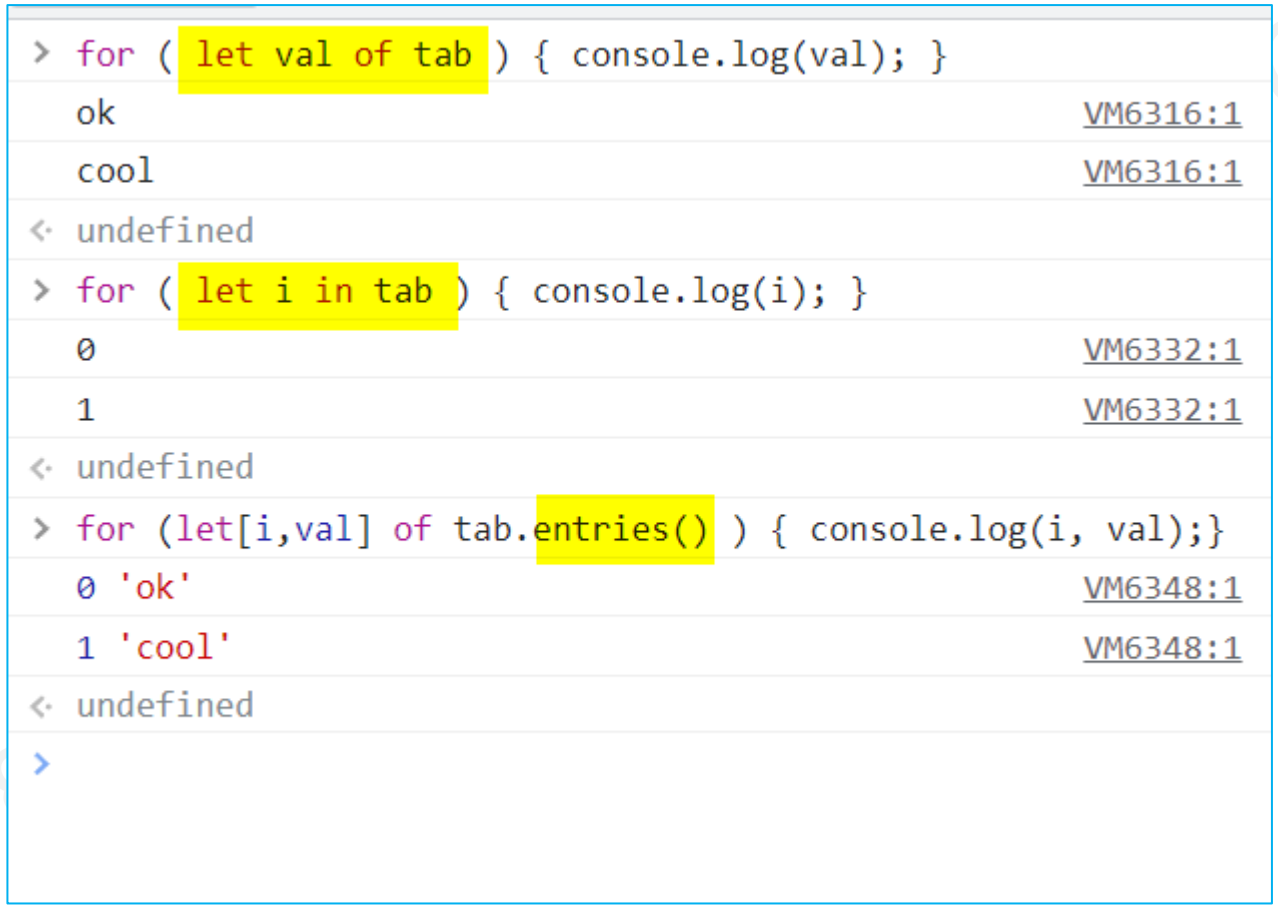
3. L'écriture simplifiée des FAT ARROWS
4. La string interpolation des variables avec ${maVariable} et la concatenation avec les back stits
5. Les itérateurs et les boucles FOR OF, FOR IN et FOR OF ENTRIES
6. Les spread operators pour déstructurer les tableaux
7. La notion de promesses afin de gérer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



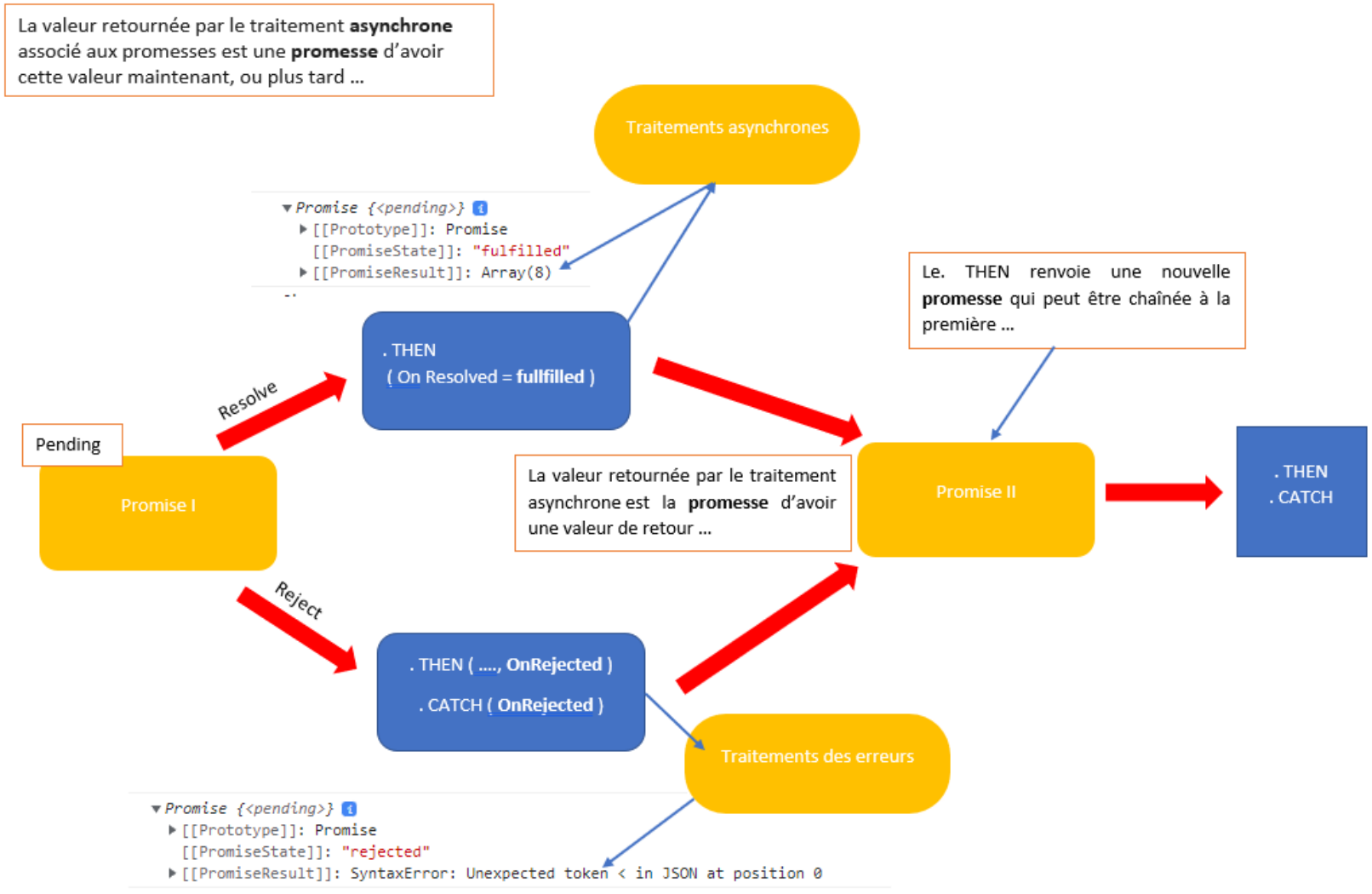
Ecmascript 2015+

```
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);
```

Formation ANGULAR

A bientôt.