ORSYS formation

Formation ANGULAR

Chapitre 3

Animé par Michel BOCCIOLESI

Table des matières

Les Formulaires

Les formulaires Angular (avec Matérial Design)

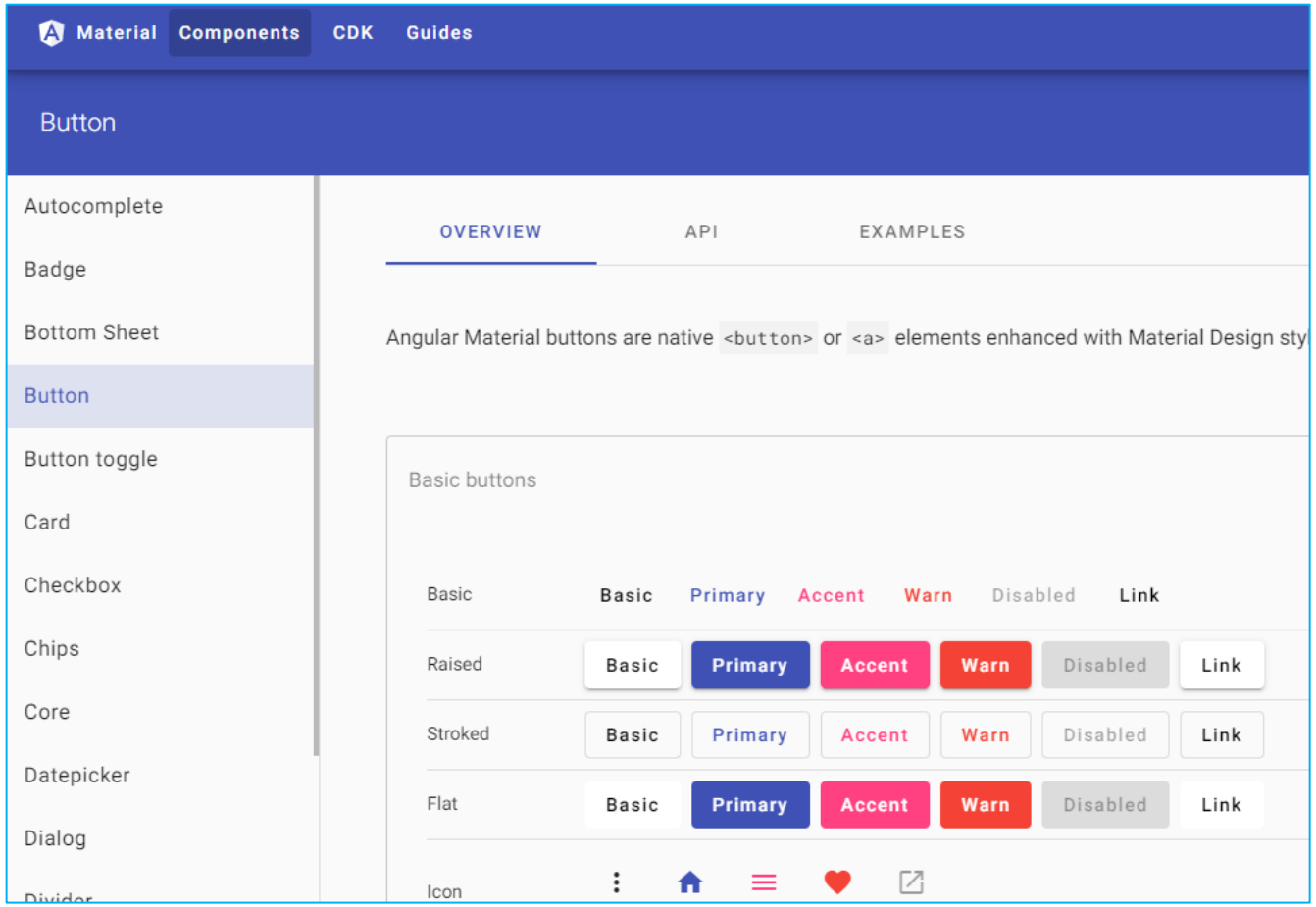
- https://material.angular.io/components/categories
- https://fonts.google.com/icons

HTML TS CSS

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

HTML TS CSS

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



Les formulaires Angular (avec Matérial Design)

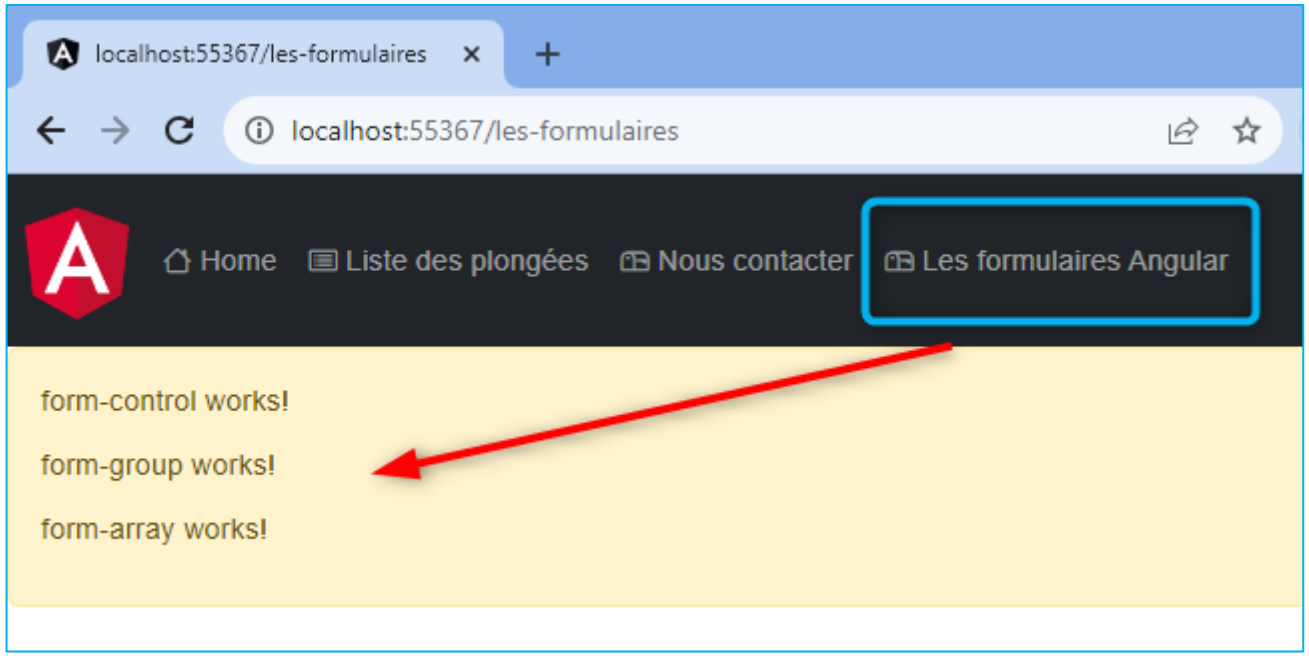
ng add @angular/material

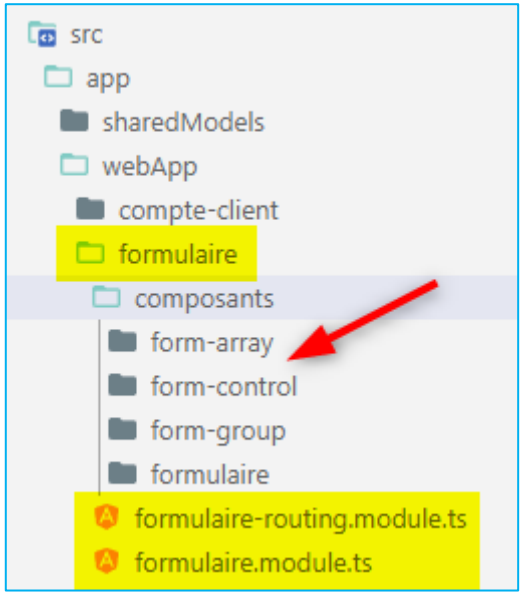
ng add @angular/localize
ng add @angular/pwa

Les formulaires Angular (avec Matérial Design)

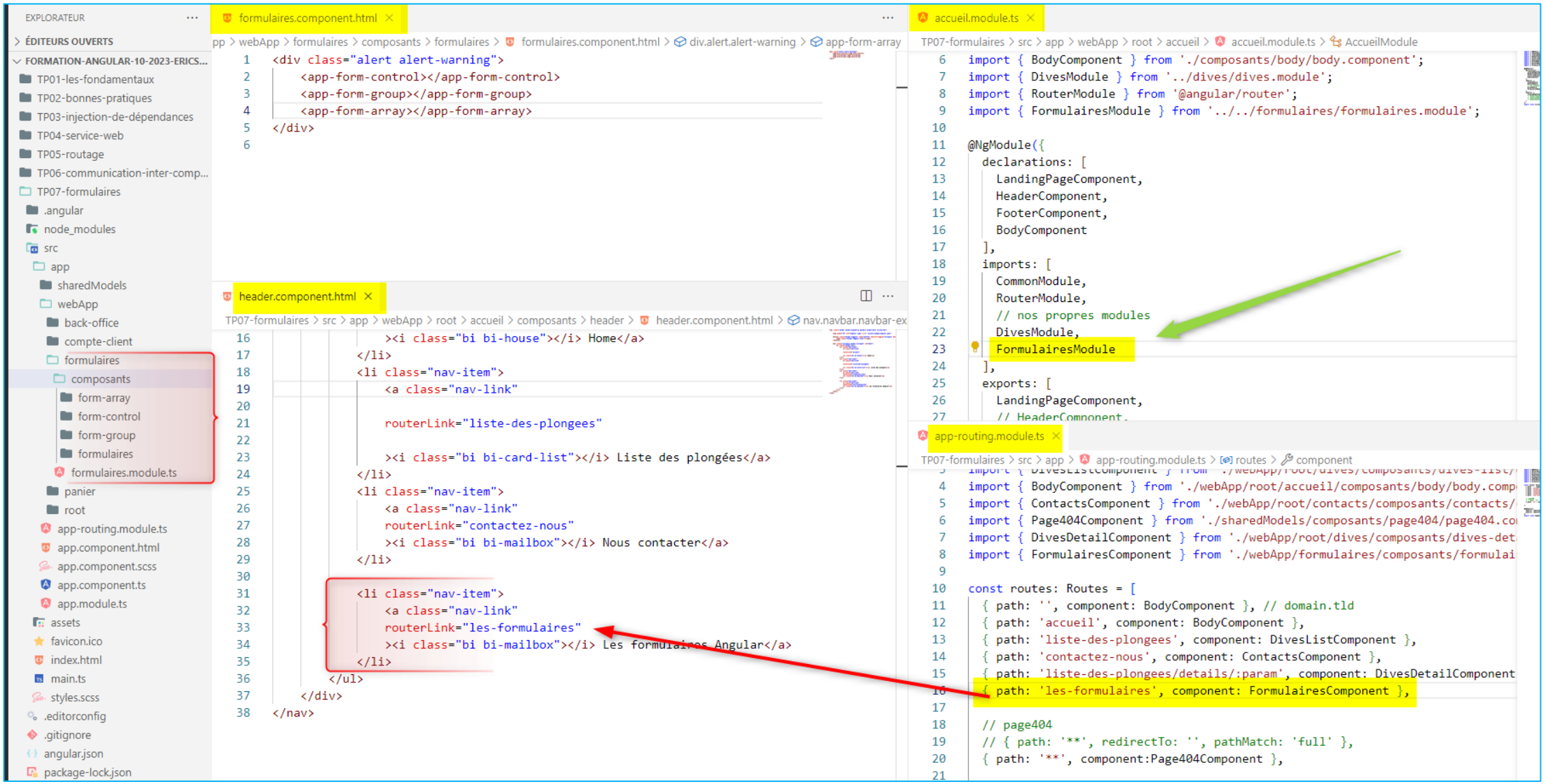
https://dev.webjs.fr/1-Angular/ressources-formation/material.module.ts
https://dev.webjs.fr/1-Angular/ressources-formation/material-all.module.ts

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



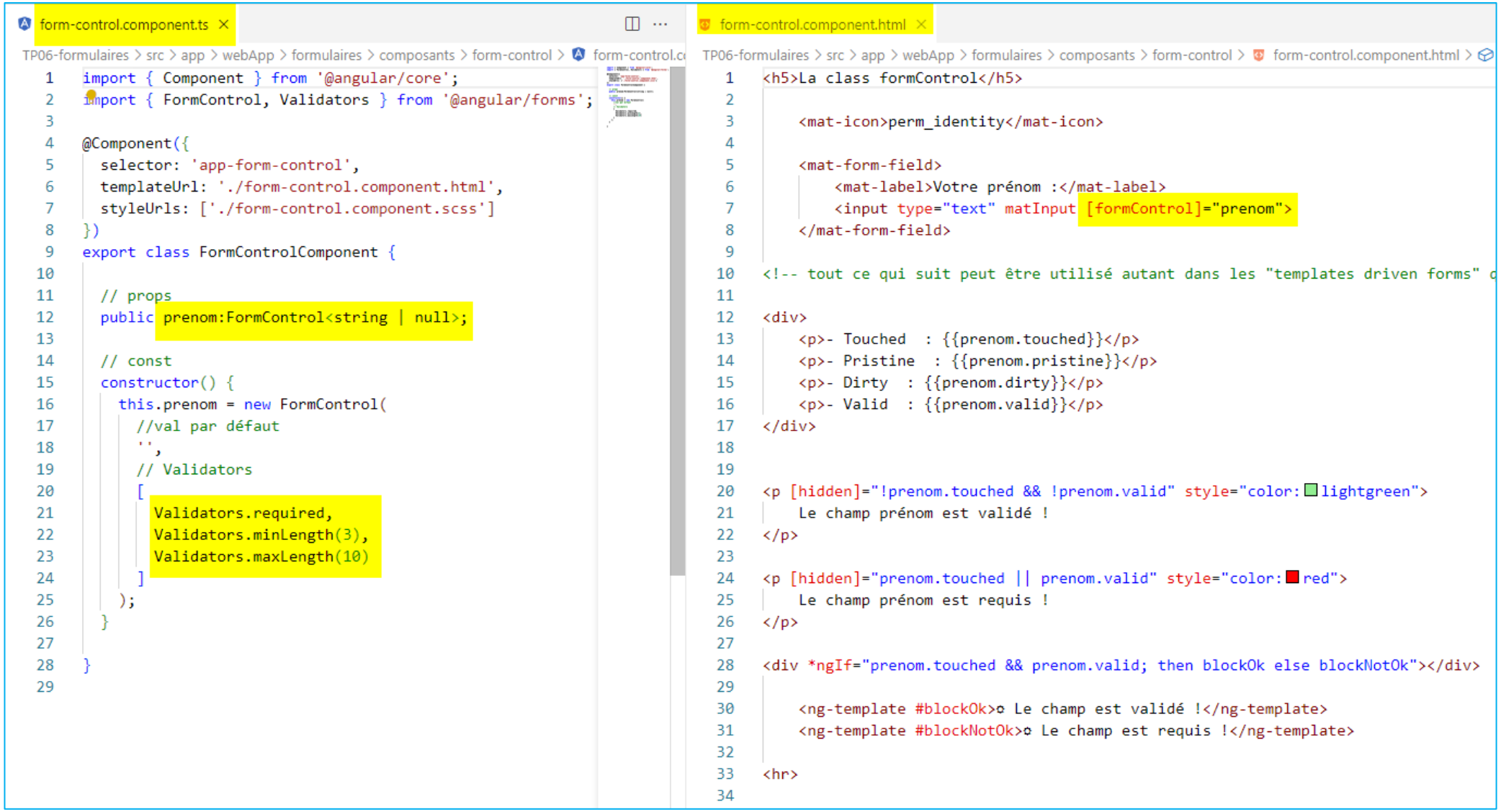


Les formulaires Angular (avec Matérial Design)



Les formulaires Angular

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



form-control.component.ts

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

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

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

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

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

}

```

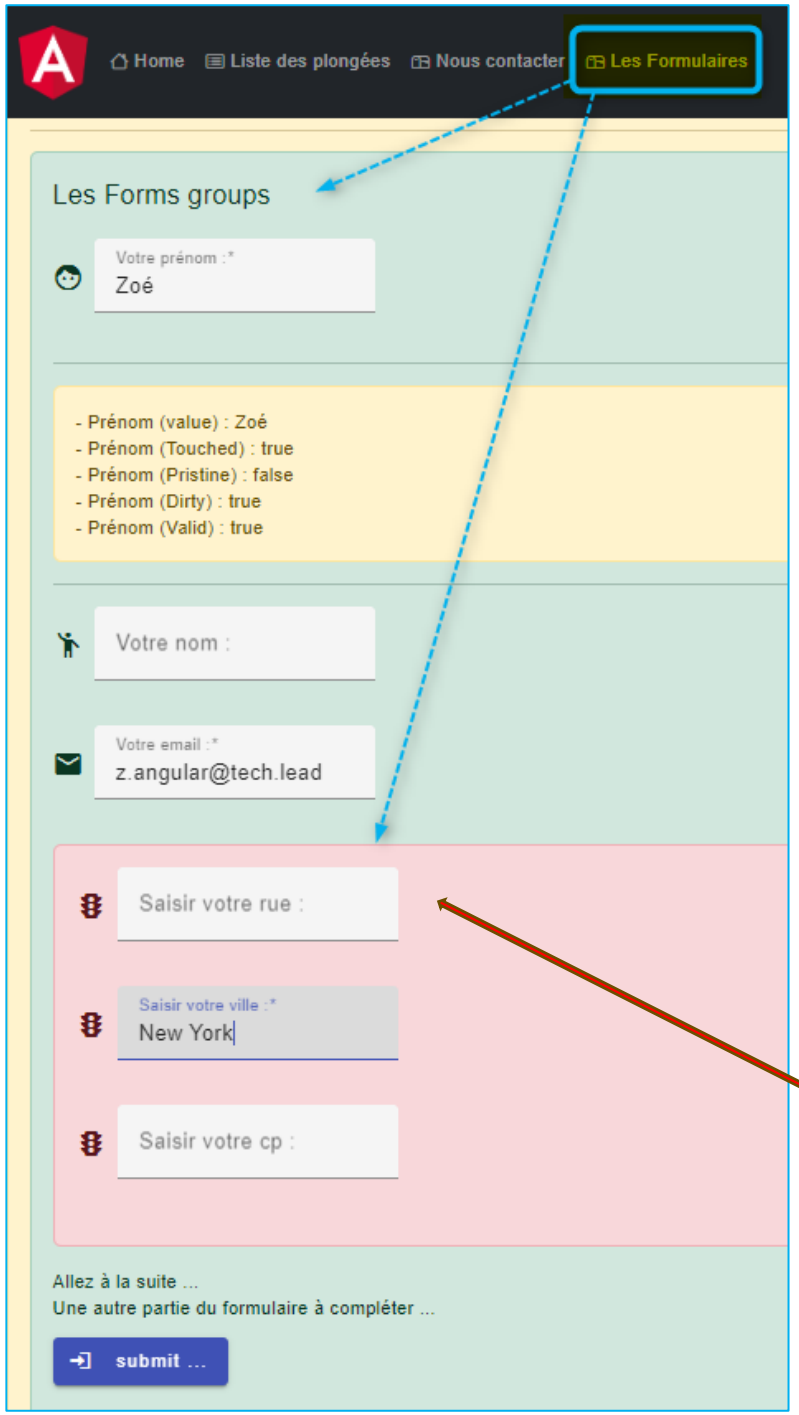
form-control.component.html

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

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

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

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

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

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


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

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

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

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

<hr>

```

La Class form-control

La class form-group



form-group.component.html

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

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

La Class formGroup

```
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

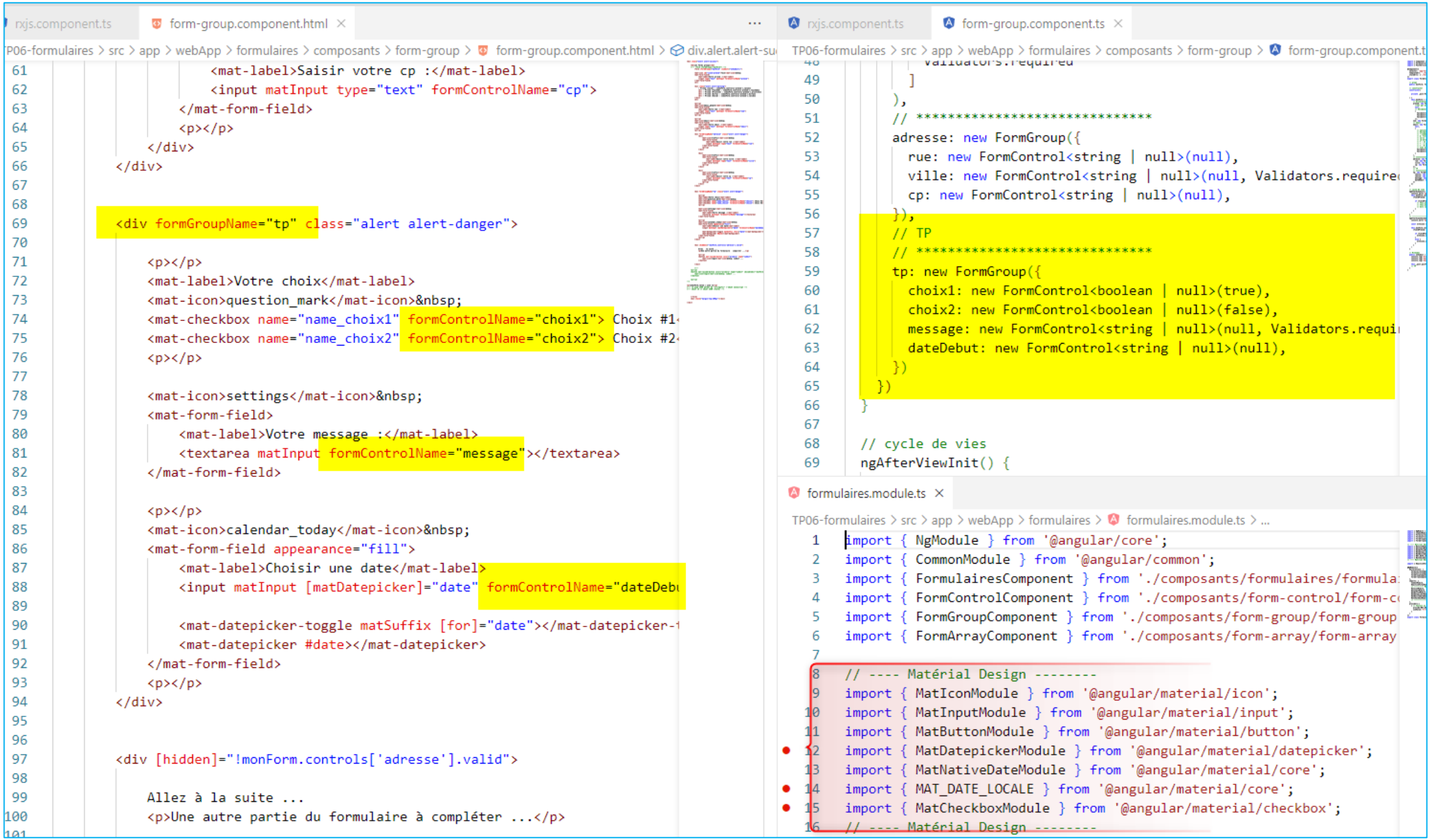
3- Créer le service « post.service.ts » qui poste les datas sur un json-server

```
"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

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

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

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

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

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

formulaires.module.ts

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

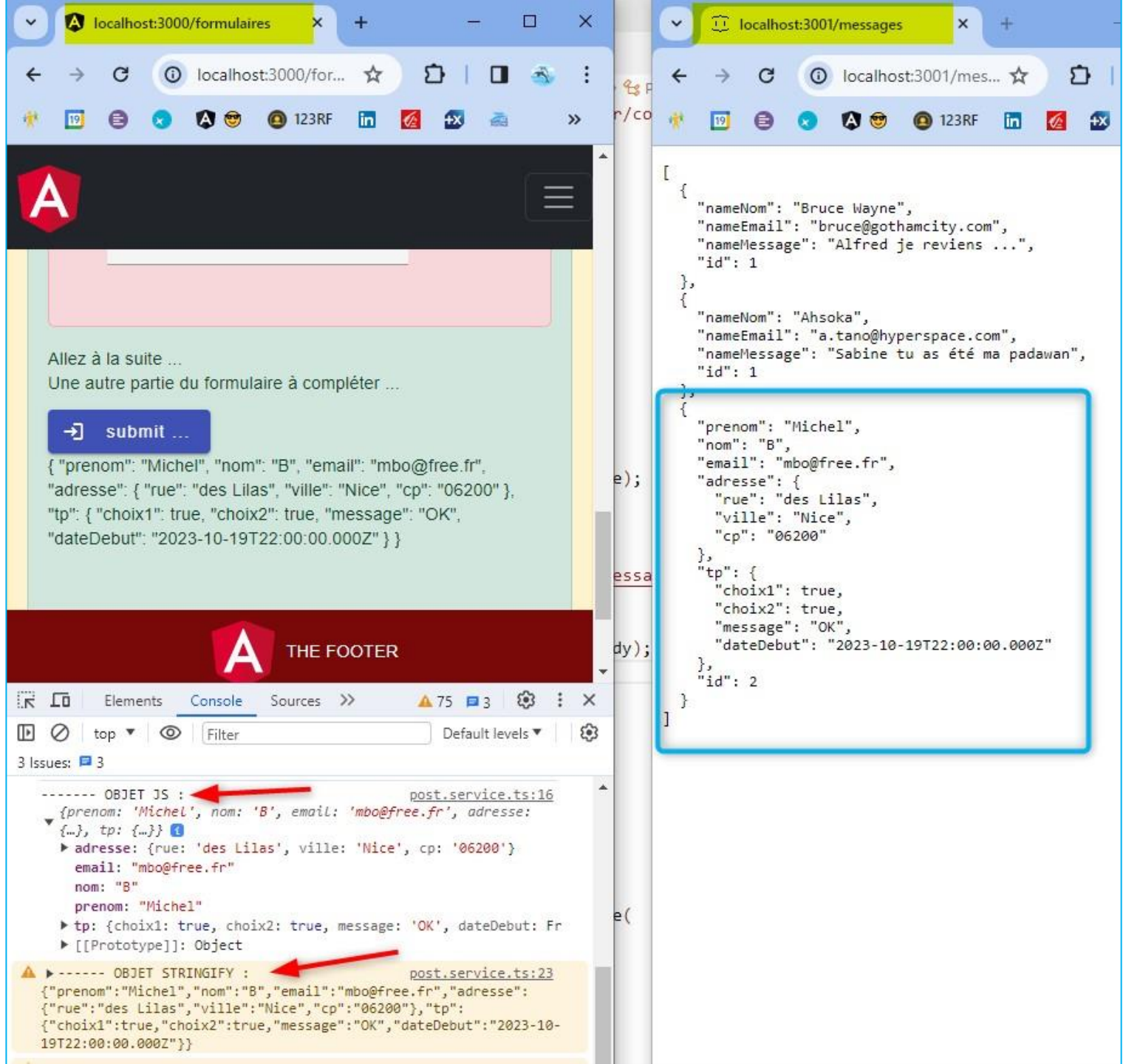
```
1   import { NgModule } from '@angular/core';
2   import { CommonModule } from '@angular/common';
3   import { FormulairesComponent } from './composants/formulaires/formula
4   import { FormControlComponent } from './composants/form-control/form-c
5   import { FormGroupComponent } from './composants/form-group/form-group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

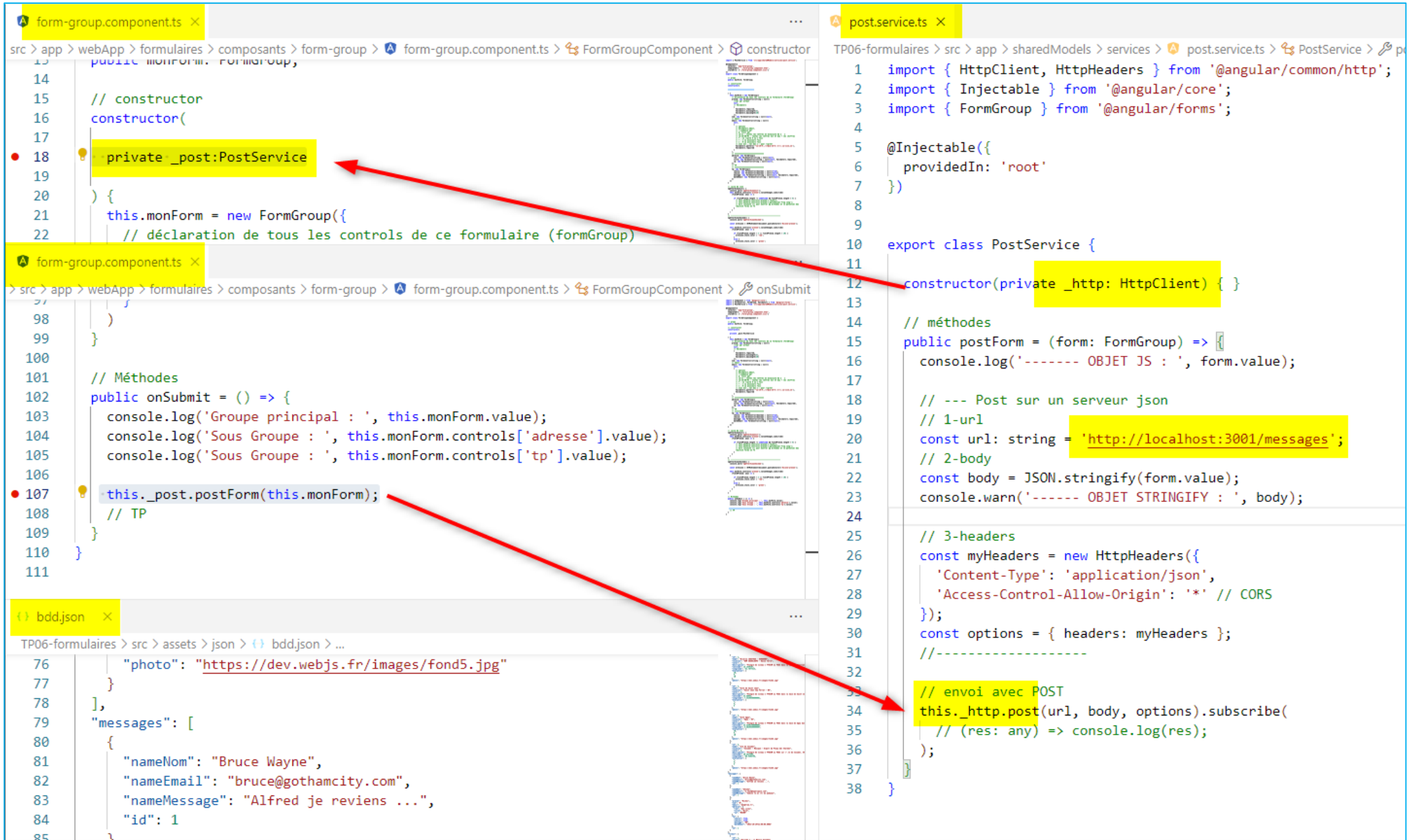
La Class formGroup

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

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

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

}
}
```

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

```
1 Issue: 1

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

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

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

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

>
```


localhost:3000/formulaires
localhost:3000/for...
Allez à la suite ...
Une autre partie du formulaire à compléter ...
submit ...
{ "prenom": "Michel", "nom": "B", "email": "mbo@free.fr",
"adresse": { "rue": "des Lilas", "ville": "Nice", "cp": "06200" },
"tp": { "choix1": true, "choix2": true, "message": "OK",
"dateDebut": "2023-10-19T22:00:00.000Z" } }
THE FOOTER
Elements
Console
Sources
75
3
top
Filter
Default levels
3 Issues: 3
------- OBJET JS :
post.service.ts:16
{prenom: 'Michel', nom: 'B', email: 'mbo@free.fr', adresse: {…}, tp: {…}}
adresse: {rue: 'des Lilas', ville: 'Nice', cp: '06200'}
email: "mbo@free.fr"
nom: "B"
prenom: "Michel"
tp: {choix1: true, choix2: true, message: 'OK', dateDebut: Fr
[[Prototype]]: Object
------ OBJET STRINGIFY :
post.service.ts:23
{"prenom":"Michel","nom":"B","email":"mbo@free.fr","adresse":{"rue":"des Lilas","ville":"Nice","cp":"06200"},"tp":{"choix1":true,"choix2":true,"message":"OK","dateDebut":"2023-10-19T22:00:00.000Z"}}
localhost:3001/messages
localhost:3001/mes...
[
{
"nameNom": "Bruce Wayne",
"nameEmail": "bruce@gothamcity.com",
"nameMessage": "Alfred je reviens ...",
"id": 1
},
{
"nameNom": "Ahsoka",
"nameEmail": "a.tano@hyperspace.com",
"nameMessage": "Sabine tu as été ma padawan",
"id": 1
},
{
"prenom": "Michel",
"nom": "B",
"email": "mbo@free.fr",
"adresse": {
"rue": "des Lilas",
"ville": "Nice",
"cp": "06200"
},
"tp": {
"choix1": true,
"choix2": true,
"message": "OK",
"dateDebut": "2023-10-19T22:00:00.000Z"
},
"id": 2
}
]


form-group.component.ts
src > app > webApp > formulaires > composants > form-group > form-group.component.ts > FormGroupComponent > constructor
// constructor
constructor(
private _post:PostService
) {
this.monForm = new FormGroup({
// déclaration de tous les controls de ce formulaire (formGroup)
form-group.component.ts
src > app > webApp > formulaires > composants > form-group > form-group.component.ts > FormGroupComponent > onSubmit
// Méthodes
public onSubmit = () => {
console.log('Groupe principal : ', this.monForm.value);
console.log('Sous Groupe : ', this.monForm.controls['adresse'].value);
console.log('Sous Groupe : ', this.monForm.controls['tp'].value);
this._post.postForm(this.monForm);
// TP
bdd.json
TP06-formulaires > src > assets > json > bdd.json > ...
"photo": "https://dev.webjs.fr/images/fond5.jpg"
"messages": [
"nameNom": "Bruce Wayne",
"nameEmail": "bruce@gothamcity.com",
"nameMessage": "Alfred je reviens ...",
"id": 1
post.service.ts
TP06-formulaires > src > app > sharedModels > services > post.service.ts > PostService
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { FormGroup } from '@angular/forms';
@Injectable({
providedIn: 'root'
})
export class PostService {
constructor(private _http: HttpClient) { }
// méthodes
public postForm = (form: FormGroup) => {
console.log('------- OBJET JS : ', form.value);
// --- Post sur un serveur json
// 1-url
const url: string = 'http://localhost:3001/messages';
// 2-body
const body = JSON.stringify(form.value);
console.warn('------ OBJET STRINGIFY : ', body);
// 3-headers
const myHeaders = new HttpHeaders({
'Content-Type': 'application/json',
'Access-Control-Allow-Origin': '*' // CORS
});
const options = { headers: myHeaders };
//--------------------
// envoi avec POST
this._http.post(url, body, options).subscribe(
// (res: any) => console.log(res);
);
}

La Class formArray

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

```ts
export class FormArrayComponent {

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

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

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

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

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

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

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

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

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

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

Les Tests Unitaires

1Er exemple : Apprendre de l'existant

app.component.ts

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

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

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

```

app.component.html

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

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

app.component.spec.ts

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

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

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

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


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


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

```

2nd exemple : Injection de services

warn.service.ts

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

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

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

operations.service.ts

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

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

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

  constructor(
    private _warn: WarnService
  ) { }

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

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

operations.service.spec.ts

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

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


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

  // 2- liste des spécifications

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

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

  });

  // 4- duplication du premier IT

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

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

  });


})
```

2nd exemple : Injection de services

warn.service.ts

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

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

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

operations.service.ts

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

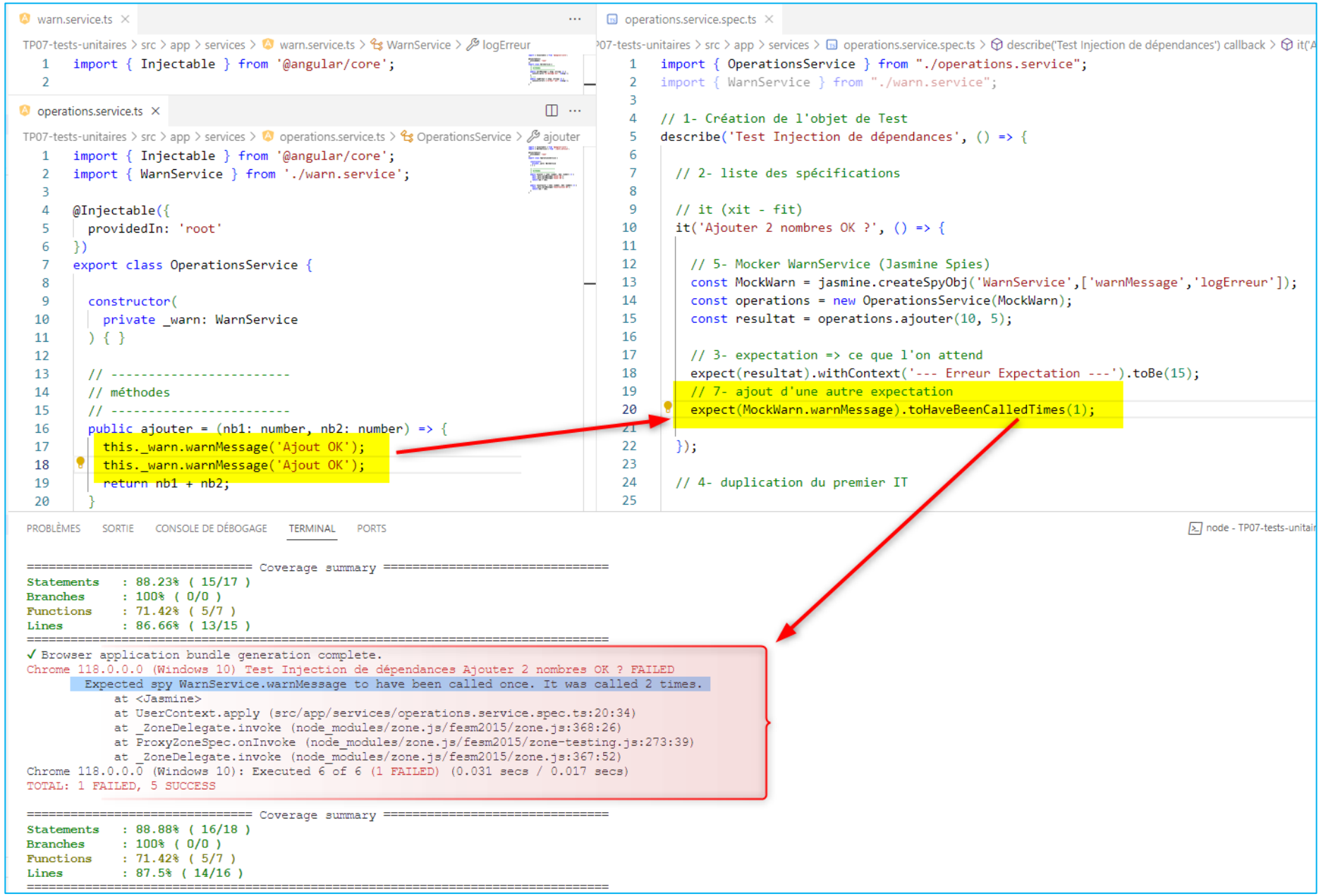
```ts
import { Injectable } from '@angular/core';
import { WarnService } from './warn.service';

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

  constructor(
    private _warn: WarnService
  ) { }

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

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

operations.service.spec.ts

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

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

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

  // 2- liste des spécifications

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

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

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

  });

  // 4- duplication du premier IT

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

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

  });


})
```

2nd exemple : Injection de services



warn.service.ts

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

operations.service.ts

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

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

  constructor(
    private _warn: WarnService
  ) { }

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

operations.service.spec.ts

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

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

  // 2- liste des spécifications

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

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

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

  });

  // 4- duplication du premier IT
```

PROBLÈMES SORTIE CONSOLE DE DÉBOGAGE TERMINAL PORTS

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

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

2nd exemple : Injection de services

warn.service.ts

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

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

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

operations.service.ts

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

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

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

  constructor(
    private _warn: WarnService
  ) { }

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

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

operations.service.spec.ts

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

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

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

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

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

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

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

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

2nd exemple : Injection de services

warn.service.ts

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

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

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

operations.service.ts

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

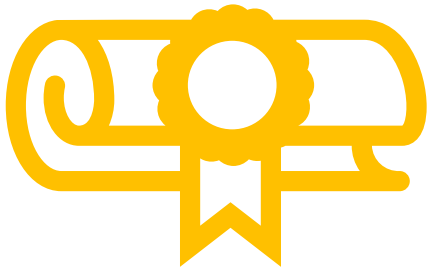
```ts
  // ------------------------
  // méthodes
  // ------------------------
  public ajouter = (nb1: number, nb2: number) => {
    this._warn.warnMessage('Ajout OK');
    // this._warn.warnMessage('Ajout OK');
    return nb1 + nb2;
  }

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

operations.service.spec.ts

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

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

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

Rxjs – Observables Approfondissement



RXJS et les OBSERVABLES

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

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

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

- Cela offre beaucoup de souplesse dans le traitement des données et le gestion des requêtes asynchrones.

RXJS et les OBSERVABLES

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

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

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

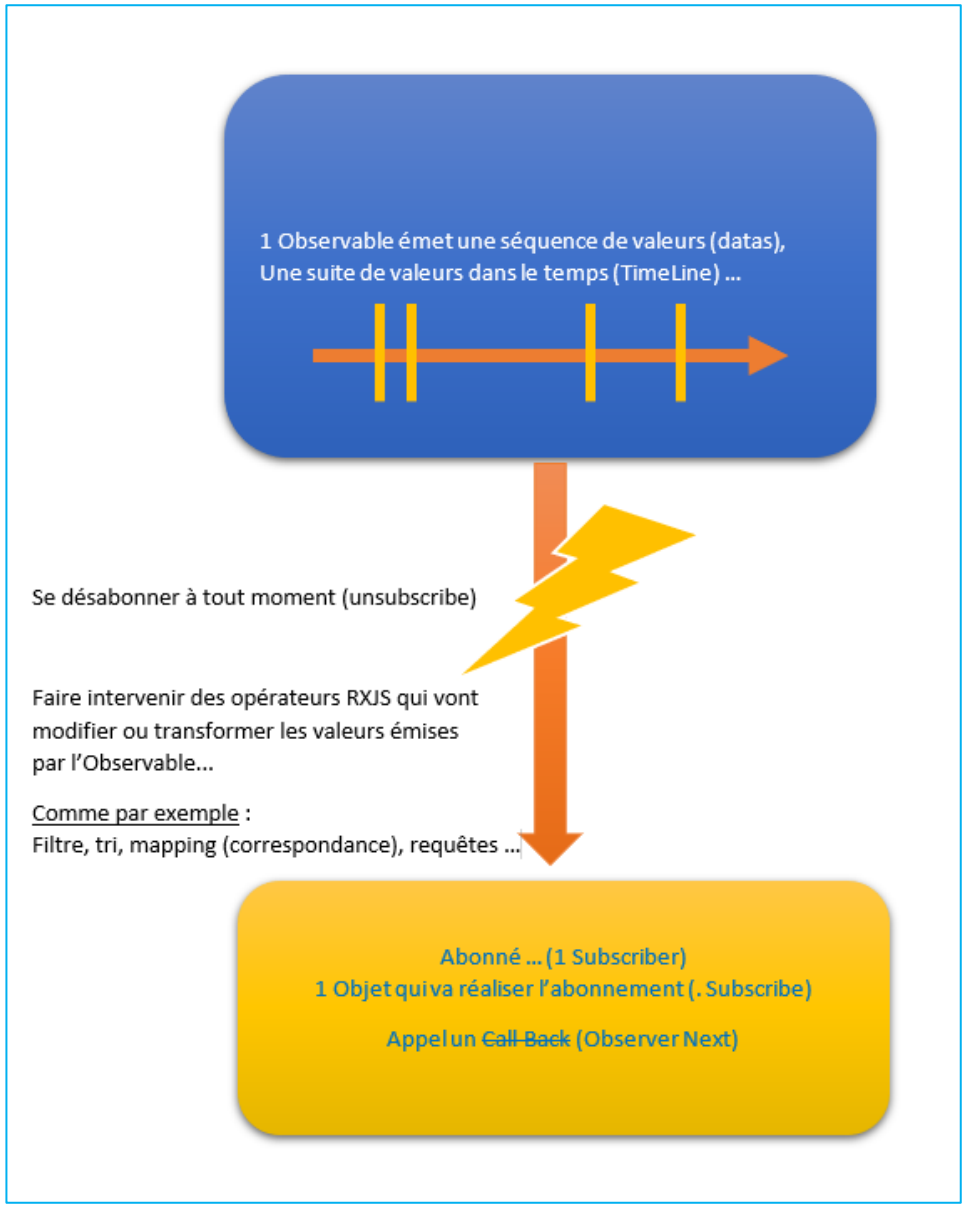
Opérateurs : des fonctions tap, map, merge, filter qui vont pouvoir modifier/transformer ce flux de valeurs émises par l'observable

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

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

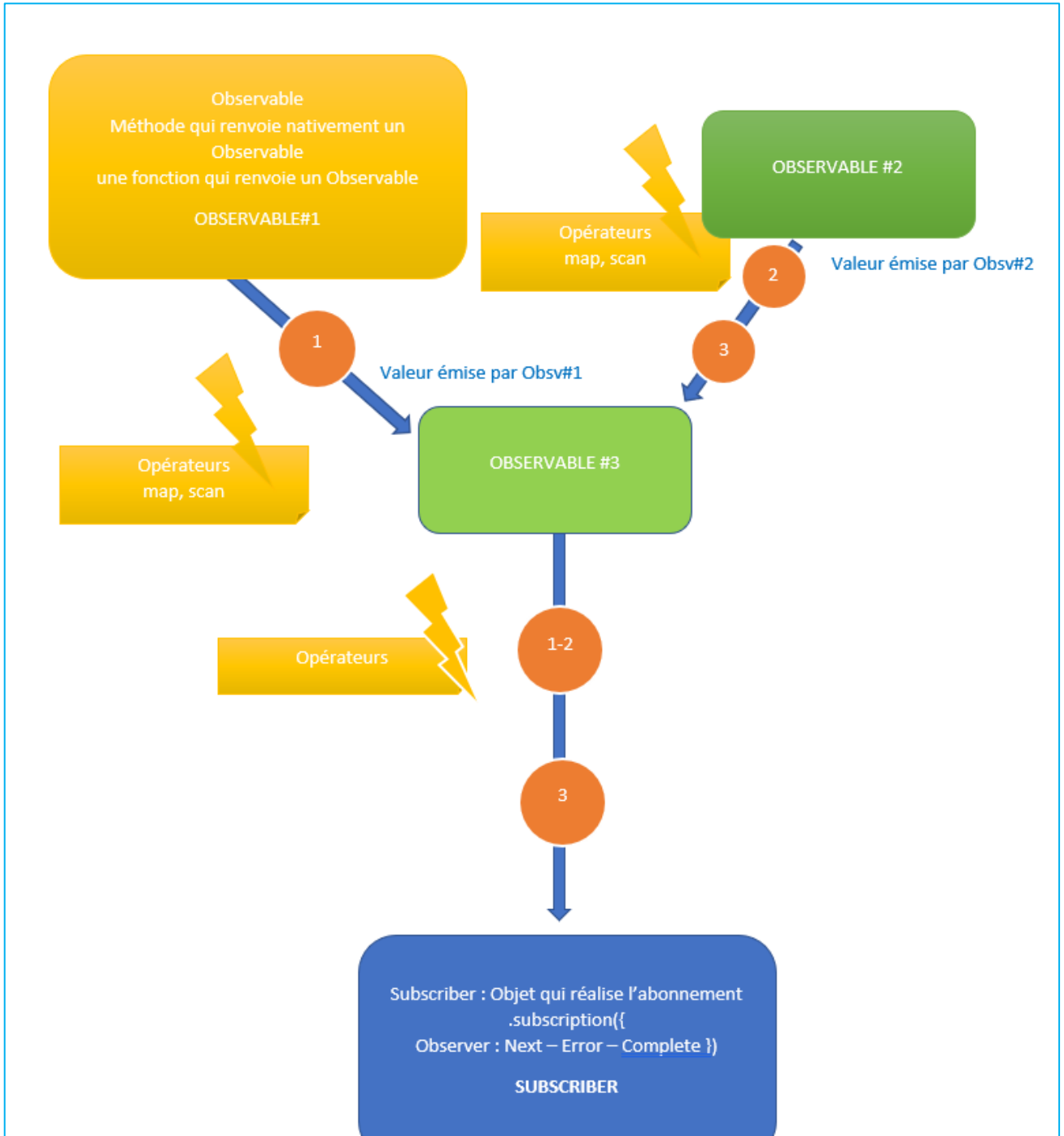
Les séquences de valeurs émises sont facilement annulables !

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

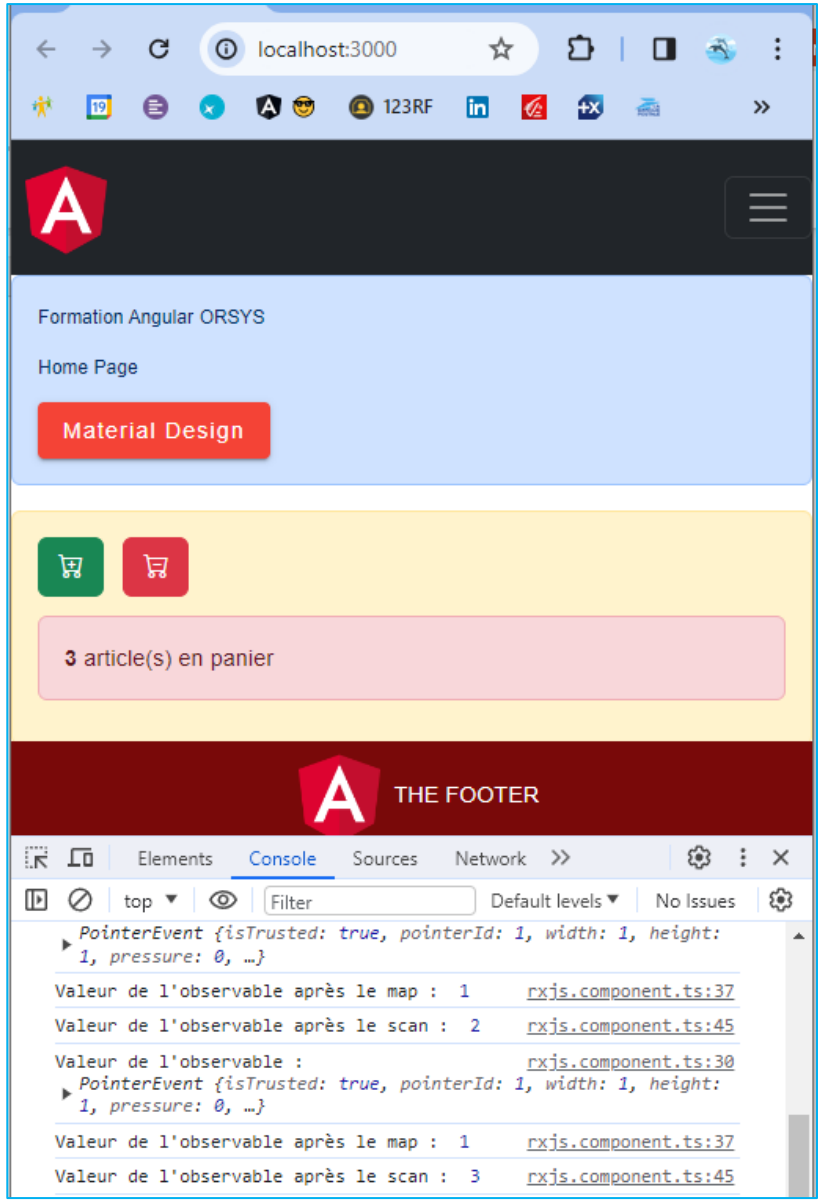


RXJS et les OBSERVABLES

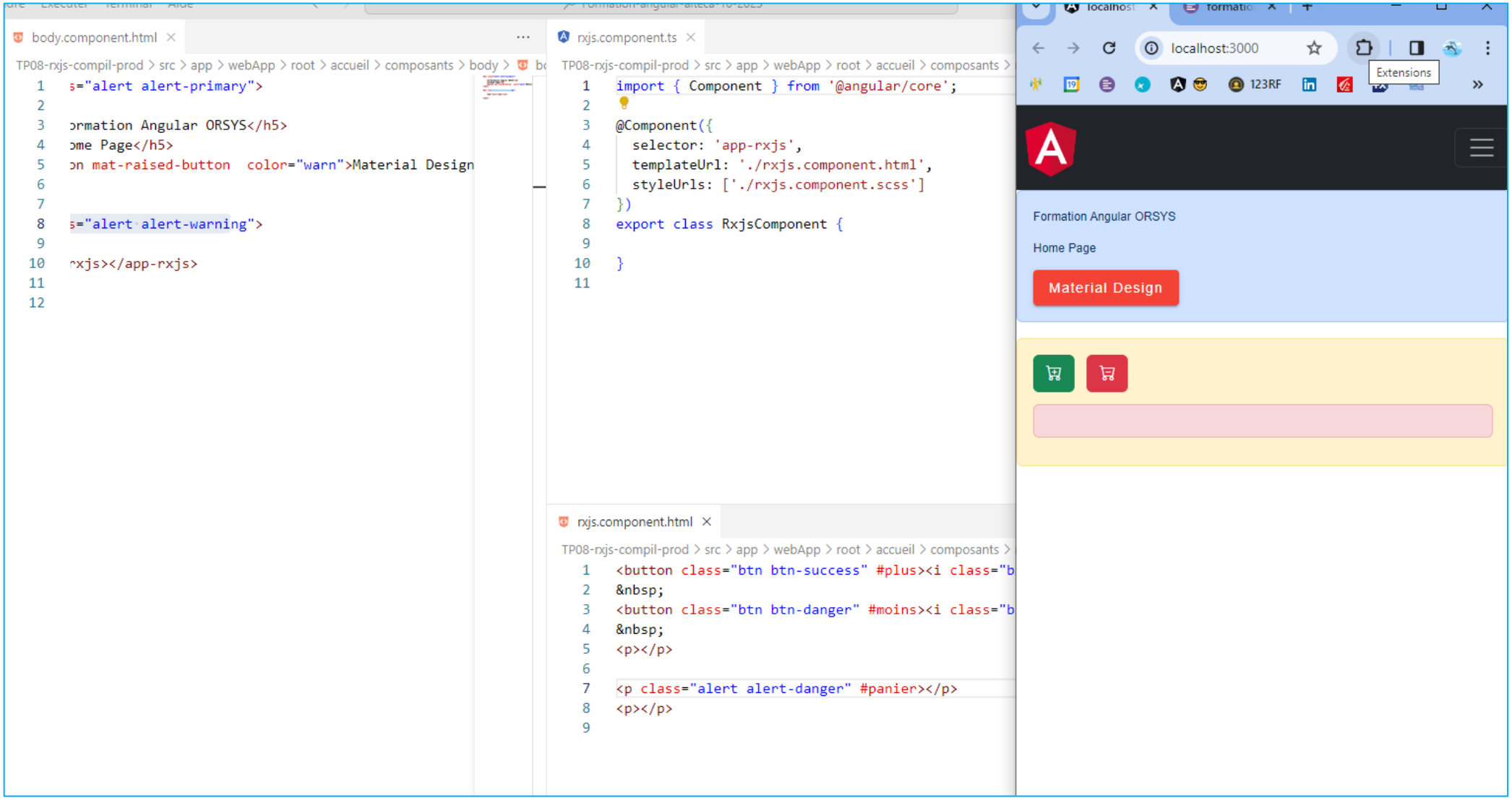
On peut facilement les recomposer
ou les recombiner pour former
une nouvelle séquence de valeurs !



RXJS et les OBSERVABLES



RXJS et les OBSERVABLES



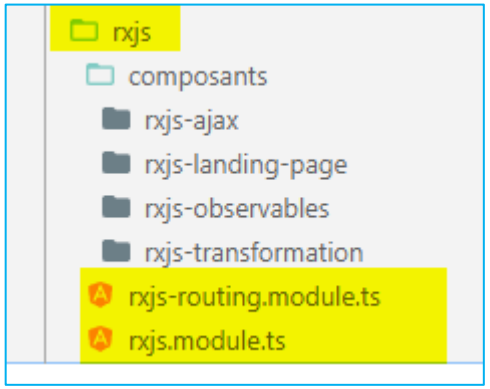
```
// ----------------------------------------
// création de l'observable #1 : PLUS
// ----------------------------------------
const plus$ = fromEvent(this.eltPlus.nativeElement, 'click')
  .pipe(
    startWith(0),
    tap(
      (valObs) => console.log(`Valeur de l'observable : `, valObs)
      //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

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

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

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

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

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

RXJS et les OBSERVABLES

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

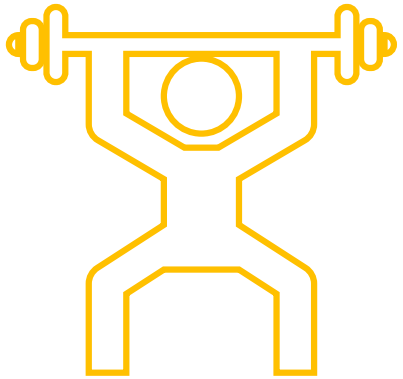


```html
<button class="btn btn-warning" (click)="createObservable1()">Créer un Observable</button>
<ul id="formations" class="list-group"></ul>

<p></p>


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

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

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

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

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

```ts
// ------------------------------------------
public createObservable2: any = () => {
  // tableau à 2 entrées
  const formationsArray2: Formation[] = [
    { cours: 'NG 16', duree: 4 },
    { cours: 'REACT 17', duree: 3 },
    { cours: 'VUE*', duree: 2 },
    { cours: 'VUE', duree: 4 },
    { cours: 'ECMA SCRIPT*', duree: 2 },
    { cours: 'TYPESCRIPT', duree: 5 }
  ];
  // console.table(formationsArray2);
  // ------------------------------------------
  // création d'un observable SANS le nommer
  from(formationsArray2)
    .pipe(
      // permet d'enchainer les opérateurs RXJS
      tap(
        // observer NEXT
        (formation: Formation) => console.table(formation)
      ),
      // first(),
      // last(),
      // first(
      //   // prédicat === patern
      //   (formation:Formation) => {
      //     return formation.duree ===2 ;
      //   }
      // ),
      delay(3000),
      delay(3000),
      filter(
        (formation: Formation) => {
          return formation.duree === 4;
        }
      ),
      tap(
        // observer NEXT
        (formation: Formation) => {
          console.warn(formation);
          const eltLi = <HTMLElement>document.createElement('li');
          eltLi.innerHTML = `${formation.cours} : ${formation.duree} jours.`;
          eltLi.className = 'list-group-item';
          // nativeElement est obligatoire ici car on récupère l'élément par Vie
          this.eltUl2.nativeElement.appendChild(eltLi);
        }
      ),
      catchError(
        // capturer les erreurs de l'observable
        (e: any): any => {
          console.log(e);

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

Les versions marquantes !



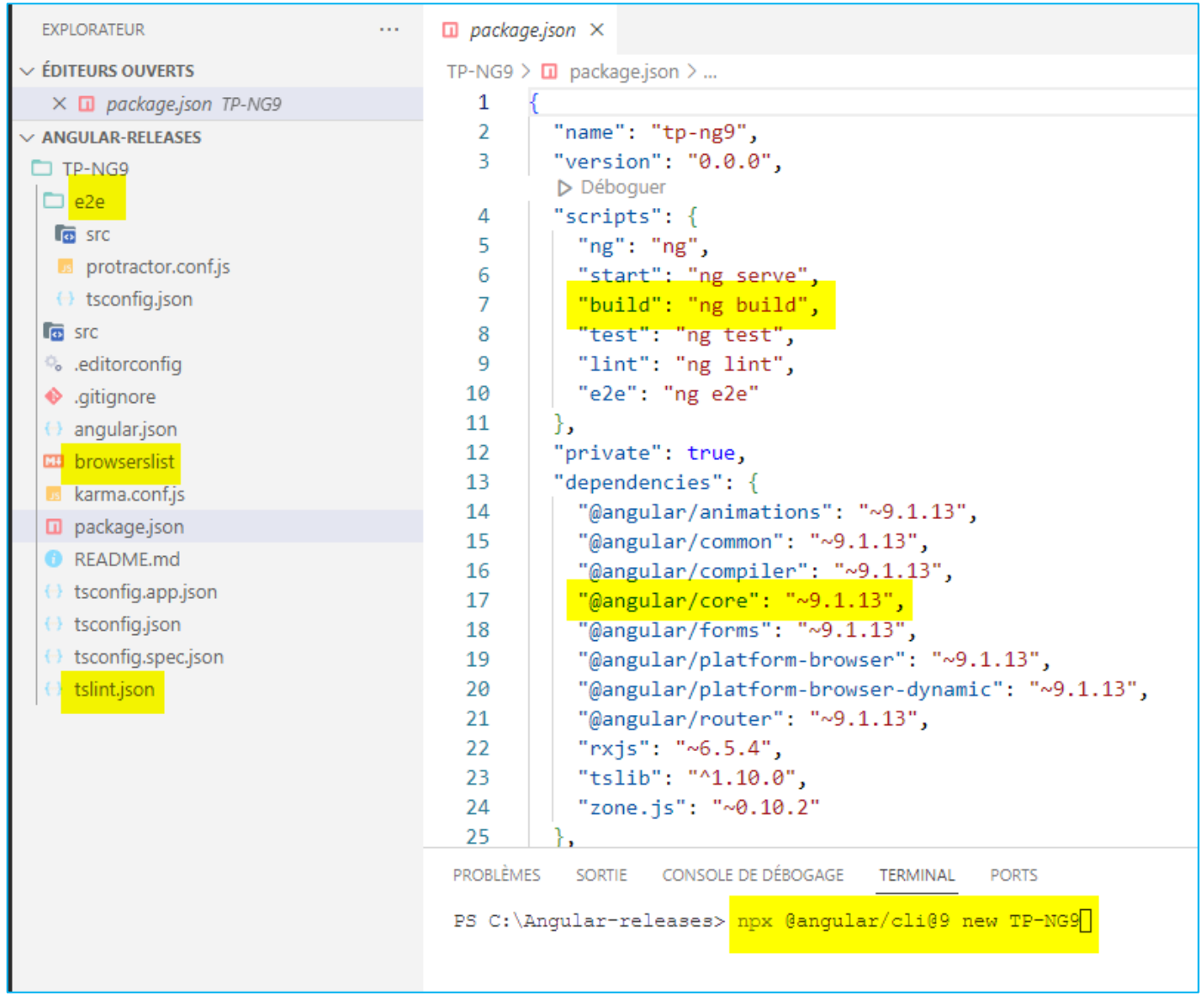
Les versions marquantes Angular

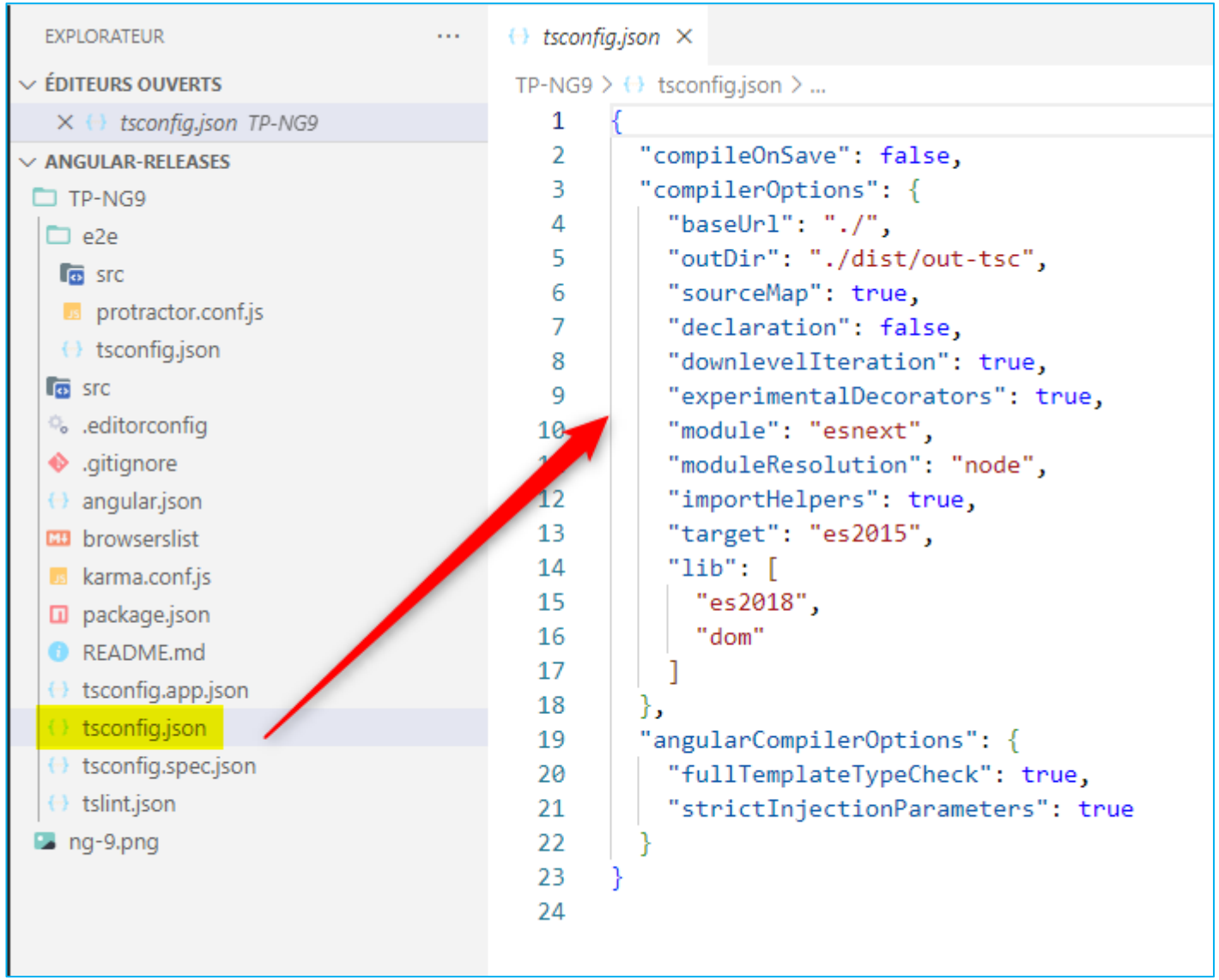
- Version 9 (6 février 2020)

  - Nouveau compiler et nouveau moteur de rendu Ivy
  - Réduction notable de la taille des bundles (build)
  - Optimisation de la compilation (temps, débug) AOT

  activé par défaut avec ng serve => remontée des erreurs en mode dév …

  - composant YouTube
  https://github.com/angular/components/tree/main/src/youtube-player
  - composant Google Maps
  https://github.com/angular/components/blob/main/src/google-maps/README.md


EXPLORATEUR
ÉDITEURS OUVERTS
package.json TP-NG9
ANGULAR-RELEASES
TP-NG9
e2e
src
protractor.conf.js
tsconfig.json
src
.editorconfig
.gitignore
angular.json
browserslist
karma.conf.js
package.json
README.md
tsconfig.app.json
tsconfig.json
tsconfig.spec.json
tslint.json
package.json
TP-NG9 > package.json > ...
{
"name": "tp-ng9",
"version": "0.0.0",
Déboguer
"scripts": {
"ng": "ng",
"start": "ng serve",
"build": "ng build",
"test": "ng test",
"lint": "ng lint",
"e2e": "ng e2e"
},
"private": true,
"dependencies": {
"@angular/animations": "~9.1.13",
"@angular/common": "~9.1.13",
"@angular/compiler": "~9.1.13",
"@angular/core": "~9.1.13",
"@angular/forms": "~9.1.13",
"@angular/platform-browser": "~9.1.13",
"@angular/platform-browser-dynamic": "~9.1.13",
"@angular/router": "~9.1.13",
"rxjs": "~6.5.4",
"tslib": "^1.10.0",
"zone.js": "~0.10.2"
},
PROBLÈMES
SORTIE
CONSOLE DE DÉBOGAGE
TERMINAL
PORTS
PS C:\Angular-releases> npx @angular/cli@9 new TP-NG9


EXPLORATEUR
ÉDITEURS OUVERTS
tsconfig.json TP-NG9
ANGULAR-RELEASES
TP-NG9
e2e
src
protractor.conf.js
tsconfig.json
src
.editorconfig
.gitignore
angular.json
browserslist
karma.conf.js
package.json
README.md
tsconfig.app.json
tsconfig.json
tsconfig.spec.json
tslint.json
ng-9.png
tsconfig.json
TP-NG9 > tsconfig.json > ...
{
"compileOnSave": false,
"compilerOptions": {
"baseUrl": "./",
"outDir": "./dist/out-tsc",
"sourceMap": true,
"declaration": false,
"downlevelIteration": true,
"experimentalDecorators": true,
"module": "esnext",
"moduleResolution": "node",
"importHelpers": true,
"target": "es2015",
"lib": [
"es2018",
"dom"
]
},
"angularCompilerOptions": {
"fullTemplateTypeCheck": true,
"strictInjectionParameters": true
}
}

Les versions marquantes Angular

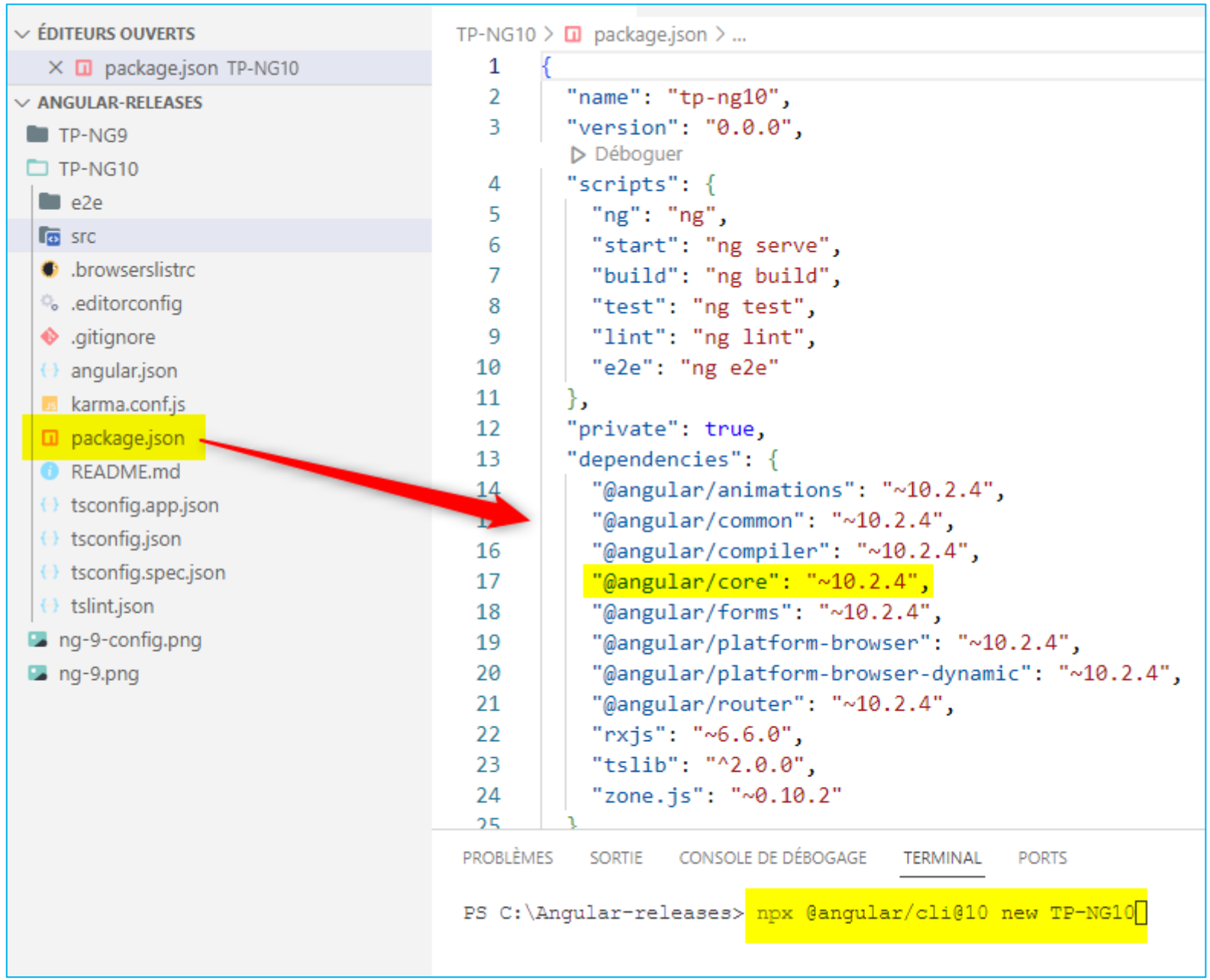
•Version 10 (24 juin 2020)

- Mode strict (typage obligatoire)
  ng new TP01 --strict

Rappels :

- basic : pas de vérification de types
- full : les props utilisées dans la Vue doivent être correctement définies dans le TS

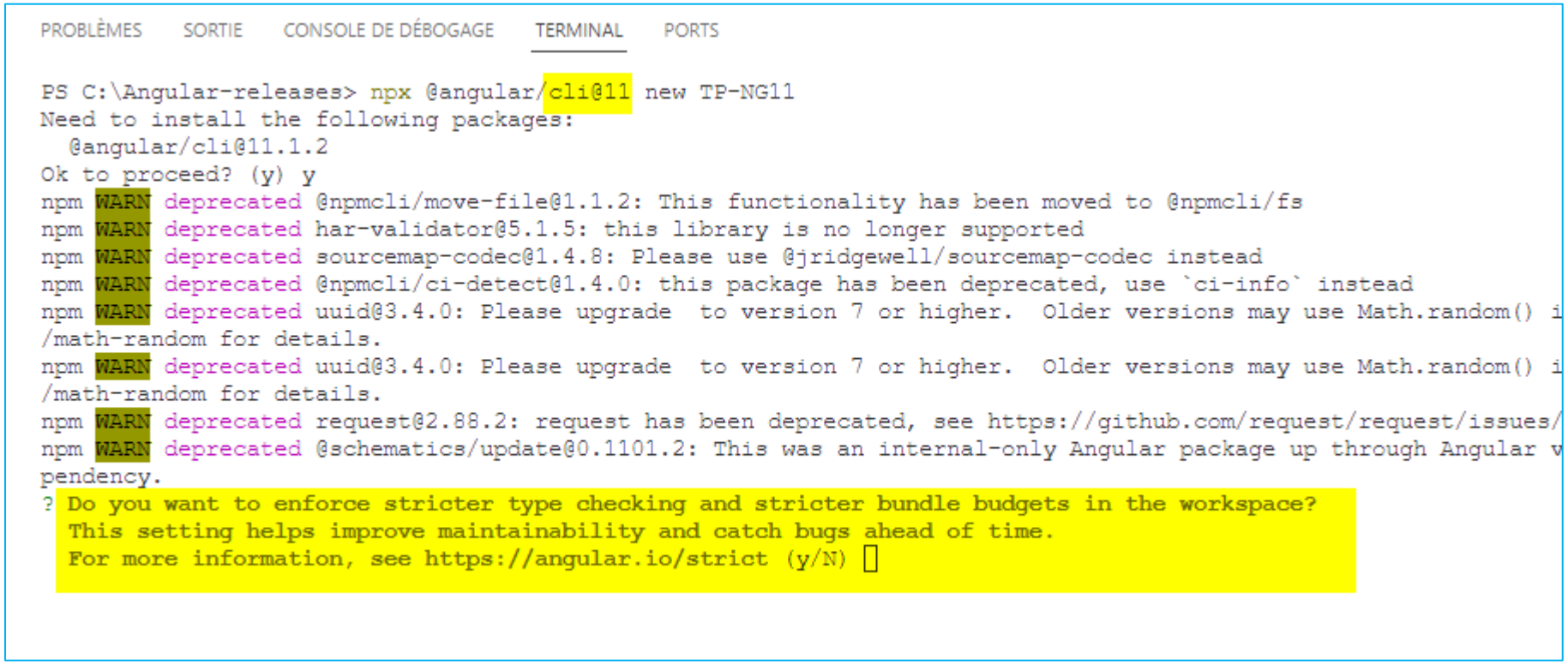
- nouveau datePicker dans Angular Matérial


ÉDITEURS OUVERTS
package.json TP-NG10
ANGULAR-RELEASES
TP-NG9
TP-NG10
e2e
src
.browserslistrc
.editorconfig
.gitignore
angular.json
karma.conf.js
package.json
README.md
tsconfig.app.json
tsconfig.json
tsconfig.spec.json
tslint.json
ng-9-config.png
ng-9.png
TP-NG10 > package.json > ...
{
"name": "tp-ng10",
"version": "0.0.0",
Déboguer
"scripts": {
"ng": "ng",
"start": "ng serve",
"build": "ng build",
"test": "ng test",
"lint": "ng lint",
"e2e": "ng e2e"
},
"private": true,
"dependencies": {
"@angular/animations": "~10.2.4",
"@angular/common": "~10.2.4",
"@angular/compiler": "~10.2.4",
"@angular/core": "~10.2.4",
"@angular/forms": "~10.2.4",
"@angular/platform-browser": "~10.2.4",
"@angular/platform-browser-dynamic": "~10.2.4",
"@angular/router": "~10.2.4",
"rxjs": "~6.6.0",
"tslib": "^2.0.0",
"zone.js": "~0.10.2"
}
PROBLÈMES
SORTIE
CONSOLE DE DÉBOGAGE
TERMINAL
PORTS
PS C:\Angular-releases> npx @angular/cli@10 new TP-NG10

Les versions marquantes Angular

•Version 11 (17 novembre 2020)

- Mode strict est proposé lors de la création de l'appli



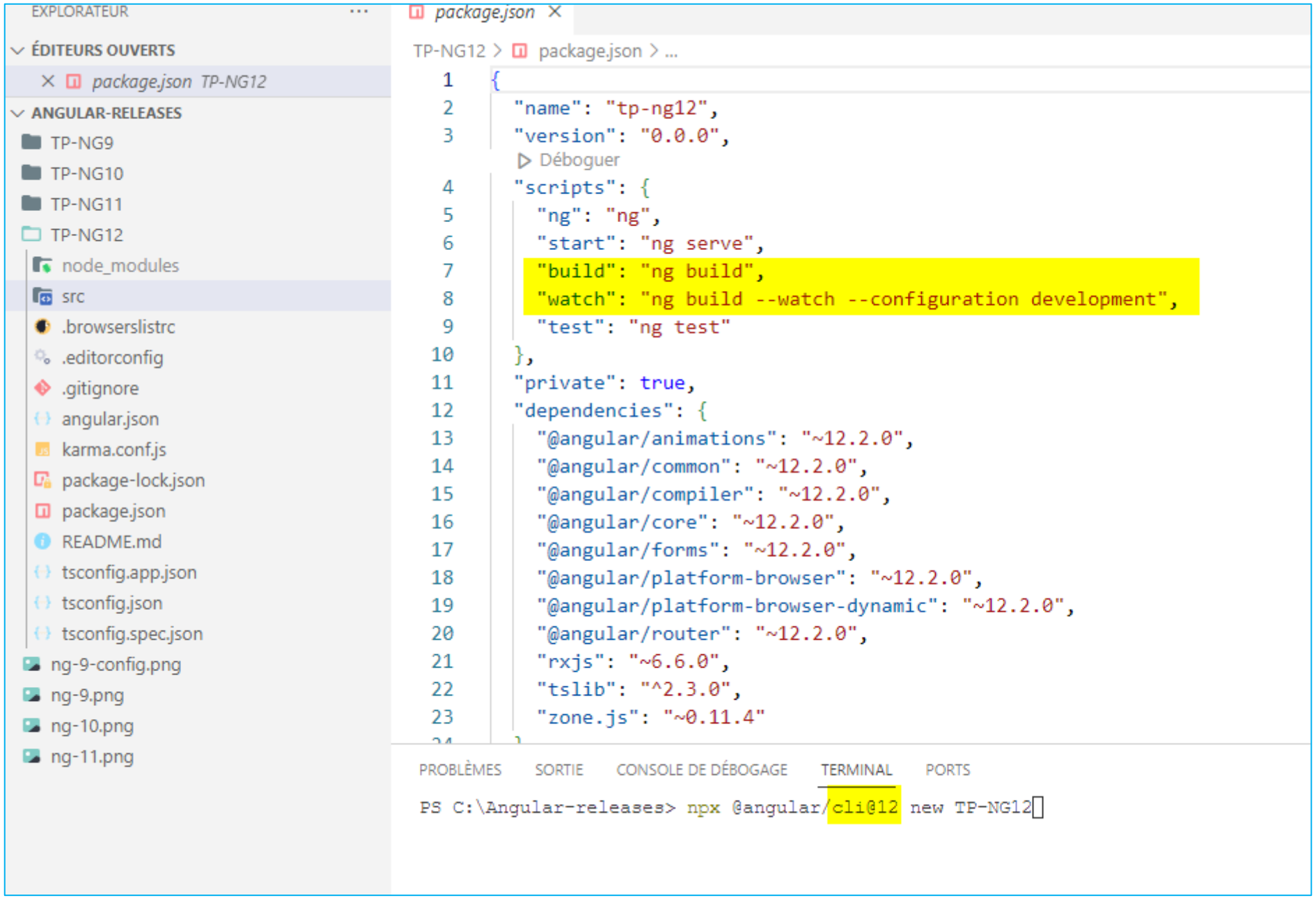
Les versions marquantes Angular

•Version 12 (19 mai 2021)

- Abandon progressif de la compatibilité et support IE

validé avec NG13

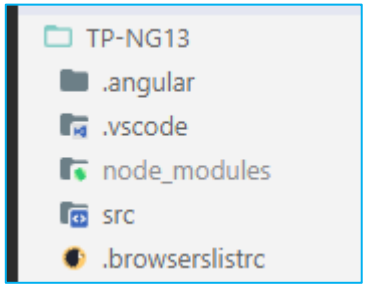
- Fin du support de e2e, protractor..
- TSlint déprécié depuis 2019 est remplacé par ESlint
- script watch qui remplace et améliore build
- build est désormais le mode prod


EXPLORATEUR
ÉDITEURS OUVERTS
package.json TP-NG12
ANGULAR-RELEASES
TP-NG9
TP-NG10
TP-NG11
TP-NG12
node_modules
src
.browserslistrc
.editorconfig
.gitignore
angular.json
karma.conf.js
package-lock.json
package.json
README.md
tsconfig.app.json
tsconfig.json
tsconfig.spec.json
ng-9-config.png
ng-9.png
ng-10.png
ng-11.png
package.json
TP-NG12 > package.json > ...
{
"name": "tp-ng12",
"version": "0.0.0",
Déboguer
"scripts": {
"ng": "ng",
"start": "ng serve",
"build": "ng build",
"watch": "ng build --watch --configuration development",
"test": "ng test"
},
"private": true,
"dependencies": {
"@angular/animations": "~12.2.0",
"@angular/common": "~12.2.0",
"@angular/compiler": "~12.2.0",
"@angular/core": "~12.2.0",
"@angular/forms": "~12.2.0",
"@angular/platform-browser": "~12.2.0",
"@angular/platform-browser-dynamic": "~12.2.0",
"@angular/router": "~12.2.0",
"rxjs": "~6.6.0",
"tslib": "^2.3.0",
"zone.js": "~0.11.4"
PROBLÈMES
SORTIE
CONSOLE DE DÉBOGAGE
TERMINAL
PORTS
PS C:\Angular-releases> npx @angular/cli@12 new TP-NG12

Les versions marquantes Angular

•Version 13 (3 novembre 2021)

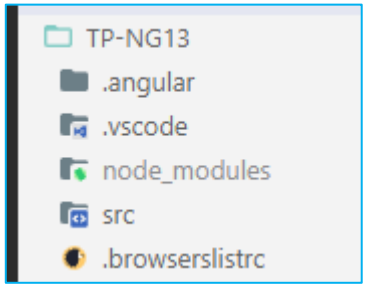
- Arrivée du cache .angular => accélération des temps de compilation
- démarrage en mode dév plus rapide
- Travail continuel d'optimisation de IVY et ViewEngine



Les versions marquantes Angular

•Version 14 (11 juin 2022)

- FormControl enfin typé
- composants standalone (*validé avec NG15*)
- optimisation des messages d'erreurs
- erreur banana in a box [()] ou ([])



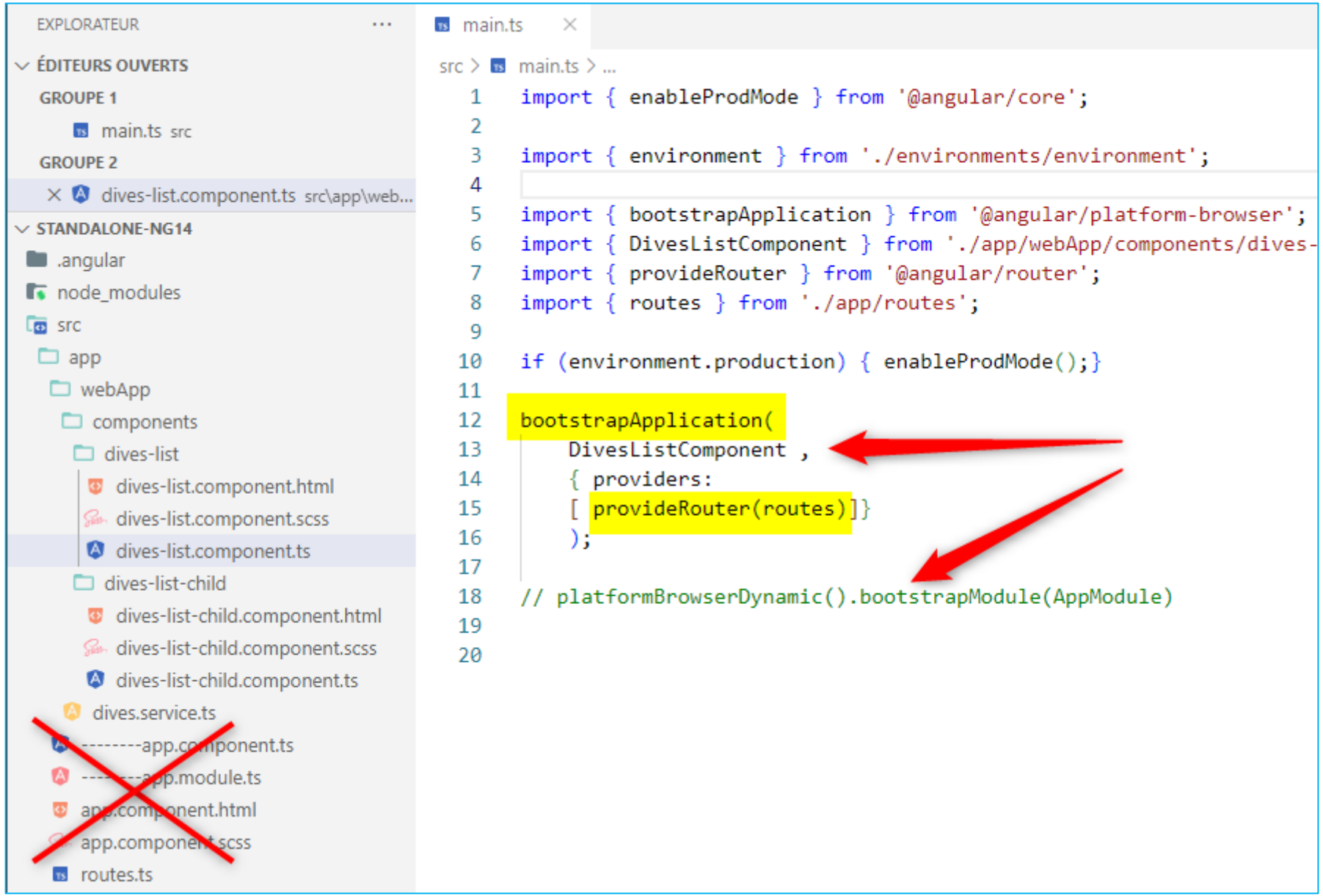
Les versions marquantes Angular

•Version 15 (16 novembre 2022)

- composants standalone

  Prop : standalone:true
- Omission des ngModules ?
- API autonomes ?

```
@Component({
  selector: 'app-root',
  // **********************
  standalone: true,
  templateUrl: './dives-list.component.html',
  styleUrls: ['./dives-list.component.scss'],
  // **********************
  imports: [
    // les autres composants doivent être également en component standalone !!
    DivesListChildComponent,
    // les pipes - les directives
    FilterPipe,
    // les modules
    CommonModule, HttpClientModule, RouterModule, FormsModule
  ],
  providers:[
    DivesService
  ]
})
```

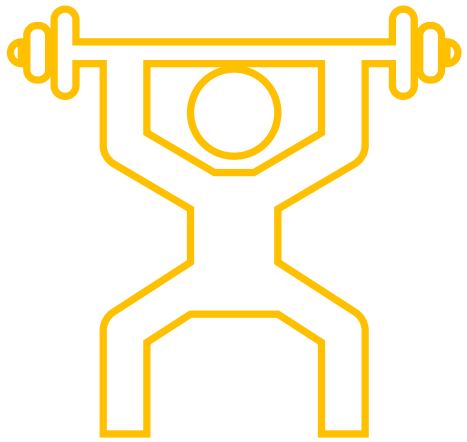

EXPLORATEUR
ÉDITEURS OUVERTS
GROUPE 1
main.ts src
GROUPE 2
dives-list.component.ts src\app\web...
STANDALONE-NG14
.angular
node_modules
src
app
webApp
components
dives-list
dives-list.component.html
dives-list.component.scss
dives-list.component.ts
dives-list-child
dives-list-child.component.html
dives-list-child.component.scss
dives-list-child.component.ts
dives.service.ts
app.component.ts
app.module.ts
app.component.html
app.component.scss
routes.ts
main.ts
src > main.ts > ...
1 import { enableProdMode } from '@angular/core';
2
3 import { environment } from './environments/environment';
4
5 import { bootstrapApplication } from '@angular/platform-browser';
6 import { DivesListComponent } from './app/webApp/components/dives-
7 import { provideRouter } from '@angular/router';
8 import { routes } from './app/routes';
9
10 if (environment.production) { enableProdMode();}
11
12 bootstrapApplication(
13 DivesListComponent ,
14 { providers:
15 [provideRouter(routes)]}
16);
17
18 // platformBrowserDynamic().bootstrapModule(AppModule)
19
20

Les versions marquantes Angular

- Version 16 (12 mai 2023)

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

Le Detection Change Mode
Nouveautés NG16 : Signal



Le Detection Change Mode Angular

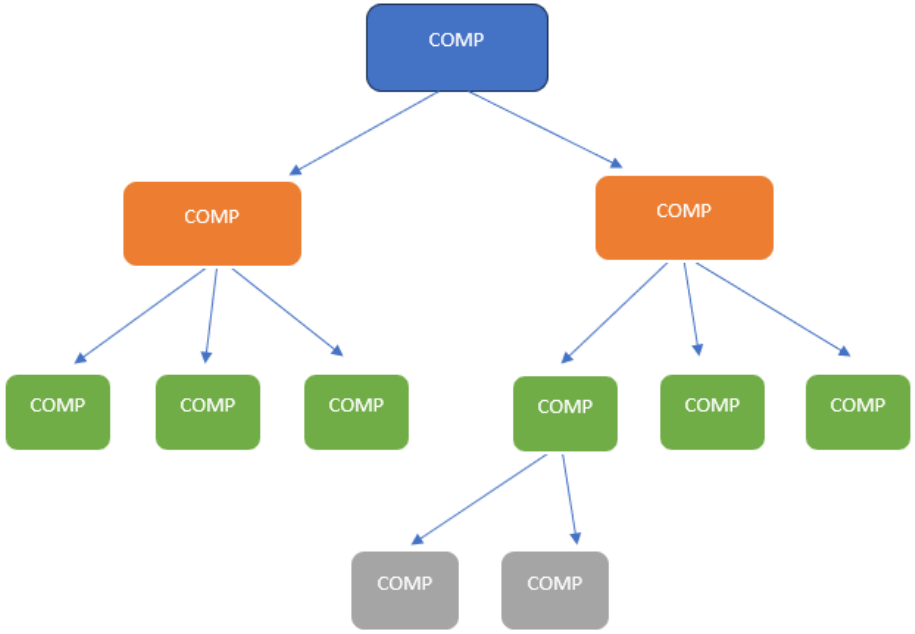
Lorsque des propriétés d'entrée (Input) ou des événements d'entrée sont déclenchés, Angular met à jour l'arbre des composants pour détecter d'éventuels changements à apporter aux affichages des Vues.

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

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

Le Detection Change Mode Angular

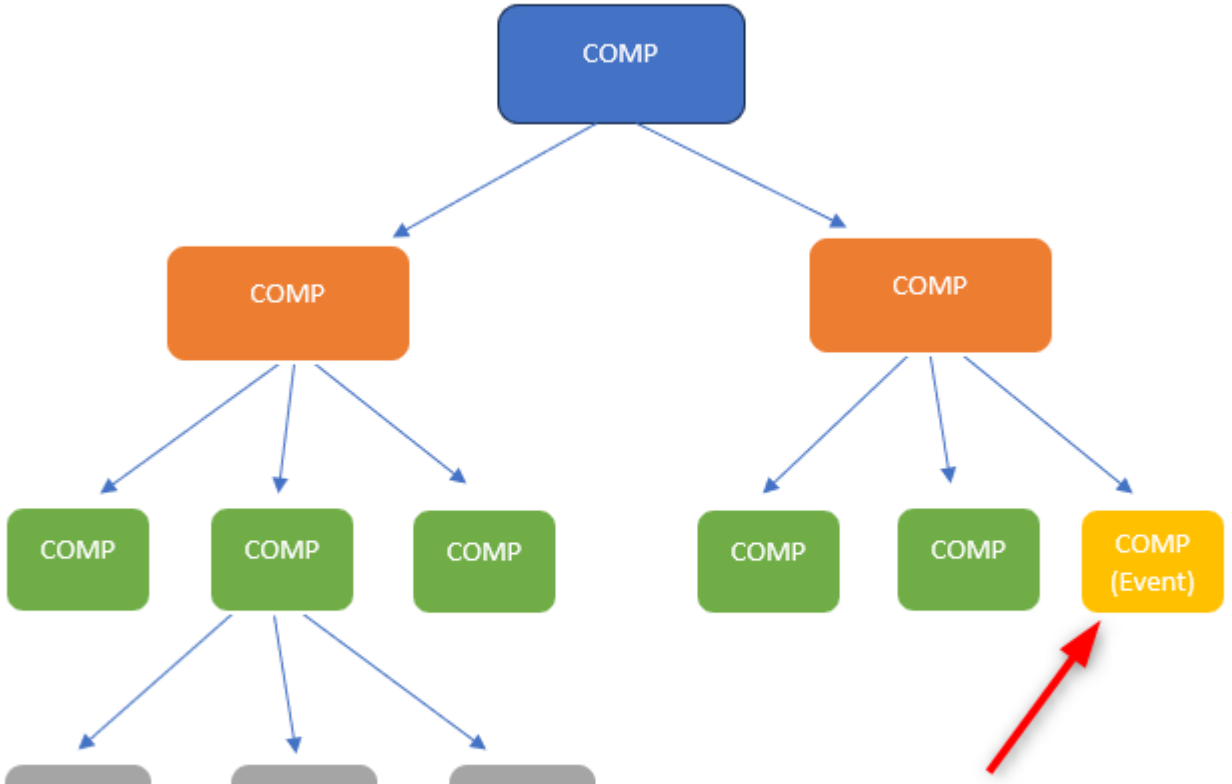
Comment Angular met à jour l'affichage des composants imbriqués par des Inputs,
dans une stratégie composants parents-enfants



Le Detection Change Mode par défaut

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

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



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

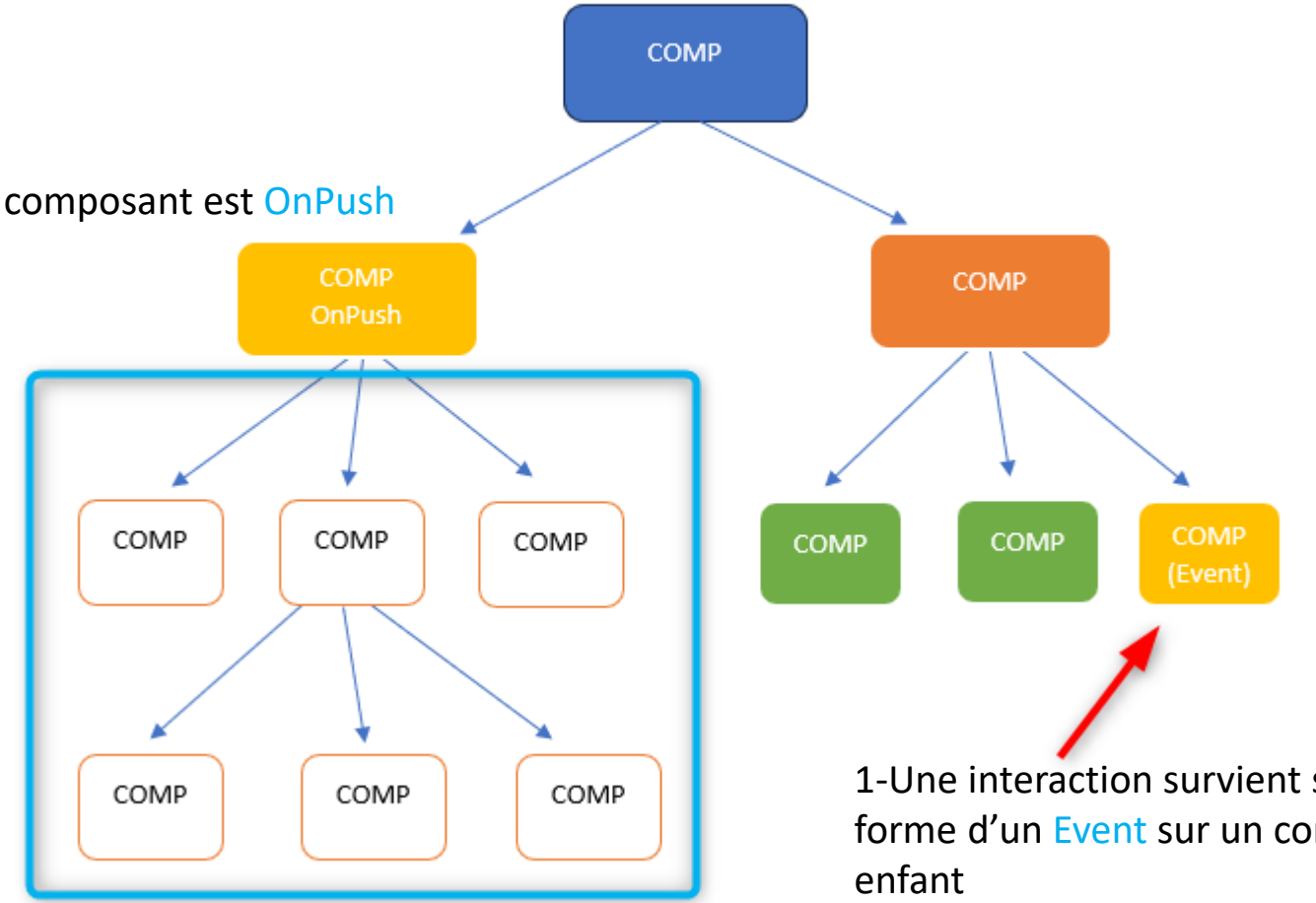
Le Detection Change Mode onPush

Angular limite le nombre de vérifications sur les enfants

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

Angular arrête son parcours…

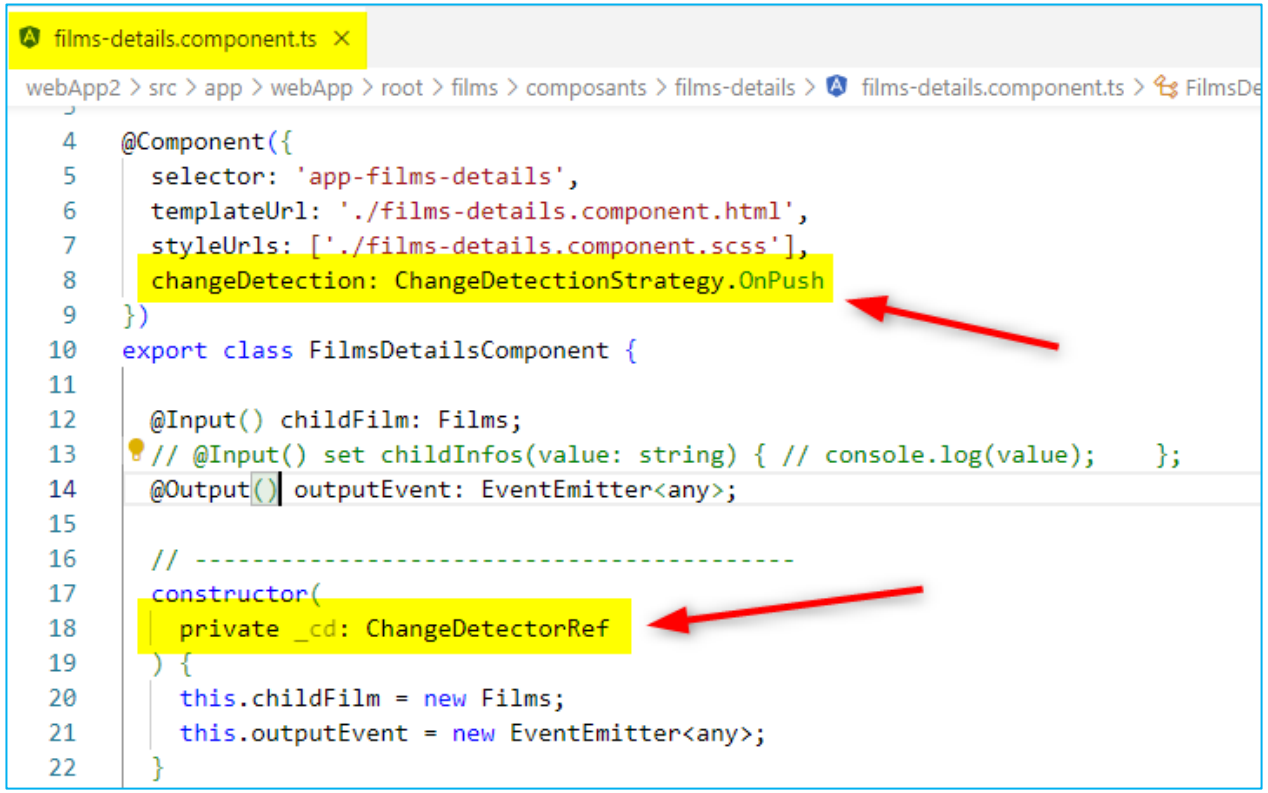
Le composant est OnPush



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

La class ChangeDetetctorRef

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



La class ChangeDetetctorRef

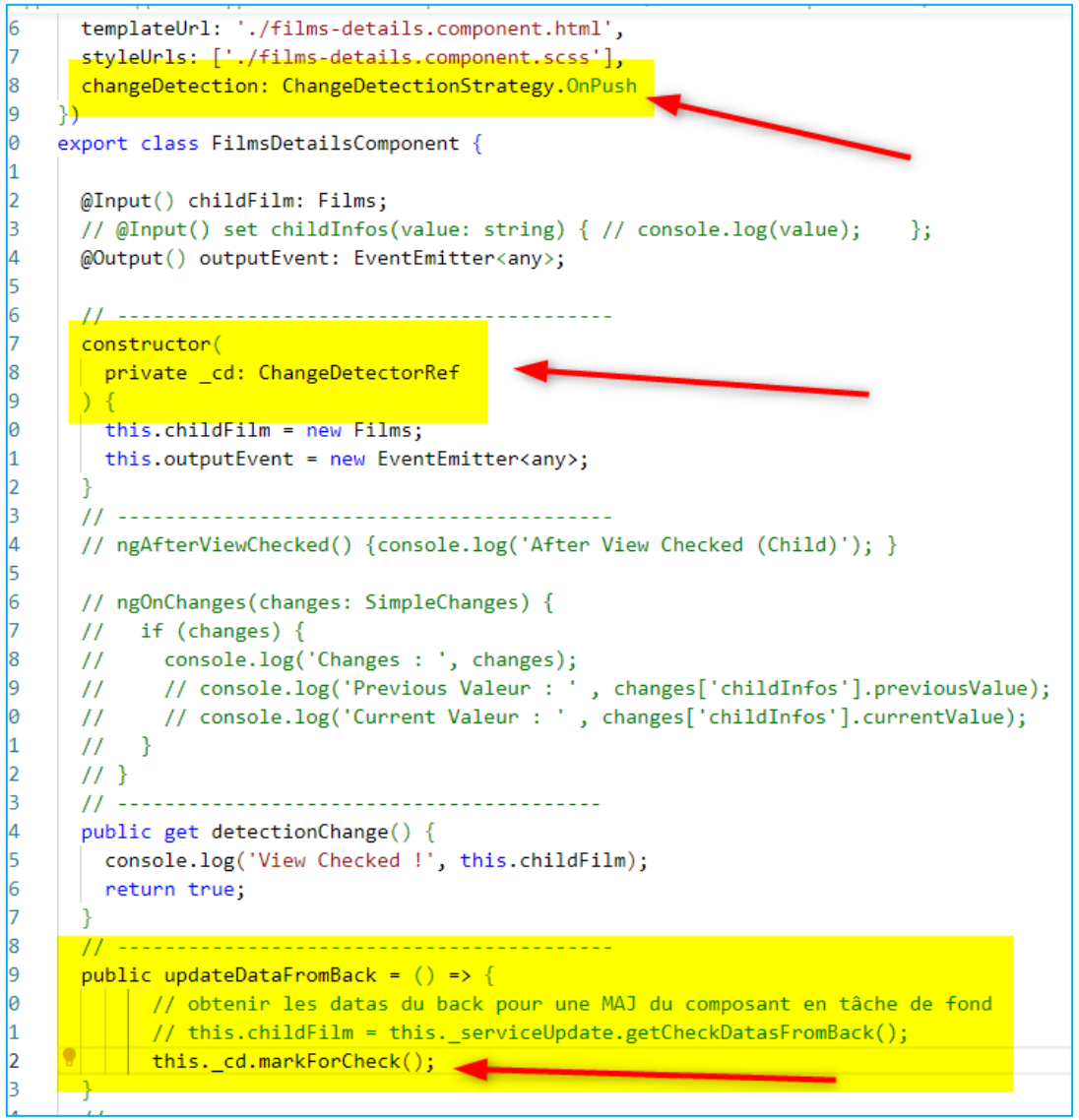
« markForCheck » indique que le composant doit être vérifié lors du prochain parcours de l'arbre des composants, même si la stratégie de mise à jour de « change detection » est OnPush

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

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

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

checkNoChanges
detach
detectChanges
markForCheck
reattach

La class ChangeDetetctorRef



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

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

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

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

Le Signal

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

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

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

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

toSignal() toObservable()

Le Signal

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

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

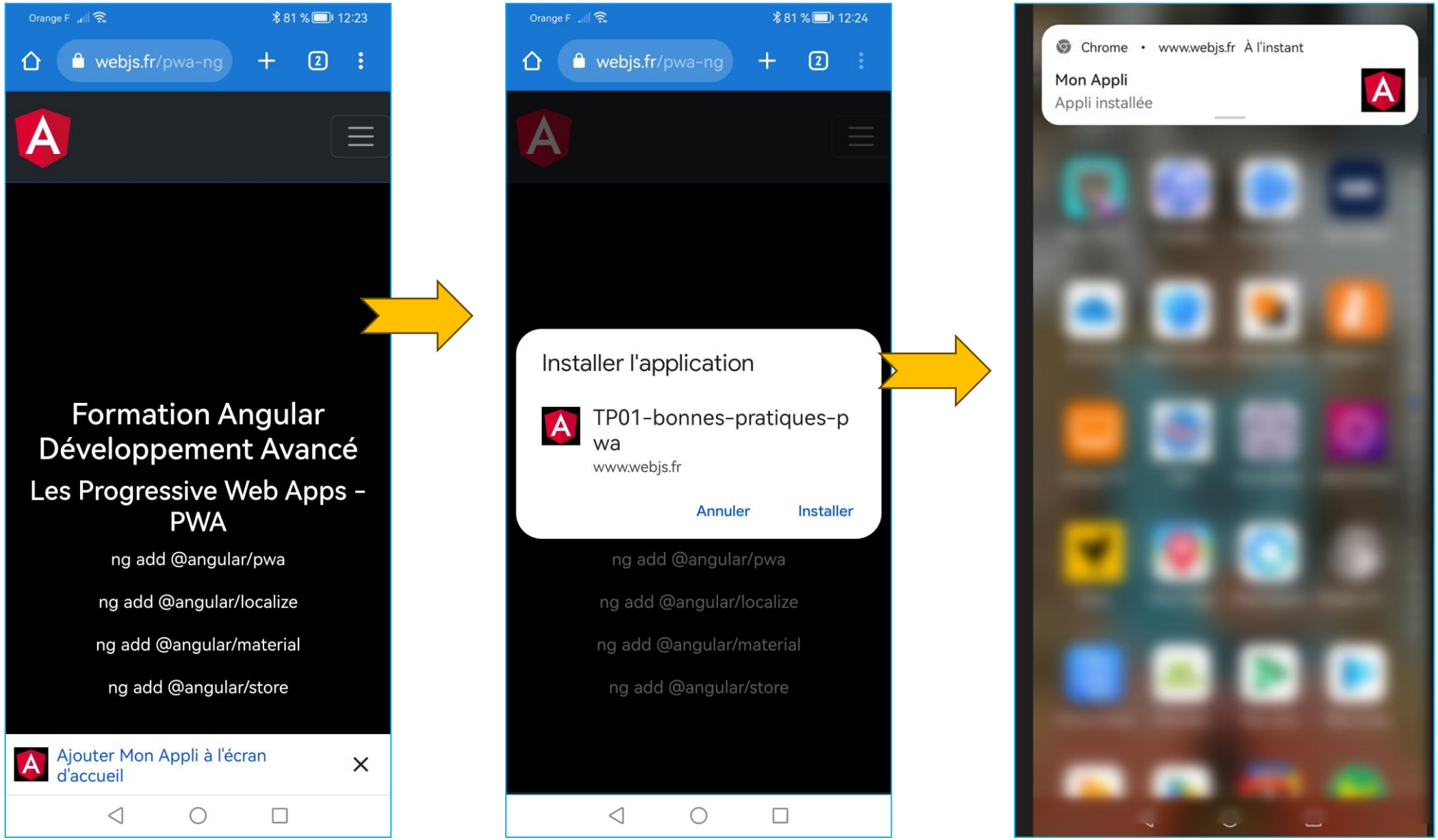
Lorsque une valeur change, le signal émet une information et l’application peut se mettre à jour de manière sélective.

PWA – les Progressive Web Apps

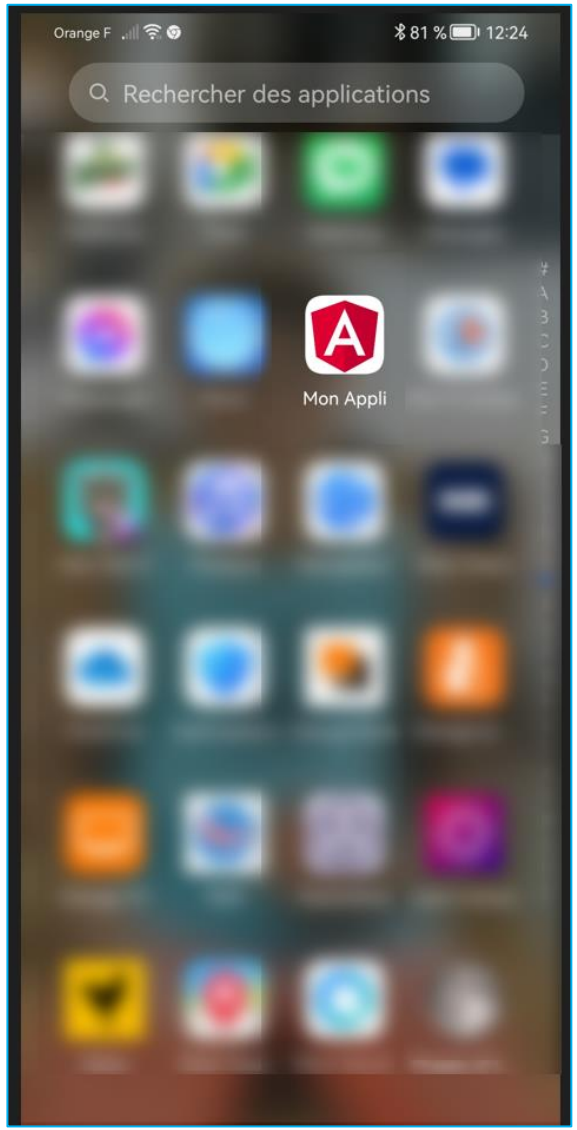
Mise en situation :
consulter cette appli depuis l'URL : https://webjs.fr/pwa-ng/

ng add @angular/pwa

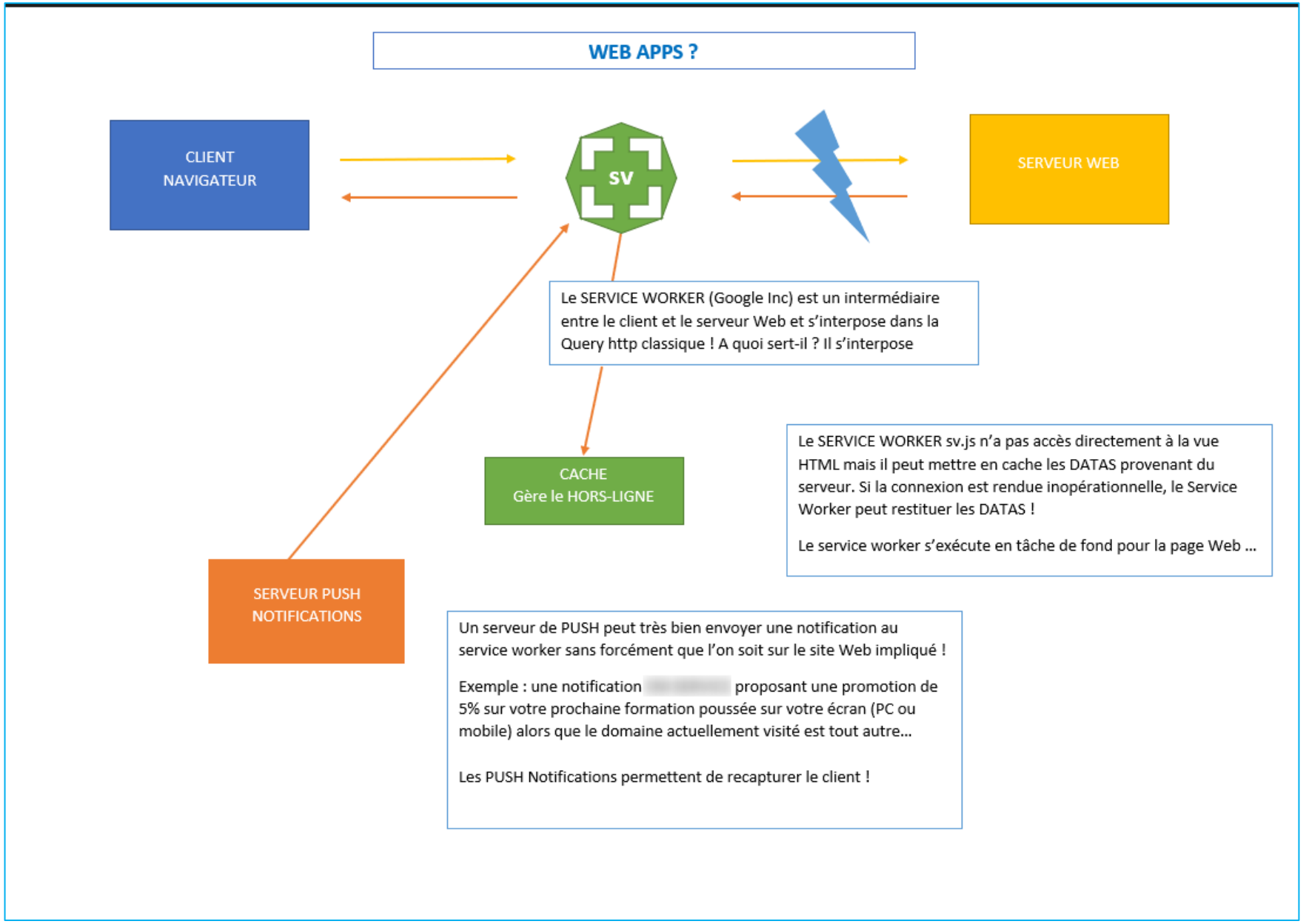
PWA – les Progressive Web Apps



PWA – les Progressive Web Apps



PWA – les Progressive Web Apps



PWA – les Progressive Web Apps