



Formation ANGULAR

Animé par Michel BOCCIOLESI

Table des matières

- Introduction :
Framework Progressif ou Orienté
Installation – configuration
- 1^{er} projet Angular : composants :
Binding – décorateurs
composants parents-enfants
directives - pipes
- Mise en place du projet « fil rouge » de la formation
Application des acquis
- Injection de dépendances
service – Requêtes HTTP
- Les cycles de vie du composant
constructor – ngOnInit – ngAfterViewInit ...
- Notion de Réactivité Angular
Méthode « Value Based » (Zone.js)
Méthode Observable Based
Le signal NG17+
- La programmation RXJS et les observables
- Le routage Angular « navigation et liens »
Concepts de base et avancés
- Communication entre composants parents et enfants
@Input() @Output()
- Les Pipes Angular
- Test Unitaires
Introduction à Karma et Jasmine
- Les formulaires Angular
Reactive Forms vs Template
- Introduction Au Signal
- Annexes :
 - Les versions marquantes **15 – 20**
 - Migration entre versions majeures
 - Rappels EcmaScript 2015+

Framework Orienté ou Progressif



Copyright Michel BOCCIOLESI - ORSYS

Framework ou librairies ?

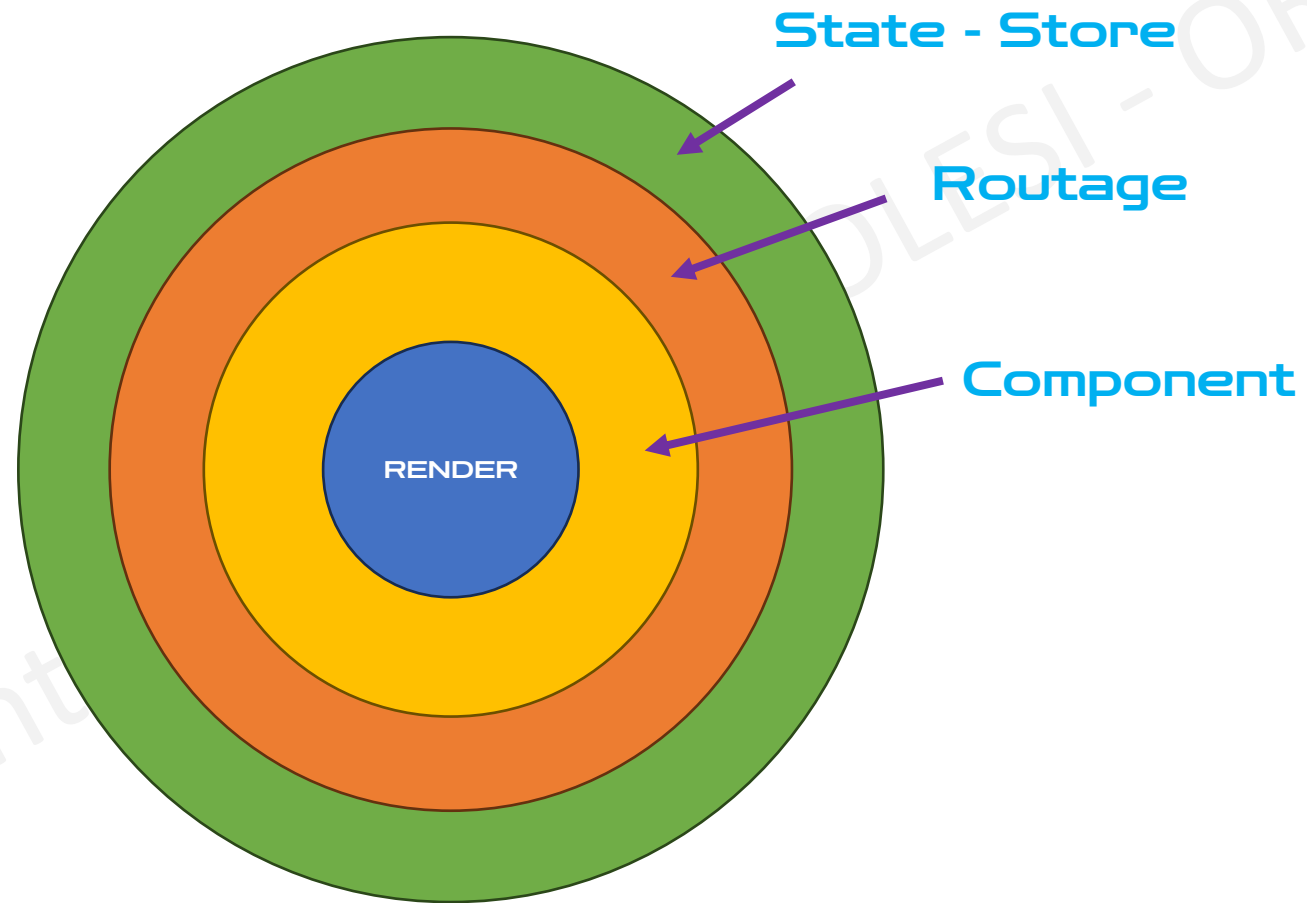
Dans le monde du développement Front, il existe plusieurs catégories de frameworks :

- [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 1^{er} 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



Copyright Michel CIOLESI - ORSYS

Installation et configuration

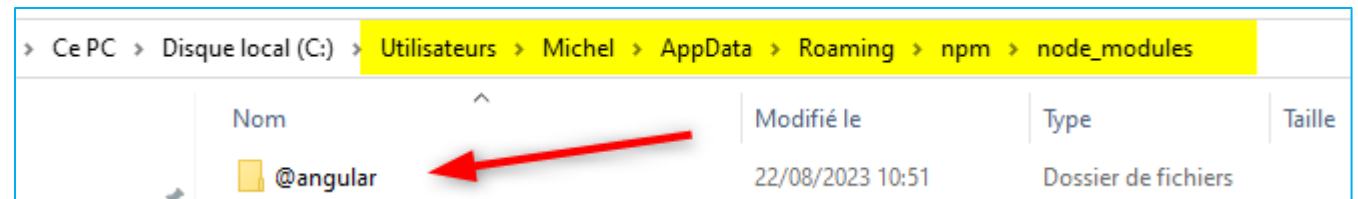
<https://nodejs.org>

<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 1^{ère} étape est de créer un projet avec la commande **npm init (-y)**

<https://nodejs.org/fr>

```
Windows PowerShell
PS C:\Users\Michel> ng version

Angular CLI
Angular CLI: 18.2.5
Node: 20.12.1
Package Manager: npm 10.8.1
OS: win32 x64

Angular: undefined
...

Package                                Version
-----
@angular-devkit/architect              0.1802.5 (cli-only)
@angular-devkit/core                   18.2.5 (cli-only)
@angular-devkit/schematics             18.2.5 (cli-only)
@schematics/angular                   18.2.5 (cli-only)

PS C:\Users\Michel>
```

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



Copyright Michel BOCCIOLESI - ORSYS

Framework ou librairies ?

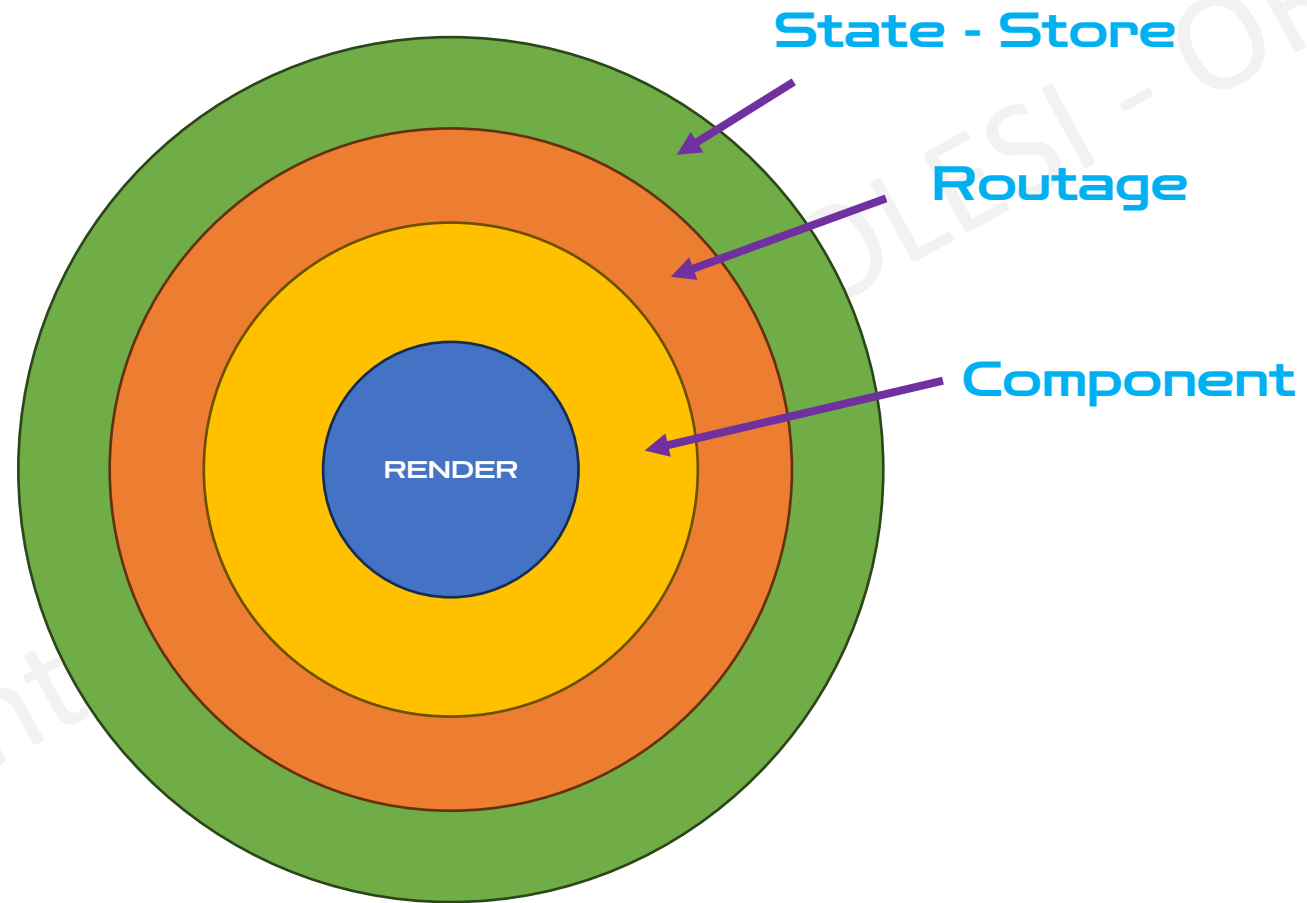
Dans le monde du développement Front, il existe plusieurs catégories de frameworks :

- [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 1^{er} 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



Copyright Michel CIOLESI - ORSYS

Installation et configuration

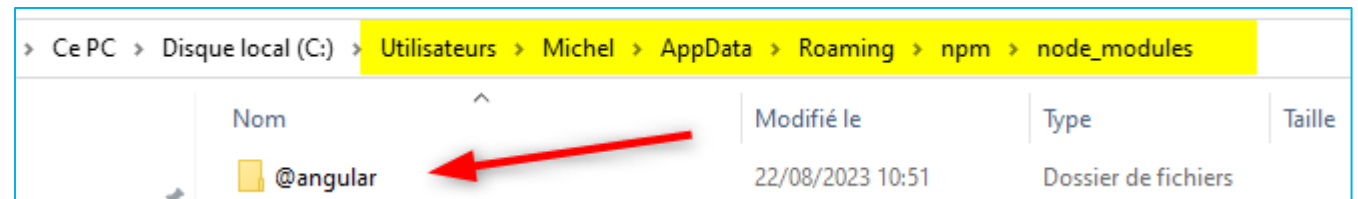
<https://nodejs.org>

<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 1^{ère} étape est de créer un projet avec la commande **npm init (-y)**

<https://nodejs.org/fr>

```
Windows PowerShell
PS C:\Users\Michel> ng version

Angular CLI
Angular CLI: 18.2.5
Node: 20.12.1
Package Manager: npm 10.8.1
OS: win32 x64

Angular: undefined
...

Package                                Version
-----
@angular-devkit/architect              0.1802.5 (cli-only)
@angular-devkit/core                  18.2.5 (cli-only)
@angular-devkit/schematics             18.2.5 (cli-only)
@schematics/angular                   18.2.5 (cli-only)

PS C:\Users\Michel>
```

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`

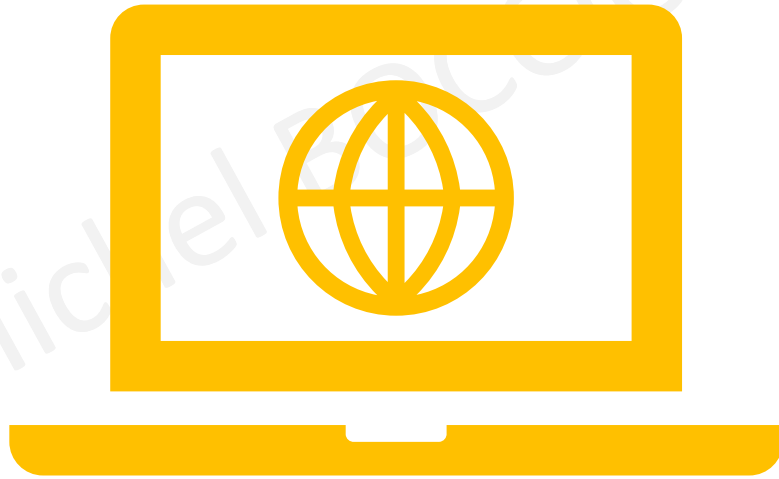
```
PS C:\Users\Michel\projet-angular-14> ls

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>
```

1^{ER} 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

```
package.json index.html angular.json x app.component.html main.ts x
```

```
-fondamentaux > angular.json > {} projects > {} TP01-les-fondamentaux > {} architect > {} build > {} options
```

```
9   "@schematics/angular:component": {
10     "style": "scss"
11   },
12   },
13   "root": "",
14   "sourceRoot": "src",
15   "prefix": "app",
16   "architect": {
17     "build": {
18       "builder": "@angular-devkit/build-angular:browser",
19       "options": {
20         "outputPath": "dist/tp01-les-fondamentaux",
21         "index": "src/index.html",
22         "main": "src/main.ts",
23         "polyfills": [
```

```
TP01-les-fondamentaux > src > main.ts
```

```
1  import { platformBrowserDynamic } from '@angular/platform-brows
2
3  // import LOCAL (import ES 2015)
4  import { AppModule } from './app/app.module';
5
6  // est le point d'entrée du projet
7  // charger ou bootstrapper le tout 1er module
8  ⚡
9  platformBrowserDynamic().bootstrapModule(AppModule)
10  | // .catch(err => console.error(err));
11
```

```
app.component.scss app.module.ts x app.component.ts x
```

```
TP01-les-fondamentaux > src > app > app.module.ts > ...
```

```
1  import { NgModule } from '@angular/core';
2  import { BrowserModule } from '@angular/platform-browser';
3
4  ⚡ import { AppRoutingModule } from './app-routing.module';
5  import { AppComponent } from './app.component';
6
7  // un décorateur transforme la class en objet NG (comp, module,
8  // service ...)
9  @NgModule({
10    declarations: [
11      // liste de tous les composants du module
12      AppComponent,
13      // PlanDeCoursComponent,
14      // DatesCoursComponent
15    ],
16    imports: [
17      // toutes les librairies nécessaires à ce module
18      BrowserModule,
19      AppRoutingModule
20    ],
21    providers: [
22      // services
23    ],
24    bootstrap: [
25      // quel est le 1er composant chargé ?
26      AppComponent
27    ]
28  })
```

```
TP01-les-fondamentaux > src > app > app.component.ts > ...
```

```
1  import { Component } from '@angular/core';
2  ⚡
3  // un décorateur transforme la class en objet NG (comp, module,
4  @Component({
5    // props ou directives
6    selector: 'app-root',
7    templateUrl: './app.component.html',
8    styleUrls: ['./app.component.scss']
9  })
10
11  // export et class sont des mots ECMAScript 2015
12  export class AppComponent {
13
14    // propriétés
15    title = 'TP01-les-fondamentaux';
16  }
17
```

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

The screenshot displays the Visual Studio Code interface for an Angular project. On the left, the Explorer sidebar shows the project structure under 'NG19', including files like 'app.component.ts', 'app.config.ts', and 'app.routes.ts'. The main editor area is divided into three panes. The top-left pane shows 'main.ts' with imports for 'bootstrapApplication', 'appConfig', and 'AppComponent', followed by a call to 'bootstrapApplication'. The top-right pane shows 'app.config.ts' with imports for 'routes' and 'provideClientHydration', and an 'appConfig' object containing 'providers'. The bottom pane shows 'app.routes.ts' with an import for 'Routes' and an empty 'routes' array. A large, semi-transparent watermark 'ESI - ORSYS' is visible in the background.

```
EXPLORATEUR
...
NG19
> .vscode
> node_modules
> public
src
  app
    app.component.css
    app.component.html
    app.component.spec.ts
    app.component.ts
    app.config.server.ts
    app.config.ts
    app.routes.ts
    index.html
    main.server.ts
    main.ts
    server.ts
    styles.css
    .editorconfig
    .aitianore

main.ts
src > main.ts > ...
1 import { bootstrapApplication } from '@angular/platform-browser';
2 import { appConfig } from './app/app.config';
3 import { AppComponent } from './app/app.component';
4
5 bootstrapApplication(AppComponent, appConfig)
6   .catch((err) => console.error(err));
7

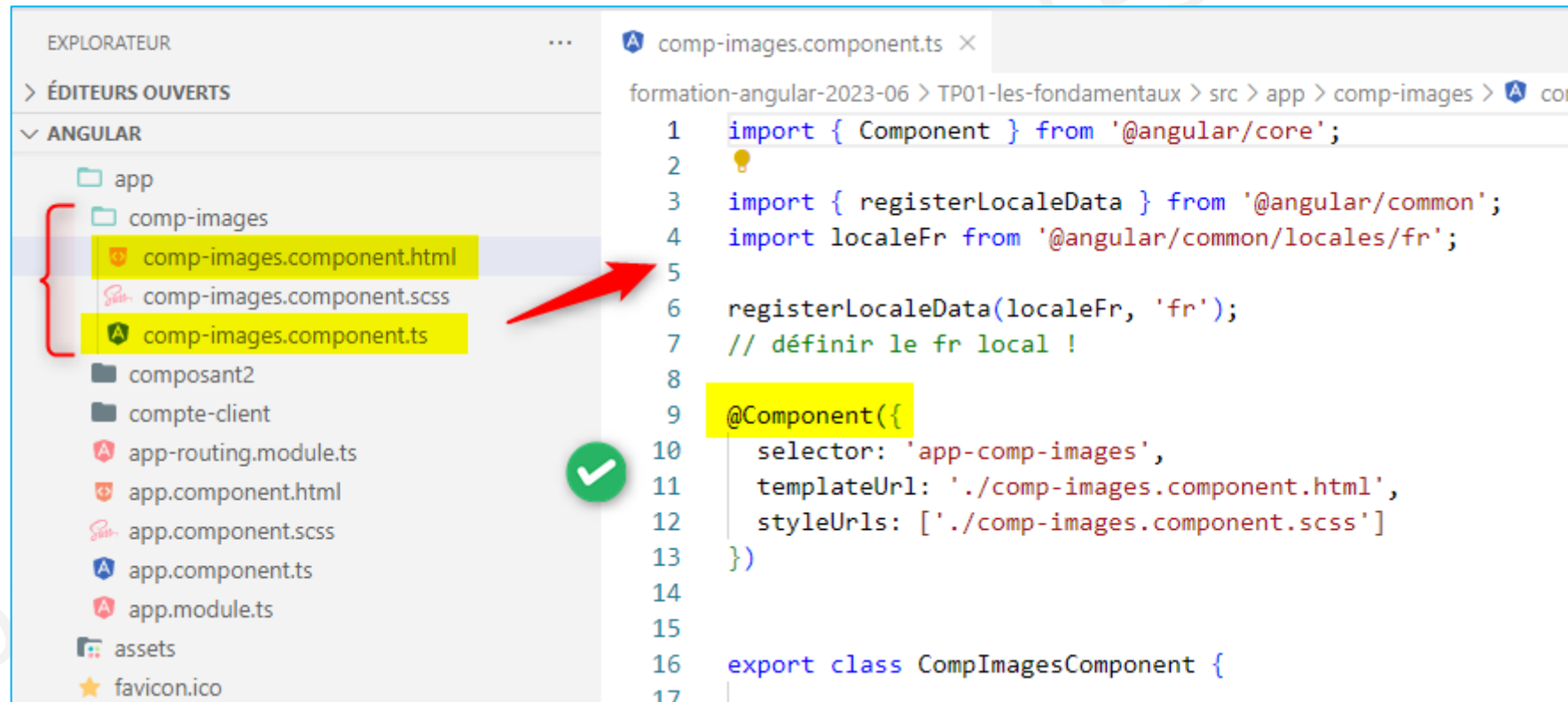
app.config.ts
src > app > app.config.ts > appConfig > providers
4 import { routes } from './app.routes';
5 import { provideClientHydration, withEventReplay } from '@angular/platform-browser';
6
7 export const appConfig: ApplicationConfig = {
8   providers: [
9     provideZoneChangeDetection({ eventCoalescing: true }),
10    provideRouter(routes),
11    provideClientHydration(withEventReplay())
12  ];
13

app.routes.ts
src > app > app.routes.ts > ...
1 import { Routes } from '@angular/router';
2
3 export const routes: Routes = [];
4
```

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 ×
formation-angular-2023-06 > TP01-les-fondamentaux > src > app > app.component.ts > ...
1 import { Component } from '@angular/core';
2
3 // un décorateur : son rôle : transformer la
  classe en ... (composant)
4
5 @Component({
6   selector: 'app-root',
7   templateUrl: './app.component.html',
8   styleUrls: ['./app.component.scss']
9 })
10
11 // une classe (ES2015) exportée
12 export class AppComponent {
13
14   // 1- déf des propriétés
15   public title: string = 'Angular';
16   public version: number;
17
18   // 2- constructeur de classe
19   constructor() {
20     this.version = 16;
21   }
22 }
23
```

```
app.component.html ×
formation-angular-2023-06 > TP01-les-fondamentaux > src > app > app.component.html > ...
1
2 <!-- string interpolation de variables {{ }} -->
3 <!-- Binding : le TS est lié à la vue HTML(template) -->
4 <!-- Single way Binding: TS => HTML -->
5 <h1>Formation {{title}} {{version}}</h1>
6
7
8 <!-- composant enfant de AppComponent -->
9 <app-composant2></app-composant2>
10
11 <hr>
12
13 <app-comp-images></app-comp-images>
14
15 <hr>
16 <h2>Espace Client ...</h2>
17 <app-landing-page></app-landing-page>
```


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

app.component.html

```
1
2 <!-- string interpolation de variables {{ }} -->
3 <!-- Binding : le TS est lié à la vue HTML(template) -->
4 <!-- Single way Binding: TS => HTML -->
5 <h1>Formation {{title}} {{version}}</h1>
6
7
8 <!-- composant enfant de AppComponent -->
9 <app-composant2></app-composant2>
10
11 <hr>
12
13 <app-comp-images></app-comp-images>
14
15 <hr>
16 <h2>Espace Client ...</h2>
17 <app-landing-page></app-landing-page>
```

composant2.component.html

```
1 <h1>Composant #2</h1>
2
```

comp-images.component.html

```
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     
19   <!-- la directive ngClass affiche une classe nommée si la condition se
20 </span>
21 </div>
```

TP01LesFondamentaux

localhost:3000

Formation Angular 16

Composant #2

Les Avatars ...



Copyright

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     
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

```
1 <h1>Formation {{title}} {{version}}</h1>
2
3
4
5
6
7
8 <!-- composant enfant de AppComponent -->
9 <app-composant2></app-composant2>
10
11 <hr>
12
13 <app-comp-images></app-comp-images>
14
15 <hr>
16 <h2>Espace Client ...</h2>
17 <app-landing-page></app-landing-page>
```

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

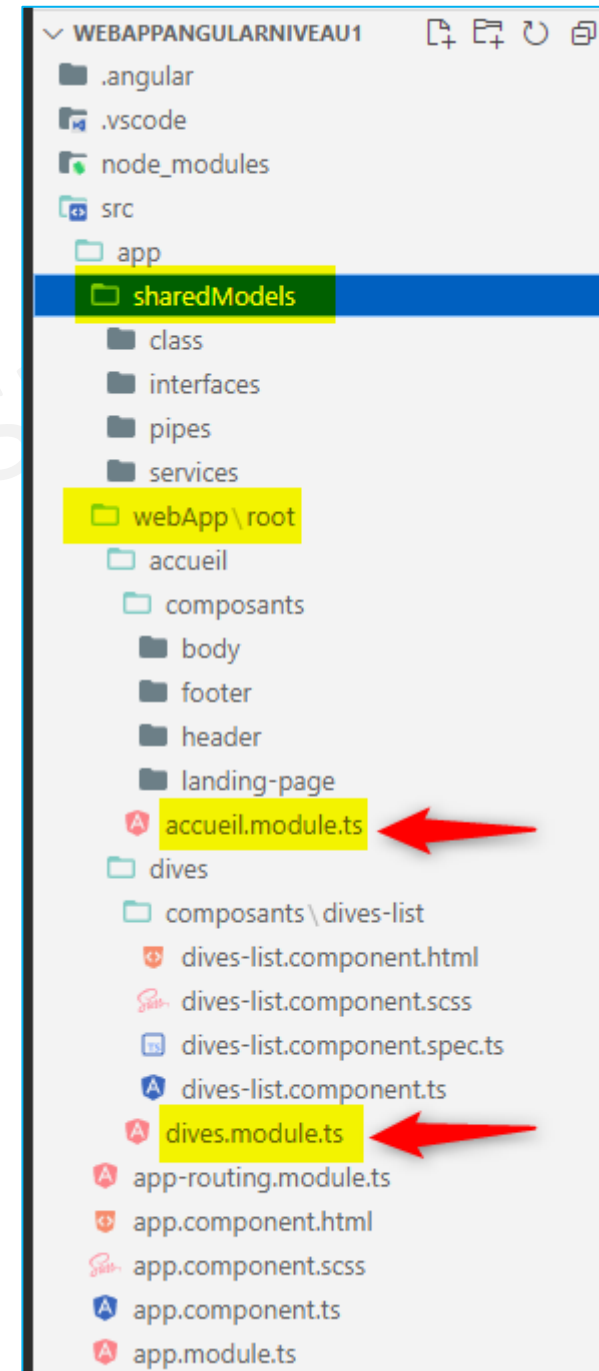
```
1 import { NgModule } from '@angular/core';
2 import { BrowserModule } from '@angular/platform-browser';
3
4 // -----
5 // IMPORT LOCAL (ressources fichiers)
6 import { AppRoutingModule } from './app-routing.module';
7 import { AppComponent } from './app.component';
8 import { FormsModule } from '@angular/forms';
9 import { Composant2Component } from './composant2/composant2.component';
10 import { CompImagesComponent } from './comp-images/comp-images.component';
11 import { HeaderComponent } from './compte-client/composants/header/header.component';
12 import { CompteClientModule } from './compte-client/compte-client.module';
13
14 // un décorateur son rôle : transformer la classe en ... (module)
15 @NgModule({
16   // objet de description
17   declarations: [
18     HeaderComponent,
19     FooterComponent,
20     LandingPageComponent
21   ],
22   imports: [
23     CommonModule
24   ],
25   exports: [
26     HeaderComponent,
27     BodyComponent,
28     FooterComponent,
29     LandingPageComponent
30   ]
31 })
32 export class CompteClientModule { }
```

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

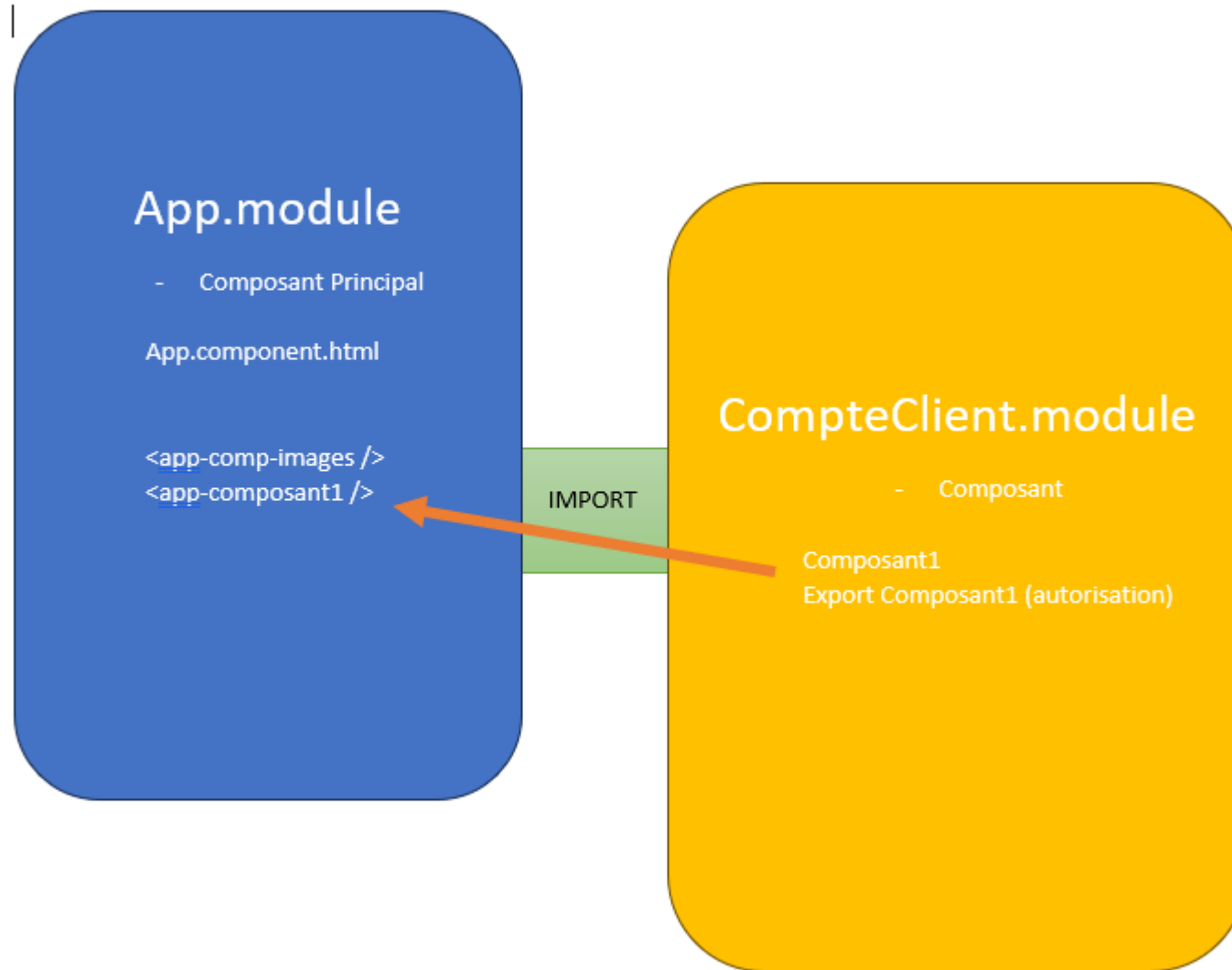
```
9   "test": "ng test",  
10  "json-server": "json-server --watch src/assets/json/bdd.json -p 3001"  
11
```

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



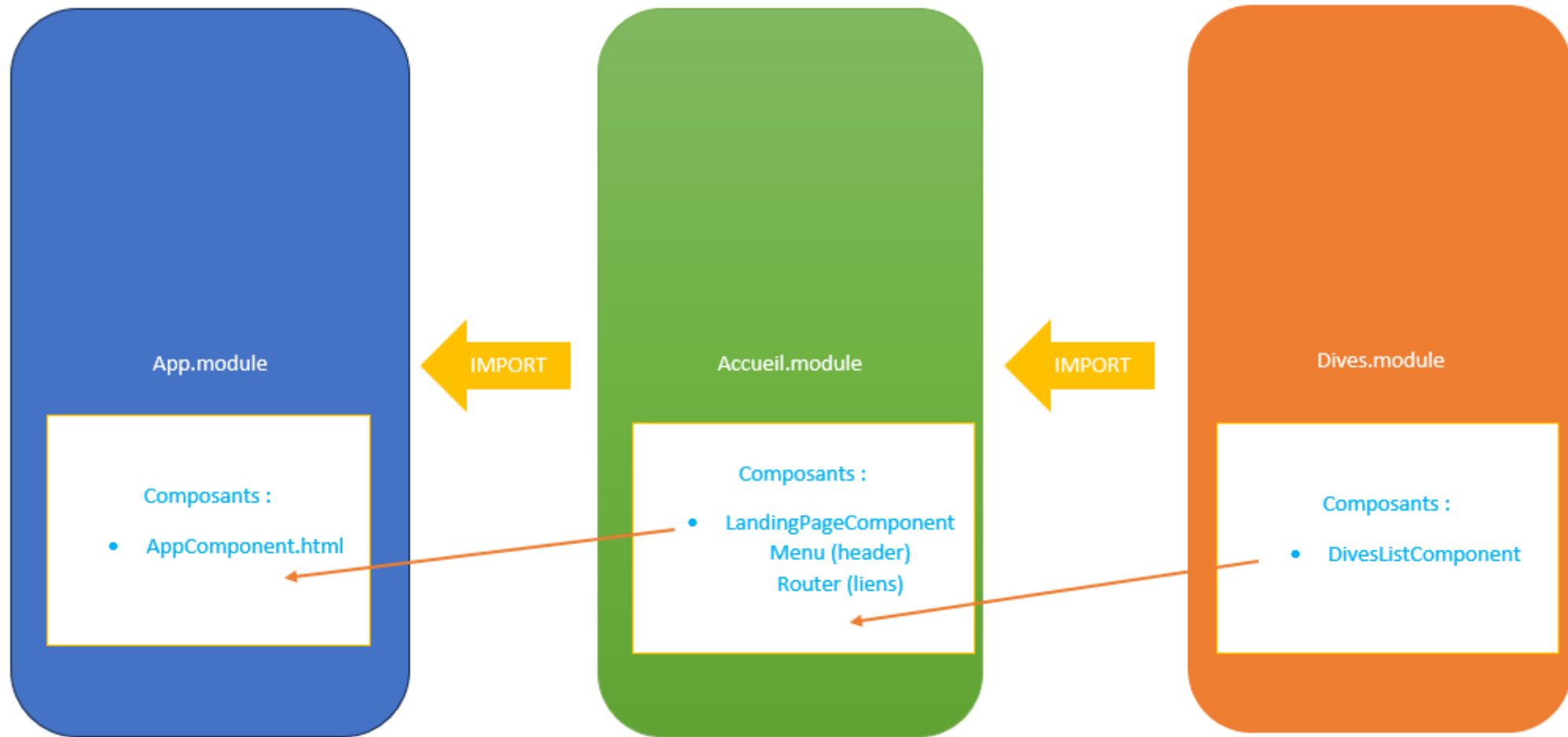
Mode Modules *
Compréhension des
anciens projets Angular

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

The image shows a VS Code editor with two files open: `package.json` and `angular.json`.

package.json (Left Pane):

```
1 {
2   "name": "tp02-bonnes-pratiques",
3   "version": "0.0.0",
4   "scripts": {
5     "ng": "ng",
6     "start": "ng serve --open --port=3000",
7     "build": "ng build",
8     "watch": "ng build --watch --configuration development",
9     "test": "ng test"
10  },
11  "private": true,
12  "dependencies": {
13    "@angular/animations": "^16.2.0",
14    "@angular/common": "^16.2.0",
15    "@angular/compiler": "^16.2.0",
16    "@angular/core": "^16.2.0",
17    "@angular/forms": "^16.2.0",
18    "@angular/platform-browser": "^16.2.0",
19    "@angular/platform-browser-dynamic": "^16.2.0",
20    "@angular/router": "^16.2.0",
21    "bootstrap": "^5.3.2",
22    "bootstrap-icons": "^1.11.1",
23    "rxjs": "~7.8.0",
24    "tslib": "^2.3.0",
25    "zone.js": "~0.13.0"
26  }
27 }
```

angular.json (Right Pane):

```
16 "architect": {
17   "build": {
18     "builder": "@angular-devkit/build-angular:browser",
19     "options": {
20       "outputPath": "dist/tp02-bonnes-pratiques",
21       "index": "src/index.html",
22       "main": "src/main.ts",
23       "polyfills": [
24         "zone.js"
25       ],
26       "tsConfig": "tsconfig.app.json",
27       "inlineStyleLanguage": "scss",
28       "assets": [
29         "src/favicon.ico",
30         "src/assets"
31       ],
32       "styles": [
33         "node_modules/bootstrap/dist/css/bootstrap.min.css",
34         "node_modules/bootstrap-icons/font/bootstrap-icons.css",
35         "src/styles.scss"
36       ],
37       "scripts": [
38         "node_modules/bootstrap/dist/js/bootstrap.min.js"
39       ]
40     }
41   }
42 }
```

A red arrow points from the `bootstrap` dependency in `package.json` to the `styles` array in `angular.json`.

TP : single et double way binding

```
1 import { Component } from '@angular/core';
2
3 Codiumate: Options | Test this class
4 @Component({
5   selector: 'app-home-page',
6   templateUrl: './home-page.component.html',
7   styleUrls: ['./home-page.component.scss']
8 })
9 export class HomePageComponent {
10   // propriétés
11   public codeCours:string='VUE';
12
13   public result:string='ok';
14
15   // méthodes
16   Codiumate: Options | Test this method
17   public methodClic(){
18     this.result='Vous avez cliqué !';
19   }
20 }
21
```

```
Go to component
1 <div class="alert alert-primary">
2   <h3>HOME PAGE</h3>
3
4   <!-- single way Binding : HTML => TS -->
5   <button class="btn btn-warning" (click)="methodClic()">
6     Cliquez pour tester le single way binding...
7   </button>
8
9   <p>{{result}}</p>
10
11   <!-- EX0 :
12     *ngIf *ngSwitchCase
13     @if() Control flow NG>17+
14
15   si le codeCours est égal NG => Vous avez choisi
16   Formation Angular
17   Sinon ..... REACT => .... React
18   Sinon ..... VUE => ..... Vue
19   Autres => Choisir une formation
20   -->
21
22 </div>
```

TP :

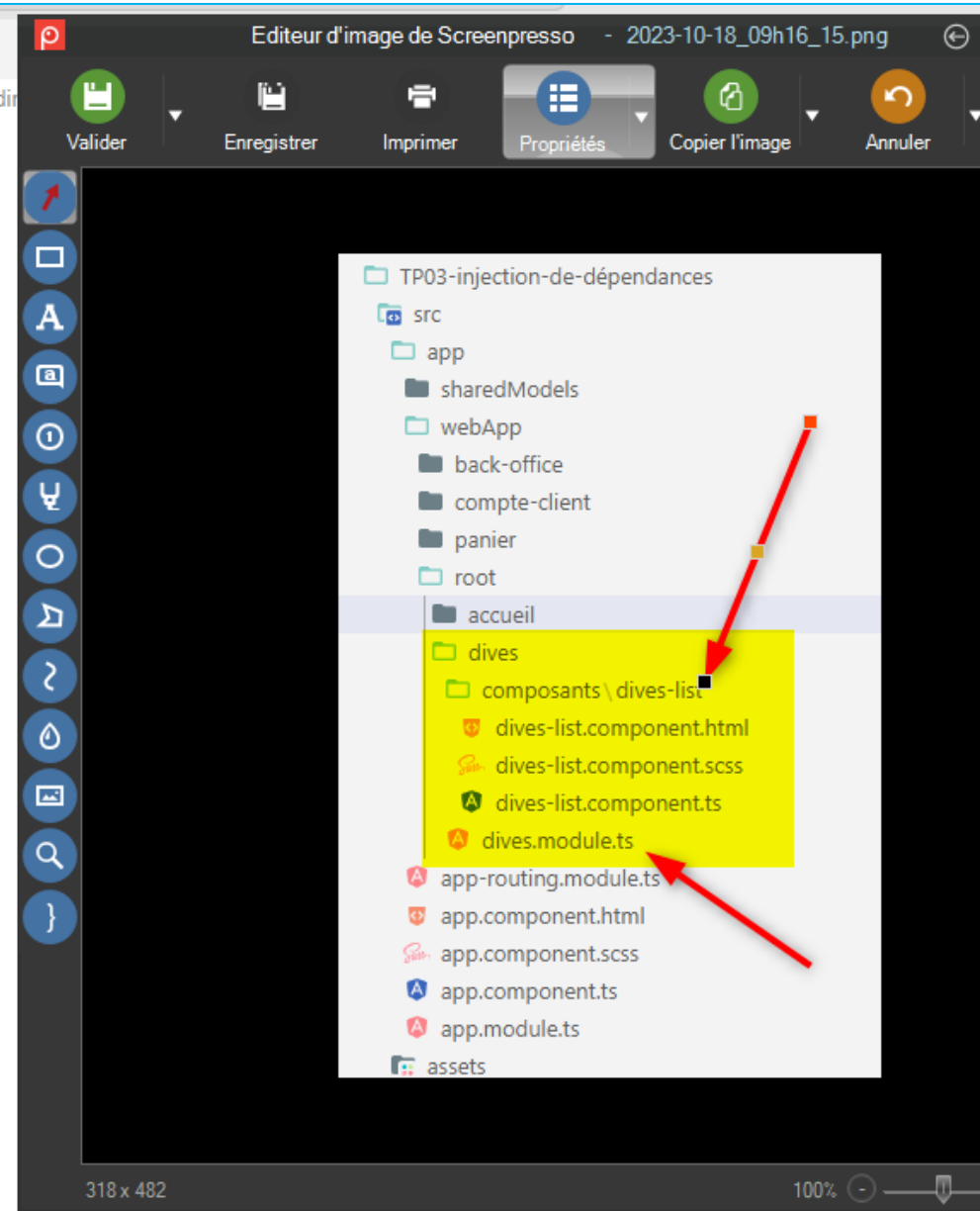
```
home-page.component.ts x
app > webApp > root > accueil > components > home-page > home-page.component.ts
1 import { Component } from '@angular/core';
2
3 Codiumate: Options | Test this class
4 @Component({
5   selector: 'app-home-page',
6   templateUrl: './home-page.component.html',
7   styleUrls: ['./home-page.component.scss']
8 })
9 export class HomePageComponent {
10   // propriétés
11   public codeCours:string='VUE';
12
13   public result:string='';
14
15   // méthodes
16   Codiumate: Options | Test this method
17   public methodClic(){
18     this.result='Vous avez cliqué !';
19   }
20 }
21
```

```
home-page.component.html x
TP02 > src > app > webApp > root > accueil > components > home-page > home-page.component.html > div,ale
Go to component
1 <div class="alert alert-primary">
2   <h3>HOME PAGE</h3>
3
4   <!-- single way Binding : HTML => TS -->
5
6   <button class="btn btn-primary" (click)="methodClic()">
7     Cliquez pour tester le single way binding...
8   </button>
9   <p>{{result}}</p>
10
11 <p></p>
12
13 <select class="form-control" [(ngModel)]="codeCours">
14   <option value="">Choisir pour tester le double way binding</option>
15   <option value="NG">Angular ...</option>
16   <option value="REACT">React ...</option>
17   <option value="VUE">Vue ...</option>
18 </select>
19
20 <p></p>
21
22 <div class="alert alert-warning">
23
24   @if (codeCours==="NG") {
25     <span>Vous avez choisi Angular</span>
26   }
27   @else if (codeCours==="REACT") {
28     <span>Vous avez choisi React</span>
29   }
30   @else if (codeCours==="VUE") {
31     <span>Vous avez choisi Vue</span>
32   }
33   @else {
34     <span>Choisissez une formation</span>
35   }
36
```

Arborescence de dives

```
landing-page.component.html x
TP02-bonnes-pratiques > src > app > webApp > root > accueil > composants > landing-page > landing-page.component.html
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

The image displays the file structure and code of an Angular application, illustrating the 'Arborescence de dives' (Dives tree structure).

Left Pane (accueil.module.ts):

```
TP02-bonnes-pratiques > src > app > webApp > root > accueil > accueil.module.ts > AccueilModule
6 import { FooterComponent } from './composants/footer/footer.component';
7 import { DivesModule } from '../dives/divesModule';
8
9
10
11 @NgModule({
12   declarations: [
13     LandingPageComponent,
14     HeaderComponent,
15     BodyComponent,
16     FooterComponent
17   ],
18   imports: [
19     CommonModule,
20     // import de nos modules
21     DivesModule
22 ]})
```

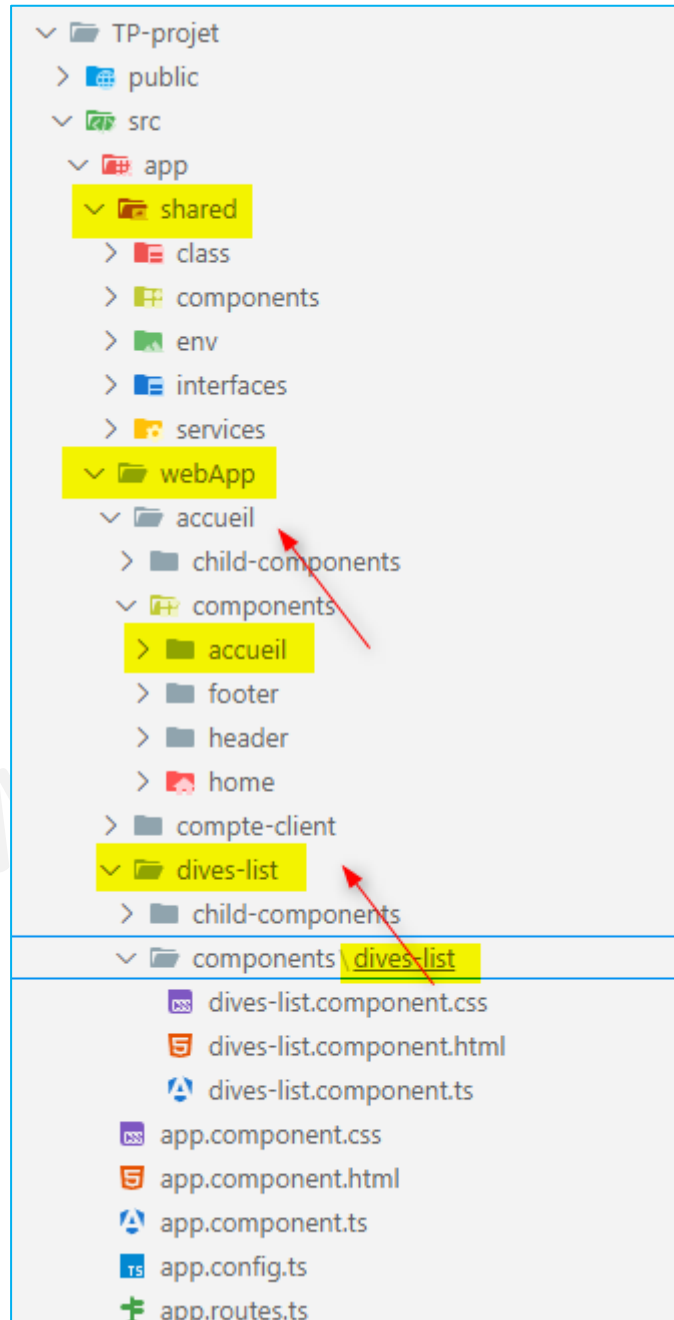
Right Pane (divesModule.ts):

```
TP02-bonnes-pratiques > src > app > webApp > root > dives > dives.module.ts > DivesModule
1 import { NgModule } from '@angular/core';
2 import { CommonModule } from '@angular/common';
3 import { DivesListComponent } from './composants/dives-list/dives-list.component';
4
5
6
7 @NgModule({
8   declarations: [
9     DivesListComponent
10  ],
11   imports: [
12     CommonModule
13  ],
14   exports: [
15     DivesListComponent
16  ]
17 })
18 export class DivesModule
19
```

Browser (localhost:3000):

dives-list works!

Arborescence en Mode Standalone 😊



app.module.ts

```
TP03-injection-de-dépendances > src > app > app.module.ts
1 import { NgModule } from '@angular/core';
2 import { BrowserModule } from '@angular/platform-browser';
3
4 import { AppRoutingModule } from './app-routing.module';
5 import { AppComponent } from './app.component';
6 import { AccueilModule } from './webApp/accueil/accueil.module';
7
8 @NgModule({
9   declarations: [
10     AppComponent
11   ],
12   imports: [
13     BrowserModule,
14     AppRoutingModule,
15     AccueilModule
16   ],
17   providers: [],
18   bootstrap: [AppComponent]
19 })
20 export class AppModule { }
21
```

landing-page.component.html

```
TP03-injection-de-dépendances > src > app > webApp > root > accueil > composants > landing-page > landing-page.component.html
1 <app-header></app-header>
2 <!-- <app-body></app-body> -->
3 <app-dives-list></app-dives-list>
4 <app-footer></app-footer>
```

accueil.module.ts

```
TP03-injection-de-dépendances > src > app > webApp > root > accueil > accueil.module.ts
1 import { NgModule } from '@angular/core';
2 import { CommonModule } from '@angular/common';
3 import { LandingPageComponent } from './composants/landing-page/landing-page.component';
4 import { HeaderComponent } from './composants/header/header.component';
5 import { FooterComponent } from './composants/footer/footer.component';
6 import { BodyComponent } from './composants/body/body.component';
7 import { DivesModule } from '../dives/dives.module';
8
9 @NgModule({
10   declarations: [
11     LandingPageComponent,
12     HeaderComponent,
13     FooterComponent,
14     BodyComponent
15   ],
16   imports: [
17     CommonModule,
18     DivesModule
19   ],
20   exports: [
21     LandingPageComponent,
22     HeaderComponent,
23     FooterComponent,
24     BodyComponent
25   ]
26 })
27 export class AccueilModule { }
```

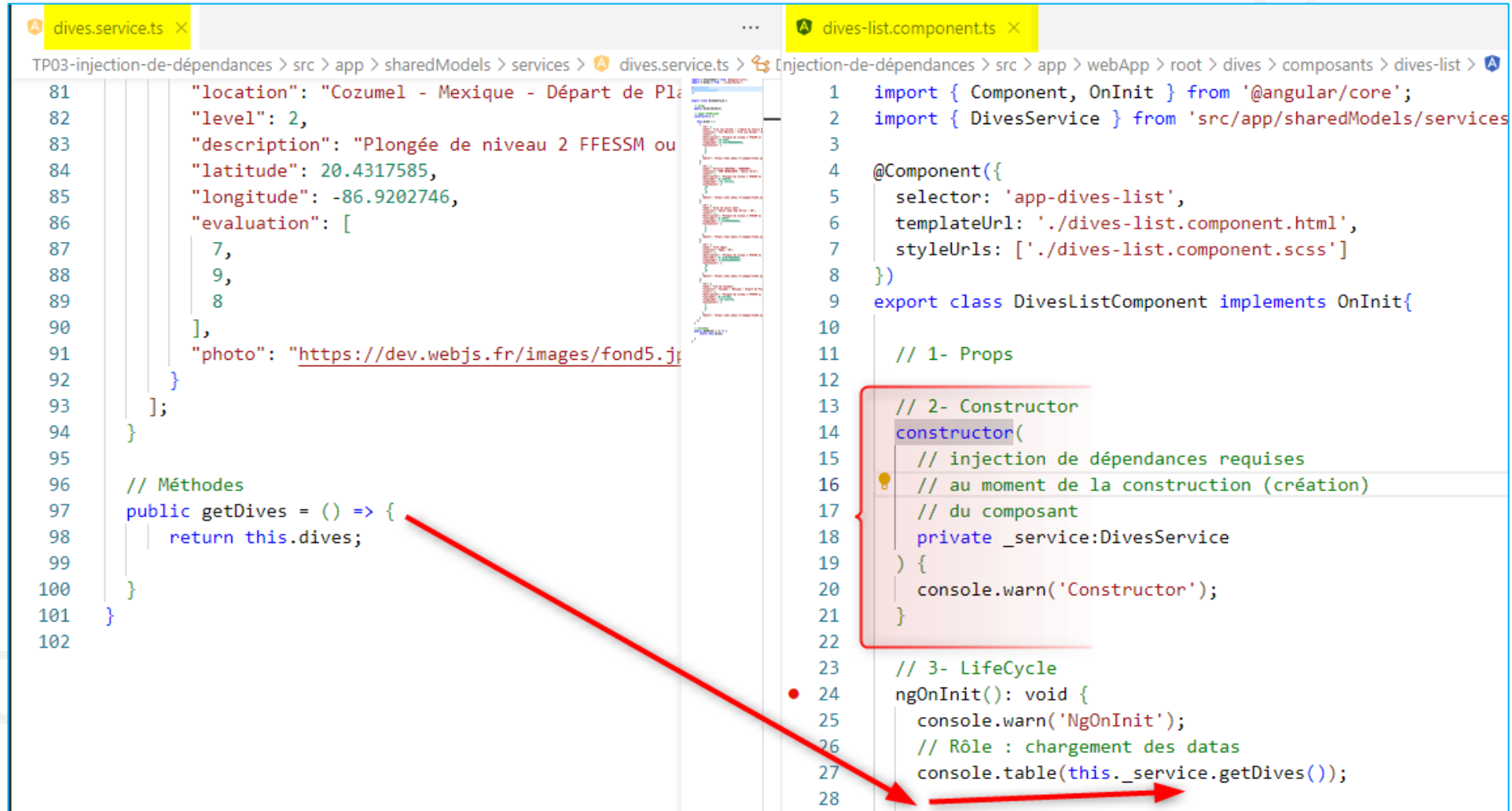
dives.module.ts

```
TP03-injection-de-dépendances > src > app > webApp > root > dives > dives.module.ts
1 import { NgModule } from '@angular/core';
2 import { CommonModule } from '@angular/common';
3 import { DivesListComponent } from './composants/dives-list/dives-list.component';
4
5 @NgModule({
6   declarations: [
7     DivesListComponent
8   ],
9   imports: [
10     CommonModule
11   ],
12   exports: [
13     DivesListComponent
14   ]
15 })
16 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



```
dives.service.ts
81  "location": "Cozumel - Mexique - Départ de Pl
82  "level": 2,
83  "description": "Plongée de niveau 2 FFESSM ou
84  "latitude": 20.4317585,
85  "longitude": -86.9202746,
86  "evaluation": [
87    7,
88    9,
89    8
90  ],
91  "photo": "https://dev.webjs.fr/images/fond5.jp
92  }
93  ];
94  }
95
96  // Méthodes
97  public getDives = () => {
98    return this.dives;
99  }
100 }
101 }
102 }

dives-list.component.ts
1  import { Component, OnInit } from '@angular/core';
2  import { DivesService } from 'src/app/sharedModels/services
3
4  @Component({
5    selector: 'app-dives-list',
6    templateUrl: './dives-list.component.html',
7    styleUrls: ['./dives-list.component.scss']
8  })
9  export class DivesListComponent implements OnInit{
10
11    // 1- Props
12
13    // 2- Constructor
14    constructor(
15      // injection de dépendances requises
16      // au moment de la construction (création)
17      // du composant
18      private _service:DivesService
19    ) {
20      console.warn('Constructor');
21    }
22
23    // 3- Lifecycle
24    ngOnInit(): void {
25      console.warn('NgOnInit');
26      // Rôle : chargement des datas
27      console.table(this._service.getDives());
28    }
```


L'injection de dépendances

1^{er} exemple : le service

```
dives-list.component.ts x
App > root > dives > composants > dives-list > dives-list.component.ts > DivesListComponent
16 // @ViewChild('btnPlus') eltBtn!: ElementRef;
17 @ViewChildren('btnPlus') eltBtnCollection!: QueryList<ElementRef>;
18
19 // 2- const
20 // 1'injection de dépendances permet de lier un service
21
22 constructor(
23     private _service: DivesService,
24     private _title: Title,
25     private _meta: Meta
26 ) {
27     console.warn('Constructor');
28     this.dives = [];
29     this._title.setTitle('Liste des Plongées SEO ....');
30     this._meta.addTags([
31         { name: 'description', content: 'Texte de description' },
32         { name: 'author', content: 'MB CREATION WEB' }
33     ]);
34 }
35
36 // -----
37 // 3- lifeCycle
38 // -----
39 ngOnInit(): void {
40     // Chargement des datas
41     console.warn('NgOnInit');
42     this._service.getDives().subscribe(
43         // .subscribe(
44         //     // next seulement ...
45         //     (datas: Dives[]) => {
46         //         this.dives = datas;
47         //     }
48         // );
49
50     .subscribe({
51         next: // Call Back OK
52         (datas: Dives[]) => {
53             this.dives = datas;
54         },
55         error:
56         e => console.log(e)
57     },
58     complete:
59     () => console.log('Observer Complete')
60 );
61 }

dives.service.ts x
TP03-services-http > src > app > sharedModels > services > dives.service.ts > DivesService > getDives
1 import { Injectable } from '@angular/core';
2 import { Dives } from '../class/dives';
3 import { HttpClient } from '@angular/common/http';
4 import { Observable, tap, map } from 'rxjs';
5
6 @Injectable({
7     providedIn: 'root'
8 })
9
10 export class DivesService {
11
12     // 1- props
13     private dives: Dives[];
14
15     // 2- const
16     constructor(private _http: HttpClient) {
17         this.dives = [];
18     }
19
20     // 3- Méthodes
21     public getDives = (): Observable<Dives[]> => {
22
23         const API_URL = 'http://localhost:3001/dives';
24         // const API_URL = 'https://dev.webjs.fr/dives.json';
25
26         return this._http.get<Dives[]>(API_URL).pipe(
27             // le pipe permet d'enchaîner les opérateurs RXJS
28             // 1 opérateur RXJS est une fonction qui modifie
29             // ou transforme la séquence(stream ou flux) de valeurs émises par l'observable
30
31             tap(
32                 // opérateur de debug => console
33                 (response: any) => {
34                     console.log('----- Réponse depuis le service : ', response);
35                 }
36             ),
37             map(
38                 (datas: Dives[]) => {
39                     return datas.filter(
40                         (data: Dives) => {
41                             // return data.id===5
42                             return data.id >= 1
43                         }
44                     );
45                 }
46             )
47         );
48     }
49 }
```

Life Cycle du composant constructor & ngOnInit

ng g(enerate) s(ervice) dives

```
divs-list.component.ts 3  dives.service.ts X
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 B"
```

```
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

The image shows a code editor with two files open: `dives-list.component.ts` and `dives-list.component.html`. The left pane shows the TypeScript component file, and the right pane shows the HTML template file. Red arrows indicate the flow of data and dependencies between the two files.

dives-list.component.ts

```
1 import { Component, OnInit } from '@angular/core';
2 import { Dives } from 'src/app/sharedModels/class/dives';
3 import { DivesService } from 'src/app/sharedModels/service/dives';
4
5 @Component({
6   selector: 'app-dives-list',
7   templateUrl: './dives-list.component.html',
8   styleUrls: ['./dives-list.component.scss']
9 })
10 export class DivesListComponent implements OnInit {
11
12   // 1- Props
13   .....
14
15   // 2- Constructor
16   constructor(
17     // injection de dépendances requises
18     // au moment de la construction (création)
19     // du composant
20     private _service:DivesService
21   ) {
22     console.warn('Constructor');
23     .....
24   }
25
26   // 3- Lifecycle
27   ngOnInit(): void {
28     console.warn('NgOnInit');
29     // Rôle : chargement des datas
30     console.table(this._service.getDives());
31     .....
32 }
```

dives-list.component.html

```
1 <h1 class="alert alert-primary">Liste des Plongées</h1>
2
3 <div class="row">
4   <div class="col-md-3" *ngFor=".....">
5     <div class="card">
6       <!-- récupérer chaque image -->
7       
8       <div class="card-body">
9         <h3 class="card-title">Nom : .....</h3>
10        <p class="card-text">Description : .....</p>
11
12        <!-- cibler un élément du dom en template référence # -->
13        <button class="btn btn-primary" #btnPlus>En Savoir Plus</button>
14      </div>
15    </div>
16  </div>
17</div>
```

Red arrows indicate the flow of data and dependencies:

- From the `ngOnInit` method in `dives-list.component.ts` to the `*ngFor` loop in `dives-list.component.html`.
- From the `constructor` method in `dives-list.component.ts` to the `#btnPlus` button in `dives-list.component.html`.

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



Copyright Michel POCCIOLESI - ORSYS

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 Change¹** » ou « **Dirty Checking²** » 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 dev. 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

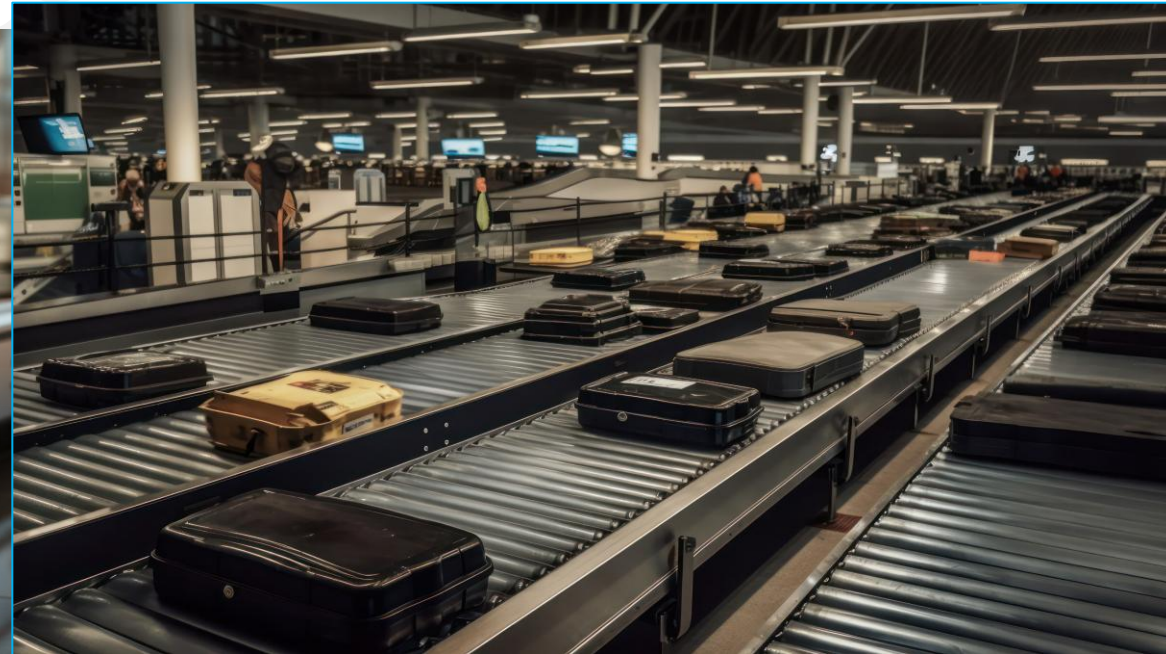
Niveau 1



Copyright Michel PROCIOSI - ORSYS

RXJS et les OBSERVABLES

- 1 Observable est un objet qui émet une suite (une séquence) de valeurs (datas) dans le temps.
Exemple : une requête HTTP
- On pourrait comparer cette suite de valeurs émises aux valises et bagages qui défilent sur un tapis roulant
- Chaque bagage a une destination mais va peut être croiser d'autres bagages qui n'ont pas la même destination mais empruntent le même tapis
- Ces valeurs peuvent être reçues par un poste de tri, interceptées, regroupées en fonction de critères et réaiguillées dynamiquement.
- On peut également stopper le tapis roulant
- Cela offre beaucoup de souplesse dans le traitement des données et la gestion des requêtes asynchrones.



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)

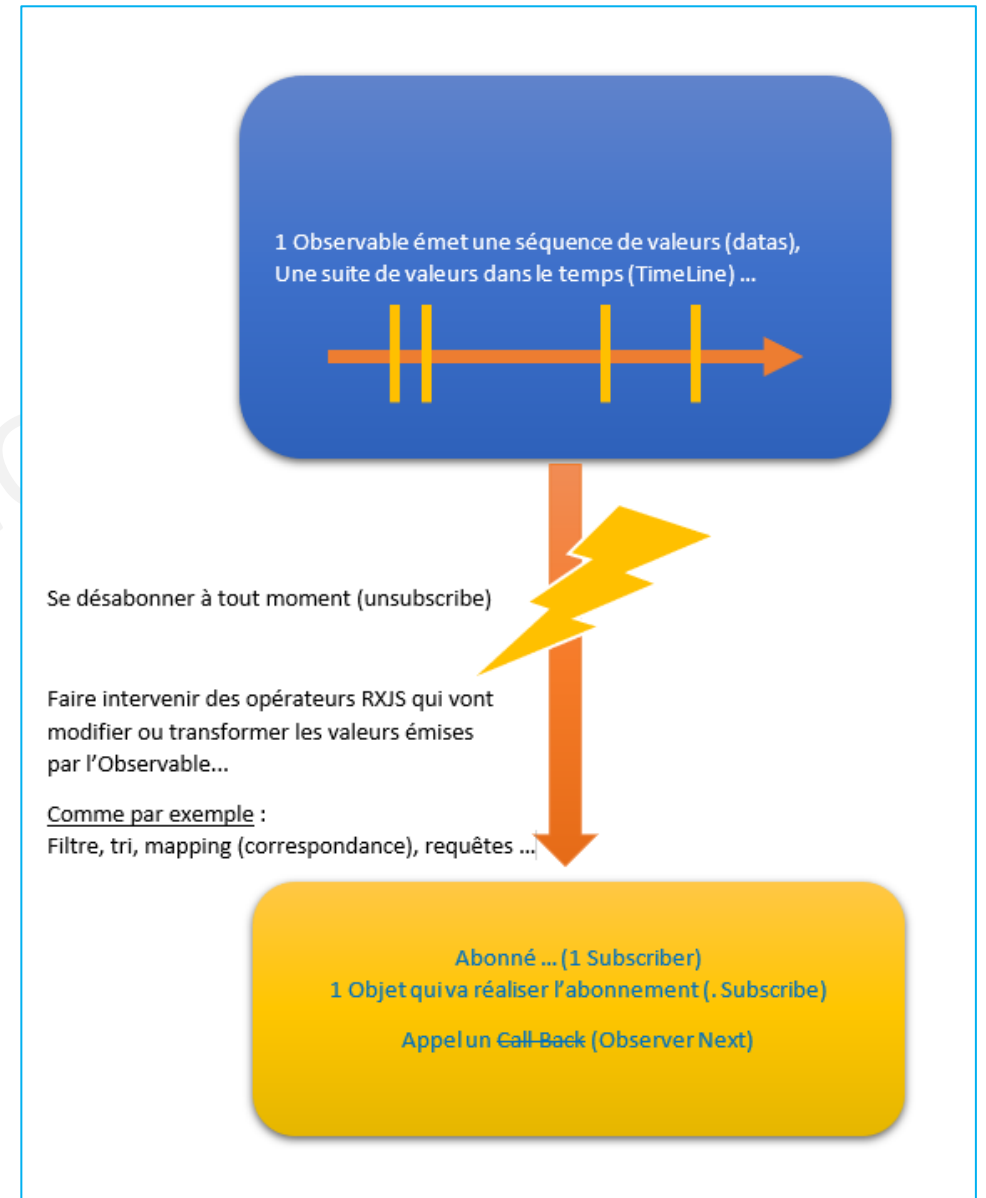
Opérateurs : des fonctions tap, map, merge, filter qui vont pouvoir modifier/transformer ce flux de valeurs émises par l'observable

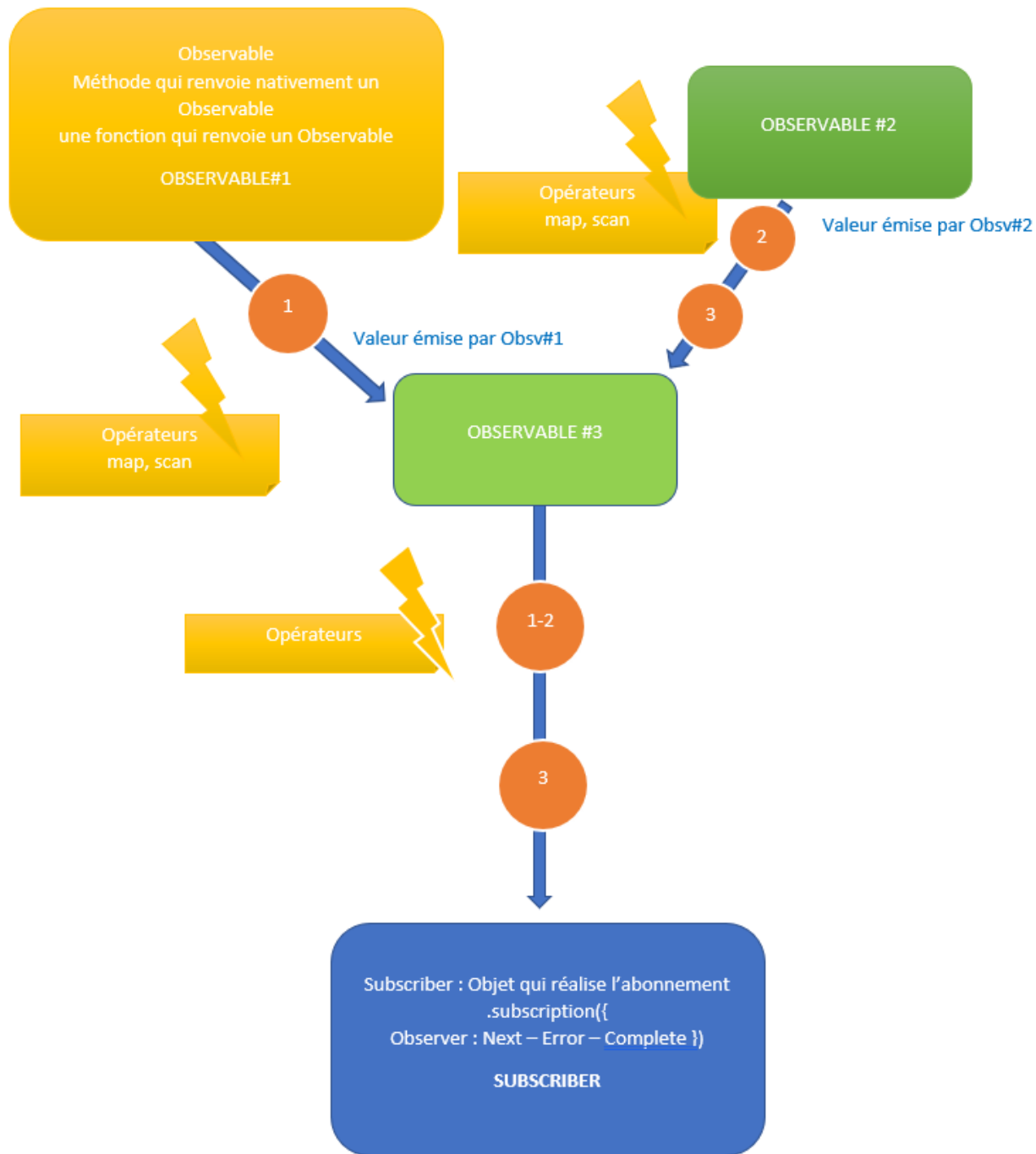
Dans le subscribe => Les observers (fcts de call back) qui se définissent dans le subscribe : Next - Error – Complete

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 recombinaison pour former une nouvelle séquence de valeurs !





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

Routage



Copyright Michel BOCCIOLESI - ORSYS

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

The image shows a code editor with four files open, illustrating the basic concepts of Angular routing:

- app-routing.module.ts**: Defines the routes for the application. A route is defined for the path `'liste-des-plongees'` with the component `DivesListComponent`.

```
1 import { NgModule } from '@angular/core';
2 import { RouterModule, Routes } from '@angular/router';
3
4 // import des composants
5 import { DivesListComponent } from '../webApp/root/dives/composants/dives-list/';
6 import { BodyComponent } from '../webApp/root/accueil/composants/body/body.comp';
7
8 const routes: Routes = [
9   { path: '', component: BodyComponent },
10   { path: 'liste-des-plongees', component: DivesListComponent },
11 ];
12
13 @NgModule({
14   imports: [RouterModule.forRoot(routes)],
15   exports: [RouterModule]
16 })
17 export class AppRoutingModule { }
18
19
```
- header.component.html**: Contains the navigation menu. A link is defined with the `routerLink` attribute pointing to the `'liste-des-plongees'` route.

```
4
5
6 <button class="navbar-toggler" type="button"
7   <span class="navbar-toggler-icon"></span>
8 </button>
9
10 <div class="collapse navbar-collapse" id="men
11   <ul class="navbar-nav">
12     <li class="nav-item">
13       <a class="nav-link"><i class="bi
14     </li>
15     <li class="nav-item">
16       <a class="nav-link"
17         routerLink="liste-des-plongees"
18     </li>
19     <li class="nav-item">
20       <a class="nav-link"><i class="bi bi-card-list"></i>
21     </li>
22     <li class="nav-item">
23       <a class="nav-link"><i class="bi
24     </li>
25   </ul>
26 </div>
27 </nav>
28
```
- landing-page.component.html**: Contains the `<router-outlet>` directive, which renders the component specified in the route configuration.

```
1 <!-- <div class="container"> -->
2   <app-header></app-header>
3
4   <div class="alert alert-warning">
5     <!-- comparé à un composant angular -->
6     <router-outlet></router-outlet>
7   </div>
8
9
10
11 <app-footer></app-footer>
12
```
- accueil.module.ts**: Defines the module for the landing page, importing the `RouterModule`.

```
11
12 @NgModule({
13   declarations: [
14     LandingPageComponent,
15     HeaderComponent,
16     FooterComponent,
17     BodyComponent
18   ],
19   imports: [
20     CommonModule,
21     RouterModule,
22   ]
23 })
24 export class AccueilModule { }
25
```

Red arrows indicate the flow of data: from the route definition in `app-routing.module.ts` to the `routerLink` attribute in `header.component.html`, and then to the `<router-outlet>` directive in `landing-page.component.html`.

Routing : Concepts de base

app-routing.module.ts

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

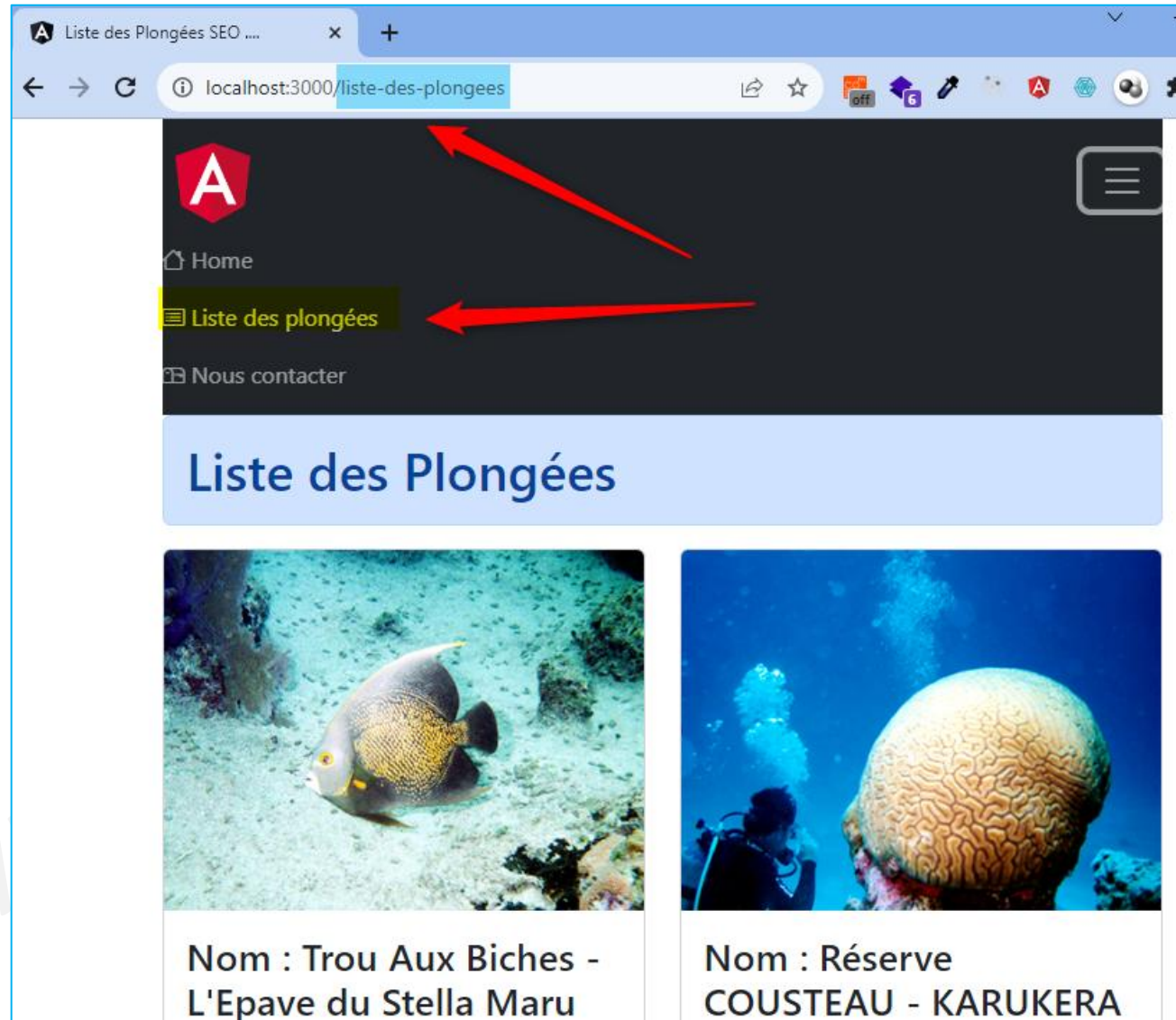
```
1 import { NgModule } from '@angular/core';
2 import { RouterModule, Routes } from '@angular/router';
3 import { BodyComponent } from '../webApp/root/accueil/composants/body/body.component';
4 import { DivesListComponent } from '../webApp/root/dives/composants/dives-list/dives-list.component';
5 import { Page404Component } from '../sharedModels/composants/page404.component';
6 import { ContactsComponent } from '../webApp/root/form/composants/contacts/contacts.component';
7 import { DivesDetailComponent } from '../webApp/root/dives/composants/dives-detail/dives-detail.component';
8
9 const routes: Routes = [
10   { path: '', component: BodyComponent },
11   // domain.tld/ (page de démarrage)
12   // { path: 'accueil', component: BodyComponent }, // domain.tld/accueil
13   { path: 'accueil', redirectTo: '', pathMatch: 'full' },
14   // { path: 'nouvelle-url', component: DivesListComponent },
15   // { path: 'ancienne-url-seo-déjà-référencée',
16   //   redirectTo: 'nouvelle-url', pathMatch: 'full' },
17   { path: 'contactez-nous', component: ContactsComponent },
18   { path: 'liste-des-plongees', component: DivesListComponent },
19   { path: 'liste-des-plongees/:id', component: DivesDetailComponent },
20
21   // Gestion des erreurs : toujours en dernier !!
22   // { path: '**', redirectTo: '', pathMatch: 'full' }
23   { path: '**', component: Page404Component }
24 ];
25
26 @NgModule({
27   imports: [RouterModule.forRoot(routes)],
28   exports: [RouterModule]
29 })
30 export class AppRoutingModule { }
```

header.component.html

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

```
1 <nav class="navbar navbar-expand-lg bg-dark navbar-dark sticky-top">
2   
3
4   <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-
5     <span class="navbar-toggler-icon"></span>
6   </button>
7
8   <div class="collapse navbar-collapse" id="menu">
9     <ul class="navbar-nav">
10       <li class="nav-item">
11         <a class="nav-link"
12           routerLink="accueil"
13           ><i class="bi bi-house"></i> Home</a>
14       </li>
15
16       <li class="nav-item">
17         <!-- <a class="nav-link"
18           [routerLink]="['liste-des-plongees']"
19           ><i class="bi bi-card-list"></i> Liste des plongées</a> -->
20         <a class="nav-link"
21           routerLink="liste-des-plongees"
22           ><i class="bi bi-card-list"></i> Liste des plongées</a>
23       </li>
24
25       <li class="nav-item">
26         <a class="nav-link"
27           routerLink="contactez-nous"
28           ><i class="bi bi-mailbox"></i> Nous contacter</a>
29       </li>
30     </ul>
31   </div>
32 </nav>
```

Routing : Concepts de base



Routing : Concepts de base

The image displays three code editors side-by-side, illustrating the configuration of Angular routing. Red arrows indicate the relationship between the HTML templates and the routing module.

landing-page.component.html

```
1 <div style="min-height:100vh;" class="alert alert-warning">
2   <app-header></app-header>
3   <!-- <app-body></app-body> -->
4   <!-- <app-dives-list></app-dives-list> -->
5
6   <!-- router-outlet primary -->
7   <div class="alert alert-primary">
8     <router-outlet></router-outlet>
9   </div>
10
11   <!-- router-outlet autres -->
12   <div class="alert alert-success">
13     <router-outlet name="outlet-2"></router-outlet>
14   </div>
15 </div>
16 <app-footer></app-footer>
```

app-routing.module.ts

```
16 domain.tld/accueil
17 // { path:'accueil' , redirectTo:'',
18   pathMatch:'full'},
19 // { path:'nouvelle-url',
20   component:DivesListComponent},
21 // { path:'ancienne-url-seo-déjà-référencée',
22   redirectTo:'nouvelle-url', pathMatch:'full'},
23 { path: 'liste-des-plongees', component:
24   DivesListComponent },
25 { path: 'liste-des-plongees/:id', component:
26   DiveDetailComponent },
27 { path: 'routing-angular', component:
28   CompRoutingOutletComponent, outlet: 'outlet-2',
29   canActivate: [RouteGuardService] },
30
31 // 1ère manière de faire : charger un composant depuis
32 // un module qui a son propre système de routage forChild
33 // {
34 //   path: 'mon-compte-client',
35 //   component: ClientLandingPageComponent,
36 //   children: clientRoutes
37 // },
38
39 // 2nde manière optimisée : lazy loading avec ES2015
40 // (promesses-import)
41 // {
42 //   path: 'mon-compte-client',
43 //   component: ClientLandingPageComponent,
44 //   loadChildren:
45 //     () => import('./webApp/compte-client/
46 //       compte-client.module').then(
```

header.component.html

```
16 Liste des plongées
17 </a>
18 </li>
19
20 <li class="nav-item">
21   <a class="nav-link" routerLink="mon-compt
22 </li>
23
24 <li class="nav-item">
25   <a class="nav-link" [routerLink]="[{
26     outlets : {
27       'outlet-2': 'routing-angular'
28     }]
29   >
30
31 [skipLocationChange]="true"><i class="bi bi-m
32 </li>
33
34 <li class="nav-item">
35   <a class="nav-link"><i class="bi bi-mailb
36 </li>
37 </ul>
38 </div>
39 </nav>
```

Red arrows point from the `<router-outlet name="outlet-2">` in the first editor to the `outlet: 'outlet-2',` in the second editor, and from the `[routerLink]="[{ outlets: { 'outlet-2': 'routing-angular' } }]"` in the third editor to the `routing-angular` path in the second editor.

Routing : Concepts avancés – QueryParams - Router



Nom : Trou Aux Biches - L'Epave du Stella Maru

Description : Plongée de niveau 2 FFESSM ou PADI au arge de la barrière de Corail, départ à 8h30. Profondeur -30 mètres. Rascasses Volantes, murènes ...

[Plus d'infos](#)



Liste des Plongées SEO ... x +

localhost:3000/liste-des-plongees

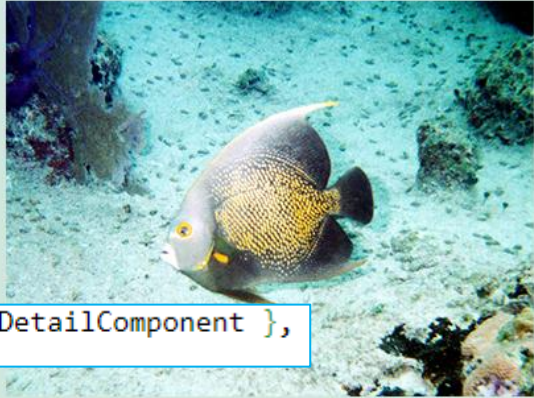
Plongée N° 1

Nom : Trou Aux Biches - L'Epave du Stella Maru

Description : Plongée de niveau 2 FFESSM ou PADI au arge de la barrière de Corail, départ à 8h30. Profondeur -30 mètres. Rascasses Volantes, murènes ...

Latitude : -20.0363

Longitude : 57.544700000000034



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

Routing : Concepts avancés – QueryParams - Router

The image shows a code editor with two files open. The left file, `dives-list.component.html`, contains the following HTML code:

```
1 <h1 class="alert alert-primary">Liste des Plongées</h1>
2
3 <div class="row">
4
5   <div class="col-md-6" *ngFor="let dive of dives">
6
7     <div class="card">
8       <!-- récupérer chaque image -->
9       
10      <div class="card-body">
11        <h3 class="card-title">Nom : {{dive.name}} </h3>
12        <p class="card-text">Description : {{dive.description}}</p>
13
14        <!-- <a
15          [routerLink]="[dive.id]"
16          [queryParams]="{dive.name}"
17        > -->
18        <!-- on interprète un des params de l'objet -->
19        <a
20          [routerLink]="[dive.id]"
21          [queryParams]="dive"
22          [skipLocationChange]="true"
23        >
24
25        <!-- on passe tout l'objet -->
26        <button class="btn btn-primary" #btnPlus>En Savoir Plus</button>
27      </a>
28    </div>
29  </div>
30 </div>
31 </div>
```

The right file, `dives-detail.component.ts`, contains the following TypeScript code:

```
1 import { Component, OnInit } from '@angular/core';
2 import { ActivatedRoute, Router } from '@angular/router';
3 import { Dives } from 'src/app/sharedModels/class/dives';
4
5 @Component({
6   selector: 'app-dives-detail',
7   templateUrl: './dives-detail.component.html',
8   styleUrls: ['./dives-detail.component.scss']
9 })
10 export class DivesDetailComponent implements OnInit {
11   // 1- props
12   public title: string;
13   public dive: Dives;
14
15   // 2- const
16   constructor(
17     private _routeActive: ActivatedRoute,
18     private _router: Router
19   ) {
20     this.title = '';
21     this.dive = new Dives();
22   }
23
24   // 3- lifeCycle
25   ngOnInit(): void {
26
27     const id = this._routeActive.snapshot.params['id'];
28     console.log(id);
29     this.title = `Plongée N° ${id}`;
30     // re-consommer le service ?
31     // ré-exploiter les datas stockées en storage ou en indexedDB
32     // Evidemment utiliser les fonctions de routage d'Angular ...
33   }
```

A red arrow points from the `[routerLink]` and `[queryParams]` attributes in the HTML file to the `ActivatedRoute` and `Router` imports in the TypeScript file.

Routing : Concepts avancés – QueryParams - Router

The image displays a code editor with five files open, illustrating advanced Angular routing concepts, specifically the use of QueryParams.

dives-list.component.html

```
1 <h1 class="alert alert-primary">Liste des Plongées</h1>
2
3 <div class="row">
4
5   <div class="col-md-6" *ngFor="let dive of dives">
6
7     <div class="card">
8       <!-- récupérer chaque image -->
9       
10      <div class="card-body">
11        <h3 class="card-title">Nom : {{dive.name}} </h3>
12        <p class="card-text">Description : {{dive.description}}</p>
13
14        <!-- [routerLink]="[dive.id]" -->
15        <!-- Ecriture simplifiée du routerLink => routerLink="{{dive.id}}<br>
16        <!-- routerLink="details/{{dive.id}}" -->
17        <!-- en chemin relatif -->
18        <button
19
20          routerLink="/liste-des-plongees/details/{{dive.id}}"
21          [queryParams]="{dive: dive.id}"
22          [skipLocationChange]="true"
23
24          class="btn btn-primary" #btnPlus>En Savoir Plus</button>
25
26      </div>
27    </div>
28  </div>
```

dives-details.component.ts

```
1 import { Component, OnInit } from '@angular/core';
2 import { ActivatedRoute } from '@angular/router';
3
4 @Component({
5   selector: 'app-dives-details',
6   templateUrl: './dives-details.component.html',
7   styleUrls: ['./dives-details.component.scss']
8 })
9 export class DivesDetailsComponent implements OnInit {
10
11   // -----
12   constructor(
13     private _routeActive: ActivatedRoute
14   ) {
15   }
16   // -----
17   ngOnInit() {
18     const id=this._routeActive.snapshot.params['param'];
19     console.clear();
20     console.log('ID récupéré : ', id);
21     // TP : passer l'ID à la Vue
22     // -----
23     this._routeActive.queryParams
24       .subscribe(
25         (params:any) => {
26           console.log('QueryParams : ', params);
27           // TP : passer l'objet récupéré à la vue
28         }
29       )
30   }
31 }
```

app-routing.module.ts

```
10 const routes: Routes = [
11
12   { path: '', component: BodyComponent }, // domain.tld/ (page de démarrage)
13   { path: 'liste-des-plongees', component: DivesListComponent },
14
15   // { path:'accueil', component:BodyComponent}, // domain.tld/accueil
16
17   // { path:'nouvelle-url', component:DivesListComponent},
18   // { path:'ancienne-url-seo-déjà-référencée', redirectTo:'nouvelle-url' },
19
20   { path: 'liste-des-plongees/details/:param', component: DivesDetailsComponent },
21
22   { path: '**', component: Page404Component }
23 ]
```

bdd.json

```
1 {
2   "dives": [
3     {
4       "id": 1,
5       "name": "Trou Aux Biches - L'Epave du Stella Maru",
6       "location": "Ile Maurice - Trou aux Biches - Nord/Ouest",
7       "level": 2,
8       "description": "Plongée de niveau 2 FFESSM ou PADI au arge de l'île",
9       "latitude": -20.0363,
10      "longitude": 57.544700000000034,
11      "evaluation": [
```

dives.module.ts

```
1 import { NgModule } from '@angular/core';
2 import { CommonModule } from '@angular/common';
3 import { DivesListComponent } from './dives-list.component';
4 import { HttpClientModule } from '@angular/common/http';
5 import { FormsModule } from './forms.module';
6 import { RouterModule } from '@angular/router';
7 import { DivesDetailsComponent } from './dives-details.component';
8
9
10
11 @NgModule({
12   declarations: [
```

Routing : Concepts avancés – QueryParams - Router

divs-details.component.ts

```
21 ngOnInit() {
22   const id = this._routeActive.snapshot.params['param'];
23   console.clear();
24   console.log('ID récupéré :', id);
25   // TP : passer l'ID à la Vue
26   // -----
27   this._routeActive.queryParams
28     .subscribe(
29       (params: any) => {
30         console.log('QueryParams : ', params);
31         // TP : passer l'objet récupéré à la vue
32         this.dive = params;
33       }
34     )
35 }
36 // -- Méthodes
37 public goBack = () => {
38   this._router.navigateByUri('liste-des-plongees');
39 }
40
41 public goHome = () => {
42   this._router.navigate(
43     [''],
44     {
45       queryParams: {
46         page: 'details-plongée',
47         datas: 15
48       }
49     }
50 )
51 }
```

divs-details.component.html

```
1 <h1>Détail de la plongée N° {{dive.id}}</h1>
2 <!-- TP afficher l'objet passé par routage -->
3
4 <!-- Single way binding -->
5 <!-- TS => HTML : [directive] et {{string interpolation}} -->
6 <!-- HTML => TS : (event du DOM) par ex : (click)-->
7 <p>
8   <button class="btn btn-warning" (click)="goBack()">Go Back</button>
9
10  <button class="btn btn-danger" routerLink="/liste-des-plongees">RouterLink</button>
11 </p>
12 <p>
13   <button class="btn btn-success" (click)="goHome()">Go Home</button>
14 </p>
```

localhost

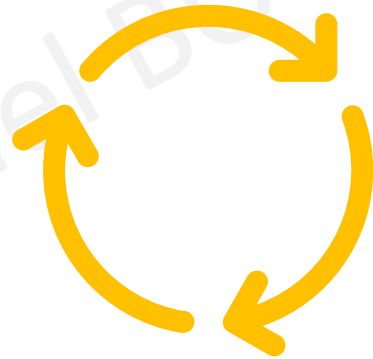
Liste des plongées

localhost:3000/?page=details-plongée&datas=15

Home Liste des plongées Nous contacter

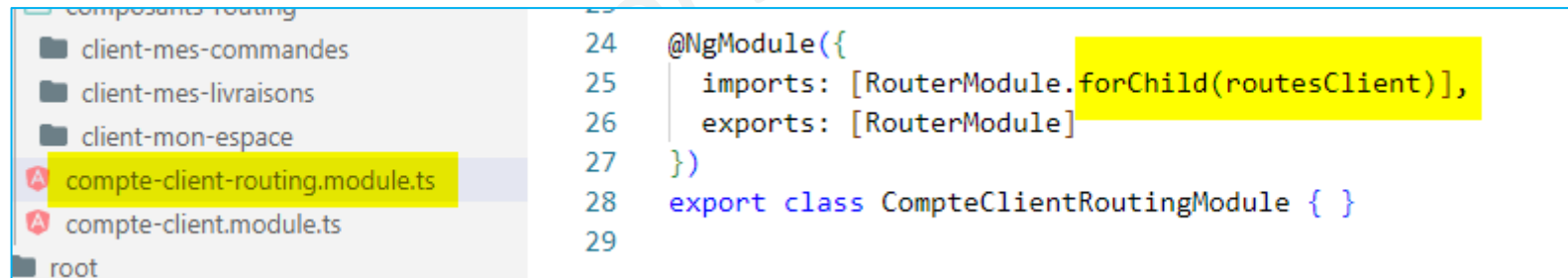
THE BODY

Routage Avancé
Lazy Loading
Optimisation du build



Routing : Concepts avancés – Le Lazy Loading

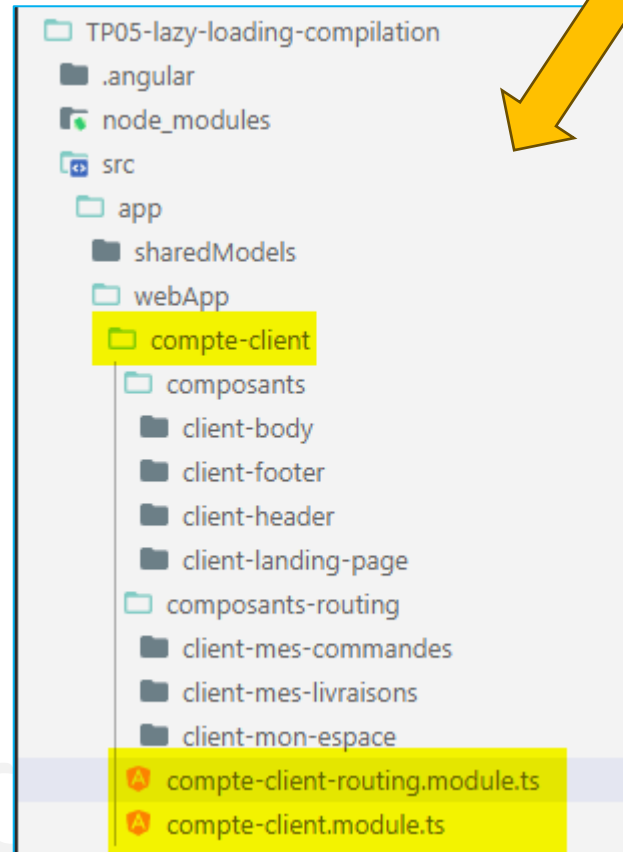
Créer le module compte-client avec l'option « --routing »
Cette option crée un module de routage spécifique
forChild



```
24 @NgModule({
25   imports: [RouterModule.forChild(routesClient)],
26   exports: [RouterModule]
27 })
28 export class CompteClientRoutingModule { }
29
```

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

```
composants-routing
├── client-mes-commandes
├── client-mes-livraisons
├── client-mon-espace
├── compte-client-routing.module.ts
├── compte-client.module.ts
└── root
```

```
24 @NgModule({
25   imports: [RouterModule.forChild(routesClient)],
26   exports: [RouterModule]
27 })
28 export class CompteClientRoutingModule { }
29
```

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\Injection-angular-2023-06\TP05-lazy-loading-compilation> npm run build

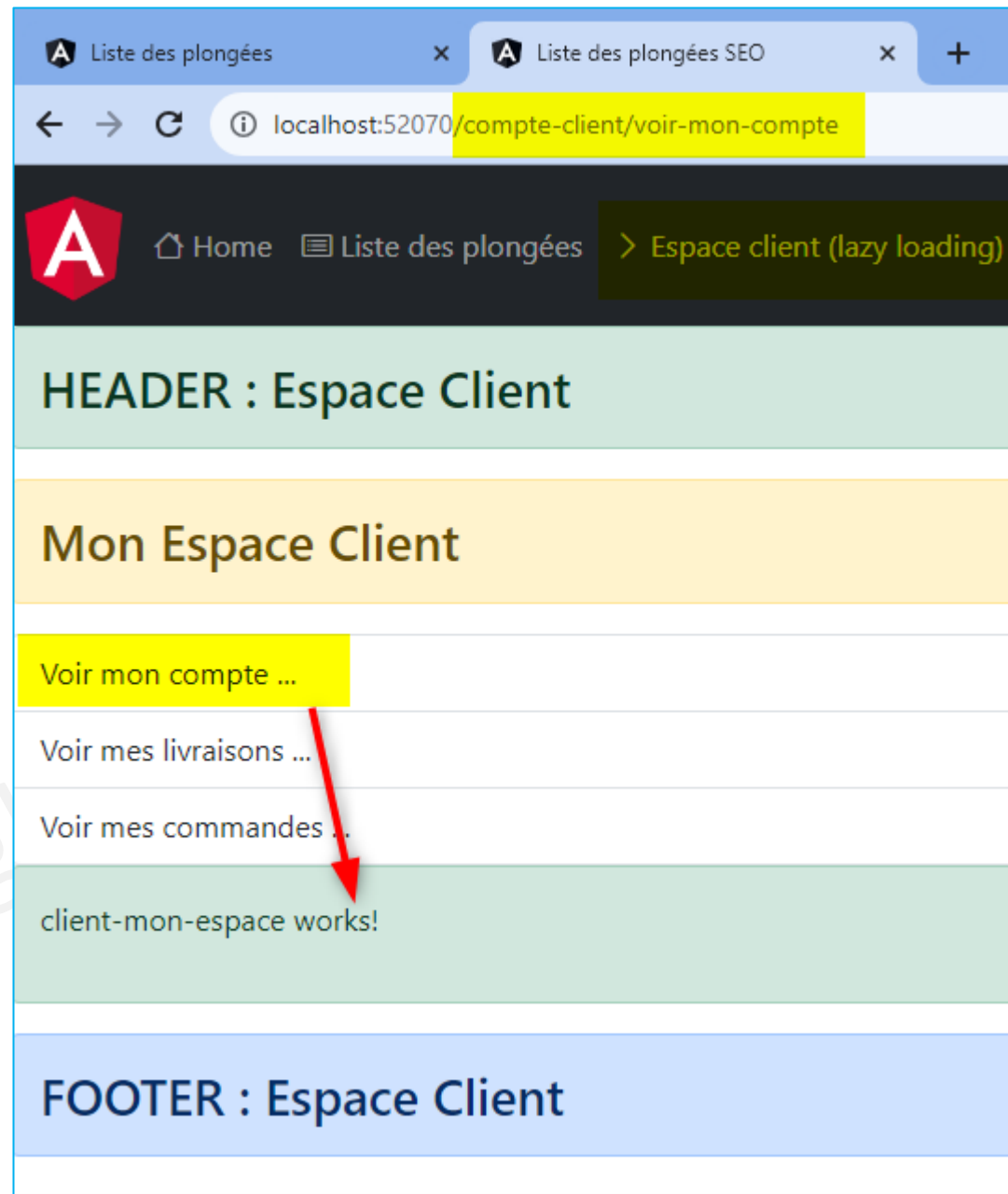
> tp02-bonnes-pratiques-injection-dependances@0.0.0 build
> ng build --stats-json

✓ 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

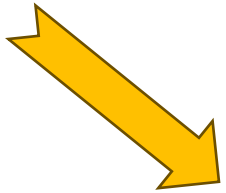
Routing : Concepts avancés – Le Lazy Loading



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)

The image shows a code editor with four files open, illustrating the setup for a client application without lazy loading.

client-landing-page.component.html

```
1 <div class="alert alert-primary">
2   <app-client-header></app-client-header>
3   <hr>
4   <app-client-body></app-client-body>
5   <hr>
6   <app-client-footer></app-client-footer>
7 </div>
```

client-body.component.html

```
1 <ul class="list-group">
2   <li class="list-group-item">
3     <a class="nav-link"
4       [routerLink]="['mon-espace']"
5       [routerLinkActive]="['lien-actif']">
6       Mon espace Client</a>
7   </li>
8   <li class="list-group-item">
9     <a class="nav-link"
10      routerLink="mes-commandes"
11      [routerLinkActive]="['lien-actif']">
12      Mes commandes</a>
13   </li>
14   <li class="list-group-item">
15     <a class="nav-link"
16      [routerLink]="['mes-livraisons']"
17      [routerLinkActive]="['lien-actif']">
18      Mes livraisons</a>
19   </li>
20 </ul>
21
22 <p class="alert alert-success">
23   <router-outlet></router-outlet>
24 </p>
25
```

accueil.module.ts

```
11 // import { CompteClientModule } from '../..formation/compte-client,
12
13 @NgModule({
14   declarations: [
15     LandingPageComponent,
16     HeaderComponent,
17     BodyComponent,
18     FooterComponent
19   ],
20   imports: [
21     CommonModule,
22     RouterModule,
23     // nos modules
24     FilmsModule,
25     FormsReactiveModule,
26     // CompteClientModule,
27     RxjsModule
28   ],
29   exports: [
30     LandingPageComponent,
31     HeaderComponent,
32     BodyComponent
33   ]
34 })
35 export { AccueilModule };
```

compte-client-routing.module.ts

```
1 import { NgModule } from '@angular/core';
2 import { RouterModule, Routes } from '@angular/router';
3 import { ClientMonEspaceComponent } from '../composants-routing/client-mon-espace/client-mon-espace.component';
4 import { ClientCommandesComponent } from '../composants-routing/client-commandes/client-commandes.component';
5 import { ClientLivraisonsComponent } from '../composants-routing/client-livraisons/client-livraisons.component';
6
7 export const routesClient: Routes = [
8   // const routesClient: Routes = [
9
10   { path: 'mon-espace', component: ClientMonEspaceComponent },
11   { path: 'mes-commandes', component: ClientCommandesComponent },
12   { path: 'mes-livraisons', component: ClientLivraisonsComponent },
13 ];
```

app-routing.module.ts

```
8 import { NgModule } from '@angular/core';
9 import { RouterModule, Routes } from '@angular/router';
10 import { LandingPageComponent } from '../webApp/formation/rxjs/landing-page/landing-page.component';
11 import { ClientLandingPageComponent } from '../webApp/formation/compte-client/client-landing-page/client-landing-page.component';
12
13 // -----
14 import { routesClient } from '../webApp/formation/compte-client/compte-client-routing.module';
15
16 const routes: Routes = [
17   { path: '', component: BodyComponent },
18   { path: 'accueil', component: BodyComponent },
19   { path: 'liste-des-films', component: ListeDesFilmsComponent },
20   // 1ère étape : sans lazy loading
21   // raccrocher un module qui a son propre système de routage
22   {
23     path: 'compte-client',
24     component: ClientLandingPageComponent,
25     children: routesClient
26   },
27 ];
28
29 @NgModule({
30   imports: [RouterModule.forRoot(routes)],
31   exports: [RouterModule]
32 })
33 export { AppRoutingModule };
```

Le Lazy Loading : Mise en pratique

Etape #2 (Lazy Loading)

client-landing-page.component.html

1 <div class="alert alert-primary">
2 <app-client-header></app-client-h
3 <hr>
4 <app-client-body></app-client-body
5 <hr>
6 <app-client-footer></app-client-f
7 </div>

client-body.component.html

1 <ul class="list-group">
2 <li class="list-group-item">
3 <a class="nav-link"
4 [routerLink]="['mon-espace']"
5 [routerLinkActive]="['lien-actif']"
6 >Mon espace Client
7
8 <li class="list-group-item">
9 <a class="nav-link"
10 routerLink="mes-commandes"
11 [routerLinkActive]="['lien-actif']"
12 >Mes commandes
13
14 <li class="list-group-item">
15 <a class="nav-link"
16 [routerLink]="['mes-livraisons']"
17 [routerLinkActive]="['lien-actif']"
18 >Mes livraisons
19
20
21
22 <p class="alert alert-success">
23 <router-outlet></router-outlet>
24 </p>
25

accueil.module.ts

11 // import { CompteClientModule } from '../formation/compte-client/compte-client.
12
13 @NgModule({
14 declarations: [
15 LandingPageComponent,
16 HeaderComponent,
17 BodyComponent,
18 FooterComponent
19]},
20 imports: [
21 CommonModule,
22 RouterModule,
23 // nos modules
24 FilmsModule,
25 FormsReactiveModule,
26 // CompteClientModule,
27 RxjsModule

app-routing.module.ts

26
27 // 2ème étape : Avec lazy loading (promesses ES2015 😊)
28 // {
29 // path: 'compte-client',
30 // component: ClientLandingPageComponent,
31 // loadChildren:
32 // () => import('../webApp/formation/compte-client/compte-client.module')
33 // .then(
34 // (m) => {
35 // return m.CompteClientModule
36 // }
37 //)
38 // },
39
40 // 3ème étape : Avec Lazy Loading (ES2017 : Async/await => asynchrone 😊)
41 {
42 path: 'compte-client',
43 component: ClientLandingPageComponent,
44 loadChildren:
45 async () => (await import('../webApp/formation/compte-client/compte-client.module'
46 // syntactic sugar
47 data :{
48 preload:true
49 }
50 }
51 },

compte-client-routing.module.ts

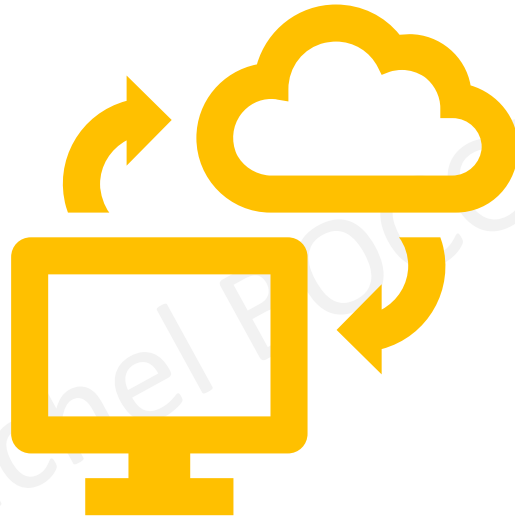
1 import { NgModule } from '@angular/core';
2 import { RouterModule, Routes } from '@angular/router';
3 import { ClientMonEspaceComponent } from './composants-routing/client-mon-espace/client-mon-espace.c
4 import { ClientCommandesComponent } from './composants-routing/client-commandes/client-commandes.com
5 import { ClientLivraisonsComponent } from './composants-routing/client-livraisons/client-livraisons.
6
7 export const routesClient: Routes = [
8 const routesClient: Routes = [
9
10 { path: 'mon-espace', component: ClientMonEspaceComponent },
11 { path: 'mes-commandes', component: ClientCommandesComponent },
12 { path: 'mes-livraisons', component: ClientLivraisonsComponent },
13];
14
15 @NgModule({
16 imports: [RouterModule.forChild(routesClient)],
17 exports: [RouterModule]
18 })

Mode standalone
1 seul fichier de routage

Le Lazy loading
se gère encore plus facilement 😊

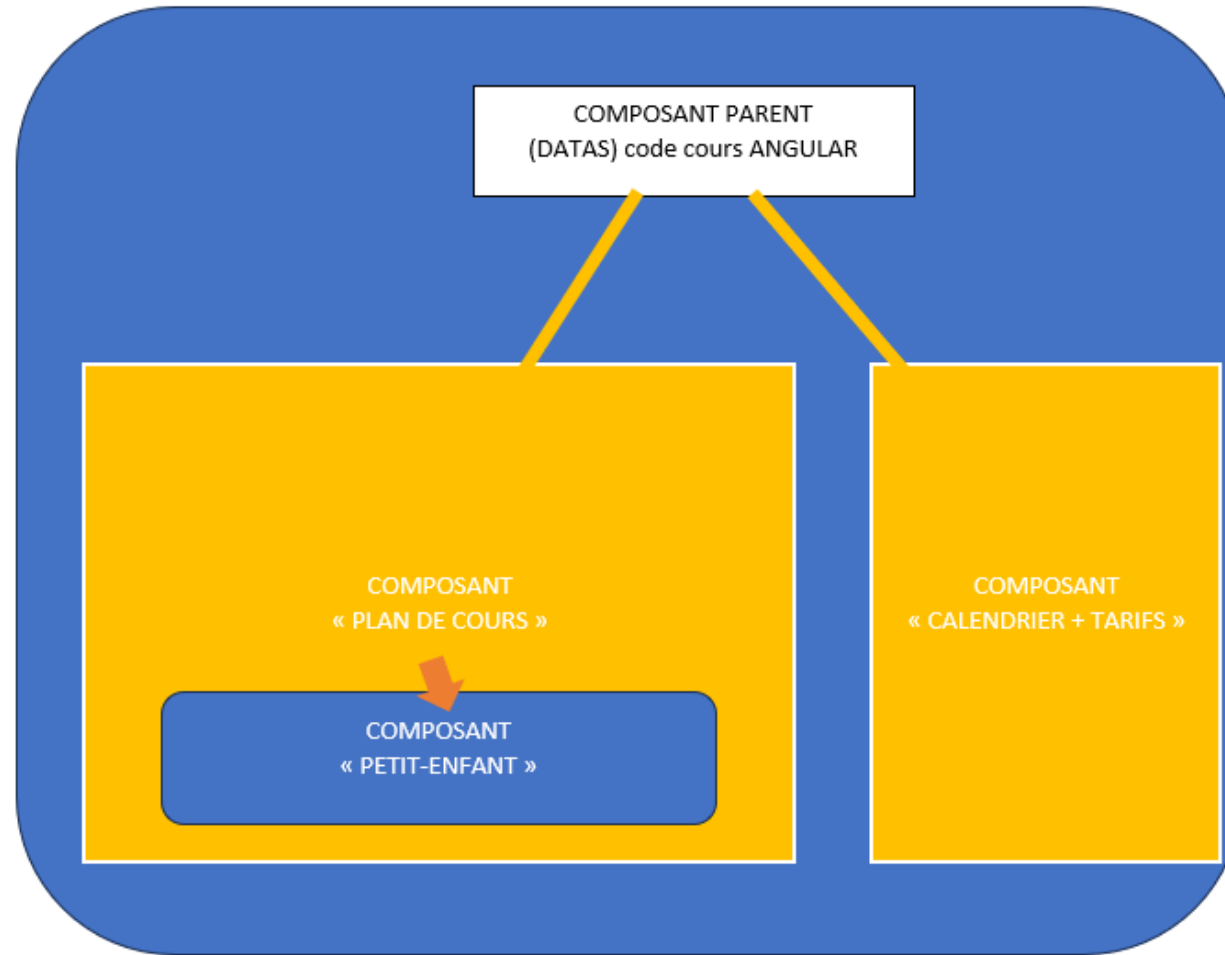
```
TS app.routes.ts X
TP05-communication-composants > src > app > TS app.routes.ts > ...
1 import { Routes } from '@angular/router';
2 import { HomeComponent } from '../webApp/accueil/home/home.component';
3 import { CommandesComponent } from '../webApp/compte-client/components-routing/commandes/commandes.component';
4 import { LivraisonsComponent } from '../webApp/compte-client/components-routing/livraisons/livraisons.component';
5 import { SignalsComponent } from '../webApp/signals/signals/signals.component';
6 import { DivesListComponent } from '../webApp/dives-list/components/dives-list/dives-list.component';
7 import { ContactsComponent } from '../webApp/contacts/components/contacts/contacts.component';
8 import { Page404Component } from '../shared/shared-components/page404.component';
9 import { DivesDetailComponent } from '../webApp/dives-list/components/dives-detail/dives-detail.component';
10 import { TpComposantsComponent } from '../webApp/tp-composants/components/tp-composants/tp-composants.component';
11
12
13 export const routes: Routes = [
14   { path: '', component: HomeComponent, title: 'Angular 18 - StandAlone Components' },
15   { path: 'liste-des-plongees', component: DivesListComponent, title: 'Liste des plongees' },
16   { path: 'liste-des-plongees/:id', component: DivesDetailComponent },
17   { path: 'les-signals-angular-17', component: SignalsComponent, title: 'Les signals Angulars 17 - trop Cool' },
18   { path: 'tp-composants', component: TpComposantsComponent, title: '😎' },
19
20   // Avec le Lazy Loading Component 😊
21   {
22     path: 'compte-client', title: 'Mon espace client - Formation Angular',
23     loadComponent:
24       () => import('../webApp/compte-client/components/compte-client/compte-client.component')
25         .then(
26           (c) => c.EntryCompteClientComponent
27         ),
28     children: [
29       { path: 'mes-commandes', component: CommandesComponent },
30       { path: 'mes-livraisons', component: LivraisonsComponent },
31     ]
32   },
33   // TP ajouter le composant ContactsComponent en lazy Loading
34   // { path: 'nous-contactez', component: ContactsComponent, title: 'Wow Génial les formulaires NG 🚗' },
35
36   // {path:'**', redirectTo:'',title:'Page Introuvable !'}
37   { path: '**', component: Page404Component, title: 'Page Introuvable !' }
38 ];
39
```

Input - Output



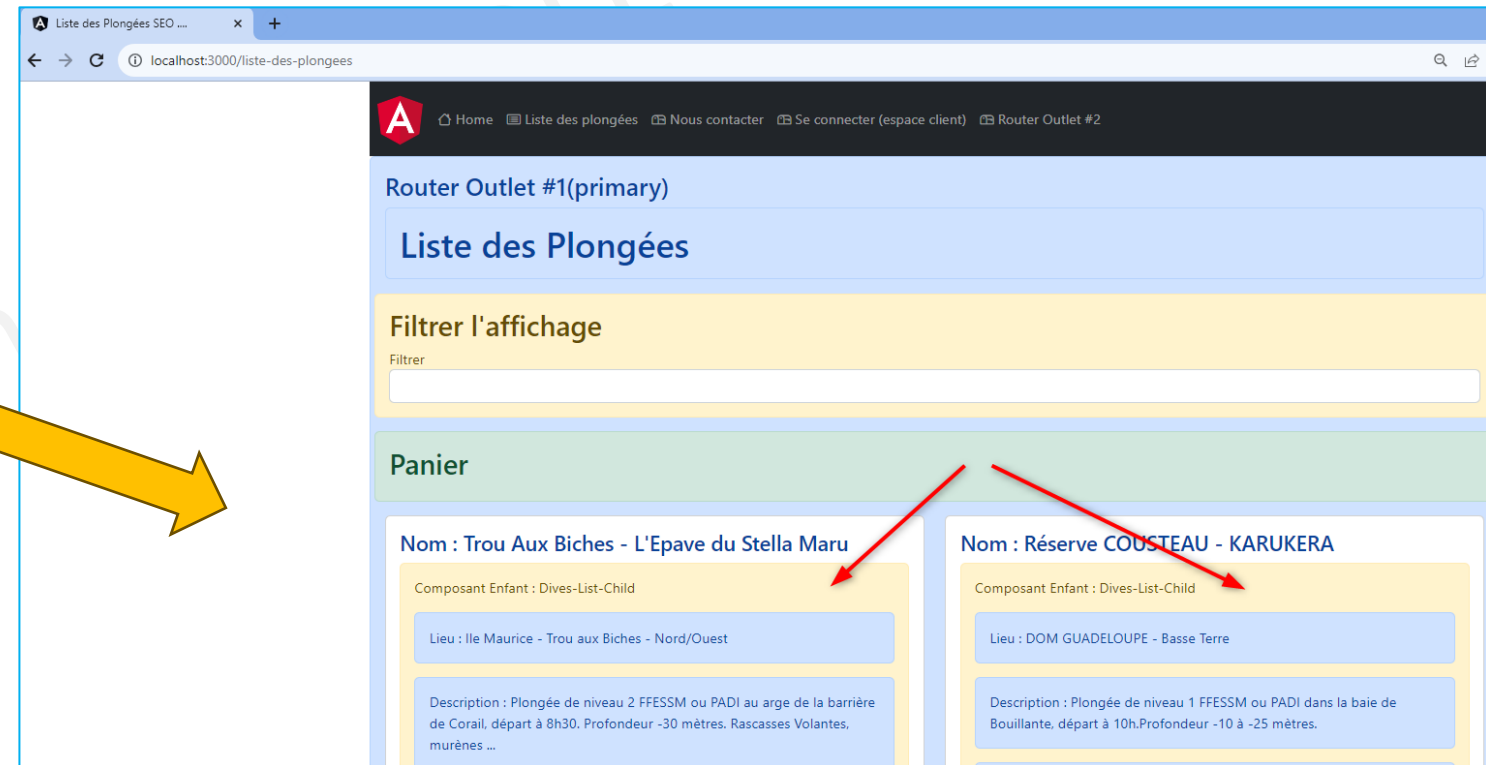
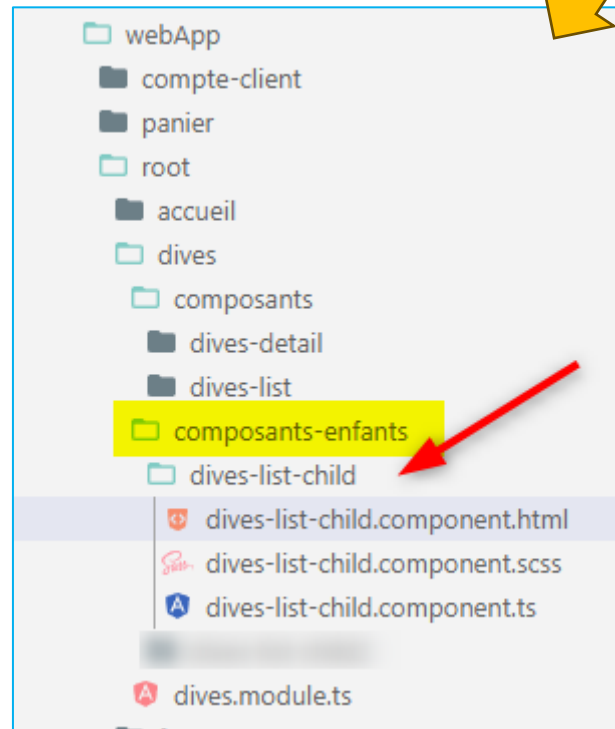
Copyright Michael F. CIOLESI - ORSYS

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 données 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
```

 [Home](#) [Liste des plongées](#) [Nous contacter](#) [Se connecter \(espace](#)

Panier

Nom : Trou Aux Biches - L'Epave du Stella Maru

Composant Enfant : Dives-List-Child

Lieu : Ile Maurice - Trou aux Biches - Nord/Ouest

Description : Plongée de niveau 2 FFESSM ou PADI au arge de la barrière de Corail, départ à 8h30. Profondeur -30 mètres. Rascasses Volantes, murènes ...

Latitude : -20.0363

Longitude : 57.544700000000034

Niveau : 2

Composant Enfant : Dives-List-Child #2



Communication : composants parents et enfants

The image displays a code editor with four files open, illustrating the communication between parent and child components in an Angular application. Red arrows highlight the flow of data and events.

dives-list.component.ts

```
114 // 4- méthodes
115 public diveSelected = (e: any) => {
116   console.log('depuis le parent $event => ',e);
117
118   let isChecked=e.paramIsChecked;
119   let diveSelect=e.paramDive;
120
121   console.log(`Plongée : ${diveSelect.name} - Etat : ${isChecked}`);
122
123   if (isChecked) {
124     this.panier.push(diveSelect);
125     console.table(this.panier);
126   } else {
127     let keyPanier = this.panier.indexOf(diveSelect);
128     if (keyPanier >= 0) {
129       this.panier.splice(keyPanier, 1);
130       console.table(this.panier);
131     }
132   }
133 }
134
135
136
```

dives-list-child.component.ts

```
9 export class DivesListChildComponent {
10
11   // les @input et les @output permettent d'établir la communication
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 template
17   @Input() inputDive!:Dives;
18   @Input() inputInfos!:string;
19   // -----
20   @Output() outputEventDive:EventEmitter<any>;
21
22   // 2- const
23   constructor(){
24     this.outputEventDive = new EventEmitter();
25   }
26
27   // 3- méthodes
28   public choixDive = (e:any) => {
29     console.clear();
30     console.log(e.target.checked);
31
32   }
33 }
34
```

dives-list-child2.component.ts

```
1 import { Component, Input } from '@angular/core';
2 import { Dives } from 'src/app/sharedModels/class';
3
4 @Component({
5   selector: 'app-dives-list-child2',
6   templateUrl: './dives-list-child2.component.html',
7   styleUrls: ['./dives-list-child2.component.scss'],
8 })
9 export class DivesListChild2Component {
10
11   @Input() inputDive!:Dives;
12
13 }
14
```

dives-list.component.html

```
41
42 <div class="card-body">
43   <h4 class="card-title">Nom : {{dive.name}} </h4>
44
45   <!-- Composant Enfant -->
46   <div class="alert alert-warning">
47     <p>Composant Enfant : Dives-List-Child</p>
48     <app-dives-list-child
49       [inputDive]="dive"
50       [inputInfos]="infos"
51       (outputEventDive)="diveSelected($event)"
52     ></app-dives-list-child>
53   </div>
54   <!-- Composant Enfant -->
55
56
```

dives-list-child.component.html

```
14
15 <p>Composant Enfant : Dives-List-Child #2</p>
16 <app-dives-list-child2
17   [inputDive]="inputDive"
18 ></app-dives-list-child2>
19
20 </div>
21 <!-- Composant Enfant -->
22
23
24
25 <p class="alert alert-primary">
26   <input type="checkbox" (change)="choixDive($event)">
27 </p>
28 <!-- $event est l'objet global evenement -->
29
```

dives-list-child2.component.html

```
1 <p class="alert alert-primary">
2   
3
4 </p>
5
```

Diagrammatical Flow:

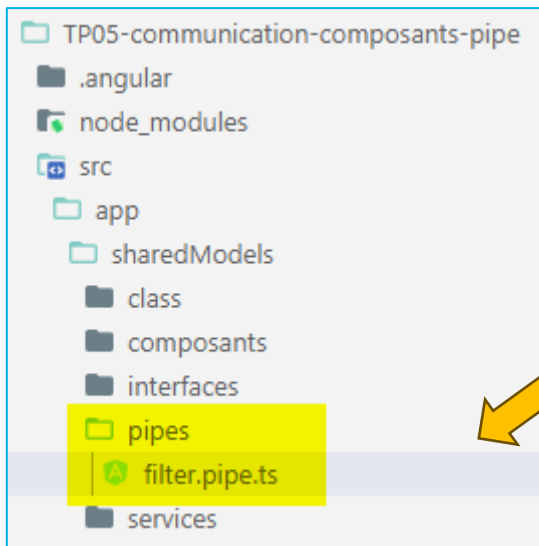
- Parent to Child 1:** A red arrow points from the `diveSelected` method in `dives-list.component.ts` to the `choixDive` method in `dives-list-child.component.ts`. Another red arrow points from the `(outputEventDive)="diveSelected($event)"` attribute in `dives-list.component.html` to the `choixDive` method in `dives-list-child.component.ts`.
- Child 1 to Child 2:** A red arrow points from the `choixDive` method in `dives-list-child.component.ts` to the `(change)="choixDive($event)"` attribute in `dives-list-child2.component.html`.

Pipes Angular



Copyright Michel ROCCHIOLESI - ORSYS

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

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

```
16 // -----
17 public dives: Dives[] = [];
18 @ViewChild('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.
28             toLocaleLowerCase()) || paramDive.description.toLocaleLowerCase().
29             includes(motif.toLocaleLowerCase())
30         }
31       )
32     }
33   }
34 }
```

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 ],
29 exports: [
```

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>Filtre 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
36       </div>
37     </div>
38   </div>
39 </div>
```



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

Nom : Site de Saint Jean

Lieu : Saint Jean Cap Ferrat - 06

Description : Plongée de niveau 2 FFESSM ou ...
lire la suite

 **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()

```
divesModule.ts
TP05-communication-composants-pipe > src > app > webApp > root > dives > dives.module.ts > DivesModule
11 import { VoirPlusPipe } from 'src/app/sharedModels/pipes/voir-plus.pipe';
12
13
14 @NgModule({
15   declarations: [
16     DivesListComponent,
17     DivesDetailComponent,
18     DivesListChildComponent,
19     DivesListChild2Component,
20     // *****
21     // on déclare AUSSI les pipes
22     FilterPipe,
23     VoirPlusPipe
24   ],
25   imports: [
26     CommonModule,
27     HttpClientModule,
28   ],
29 })
30 export class DivesModule {}

voir-plus.pipe.ts
TP05-communication-composants-pipe > src > app > sharedModels > pipes > voir-plus.pipe.ts > VoirPlusPipe > transform
1 import { Pipe, PipeTransform } from '@angular/core';
2
3 @Pipe({
4   name: 'voirPlus'
5 })
6 export class VoirPlusPipe implements PipeTransform {
7
8   transform(datas:string, nbCars:number) {
9     return `${datas.substring(0, nbCars)} ...`;
10  }
11
12 }
13

divesModule.html
TP05-communication-composants-pipe > src > app > webApp > root > dives > dives-list-child > dives-list-child.component.html > p.alert
1 <p class="alert alert-primary">Lieu : {{inputDive.location}}</p>
2
3
4 <p class="alert alert-primary">Description : {{inputDive.description | voirPlus:30}} <em>lire la
  suite</em></p>
5
6
7 <p class="alert alert-primary">Latitude : {{inputDive.latitude}}</p>
8 <p class="alert alert-primary">Longitude : {{inputDive.longitude}}</p>
9 <p class="alert alert-primary">Niveau : {{inputDive.level}}</p>
10
11 <p class="alert alert-danger">{{inputInfos}}</p>
12
13 <!-- appel enfant -->
14 <app-dives-list-child2
15   [inputDive2]="inputDive"
16 ></app-dives-list-child2>
17
18
19
20 <p class="alert alert-primary">
21   <input type="checkbox" (change)="choixDive($event)">
22   <!-- $event = EVENT GLOBAL -->
23 </p>
24
```

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



Copyright Michel ROCCIOLESI - ORSYS

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)

The image shows a Visual Studio Code interface with three main panels:

- EXPLORATEUR (Left):** Displays the project structure. The file `documentation.md` is highlighted in the `test-unitaires` folder.
- documentations.md (Middle):** Shows the content of the file with the following text:

```
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 documentations.md (Right):** Shows a rendered preview of the markdown content.

Two red arrows originate from the `documentation.md` file in the Explorer and point to the corresponding sections in the preview window: one points to the first section header and the other points to the second section header.

Savoir documenter son projet

1-Les tests Unitaires

Composants :

Pipes :

Directives :

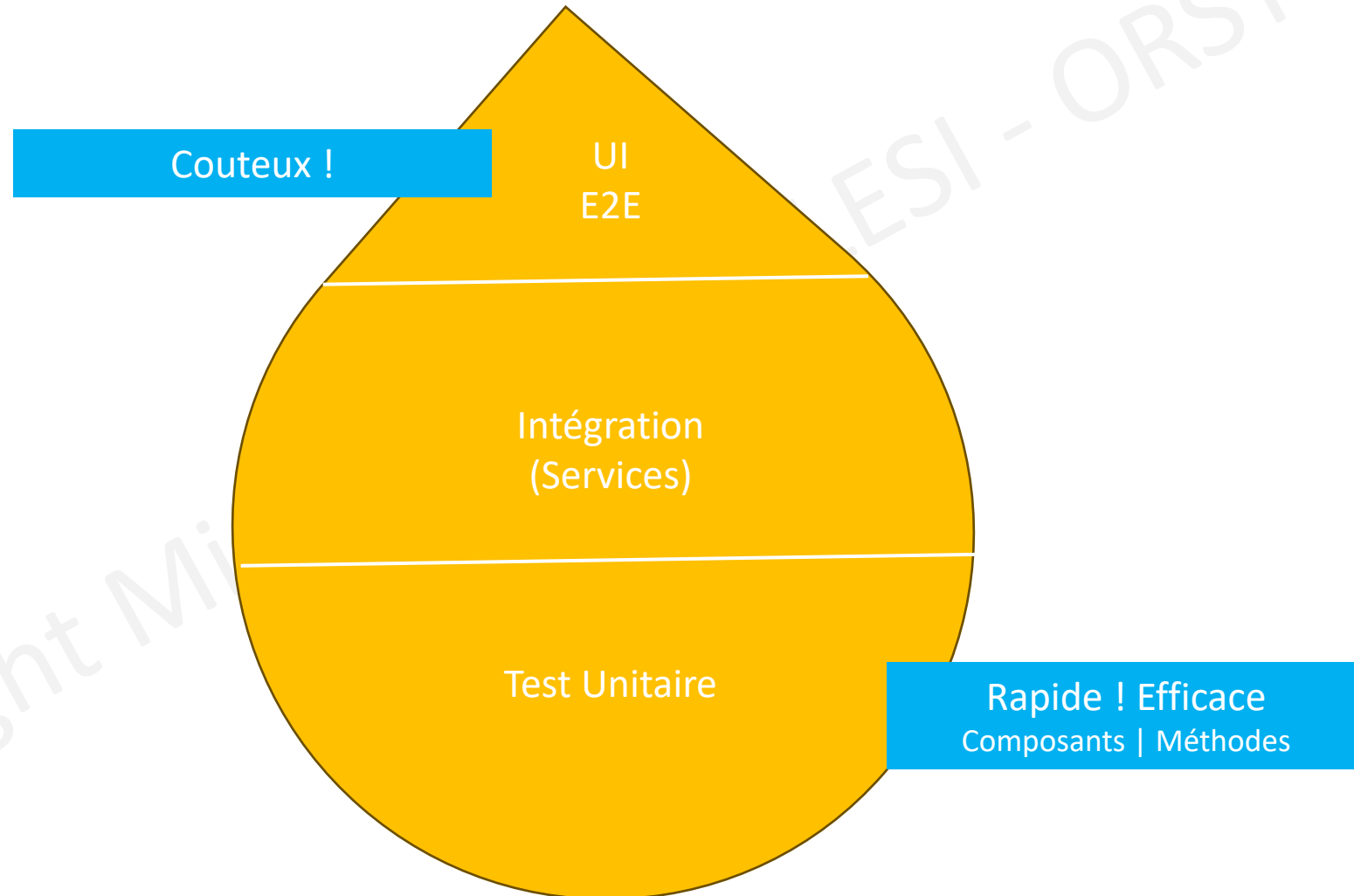
2-Les tests D'intégrations

Services :

3-Les tests End To End

Mise en Prod :

Le Test et la Documentation



1^{er} exemple : Apprendre de l'existant

app.component.ts

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

```
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-root',
5   templateUrl: './app.component.html',
6   styleUrls: ['./app.component.css']
7 })
8 export class AppComponent {
9   title = 'TU';
10 }
11
```

app.component.html

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

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

app.component.spec.ts

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

```
1 import { TestBed } from '@angular/core/testing';
2 import { AppComponent } from './app.component';
3
4 // -----
5 // describe permet de créer un Objet de Test Unitaire
6 // -----
7 describe('AppComponent', () => {
8   beforeEach(() => TestBed.configureTestingModule({
9     declarations: [AppComponent],
10   }));
11
12   // it est une fonction qui crée une spécification
13   it('should create the app', () => {
14     const fixture = TestBed.createComponent(AppComponent);
15     const app = fixture.componentInstance;
16     // une spéc (it) attend un retour (expectation)
17     expect(app).toBeTruthy(); // matchers Jasmine
18   });
19
20
21   it('should have as title 'TP07-tests-unitaires'', () => {
22     const fixture = TestBed.createComponent(AppComponent);
23     const composant = fixture.componentInstance;
24     expect(composant.title).toEqual('TU');
25   });
26
27
28   it('should render title', () => {
29     const fixture = TestBed.createComponent(AppComponent);
30     fixture.detectChanges();
31     const compiled = fixture.nativeElement as HTMLElement;
32     expect(compiled.querySelector('h1')?.textContent).toContain('TU');
33   });
34 });
35
```


2nd exemple : Injection de services

```
TP07-tests-unitaires > src > app > services > warn.service.ts > WarnService
1 import { Injectable } from '@angular/core';
2
3 @Injectable({
4   providedIn: 'root'
5 })
6 export class WarnService {
7   // -----
8   // méthodes
9   // -----
10  public warnMessage = (msg: string) => {
11    console.warn(`Le message est : ${msg}`);
12  }
13 }
```

```
TP07-tests-unitaires > src > app > services > operations.service.ts > OperationsService > ajouter
1 import { Injectable } from '@angular/core';
2 import { WarnService } from './warn.service';
3
4 @Injectable({
5   providedIn: 'root'
6 })
7 export class OperationsService {
8
9   constructor(
10     private _warn: WarnService
11   ) { }
12
13   // -----
14   // méthodes
15   // -----
16  public ajouter = (nb1: number, nb2: number) => {
17    this._warn.warnMessage('Ajout OK');
18    // this._warn.warnMessage('Ajout OK');
19    return nb1 + nb2;
20  }
21
22  public soustraire = (nb1: number, nb2: number) => {
23    this._warn.warnMessage('Soustraction OK');
24    return nb1 - nb2;
25  }
26 }
```

```
TP07-tests-unitaires > src > app > services > operations.service.spec.ts > describe('Test Injection de dépendances')
1 import { OperationsService } from './operations.service';
2 import { WarnService } from './warn.service';
3
4 // 1- Création de l'objet de Test
5 describe('Test Injection de dépendances', () => {
6
7   // 2- liste des spécifications
8
9   // it (xit - fit)
10  it('Ajouter 2 nombres OK ?', () => {
11
12    const operations = new OperationsService(new WarnService);
13    const resultat = operations.ajouter(10, 5);
14    // 3- expectation => ce que l'on attend
15    expect(resultat).withContext('--- Erreur Expectation ---').toBe(15);
16  });
17
18   // 4- duplication du premier IT
19
20  // it (xit - fit)
21  it('Soustraire 2 nombres OK ?', () => {
22
23    const operations = new OperationsService(new WarnService);
24    const resultat = operations.soustraire(10, 5);
25    // 3- expectation => ce que l'on attend
26    expect(resultat).withContext('--- Erreur Expectation ---').toBe(5);
27  });
28
29  });
30
31
32
33 })
```

2nd exemple : Injection de services

```
warn.service.ts ×
TP07-tests-unitaires > src > app > services > warn.service.ts > WarnService > logErreur
1 import { Injectable } from '@angular/core';
2
3 @Injectable({
4   providedIn: 'root'
5 })
6 export class WarnService {
7   // -----
8   // méthodes
9   // -----
10  public warnMessage = (msg: string) => {
11    console.warn(`Le message est : ${msg}`);
12  }
13  // ---
14  public logErreur = (msg: string) => {
15    console.error(`L'erreur est : ${msg}`);
16  }
17 }
```

```
operations.service.ts ×
src > app > services > operations.service.ts > OperationsService > ajouter
1 import { Injectable } from '@angular/core';
2 import { WarnService } from './warn.service';
3
4 @Injectable({
5   providedIn: 'root'
6 })
7 export class OperationsService {
8
9   constructor(
10     private _warn: WarnService
11   ) { }
12
13   // -----
14   // méthodes
15   // -----
16   public ajouter = (nb1: number, nb2: number) => {
17     this._warn.warnMessage('Ajout OK');
18     // this._warn.warnMessage('Ajout OK');
19     return nb1 + nb2;
20   }
21
22   public soustraire = (nb1: number, nb2: number) => {
```

```
operations.service.spec.ts ×
TP07-tests-unitaires > src > app > services > operations.service.spec.ts > ...
1 import { OperationsService } from './operations.service';
2 import { WarnService } from './warn.service';
3
4 // 1- Création de l'objet de Test
5 describe('Test Injection de dépendances', () => {
6
7   // 2- liste des spécifications
8
9   // it (xit - fit)
10  it('Ajouter 2 nombres OK ?', () => {
11
12    // 5- Mocker WarnService (Jasmine Spies)
13    const MockWarn = jasmine.createSpyObj('WarnService', ['warnMessage', 'logErreur']);
14    const operations = new OperationsService(MockWarn);
15    const resultat = operations.ajouter(10, 5);
16
17    // 3- expectation => ce que l'on attend
18    expect(resultat).withContext('--- Erreur Expectation ---').toBe(15);
19  });
20
21  // 4- duplication du premier IT
22
23  // it (xit - fit)
24  it('Soustraire 2 nombres OK ?', () => {
25
26    // 6- reproduire aussi ici le MockWarn
27    const MockWarn = jasmine.createSpyObj('WarnService', ['warnMessage', 'logErreur']);
28    const operations = new OperationsService(MockWarn);
29    const resultat = operations.soustraire(10, 5);
30
31    // 3- expectation => ce que l'on attend
32    expect(resultat).withContext('--- Erreur Expectation ---').toBe(5);
33  });
34
35
36
37 })
```

2nd exemple : Injection de services

The image shows a code editor with two files open: `warn.service.ts` and `operations.service.spec.ts`.

warn.service.ts

```
1 import { Injectable } from '@angular/core';
2
3
4 @Injectable({
5   providedIn: 'root'
6 })
7 export class WarnService {
8
9   constructor(
10     private _warn: WarnService
11   ) { }
12
13   // -----
14   // méthodes
15   // -----
16   public ajouter = (nb1: number, nb2: number) => {
17     this._warn.warnMessage('Ajout OK');
18     this._warn.warnMessage('Ajout OK');
19     return nb1 + nb2;
20   }
21 }
```

operations.service.spec.ts

```
1 import { OperationsService } from './operations.service';
2 import { WarnService } from './warn.service';
3
4 // 1- Création de l'objet de Test
5 describe('Test Injection de dépendances', () => {
6
7   // 2- liste des spécifications
8
9   // it (xit - fit)
10   it('Ajouter 2 nombres OK ?', () => {
11
12     // 5- Mocker WarnService (Jasmine Spies)
13     const MockWarn = jasmine.createSpyObj('WarnService', ['warnMessage', 'logErreur']);
14     const operations = new OperationsService(MockWarn);
15     const resultat = operations.ajouter(10, 5);
16
17     // 3- expectation => ce que l'on attend
18     expect(resultat).withContext('--- Erreur Expectation ---').toBe(15);
19     // 7- ajout d'une autre expectation
20     expect(MockWarn.warnMessage).toHaveBeenCalledTimes(1);
21   });
22
23   // 4- duplication du premier IT
24
25 }
```

Terminal Output

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

Red arrows indicate the flow of information: one arrow points from the `expect(MockWarn.warnMessage).toHaveBeenCalledTimes(1);` line in the spec file to the terminal error message, and another arrow points from the `this._warn.warnMessage('Ajout OK');` lines in the service file to the same error message, showing that the service is called twice instead of once.

2nd exemple : Injection de services

The image shows a code editor with three files open, illustrating service injection and testing in Angular.

warn.service.ts

```
1 import { Injectable } from '@angular/core';
2
3 @Injectable({
4   providedIn: 'root'
5 })
6 export class WarnService {
7   // -----
8   // méthodes
9   // -----
10  public warnMessage = (msg: string) => {
11    console.warn(`Le message est : ${msg}`);
12  }
13  // ---
14  public logErreur = (msg: string) => {
15    console.error(`L'erreur est : ${msg}`);
16  }
17 }
```

operations.service.spec.ts

```
1 import { OperationsService } from './operations.service';
2 import { WarnService } from './warn.service';
3
4 // Création de l'objet de Test
5 describe('Test Injection de dépendances', () => {
6
7   // -----
8   // Zone de déclarations de variables - const - objets
9   // -----
10  let MockWarn:any;
11  let operations:any;
12
13  // -----
14  // traitement spécifiques communs (beforeEach - AfterEach - beforeAll)
15  // -----
16  beforeEach(() => {
17    MockWarn = jasmine.createSpyObj('WarnService', ['warnMessage', 'logErreur']);
18    operations = new OperationsService(MockWarn);
19  });
20
21  // -----
22  // liste des spécifications
23  // -----
24
25  it('Ajouter 2 nombres OK ?', () => {
26    const resultat = operations.ajouter(10, 5);
27    expect(resultat).withContext('--- Erreur Expectation ---').toBe(15);
28    expect(MockWarn.warnMessage).toHaveBeenCalledTimes(1);
29  });
30
31  it('Soustraire 2 nombres OK ?', () => {
32    const resultat = operations.soustraire(10, 5);
33    expect(resultat).withContext('--- Erreur Expectation ---').toBe(5);
34  });
35 })
```

operations.service.ts

```
1 import { Injectable } from '@angular/core';
2 import { WarnService } from './warn.service';
3
4 @Injectable({
5   providedIn: 'root'
6 })
7 export class OperationsService {
8
9   constructor(
10     private _warn: WarnService
11   ) { }
12
13   // -----
14   // méthodes
15   // -----
16  public ajouter = (nb1: number, nb2: number) => {
17    this._warn.warnMessage('Ajout OK');
18    // this._warn.warnMessage('Ajout OK');
19    return nb1 + nb2;
20  }
21
22  public soustraire = (nb1: number, nb2: number) => {
23    this._warn.warnMessage('Soustraction OK');
24    return nb1 - nb2;
25  }
26 }
```

2nd exemple : Injection de services

```
warn.service.ts × ... operations.service.spec.ts ×
TP07-tests-unitaires > src > app > services > warn.service.ts > WarnService > warnMess
1 import { Injectable } from '@angular/core';
2
3 @Injectable({
4   providedIn: 'root'
5 })
6 export class WarnService {
7   // -----
8   // méthodes
9   // -----
10  public warnMessage = (msg: string) => {
11    console.warn(`Le message est : ${msg}`);
12  }
13  // ---
14  public logErreur = (msg: string) => {
15    console.error(`L'erreur' est : ${msg}`);
16  }
17 }

operations.service.ts × ...
TP07-tests-unitaires > src > app > services > operations.service.ts > OperationsService >
13 // -----
14 // méthodes
15 // -----
16 public ajouter = (nb1: number, nb2: number) => {
17   this._warn.warnMessage('Ajout OK');
18   // this._warn.warnMessage('Ajout OK');
19   return nb1 + nb2;
20 }
21
22 public soustraire = (nb1: number, nb2: number) => {
23   this._warn.warnMessage('Soustraction OK');
24   return nb1 - nb2;
25 }
26 }

operations.service.spec.ts ×
TP07-tests-unitaires > src > app > services > operations.service.spec.ts > describe('Test Injection de dépendances') callb
1 import { TestBed } from '@angular/core/testing';
2 import { OperationsService } from '../operations.service';
3 import { WarnService } from '../warn.service';
4
5 // Création de l'objet de Test
6 describe('Test Injection de dépendances', () => {
7   // -----
8   // Zone de déclarations de variables - const - objets
9   // -----
10  let MockWarn: any;
11  let operations: any;
12  // -----
13  // traitement spécifiques communs (beforeEach - AfterEach - beforeEach)
14  // -----
15  beforeEach(() => {
16    MockWarn = jasmine.createSpyObj('WarnService', ['warnMessage', 'logErreur']);
17    // -----
18    // Création d'un Objet complet de test et de simulation : TESTBED
19    // Créons une vraie injection de dépendances avec les providers
20    // -----
21    TestBed.configureTestingModule({
22      providers: [
23        // injection de dépendances
24        OperationsService,
25        {
26          provide: WarnService,
27          // on n'utilise pas la méthode de WarnService
28          // mais la méthode du Mock ('warnMessage', 'logErreur')
29          useValue: MockWarn
30        }
31      ]
32    });
33    // -----
34    operations = TestBed.inject(OperationsService);
35    // -----
36  });
37  // -----
38  // liste des spécifications
39  // -----
40  it('Ajouter 2 nombres OK ?', () => {
41    const resultat = operations.ajouter(10, 5);
42    expect(resultat).withContext('--- Erreur Expectation ---').toBe(15);
43    expect(MockWarn.warnMessage).toHaveBeenCalledTimes(1);
44  });
45 }
```

Les Formulaires



Les formulaires Angular (avec Matériel 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>
```

Material Components CDK Guides

Button

Autocomplete
Badge
Bottom Sheet
Button
Button toggle
Card
Checkbox
Chips
Core
Datepicker
Dialog
Divider

OVERVIEW API EXAMPLES

Angular Material buttons are native `<button>` or `<a>` elements enhanced with Material Design style.

Basic buttons

	Basic	Primary	Accent	Warn	Disabled	Link
Basic	Basic	Primary	Accent	Warn	Disabled	Link
Raised	Basic	Primary	Accent	Warn	Disabled	Link
Stroked	Basic	Primary	Accent	Warn	Disabled	Link
Flat	Basic	Primary	Accent	Warn	Disabled	Link
Icon	:	Home	Menu	Heart	Link	

Les formulaires Angular (avec Matériel 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 Matériel Design)

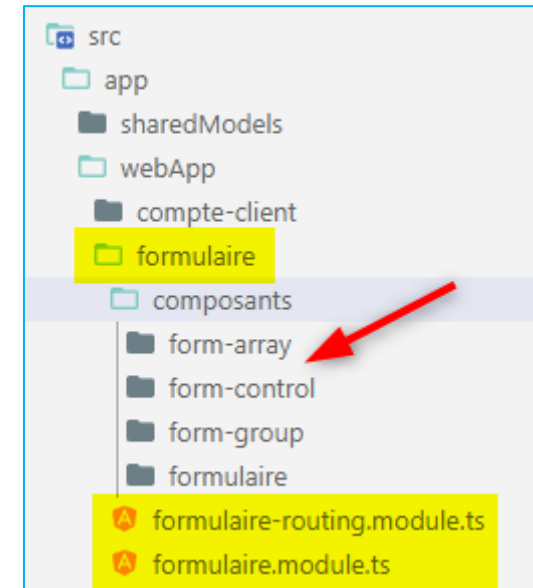
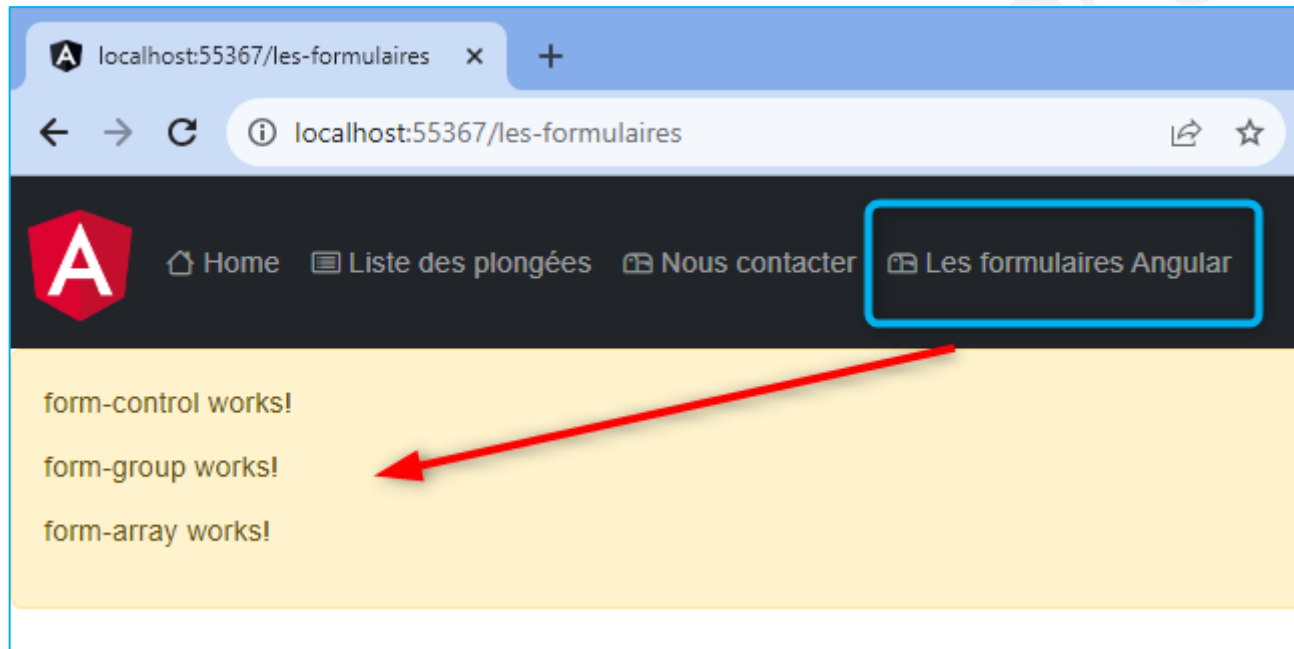
<https://dev.webjs.fr/1-Angular/ressources-formation/material.module.ts>

<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 Matériel Design)

The screenshot shows a VS Code editor with three files open:

- form.component.html**: Contains an alert message and a form with fields for 'nom', 'prenom', 'email', and 'password', along with a 'valider' button.
- header.component.html**: Contains a navigation bar with links to 'Home', 'Liste des plongées', 'Contactez-nous', and 'Les formulaires Angular'.
- app-routing.module.ts**: Contains the routing configuration, including the route for 'les-formulaires' which points to the 'FormulairesComponent'.

Red arrows highlight the 'FormulairesComponent' in the routing configuration and the 'les-formulaires' link in the header.

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 x
TP06-formulaires > src > app > webApp > formulaires > composants > form-control > form-control.c
1 import { Component } from '@angular/core';
2 import { FormControl, Validators } from '@angular/forms';
3
4 @Component({
5   selector: 'app-form-control',
6   templateUrl: './form-control.component.html',
7   styleUrls: ['./form-control.component.scss']
8 })
9 export class FormControlComponent {
10
11   // props
12   public prenom: FormControl<string | null>;
13
14   // const
15   constructor() {
16     this.prenom = new FormControl(
17       //val par défaut
18       '',
19       // Validators
20       [
21         Validators.required,
22         Validators.minLength(3),
23         Validators.maxLength(10)
24       ]
25     );
26   }
27
28 }
29

form-control.component.html x
TP06-formulaires > src > app > webApp > formulaires > composants > form-control > form-control.component.html >
1 <h5>La class formControl</h5>
2
3   <mat-icon>perm_identity</mat-icon>
4
5   <mat-form-field>
6     <mat-label>Votre prénom :</mat-label>
7     <input type="text" matInput [formControl]="prenom">
8   </mat-form-field>
9
10 <!-- tout ce qui suit peut être utilisé autant dans les "templates driven forms" d
11
12 <div>
13   <p>- Touched : {{prenom.touched}}</p>
14   <p>- Pristine : {{prenom.pristine}}</p>
15   <p>- Dirty : {{prenom.dirty}}</p>
16   <p>- Valid : {{prenom.valid}}</p>
17 </div>
18
19
20 <p [hidden]="!prenom.touched && !prenom.valid" style="color: lightgreen">
21   Le champ prénom est validé !
22 </p>
23
24 <p [hidden]="prenom.touched || prenom.valid" style="color: red">
25   Le champ prénom est requis !
26 </p>
27
28 <div *ngIf="prenom.touched && prenom.valid; then blockOk else blockNotOk"></div>
29
30   <ng-template #blockOk> Le champ est validé !</ng-template>
31   <ng-template #blockNotOk> Le champ est requis !</ng-template>
32
33 <hr>
34
```

La Class form-control

Home Liste des plongées Nous contacter **Les Formulaires**

Les Forms groups

Votre prénom :*

Zoé

- Prénom (value) : Zoé
- Prénom (Touched) : true
- Prénom (Pristine) : false
- Prénom (Dirty) : true
- Prénom (Valid) : true

Votre nom :

Votre email :*

z.angular@tech.lead

Saisir votre rue :

Saisir votre ville :*

New York

Saisir votre cp :

Allez à la suite ...

Une autre partie du formulaire à compléter ...

submit ...

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>&nbsp;  
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>&nbsp;  
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>&nbsp;  
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>&nbsp;  
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 form-group

La Class formGroup

```
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: ['']  
      })  
    });  
  }  
}
```

```
// 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  ...  rxjs.component.ts  form-group.component.ts  x
P06-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>&nbsp;
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>&nbsp;
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>&nbsp;
86    <mat-form-field appearance="fill">
87      <mat-label>Choisir une date</mat-label>
88      <input matInput [matDatepicker]="date" formControlName="dateDebut
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
TP06-formulaires > src > app > webApp > formulaires > composants > form-group > form-group.component.ts
48
49  ]
50  ),
51  // *****
52  adresse: new FormGroup({
53    rue: new FormControl<string | null>(null),
54    ville: new FormControl<string | null>(null, Validators.required),
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.required),
63    dateDebut: new FormControl<string | null>(null),
64  })
65  })
66  }
67
68  // cycle de vies
69  ngAfterViewInit() {
formulaires.module.ts  x
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/formulaires.component';
4  import { FormControlComponent } from './composants/form-control/form-control.component';
5  import { FormGroupComponent } from './composants/form-group/form-group.component';
6  import { FormArrayComponent } from './composants/form-array/form-array.component';
7
8  // ---- Matériel 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 // ---- Matériel 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>tr
54 <mat-form-fie
55 <mat-labe
56 <input ma
57 </mat-form-fi
58
59 <p></p>
60 </div>
61
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: 'gfggh', email: 'mbo@free.fr', adresse: {...}}
- ▼ adresse:
 - cp: "3"
 - rue: "2"
 - ville: "2"
 - ▶ [[Prototype]]: Object
 - email: "mbo@free.fr"
 - nom: "gfggh"
 - 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

----- 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 x
src > app > webApp > formulaires > composants > form-group > form-group.component.ts > FormGroupComponent > constructor
13 public monForm: FormGroup,
14
15 // constructeur
16 constructor(
17
18     private _post: PostService
19 ) {
20
21     this.monForm = new FormGroup({
22         // déclaration de tous les controls de ce formulaire (formGroup)

```

```

form-group.component.ts x
src > app > webApp > formulaires > composants > form-group > form-group.component.ts > FormGroupComponent > onSubmit
98
99 }
100
101 // Méthodes
102 public onSubmit = () => {
103     console.log('Groupe principal : ', this.monForm.value);
104     console.log('Sous Groupe : ', this.monForm.controls['adresse'].value);
105     console.log('Sous Groupe : ', this.monForm.controls['tp'].value);
106
107     this._post.postForm(this.monForm);
108     // TP
109 }
110
111

```

```

bdd.json x
TP06-formulaires > src > assets > json > bdd.json > ...
76     "photo": "https://dev.webjs.fr/images/fond5.jpg"
77 }
78 },
79 "messages": [
80 {
81     "nameNom": "Bruce Wayne",
82     "nameEmail": "bruce@gothamcity.com",
83     "nameMessage": "Alfred je reviens ...",
84     "id": 1
85 }

```

```

post.service.ts x
TP06-formulaires > src > app > sharedModels > services > post.service.ts > PostService > post
1 import { HttpClient, HttpHeaders } from '@angular/common/http';
2 import { Injectable } from '@angular/core';
3 import { FormGroup } from '@angular/forms';
4
5 @Injectable({
6     providedIn: 'root'
7 })
8
9
10 export class PostService {
11
12     constructor(private _http: HttpClient) {
13
14         // méthodes
15         public postForm = (form: FormGroup) => {
16             console.log('----- OBJET JS : ', form.value);
17
18             // --- Post sur un serveur json
19             // 1-url
20             const url: string = 'http://localhost:3001/messages';
21             // 2-body
22             const body = JSON.stringify(form.value);
23             console.warn('----- OBJET STRINGIFY : ', body);
24
25             // 3-headers
26             const myHeaders = new HttpHeaders({
27                 'Content-Type': 'application/json',
28                 'Access-Control-Allow-Origin': '*' // CORS
29             });
30             const options = { headers: myHeaders };
31             //-----
32
33             // envoi avec POST
34             this._http.post(url, body, options).subscribe(
35                 (res: any) => console.log(res);
36             );
37
38         }

```

La Class formArray

Récupérer les ressources en ligne :

<https://dev.webjs.fr/1-Angular/ressources-formation/reactive-forms.zip>

```
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);
    }
  }
}
```

```
<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>&nbsp;  
    <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>&nbsp;  

        <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'exécution !

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'exécuter 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 back 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!

[🏠 Home Page](#)[📄 Introduction ▾](#)[👤 Compte Client](#)[🔗 Angular datas ▾](#)[📁 Rxjs](#)[✉ Formulaires ▾](#)[🔔 Le Signal](#)[📰 Les News NG17+](#)[← Go Back](#)

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

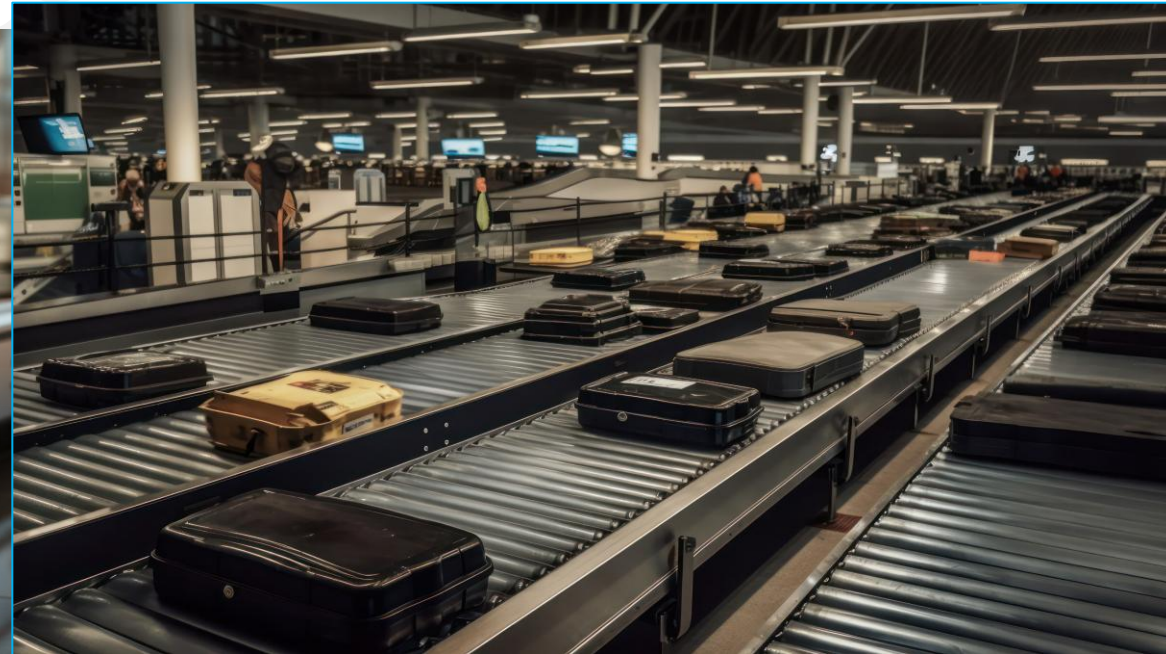
Niveau 2



Copyright Michel FACCIOLESI - ORSYS

RXJS et les OBSERVABLES

- 1 Observable est un objet qui émet une suite (une séquence) de valeurs (datas) dans le temps.
Exemple : une requête HTTP
- On pourrait comparer cette suite de valeurs émises aux valises et bagages qui défilent sur un tapis roulant
- Chaque bagage a une destination mais va peut être croiser d'autres bagages qui n'ont pas la même destination mais empruntent le même tapis
- Ces valeurs peuvent être reçues par un poste de tri, interceptées, regroupées en fonction de critères et réaiguillées dynamiquement.
- On peut également stopper le tapis roulant
- Cela offre beaucoup de souplesse dans le traitement des données et la gestion des requêtes asynchrones.



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)

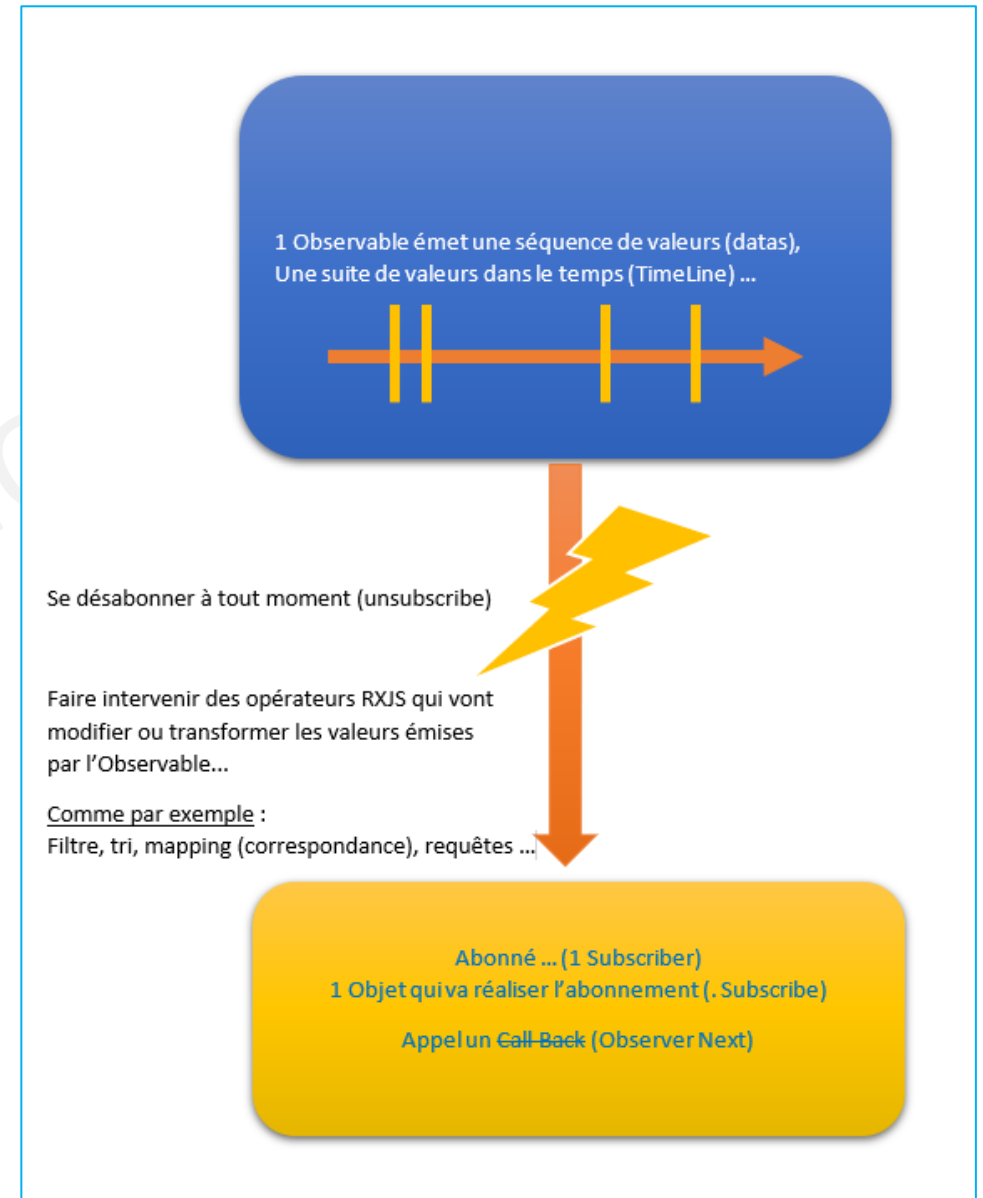
Opérateurs : des fonctions tap, map, merge, filter qui vont pouvoir modifier/transformer ce flux de valeurs émises par l'observable

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 !

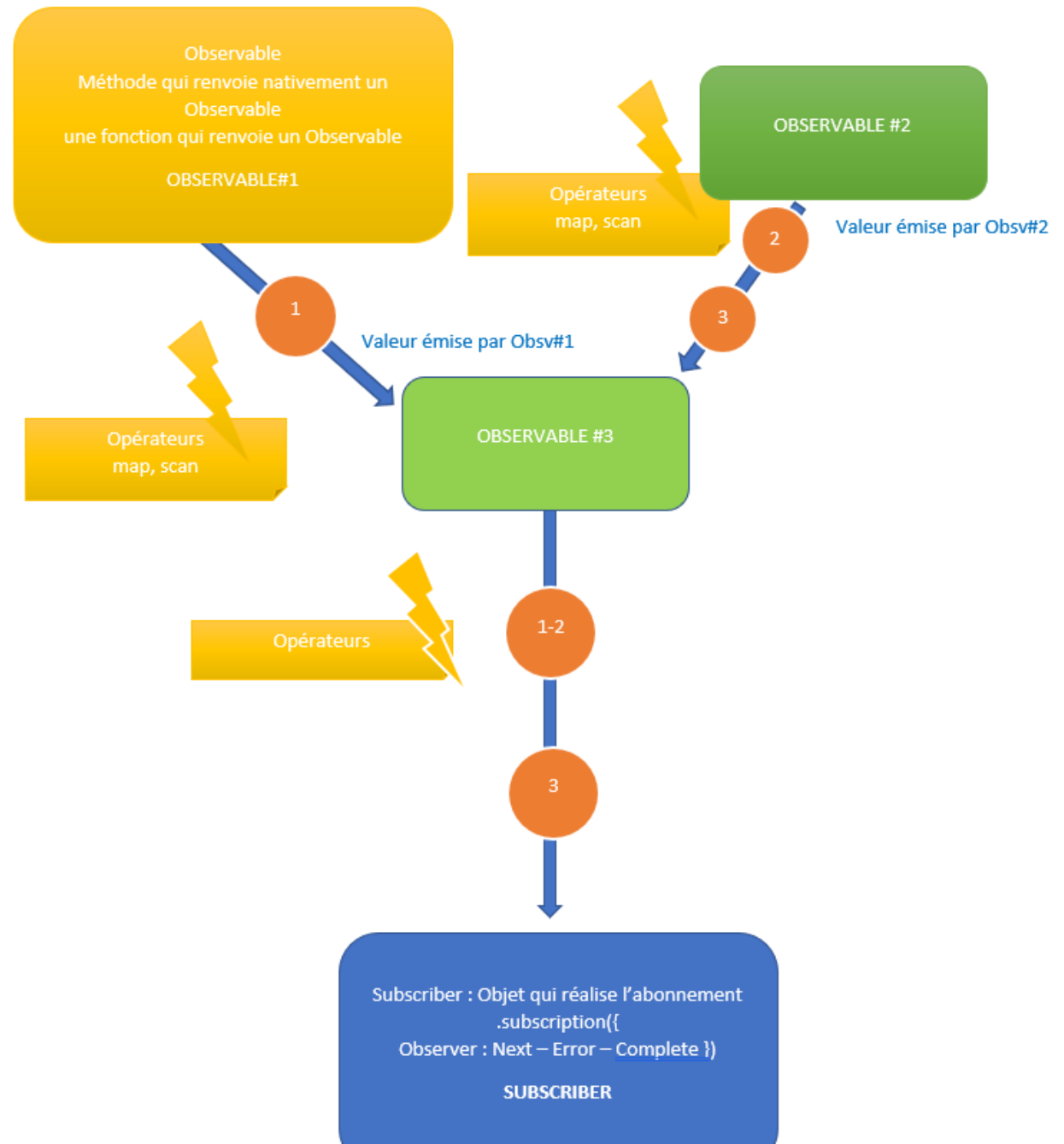
Les séquences de valeurs émises sont facilement annulables !

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

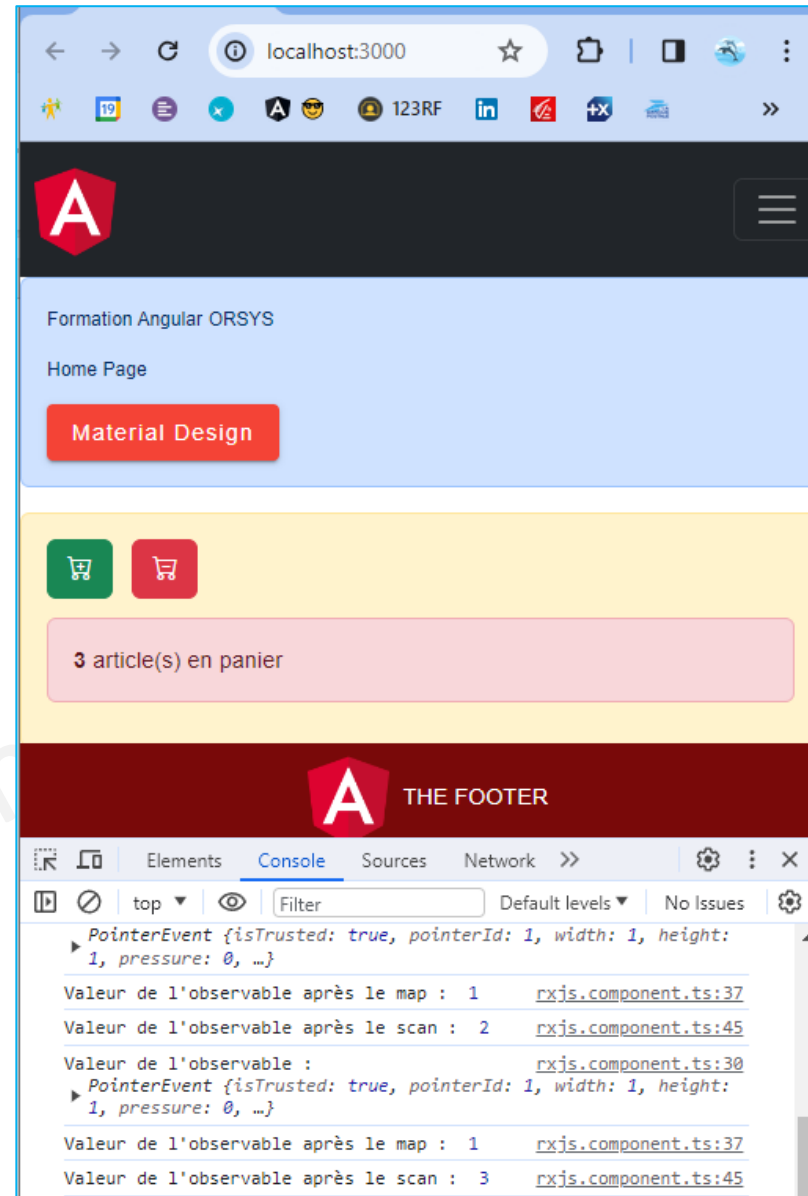


RXJS et les OBSERVABLES

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



RXJS et les OBSERVABLES



The screenshot shows a web application running on `localhost:3000`. The page has a dark header with an Angular logo and a menu icon. Below the header, there's a light blue section titled "Formation Angular ORSYS" with a "Home Page" link and a red "Material Design" button. The main content area has a yellow background and displays two shopping cart icons (green and red) and a pink box stating "3 article(s) en panier". The footer is dark red with the Angular logo and the text "THE FOOTER".

The browser's developer console is open, showing the following log messages:

- `MouseEvent {isTrusted: true, pointerId: 1, width: 1, height: 1, pressure: 0, ...}`
- Valeur de l'observable après le map : 1 `rxjs.component.ts:37`
- Valeur de l'observable après le scan : 2 `rxjs.component.ts:45`
- Valeur de l'observable : `rxjs.component.ts:30`
 - `MouseEvent {isTrusted: true, pointerId: 1, width: 1, height: 1, pressure: 0, ...}`
- Valeur de l'observable après le map : 1 `rxjs.component.ts:37`
- Valeur de l'observable après le scan : 3 `rxjs.component.ts:45`

RXJS et les OBSERVABLES

The screenshot displays an Angular development environment with three main components: a code editor on the left, a central workspace with two code files, and a browser preview on the right.

Left Panel (body.component.html):

```
1 <div class="alert alert-primary">
2
3 <h1>Formation Angular ORSYS</h1>
4 <h2>Home Page</h2>
5 <button class="btn btn-raised" color="warn">Material Design</button>
6
7
8 <div class="alert alert-warning">
9
10 </div>
11
12
```

Central Panel (rxjs.component.ts):

```
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-rxjs',
5   templateUrl: './rxjs.component.html',
6   styleUrls: ['./rxjs.component.scss']
7 })
8 export class RxjsComponent {
9
10 }
11
```

Bottom Panel (rxjs.component.html):

```
1 <button class="btn btn-success" #plus><i class="fa fa-plus"></i>
2 </button>
3 <button class="btn btn-danger" #moins><i class="fa fa-minus"></i>
4 </button>
5 <p></p>
6
7 <p class="alert alert-danger" #panier></p>
8 <p></p>
9
```

Right Panel (Browser Preview):

The browser shows the application running on `localhost:3000`. The page features a dark header with a red 'A' logo and a hamburger menu. Below the header, the text "Formation Angular ORSYS" and "Home Page" is displayed. A red button labeled "Material Design" is visible. At the bottom, there is a yellow bar containing two shopping cart icons (one green, one red) and a pink rectangular placeholder.


```

// -----
// 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`
// );
```

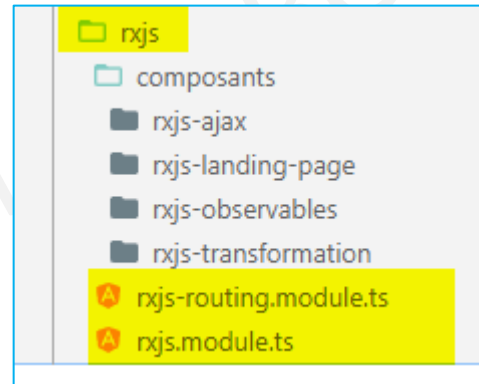
```
// -----
// Re-Composition des 2 observables : Combinaison - Fusion(merge)
// -----

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.



```

<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>

```

```

// 4- Méthodes
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')
        }
    );
}

```

```
// -----
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'enchaîner les opérateurs RXJS
      tap(
        // observer NEXT
        (formation: Formation) => console.table(formation)
      ),
      // first(),
      // last(),
      // first(
      //   // prédicat === pattern
      //   (formation: Formation) => {
      //     return formation.duree === 2;
      //   }
      // ),
      delay(3000),

```

```
128
129
130   filter(
131     (formation: Formation) => {
132       return formation.duree === 4;
133     }
134   ),
135   tap(
136     // observer NEXT
137     (formation: Formation) => {
138       console.warn(formation);
139       const eltLi = <HTMLElement>document.createElement('li');
140       eltLi.innerHTML = `${formation.cours} : ${formation.duree} jours.`;
141       eltLi.className = 'list-group-item';
142       // nativeElement est obligatoire ici car on récupère l'élément par Vie
143       this.eltUl2.nativeElement.appendChild(eltLi);
144     }
145   ),
146   catchError(
147     // capturer les erreurs de l'observable
148     (e: any): any => {
149       console.log(e);
150     }
151   )
152 )
153 .subscribe(
154   // // observer NEXT
155   // (formation: Formation) => {
156   //   console.log(formation);
157   //   const eltLi = <HTMLElement>document.createElement('li');
158   //   eltLi.innerHTML = `${formation.cours} : ${formation.duree} jours.`;
159   //   eltLi.className = 'list-group-item';
160   //   // nativeElement est obligatoire ici car on récupère l'élément par View
161   //   this.eltUl2.nativeElement.appendChild(eltLi);
162   // }
163 )
164 .unsubscribe();

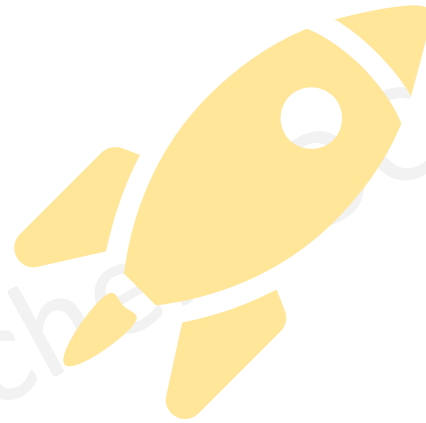
```

ANNEXES



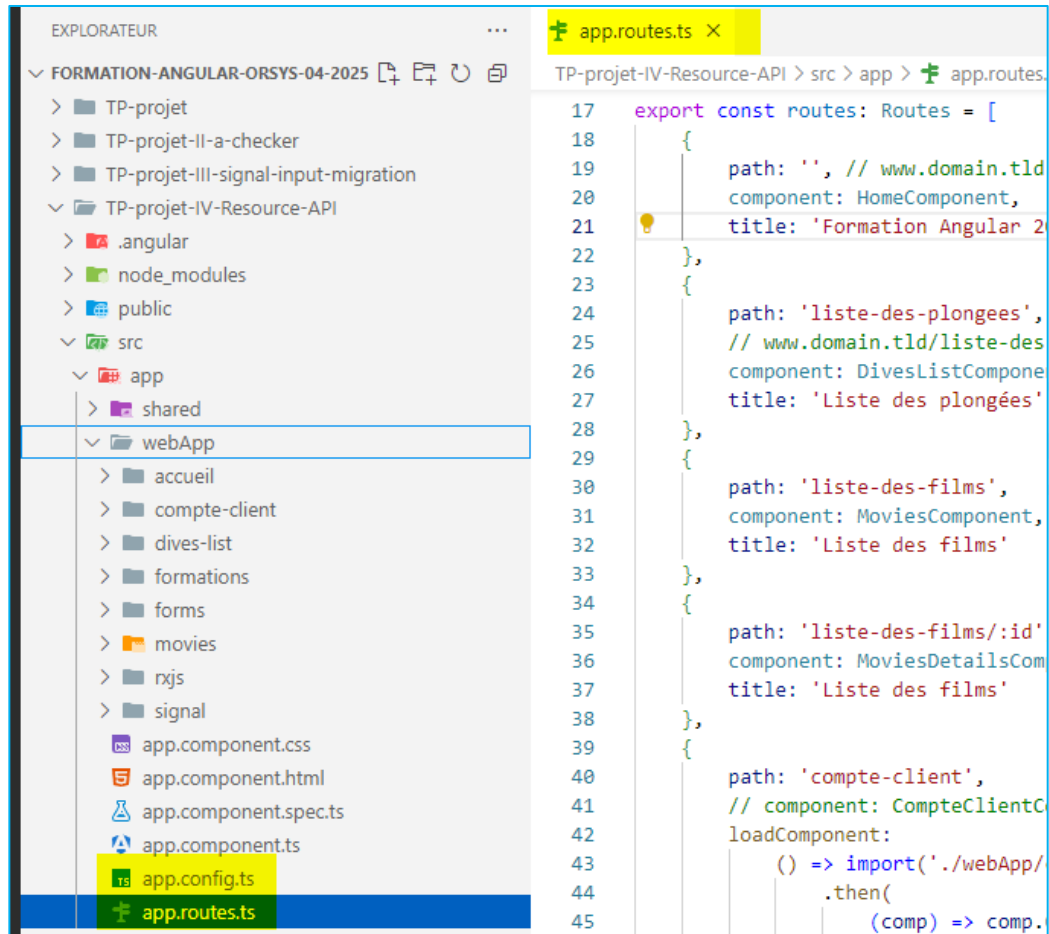
Copyright Michel BOUQUET - MOLES1 - ORSYS

Les versions marquantes 15-20



Copyright Michaël ACCIOLESI - ORSYS

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 **bootstrap** 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** ?



Copyright Michel BOCCIOLESI - ORSYS



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 ?

HTTP Queries – Angular 19+

class httpClient Angular 2+

- `private _http = inject(httpClient)`
- `this._http.get ...`

?

Angular 19

- `resource()`
- `rxResource()`
- `httpResource()`

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()`

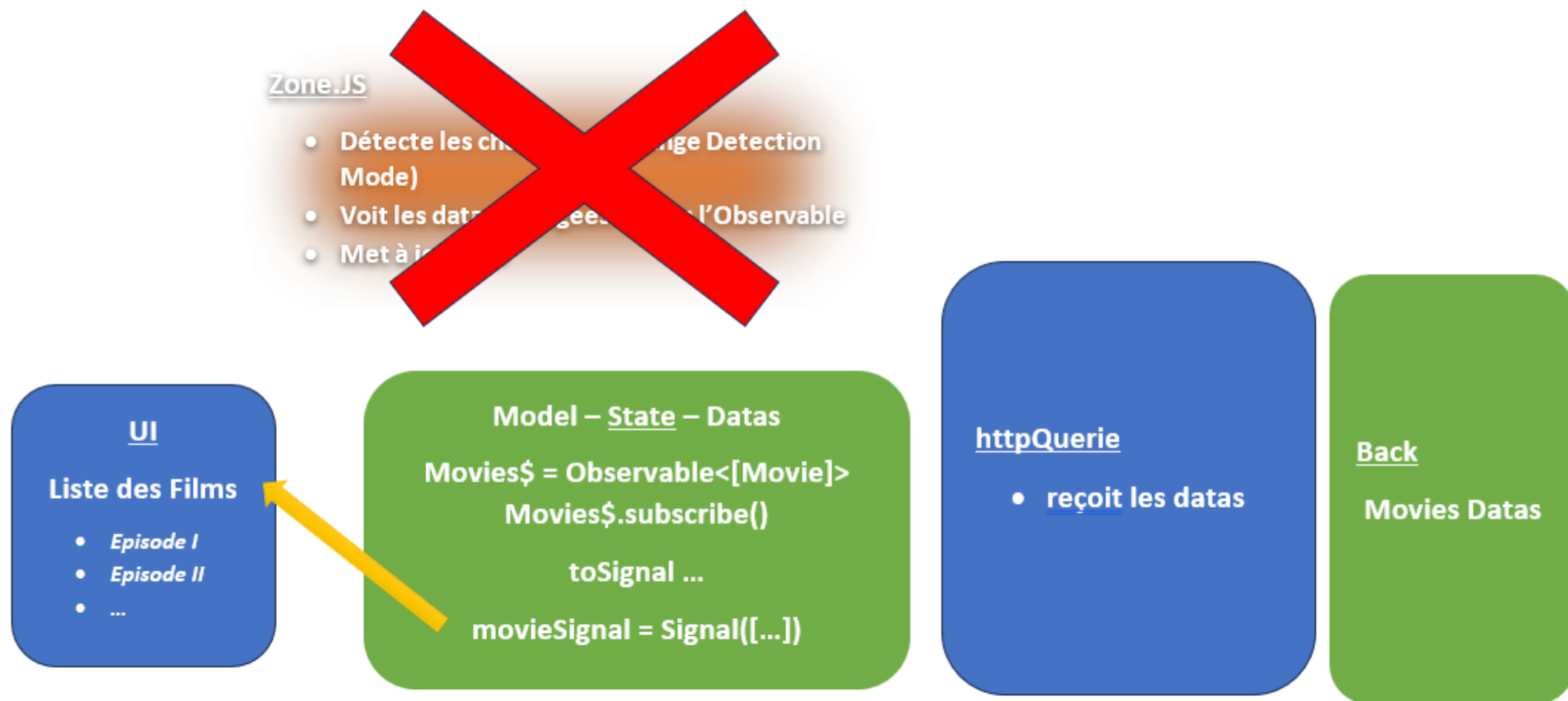
httpClient

- reçoit les datas
- émet les datas à l'Observable créé

Back

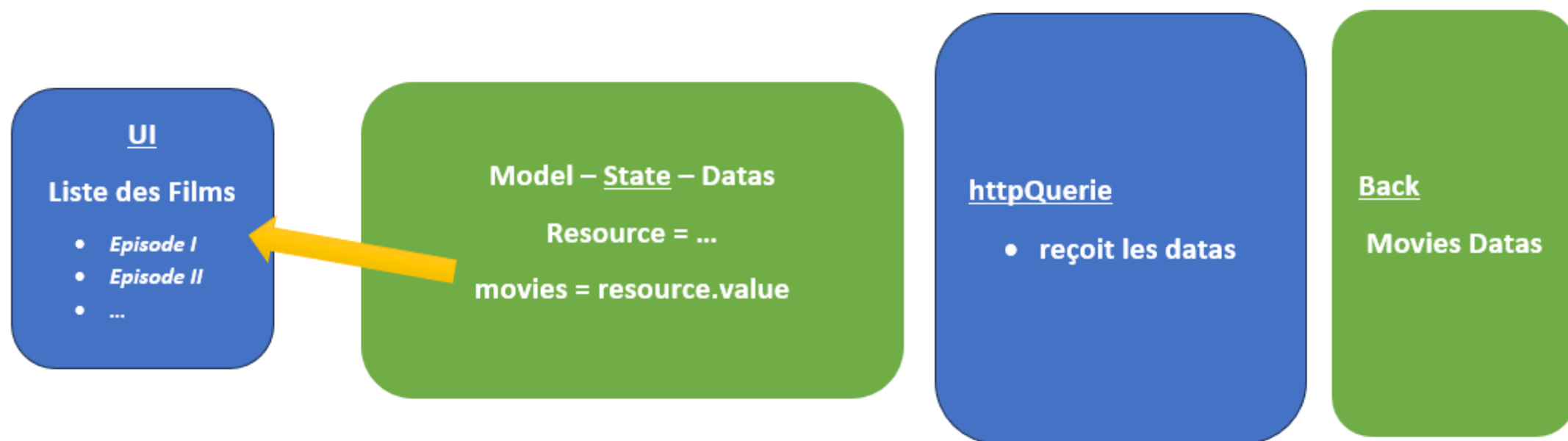
Movies Datas

SIGNAL & RESOURCE API



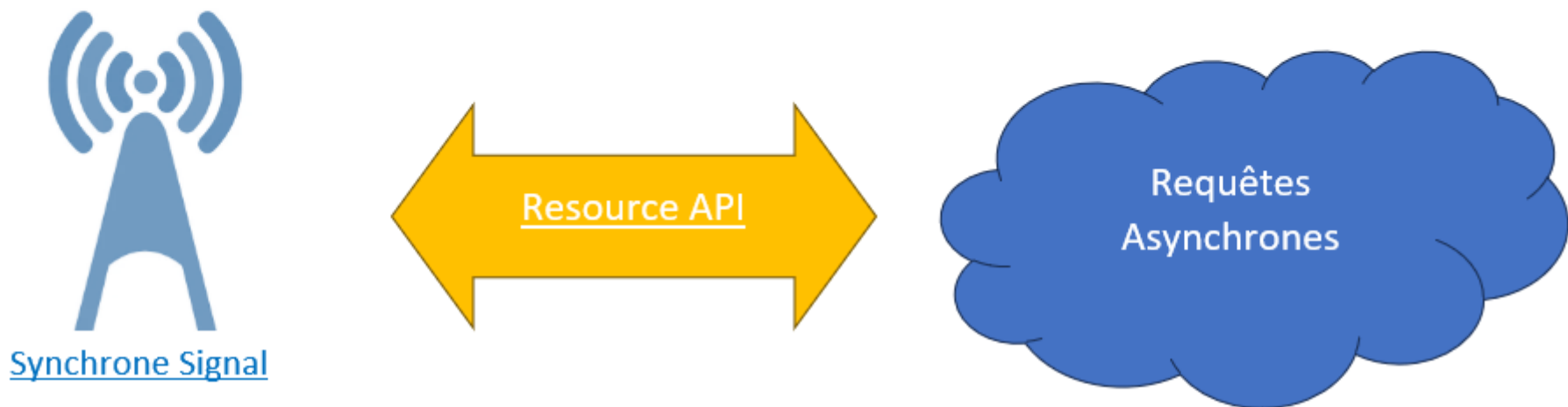
Le [signal \(SYNCHRONE\)](#) n'utilise plus Zone.JS et peut mettre à jour directement l'UI. Mais Angular 19+ a introduit une solution encore plus pratique ([Resource API](#)), qui est une sorte de pont entre la requête asynchrone http et le signal qui est synchrone ! Plus besoin de Subscription, les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

SIGNAL & RESOURCE API



Le [signal \(SYNCHRON\)](#) 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

[Home Page](#) [Compte Client](#) [Liste des plongées](#) [Liste des films Star Wars](#) [Les Observables](#) [Le Signal](#) [Resource](#) [Contacts](#) [Go Back](#)

Liste des Formations

Développement Front

Formation	Durée	Catégorie
Angular - les fondamentaux	4	Développement Front
Angular - concepts avancés	3	Développement Front
Javascript	4	Développement Front
React	3	Développement Front
Vue.JS	5	Développement Front
HTML & JS	5	Développement Front

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

```
1 import { computed, inject, Injectable, resource, signal } from '@angular/core';
2 import { HttpClient, HttpResponse } from '@angular/common/http';
3 import { FORMATIONS_API_URL } from '@shared/env/globals';
4 import { Formation } from '@shared/interfaces/Formation';
5 import { setErrorMessage } from '@shared/errors/error-http-msg';
6
7
8 @Injectable({ providedIn: 'root' })
9
10 export class FormationsService {
11
12     private _url = FORMATIONS_API_URL;
13     public formationModels = signal<string[]>([
14         'Programmation', 'Développement Front', 'Développement Back', 'Frontend', 'Backend'
15     ]);
16     public formationSelected = signal<string>('');
17     // -----
18
19     // 1- resource() avec paramètres
20
21     private formationsResource = resource({
22         request: this.formationSelected,
23         loader: (param) => fetch(`${this._url}?category=${param.request}`)
24             .then(
25                 (res: any) => {
26                     console.log(res);
27                     return res.json() as Promise<Formation[]>
28                 }
29             )
30     });
31
32
33     public formations = this.formationsResource.value;
34
35     public error = computed(
36         () => this.formationsResource.error() as HttpResponse
37     );
38
39     public errorMessage = computed(
40         () => setErrorMessage(this.error(), 'Formations');
41     );
42
43     public isLoading = this.formationsResource.isLoading;
44
45 }
```

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

```
1 import { AfterViewChecked, Component, inject } from '@angular/core';
2 import { FormsModule } from '@angular/forms';
3 import { FormationsService } from '@app/shared/services/formations.service';
4
5 @Component({
6     selector: 'app-formations',
7     imports: [FormsModule],
8     templateUrl: './formations.component.html',
9     styleUrls: ['./formations.component.css']
10 })
11 export class FormationsComponent {
12
13     private _FormationsService = inject(FormationsService);
14
15     // Signals envoyés au Template HTML
16     public formationModels = this._FormationsService.formationModels;
17     public formationsSelected = this._FormationsService.formationSelected;
18
19     public formations = this._FormationsService.formations;
20     public errorMessage = this._FormationsService.error;
21     public isLoading = this._FormationsService.isLoading;
22
23
24 }
```

Le template HTML

formations.component.html ×

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

Go to component

```
1 <div class="container alert alert-warning">
2   <h3 class="text-primary">Liste des Formations</h3>
3
4   @if (isLoading()) {
5     <div>... loading formations</div>
6   }
7   @else if (errorMessage()){
8     <div style='color: red'>An error occurred: {{ errorMessage() }}</div>
9   }
10  @else {
11    <div>
12      <select [(ngModel)]="formationsSelected" class="form-control">
13
14        <option value="" selected>Choisir une formation</option>
15
16        @for(formation of formationModels(); track formation) {
17          <option>{{ formation }}</option>
18        }
19      </select>
20    </div>
21
22    <hr>
23
24  </div>
```

```

1  <div class="container alert alert-warning">
22
23    <hr>
24
25    @if (formations()) {
26      <table class="table">
27        <thead>
28          <tr>
29            <th scope="col">Formation</th>
30            <th scope="col">Durée</th>
31            <th scope="col">Catégorie</th>
32          </tr>
33        </thead>
34
35        <tbody>
36          @for (f of formations(); track f) {
37            <tr>
38              <td> {{ f.cours }} </td>
39              <td> {{ f.duration }} </td>
40              <td> {{ f.category }} </td>
41            </tr>
42          }
43        </tbody>
44      </table>
45    }
46    @else {
47      <div>Pas de formations</div>
48    }
49
50  }
51 </div>

```

Avec RxResource (Observables et opérateurs RXJS)

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

9
10 @Injectable({ providedIn: 'root' })
11
12 export class FormationsService {
13
14   private _url = FORMATIONS_API_URL;
15   public formationModels = signal<string[]>([
16     'Programmation', 'Développement Front', 'Développement Back', 'Front CSS
17   ]);
18   public formationSelected = signal<string>('');
19   // -----
20
21   // 1- resource() avec paramètres :
22   // private formationsResource = resource({
23   //   request: this.formationSelected,
24   //   loader: (param) => fetch(`${this._url}?category=${param.request}`)
25   //   .then(
26   //     (res: any) => {
27   //       console.log(res);
28   //       return res.json() as Promise<Formation[]>;
29   //     }
30   //   )
31   // });
32
33   // 2- rxResource() avec paramètres
34   private _http = inject(HttpClient)
35   private formationsResource = rxResource({
36     request: this.formationSelected,
37     loader: (param) => this._http.get<Formation>(`${this._url}?category=${pa
38       .pipe(
39         tap(
40           (valObs) => console.log(valObs)
41         )
42       )
43   });
44
45   // -----
46   public formations = this.formationsResource.value
47
48   public error = computed(
49     () => this.formationsResource.error() as HttpErrorResponse
50   )
51   public errorMessage = computed(
```



Liste des Formations

Formation	Durée	Catégorie
C#	4	Programmation
C	5	Programmation
C++	5	Programmation

THE FOOTER - Formation Angular

ElementsConsoleSourcesNetworkPerformanceMemory>>

topFilterCustom levels2 Issues2 hidden

Angular is running in development mode.

core.mjs:20750

formations.service.ts:40

(15) [...]

(3) [...]

0: {id: '13', cours: 'C#', codeCours: 'C001', duration: 4, category: 'Programmation'}

1: {id: '14', cours: 'C', codeCours: 'C002', duration: 5, category: 'Programmation'}

2: {id: '15', cours: 'C++', codeCours: 'C003', duration: 5, category: 'Programmation'}

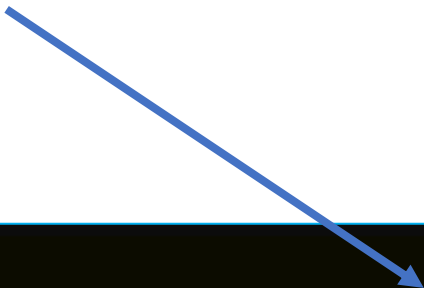
Avec httpResource (Angular 19.2)

```
formations.service.ts  package-lock.json
src > app > shared > services > formations.service.ts > FormationsService > formationsResource
12 export class FormationsService {
13     // 1- httpResource() avec paramètres
14     // private _http = inject(HttpClient)
15     // private formationsResource = rxResource({
16     //     request: this.formationSelected,
17     //     loader: (param) => this._http.get<Formation>(`${this._url}?category=${param}`),
18     //     .pipe(
19     //         tap(
20     //             (valObs) => console.log(valObs)
21     //         )
22     //     )
23     // });
24
25     // 3- httpResource() avec paramètres (Juste avec l'URL 😊)
26     // private formationsResource = httpResource<Formation>({
27     //     () => `${this._url}?category=${this.formationSelected()}`
28     // });
29
30     // 4- httpResource() avec paramètres dans un OBJET
31     private formationsResource = httpResource<Formation>({
32         url: this._url,
33         method: 'get',
34         headers: { accept: 'application/json' },
35         params: { category: this.formationSelected(), }
36     });
37
38     // -----
39     public formations = this.formationsResource.value
40
41     public error = computed(
42         () => this.formationsResource.error() as HttpErrorResponse
43     )
44     public errorMessage = computed(
45         () => setErrorMessage(this.error(), 'Formations'));
46
47     public isLoading = this.formationsResource.isLoading;
48 }
49
50
```

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 X
TP19 > src > app > app.config.ts > appConfig > providers
1 import { ApplicationConfig, provideZoneChangeDetection } from '@angular/core';
2 import { provideRouter } from '@angular/router';
3
4 import { routes } from './app.routes';
5
6 export const appConfig: ApplicationConfig = {
7   providers: [
8     provideZoneChangeDetection({ eventCoalescing: true }),
9     provideRouter(routes)
10  ];
11 };
12
```

```
app.component.ts M X
TP19 > src > app > app.component.ts > ...
1 import { NgFor } from '@angular/common';
2 import { afterRender, Component, signal } from '@angular/core';
3
4
5 @Component({
6   selector: 'app-root',
7   imports: [NgFor],
8   templateUrl: './app.component.html',
9   styleUrls: ['./app.component.css']
10 })
11 export class AppComponent {
12   // 1-props
13   public title: string = 'Les Images';
14   public imagesArray: string[] = ['avatar1.jpg', 'avatar2.jpg', 'avatar3.jpg'];
15   // Zoneless
16   public titleSignal = signal<string>('Vive le Signal');
17
18
19   constructor() {
20     afterRender(
21       () => {
22         // news NG20
23         console.log('AfterRender =>', this.title);
24       }
25     );
26   }
27 }
```

```
app.config.ts M X
TP20 > src > app > app.config.ts > ...
1 import { ApplicationConfig, provideBrowserGlobalErrorListeners } from '@angular/core';
2 import { provideRouter } from '@angular/router';
3
4 import { routes } from './app.routes';
5
6 export const appConfig: ApplicationConfig = {
7   providers: [
8     provideBrowserGlobalErrorListeners(),
9     // provideZoneChangeDetection({ eventCoalescing: true }),
10    provideZonelessChangeDetection(), // NG20
11    provideRouter(routes)
12  ];
13 };
14
```

```
app.ts M X
TP20 > src > app > app.ts > App > constructor > afterEveryRender() callback
1 import { NgFor } from '@angular/common';
2 import { afterEveryRender, Component, signal } from '@angular/core';
3
4
5 @Component({
6   selector: 'app-root',
7   imports: [NgFor],
8   templateUrl: './app.html',
9   styleUrls: ['./app.css']
10 })
11 export class App {
12   // 1-props
13   public title: string = 'Les Images';
14   public imagesArray: string[] = ['avatar1.jpg', 'avatar2.jpg', 'avatar3.jpg'];
15   // Zoneless
16   public titleSignal = signal<string>('Vive le Signal');
17
18
19   constructor(){
20     afterEveryRender(
21       () => {
22         // news NG20
23         console.log('AfterEveryRender =>', this.title);
24       }
25     );
26   }
27 }
```

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

A screenshot of the Visual Studio Code editor showing two files: `app.ts` and `app.html`. In `app.ts`, the `*ngFor` directive is imported from '@angular/common' and used in the `@Component` decorator's `imports` array. A tooltip is visible over the `*ngFor` import, displaying a deprecation warning in French: "'NgFor' est déprécié. ts(6385)" and an English message: "common_module.d-Cpp8wYHt.d.ts(613, 4): La déclaration a été marquée ici comme étant dépréciée." Below this, it shows the new `NgFor` class definition and a brief description: "A structural directive that renders a template for each item in a collection. The directive". In `app.html`, the `*ngFor` directive is used on a `span` element to iterate over `imagesArray` and render `img` tags. A tooltip is visible over the `*ngFor` attribute, displaying the message "Go to component".

```
src > app > app.ts > App
1 import { NgFor } from '@angular/common';
2 import { Component } from '@angular/core';
3
4
5 @Component({
6   selector: 'app-root',
7   imports: [NgFor],
8   templateUrl:
9   styleUrls:
10 })
11 export class
12 // 1-props
13 public tit
14 public ima
```

```
src > app > app.html > span > img
Go to component
1 <span *ngFor="let image of imagesArray">
2   
3 </span>
```

Migration entre versions majeures



Copyright Michel ROCCHIOLESI - ORSYS

- 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`

Update Guide

Select the options that match your update

Angular versions

From v. To v.



Warning: Be sure to follow the guide below to migrate your application to the new version. You can't run `ng update` to update Angular applications more than one major version at a time.

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](#)

```
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/app.component.ts (420 bytes)
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.

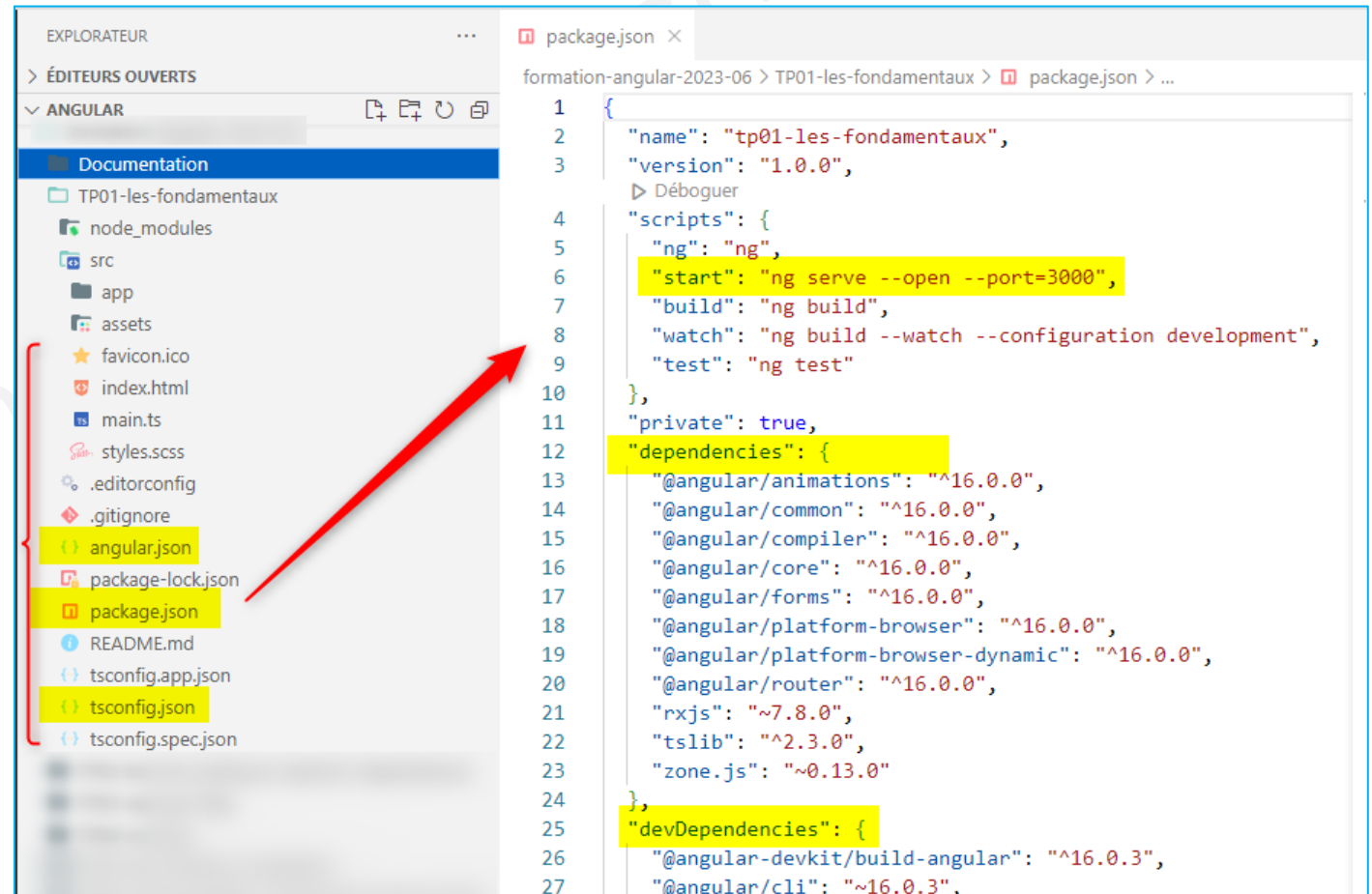
Il y a au moins 32k fichiers dans un simple projet Angular.

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



```
1 {
2   "name": "tp01-les-fondamentaux",
3   "version": "1.0.0",
4   "scripts": {
5     "ng": "ng",
6     "start": "ng serve --open --port=3000",
7     "build": "ng build",
8     "watch": "ng build --watch --configuration development",
9     "test": "ng test"
10  },
11  "private": true,
12  "dependencies": {
13    "@angular/animations": "^16.0.0",
14    "@angular/common": "^16.0.0",
15    "@angular/compiler": "^16.0.0",
16    "@angular/core": "^16.0.0",
17    "@angular/forms": "^16.0.0",
18    "@angular/platform-browser": "^16.0.0",
19    "@angular/platform-browser-dynamic": "^16.0.0",
20    "@angular/router": "^16.0.0",
21    "rxjs": "~7.8.0",
22    "tslib": "^2.3.0",
23    "zone.js": "~0.13.0"
24  },
25  "devDependencies": {
26    "@angular-devkit/build-angular": "^16.0.3",
27    "@angular/cli": "~16.0.3",
```

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

Ecmascript 2015+



```
> for (var i=0; i<=3 ; i++ ) { console.log(i); }
0 VM5352:1
1 VM5352:1
2 VM5352:1
3 VM5352:1
< undefined
> console.log(i);
4 VM5411:1
< undefined
> for (let j=0; j<=3 ; j++ ) { console.log(j); }
0 VM5535:1
1 VM5535:1
2 VM5535:1
3 VM5535:1
< undefined
> console.log(j)
✖ Uncaught ReferenceError: j is not defined VM5613:1
  at <anonymous>:1:13
>
```

Ecmascript 2015+



> for (let val of tab) { console.log(val); }	
ok	<u>VM6316:1</u>
cool	<u>VM6316:1</u>
<> undefined	
> for (let i in tab) { console.log(i); }	
0	<u>VM6332:1</u>
1	<u>VM6332:1</u>
<> undefined	
> for (let[i,val] of tab.entries()) { console.log(i, val);}	
0 'ok'	<u>VM6348:1</u>
1 'cool'	<u>VM6348:1</u>
<> undefined	
>	

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+



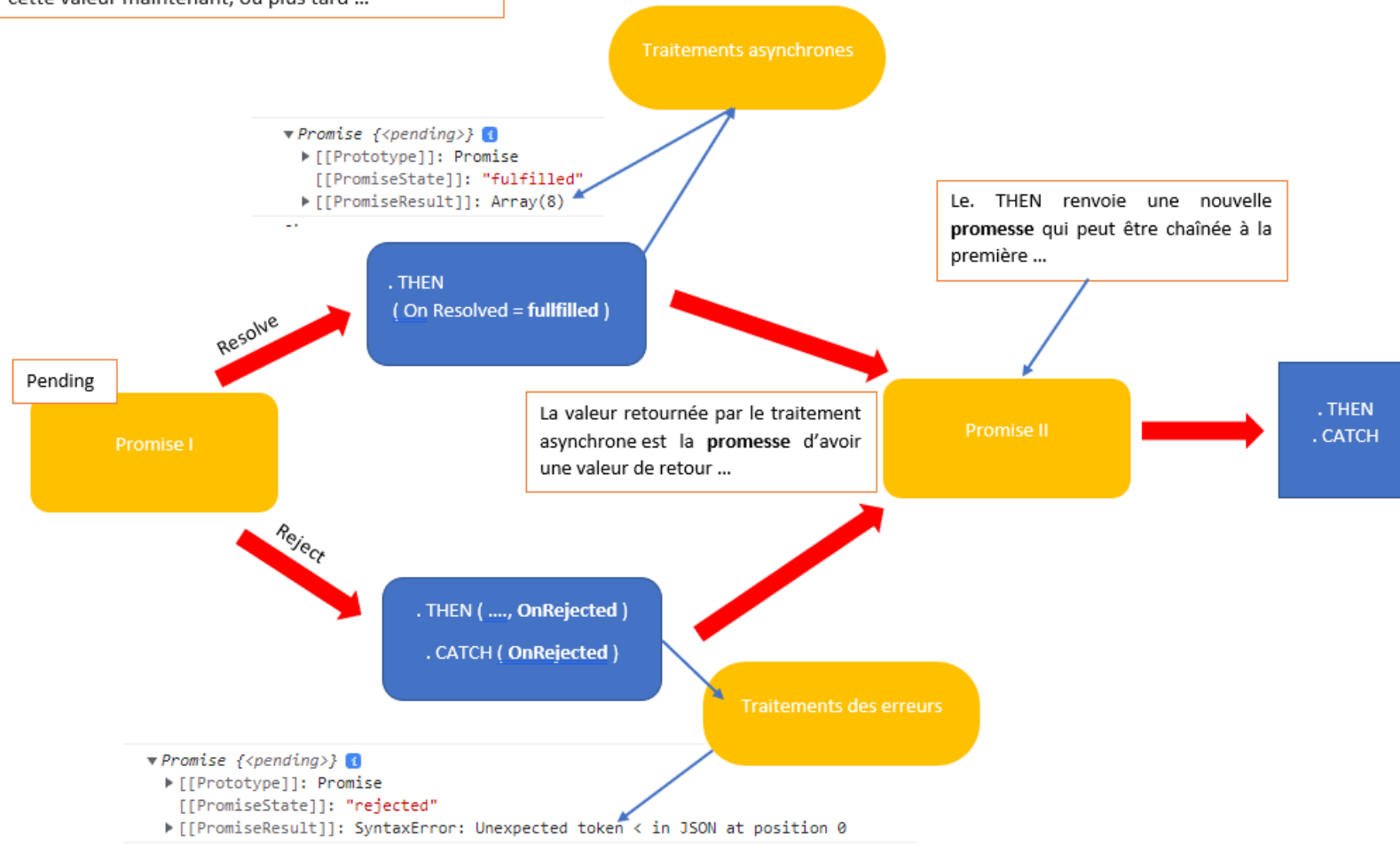
```
58 // -----  
59 let personne = {  
60   prenom: "Luke",  
61   nom: "Skywalker",  
62   // tableaux  
63   langages: ["JS", "React", "NG", "Vue"],  
64   // méthodes  
65   nomComplet() {  
66     return `${this.prenom} ${this.nom}`;  
67   }  
68 }  
69 console.log(personne.nomComplet());  
70  
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+



VS

La valeur retournée par le traitement **asynchrone** associé aux promesses est une **promesse** d'avoir cette valeur maintenant, ou plus tard ...



Ecmascript 2015+



```
1  export const fnFilm = () => {
2      // ES6 : fonction fetch
3      // simplification d'écriture d'Ajax
4      // amélioration du process JS ( Promises )
5
6      // 1- Request Query (requête Infos):
7      // const _url = 'https://test.webjs.fr/films.json';
8      const _url = 'https://dev.webjs.fr/films.json';
9      // const _url='http://127.0.0.1:3000/json/films.json'; // en local avec un serveur json
10
11     // 2- Request Init Query
12     // définir la méthode (get ou post) ... ou patch ou put ou delete)
13     const _method = 'get';
14     // définir nos entêtes HTTP
15     const _headers = new Headers();
16
17     _headers.append('Content-Type', 'text/json'); //type mime (ex:image/png)
18     // _headers.append('Access-Control-Allow-Origin', 'cors');
19
20     // 3- Ecriture du FETCH
21     // fetch renvoie une promesse
22     fetch(
23         _url,
24         {
25             method: _method,
26             headers: _headers
27         }
28     ).then(
29         // si on arrive à avoir une réponse du serveur
30         // état onFullFilled
31         (responseHTTP) => {
32             // le then récupère une réponse 200 ok 404
33             console.warn('On Fullfilled du 1er Then');
34             console.log(responseHTTP);
35             console.log(responseHTTP.body);
36         }
37     );
38 }
```



Formation ANGULAR

A bientôt.