ORSYS formation

Formation ANGULAR

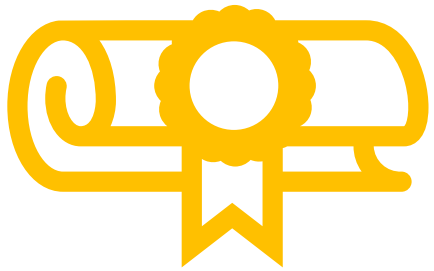
Chapitre 2

Animé par Michel BOCCIOLESI

Table des matières

Récupérer les sources du projet de formation ici

Reactive eXtensions for JavaScript
Les Observables



RXJS et les OBSERVABLES

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

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

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

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

RXJS et les OBSERVABLES

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

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

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

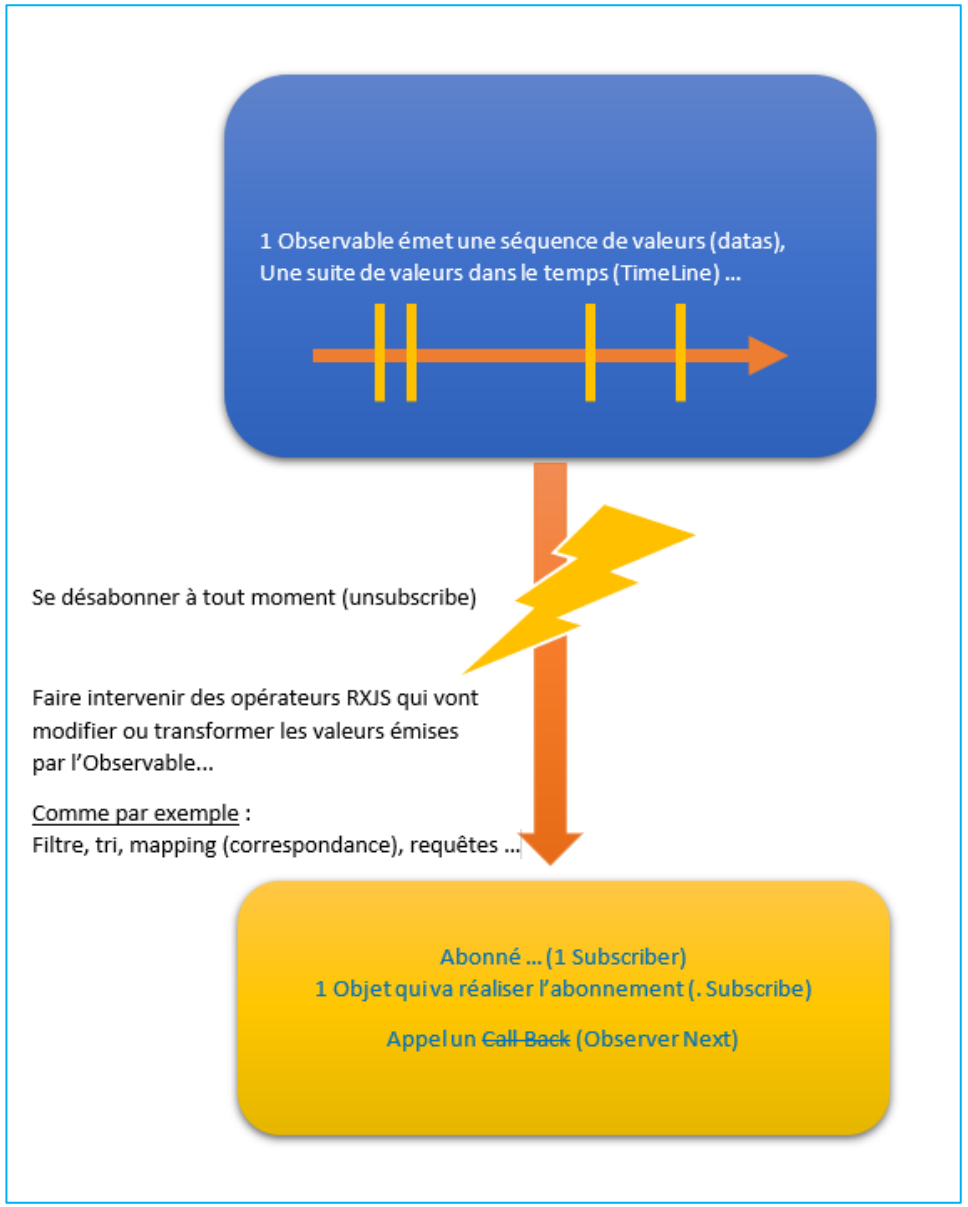
Opérateurs : des fonctions tap, map, merge, filter qui vont pouvoir modifier/transformer ce flux de valeurs émises par l'observable

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

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

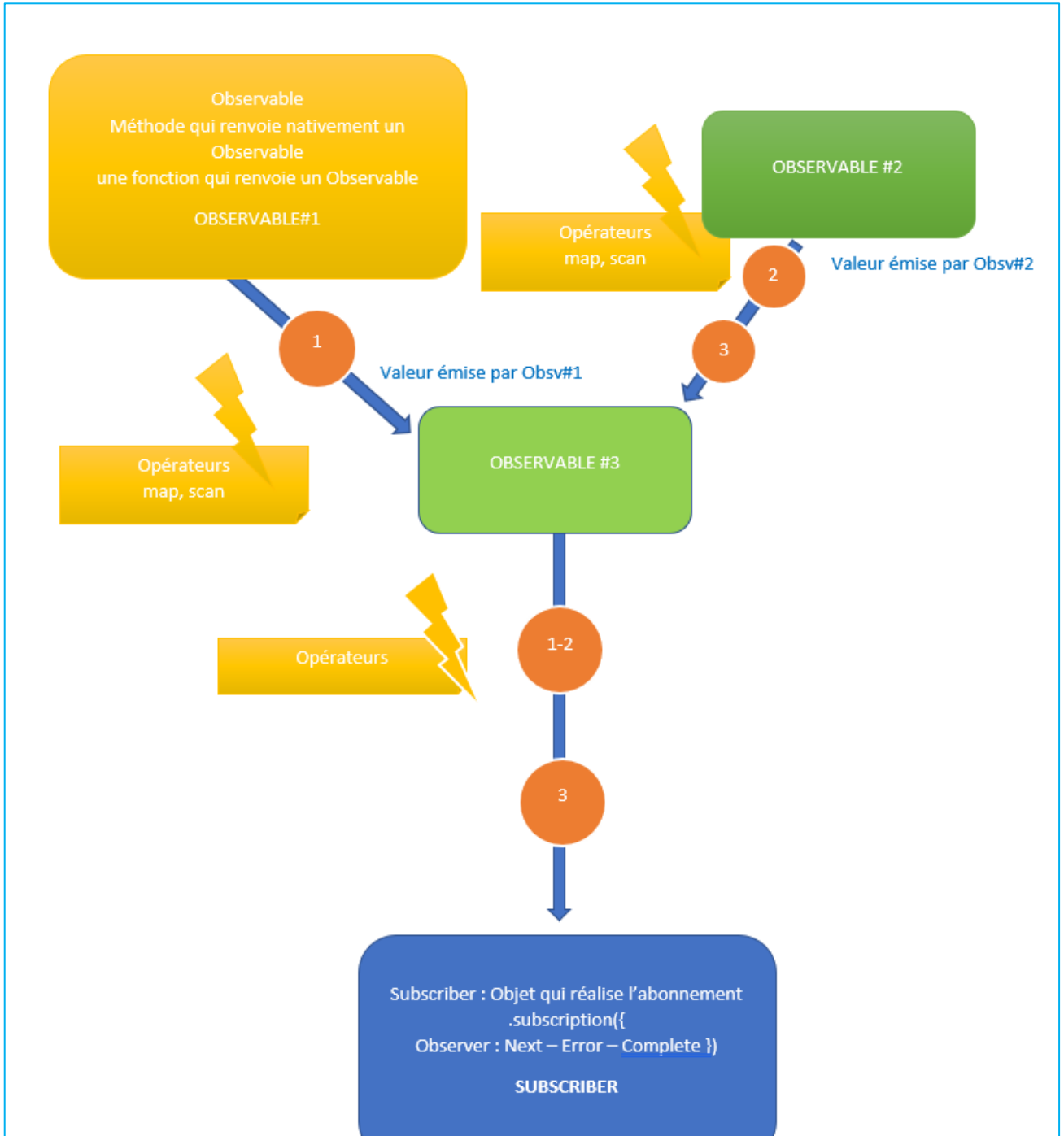
Les séquences de valeurs émises sont facilement annulables !

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

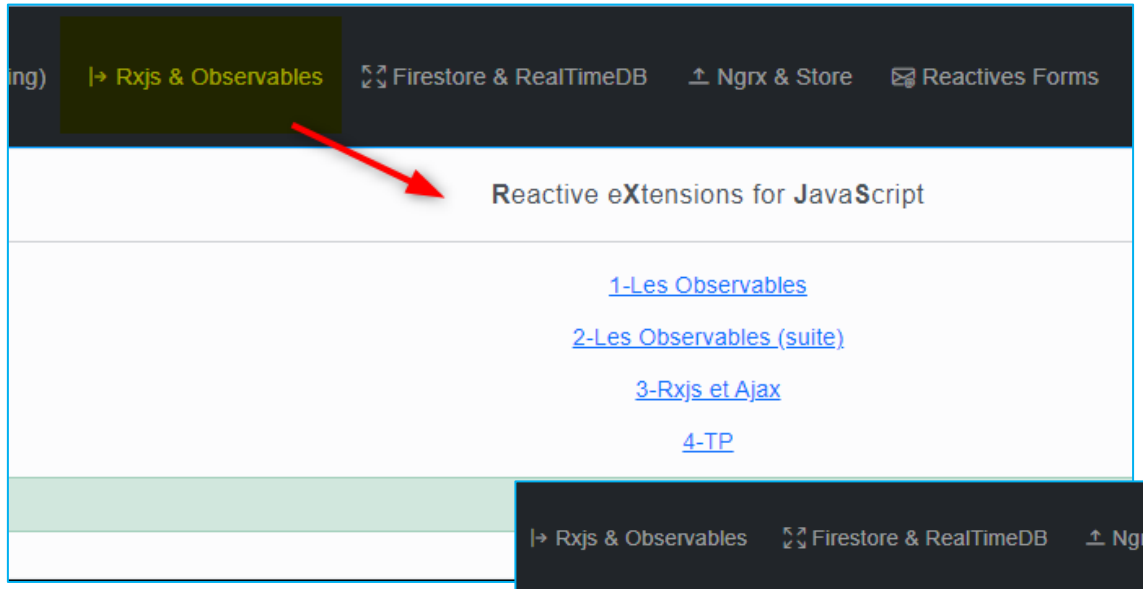


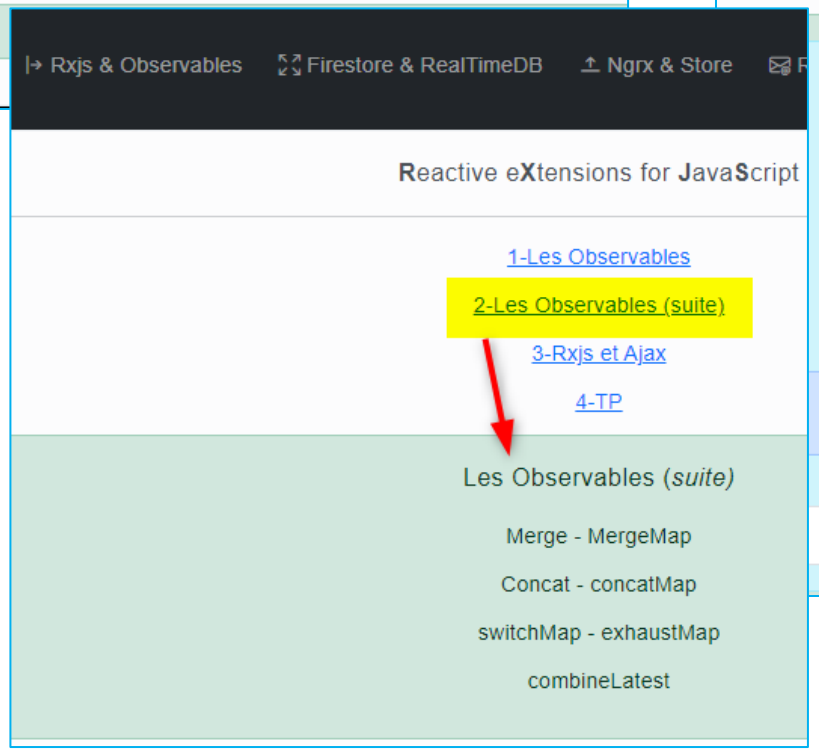
RXJS et les OBSERVABLES

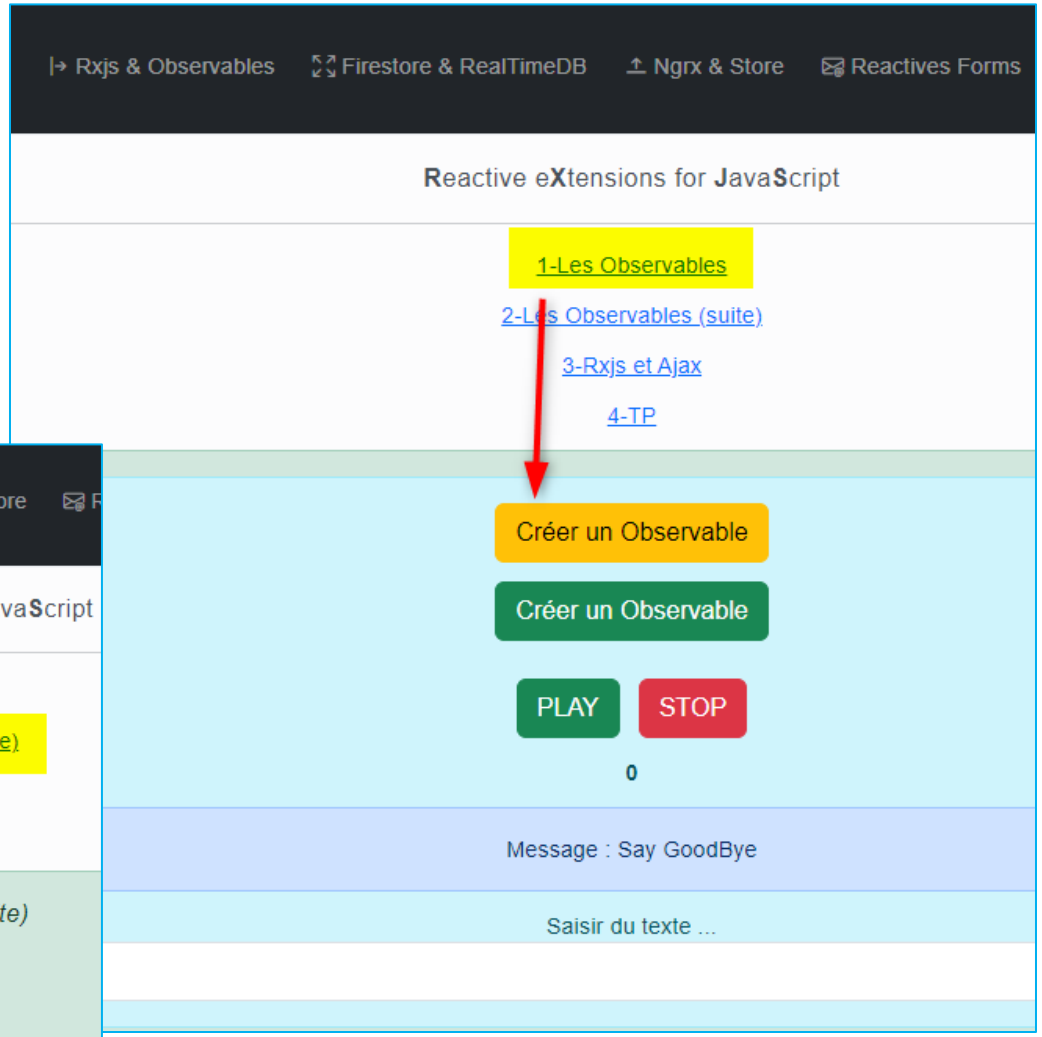
On peut facilement les recomposer
ou les recombiner pour former
une nouvelle séquence de valeurs !



TP : déployer l'arborescence en Lazy Loading RXJS





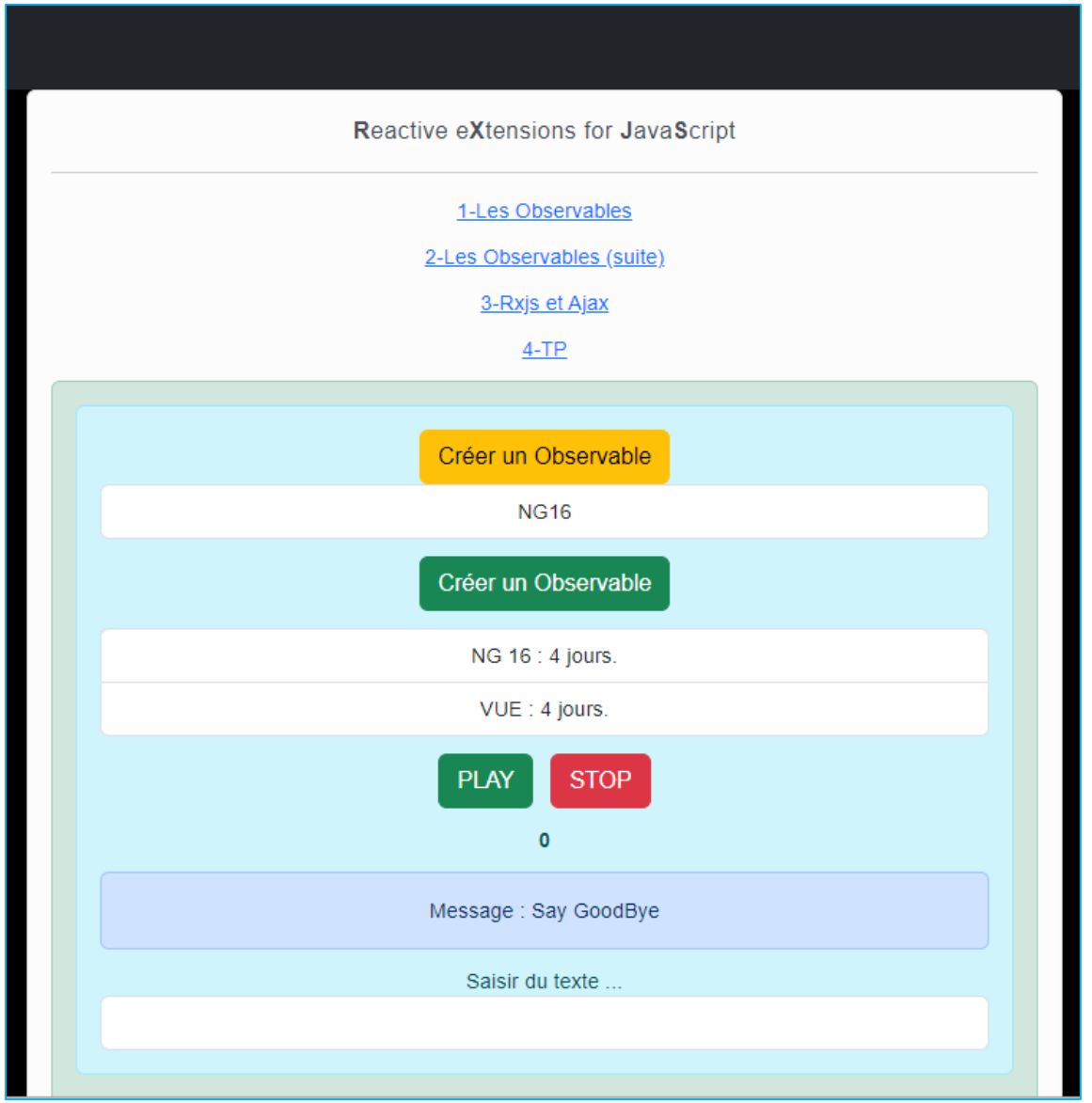


Exercice #1

observables.component.html

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

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

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

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

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

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

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

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



Exercice #1

observables.component.ts

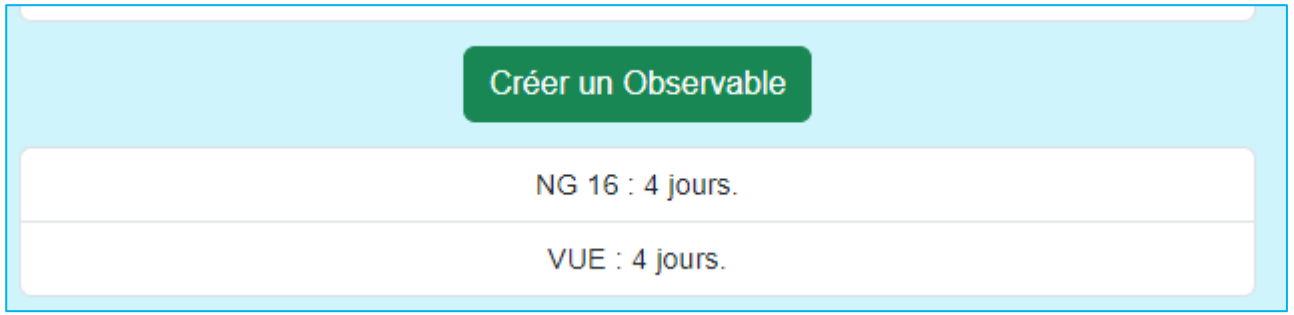
```ts
// Création de l'observable
const formation$: Observable<string> = from(formationsArray);
console.log(formation$);

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

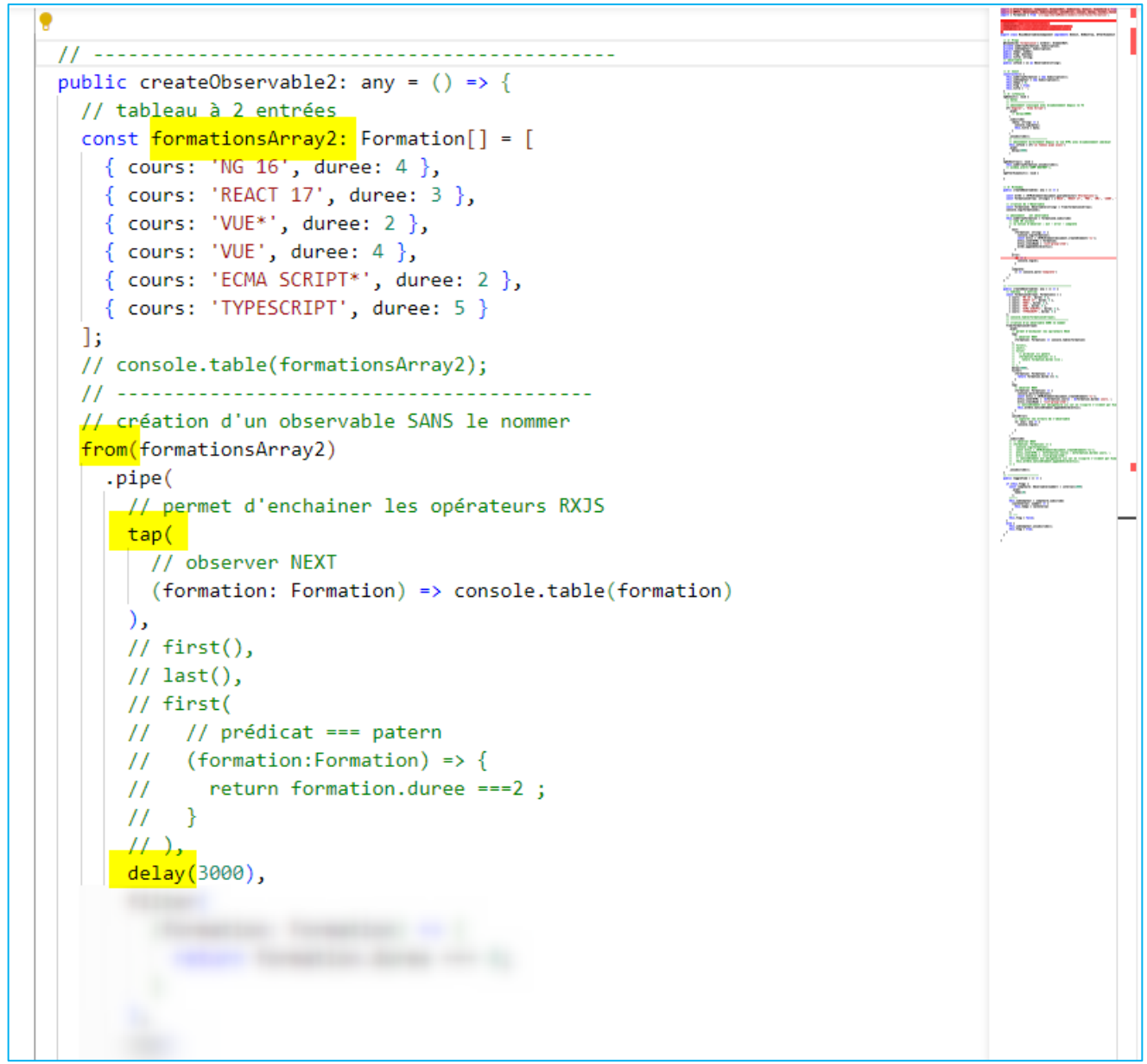
Exercice #1

observables.component.ts

TP : créer l'observable associée à la méthode createObservable2

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





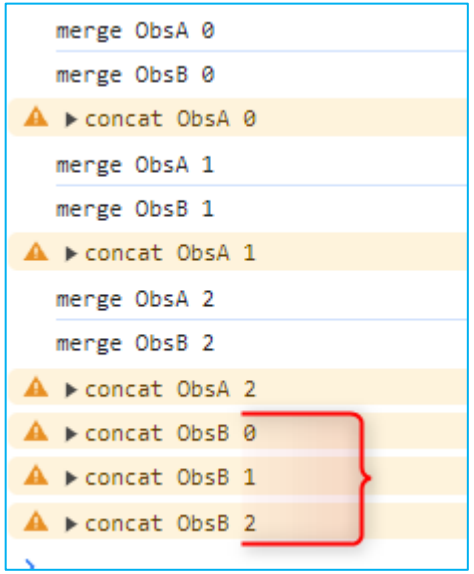
```
// ---------------------------------------------
public createObservable2: any = () => {
  // tableau à 2 entrées
  const formationsArray2: Formation[] = [
    { cours: 'NG 16', duree: 4 },
    { cours: 'REACT 17', duree: 3 },
    { cours: 'VUE*', duree: 2 },
    { cours: 'VUE', duree: 4 },
    {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

Exercice #2

Observables-suite.component

Les fonctions de regroupement et de recombinaison de séquences

```
ngOnInit() {

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

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

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

  concat(obsservableA$, obsservableB$)
    // émission  après complete
    .subscribe(
      (valObs) => console.warn('concat', valObs)
    );

}
```



Exercice #2

Observables-suite.component

Les fonctions de regroupement et de recombinaison de séquences

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

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

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

Exercice #2

Observables-suite.component

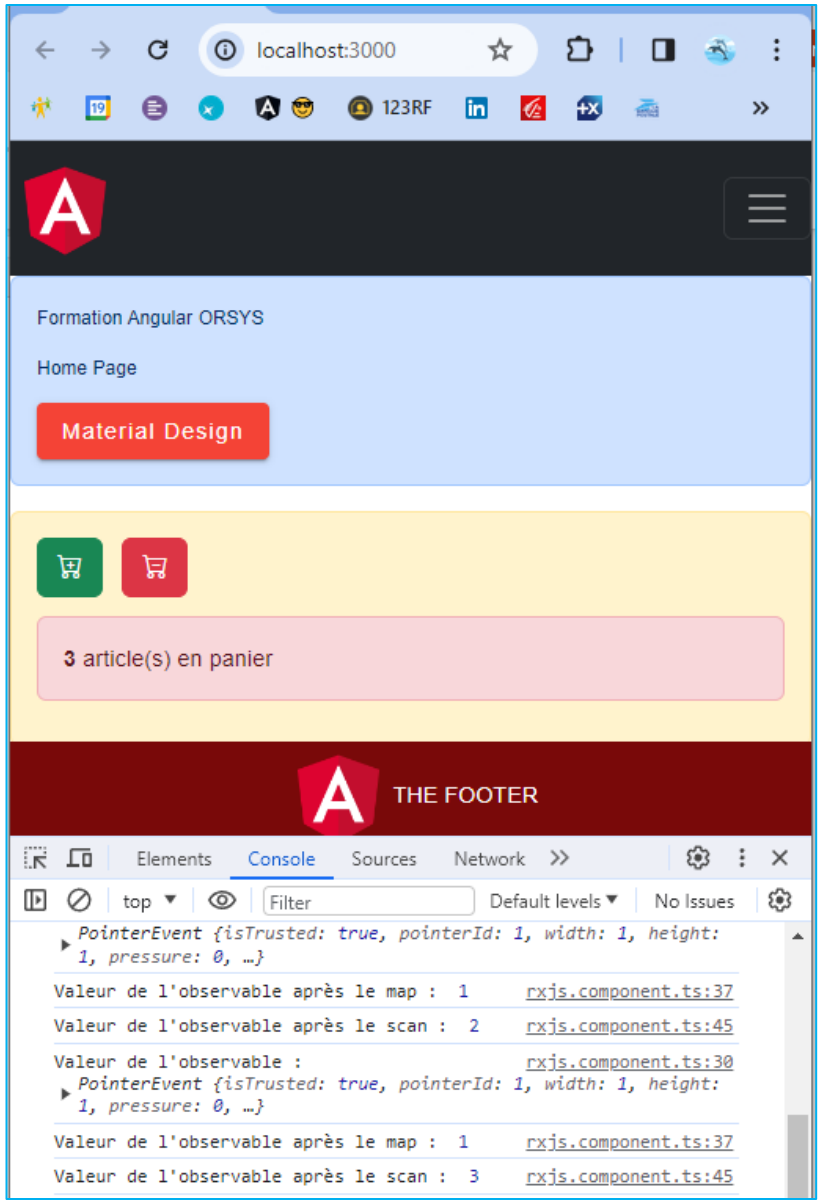
Les fonctions de regroupement et de recombinaison de séquences

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

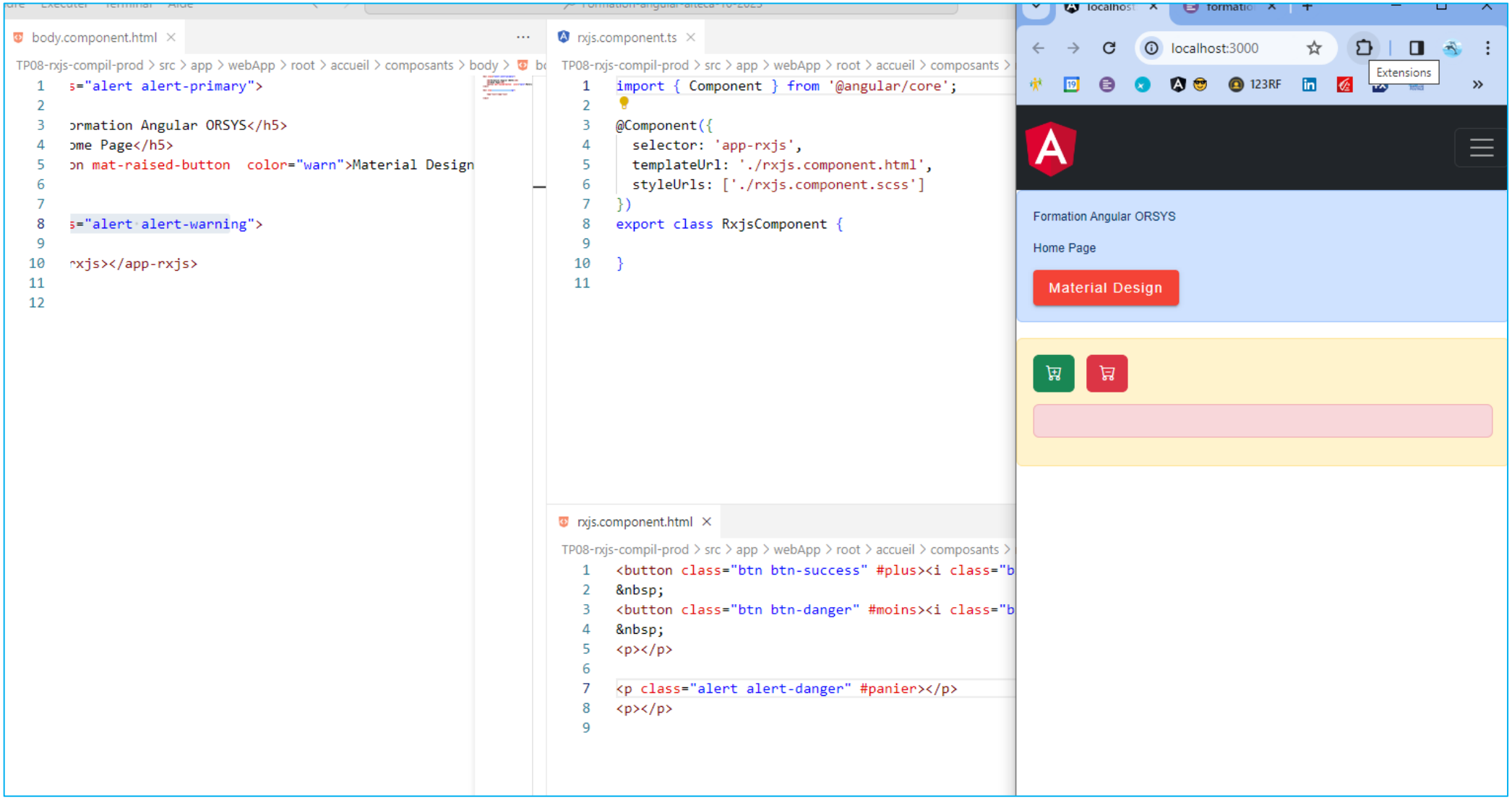
```
// ---------------------------------------------------
of(1, 2, 3)
  .pipe(
    switchMap(valObs => this._http.get<any[]>(`${API_URL}/${valObs}`))
  )
  .subscribe(
    (valObs) => console.warn('switchmap : ', valObs)
  );

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

Intéressant : les requêtes précédentes ou suivantes sont stoppées

TP (Regroupement avec combinelatest)



RXJS et les OBSERVABLES

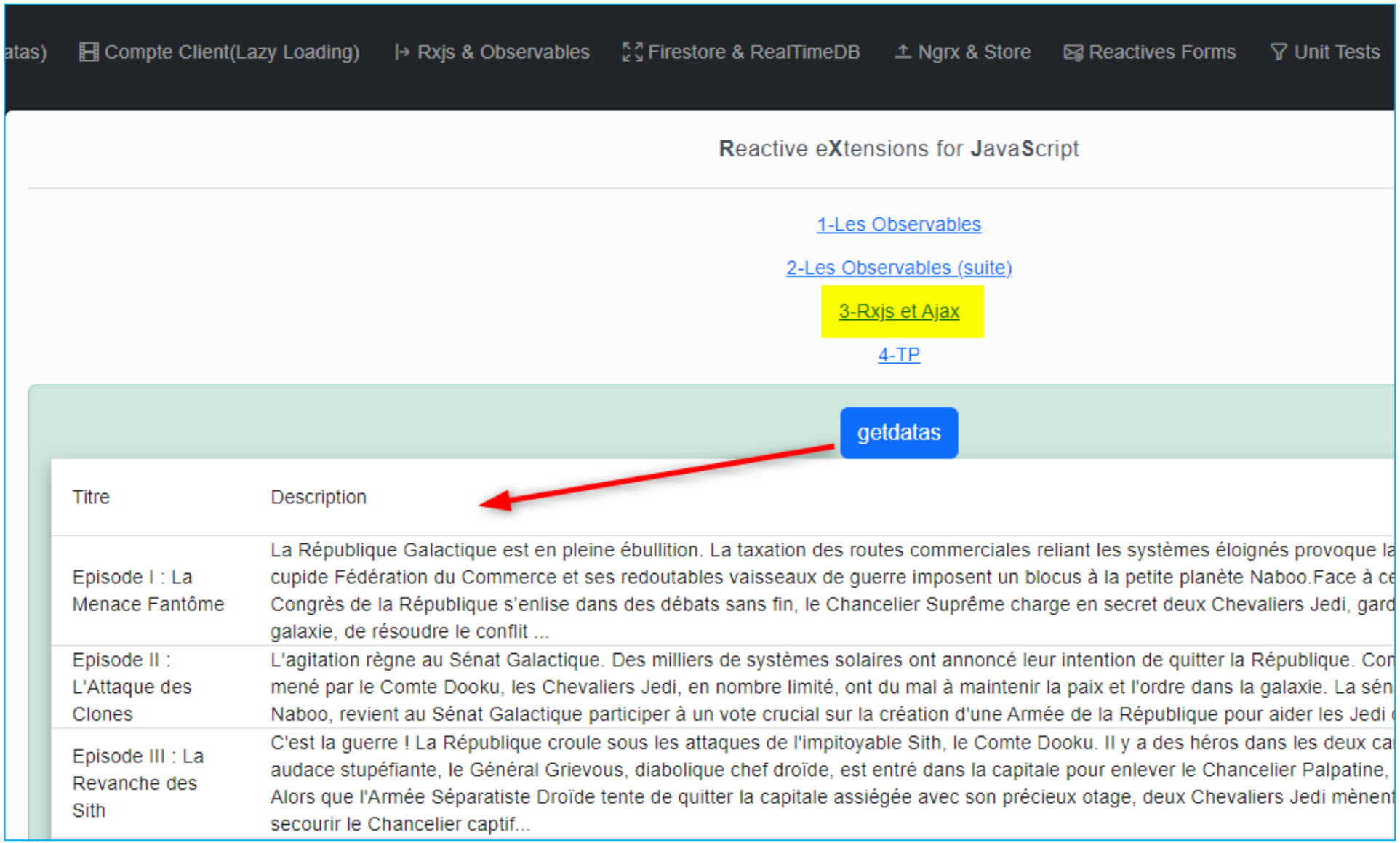


```
// ----------------------------------------
// création de l'observable #1 : PLUS
// ----------------------------------------
const plus$ = fromEvent(this.eltPlus.nativeElement, 'click')
  .pipe(
    startWith(0),
    tap(
      (valObs) => console.log(`Valeur de l'observable : `, valObs)
      //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

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

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

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

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

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

TP (Requêtes Ajax : forkJoin)



TP (Requêtes Ajax : forkJoin)

ajax.component.html | rxjs.module.ts

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

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

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

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


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


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

</table>
```

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

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

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

Les Formulaires

Les formulaires Angular (avec Matérial Design)

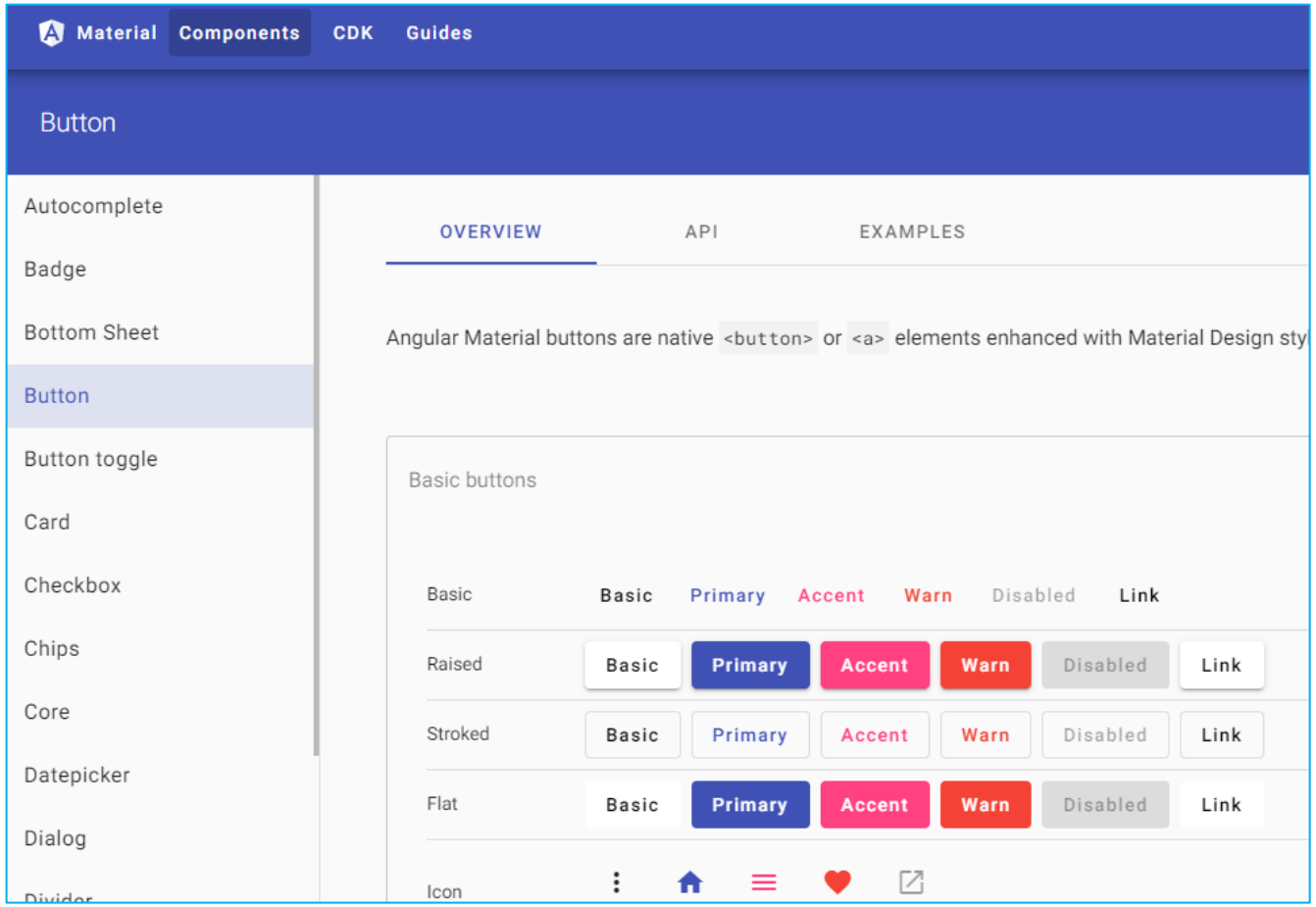
- https://material.angular.io/components/categories
- https://fonts.google.com/icons

HTML TS CSS

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

HTML TS CSS

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



Les formulaires Angular (avec Matérial Design)

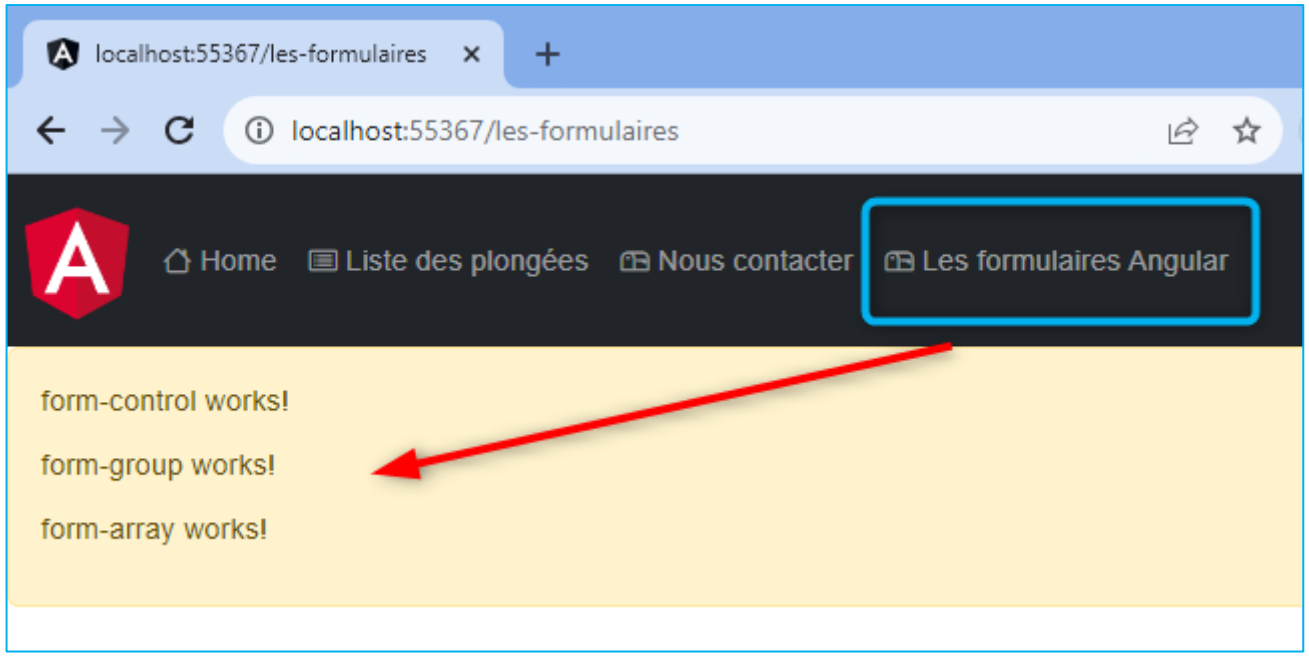
ng add @angular/material

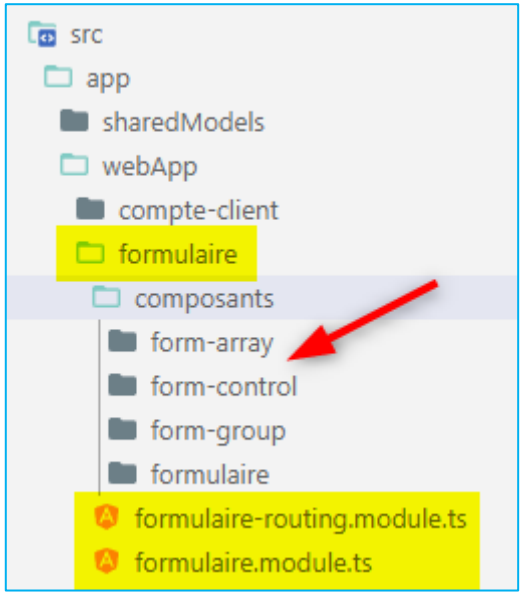
ng add @angular/localize
ng add @angular/pwa

Les formulaires Angular (avec Matérial Design)

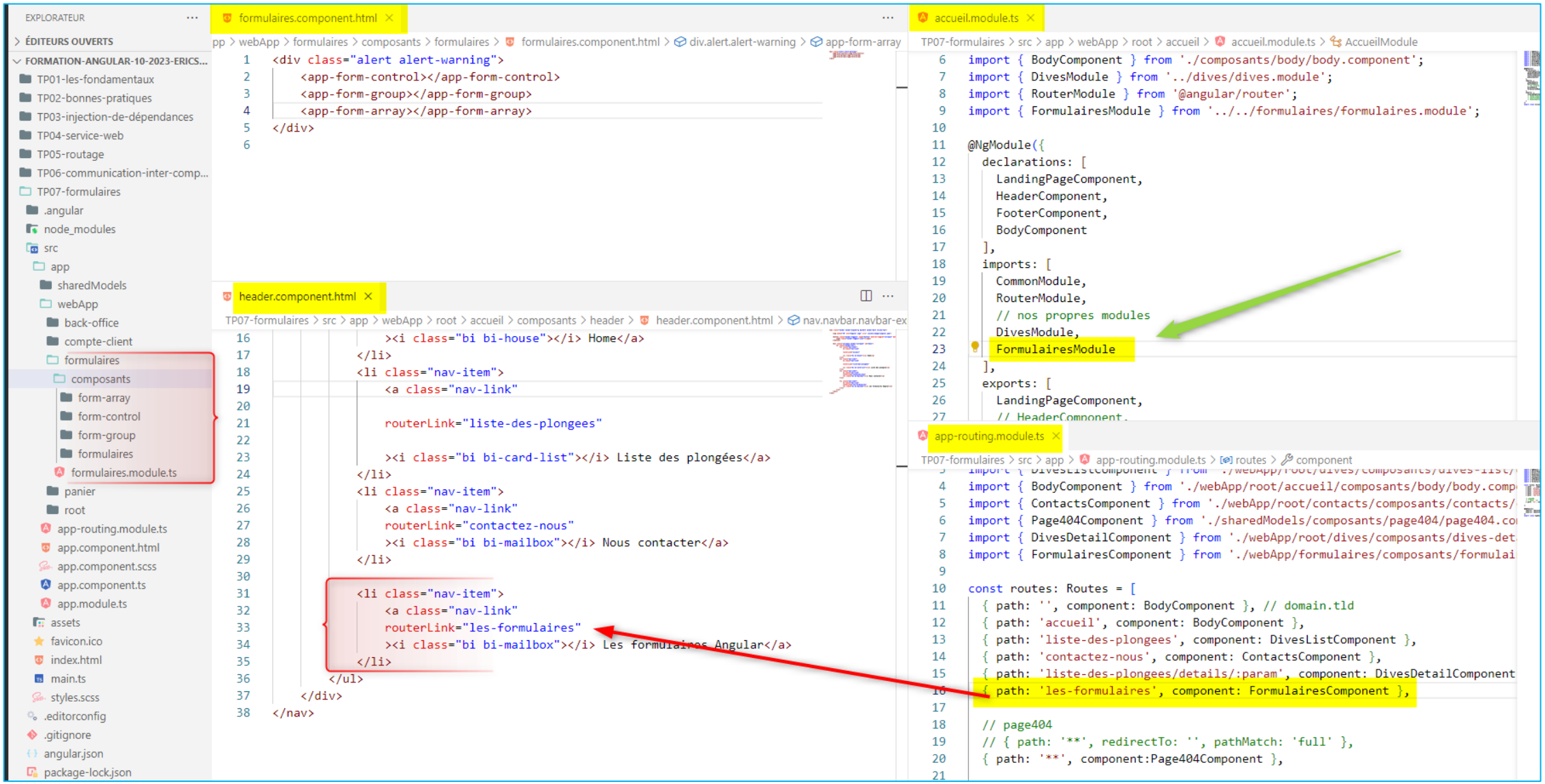
https://dev.webjs.fr/1-Angular/ressources-formation/material.module.ts
https://dev.webjs.fr/1-Angular/ressources-formation/material-all.module.ts

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



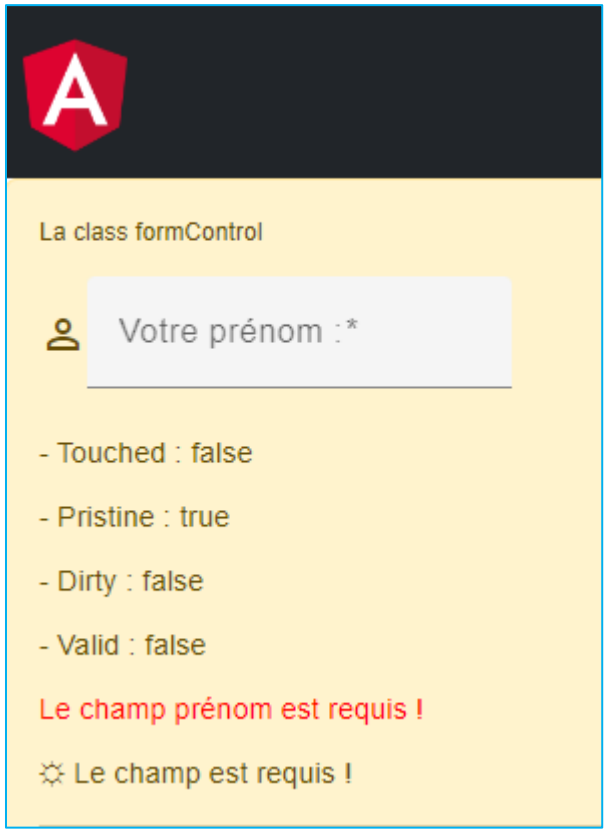


Les formulaires Angular (avec Matérial Design)

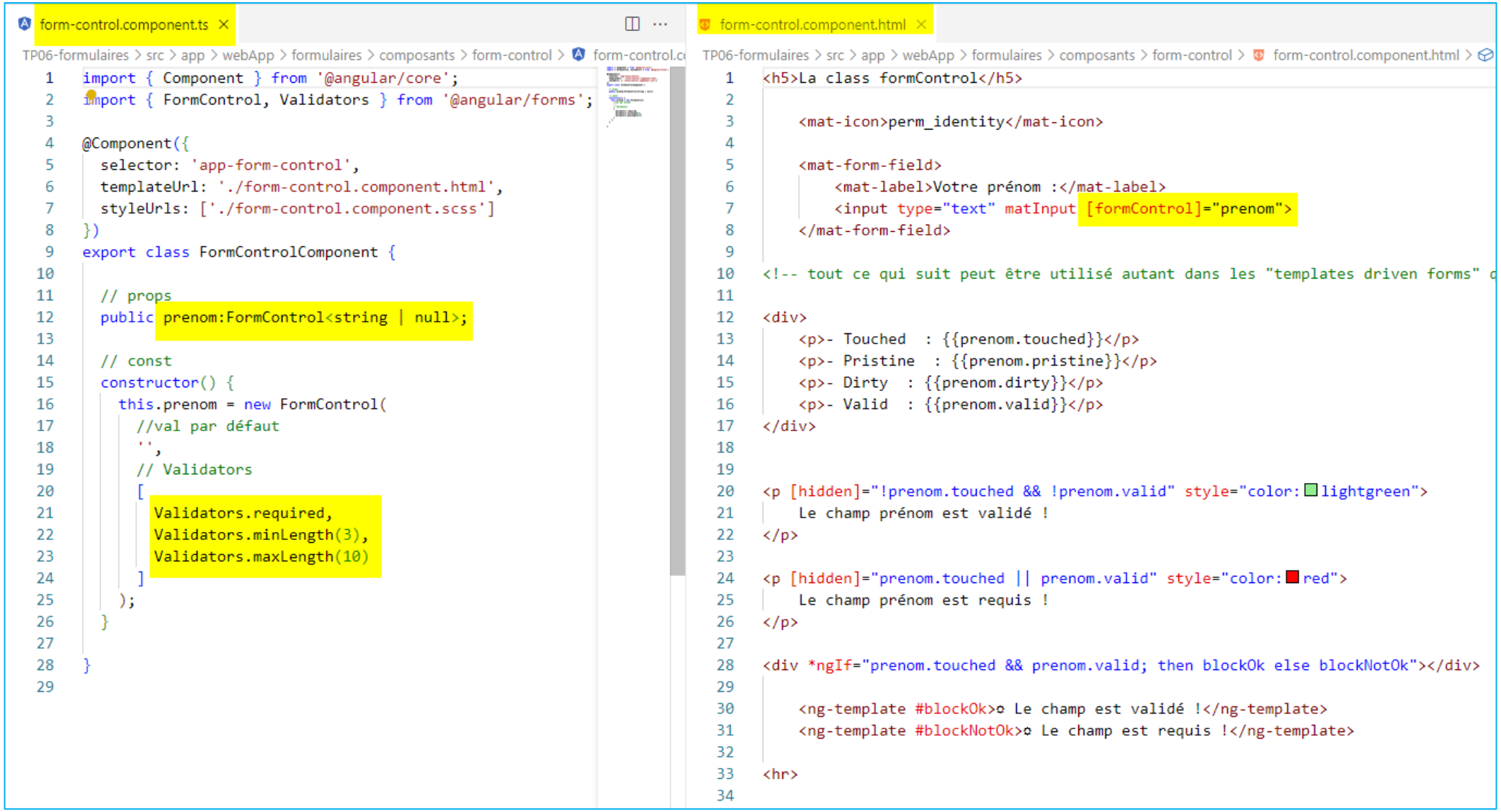


Les formulaires Angular

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



La Class form-control



form-control.component.ts

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

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

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

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

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

}

```

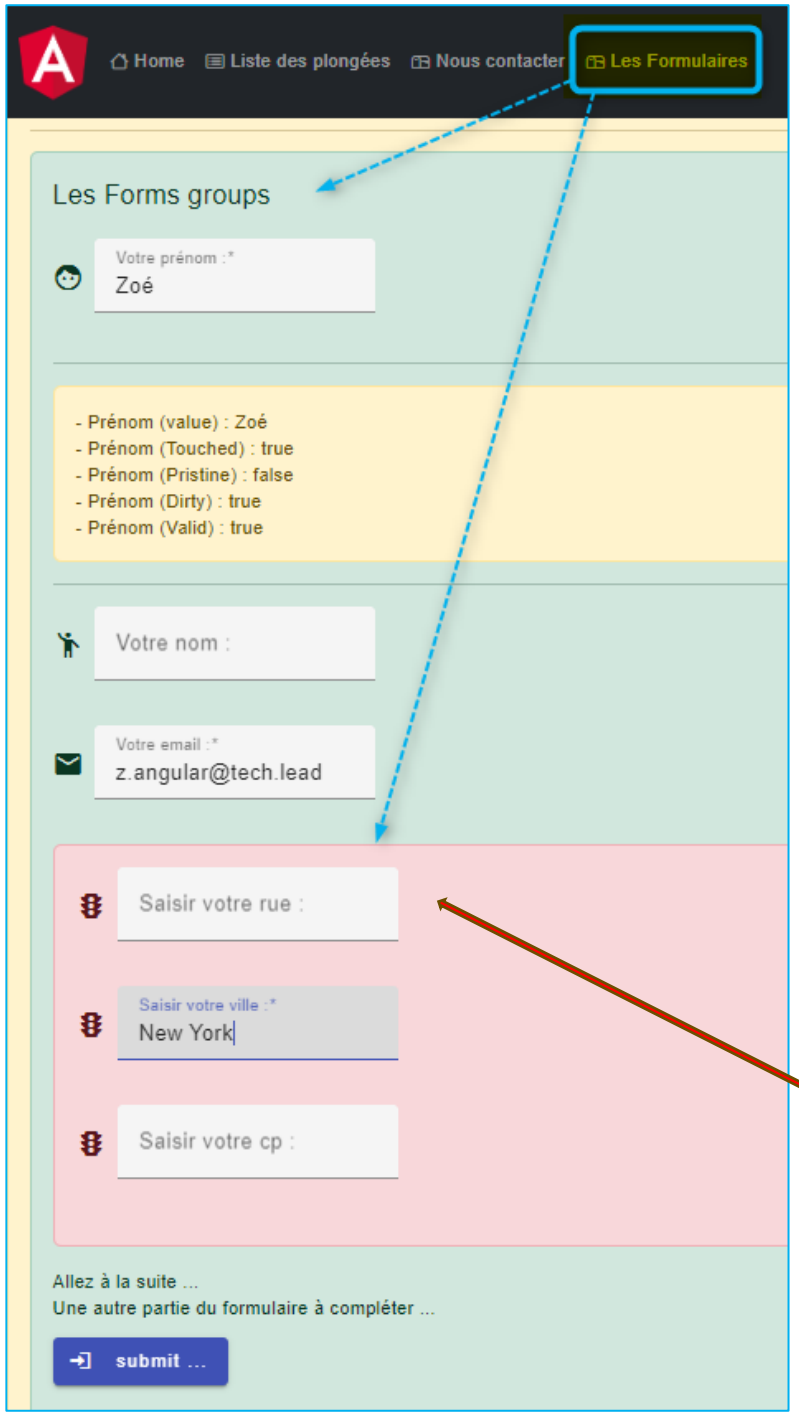
form-control.component.html

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

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

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

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

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

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


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

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

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

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

<hr>

```

La Class form-control

La class form-group



form-group.component.html

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

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

La Class formGroup

```
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

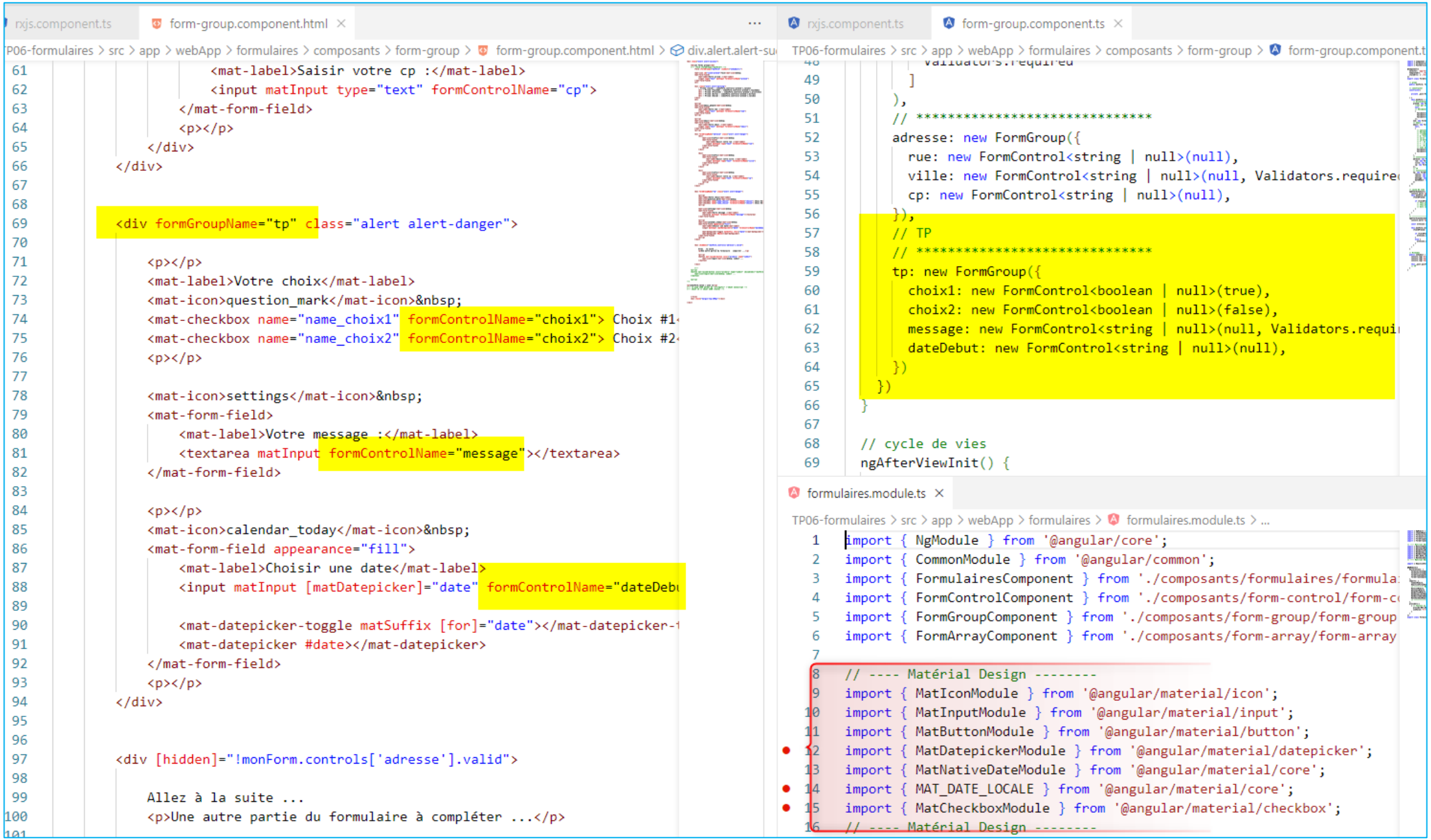
3- Créer le service « post.service.ts » qui poste les datas sur un json-server

```
"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 > ...

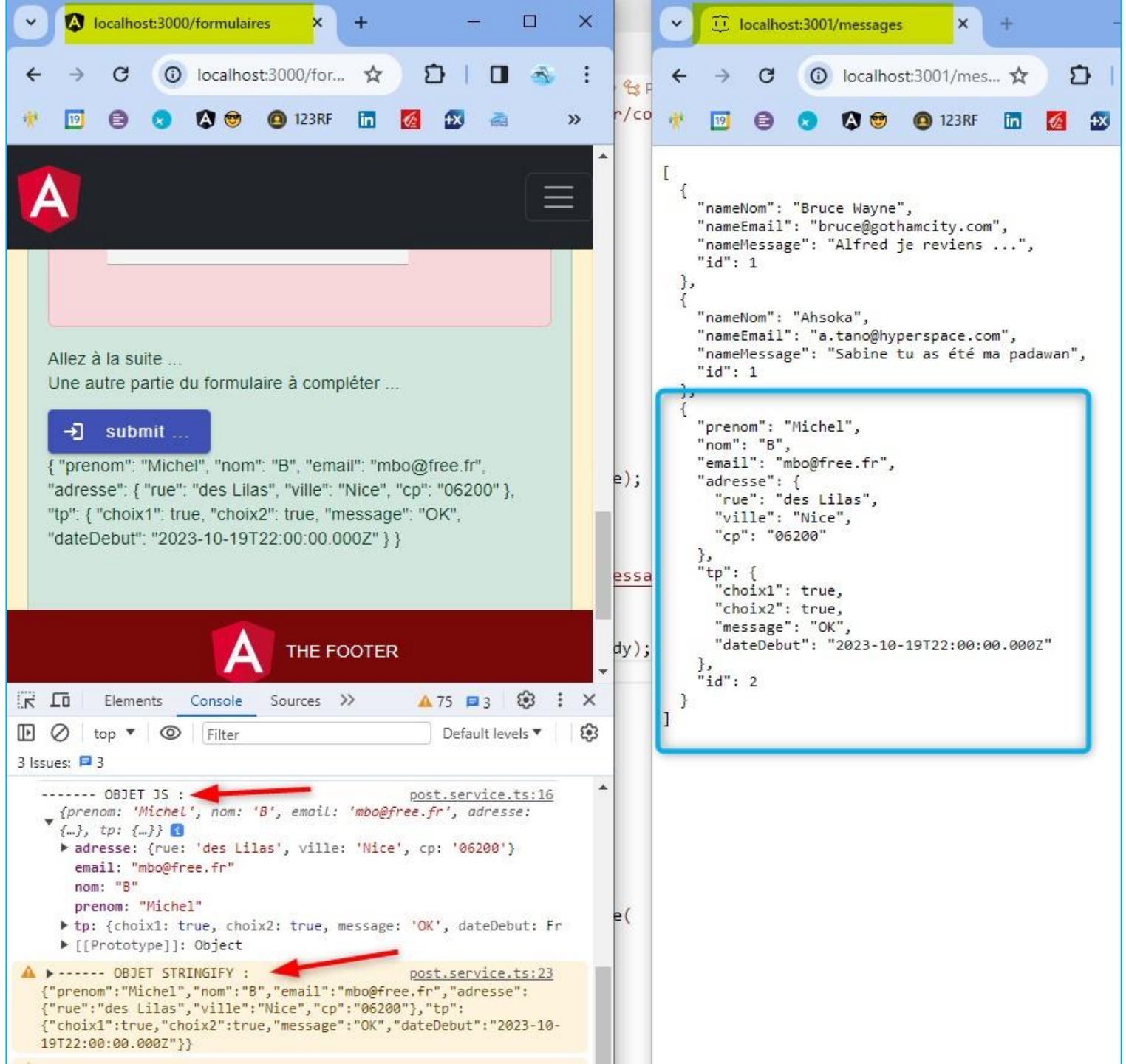
```
1   import { NgModule } from '@angular/core';
2   import { CommonModule } from '@angular/common';
3   import { FormulairesComponent } from './composants/formulaires/formula
4   import { FormControlComponent } from './composants/form-control/form-c
5   import { FormGroupComponent } from './composants/form-group/form-group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

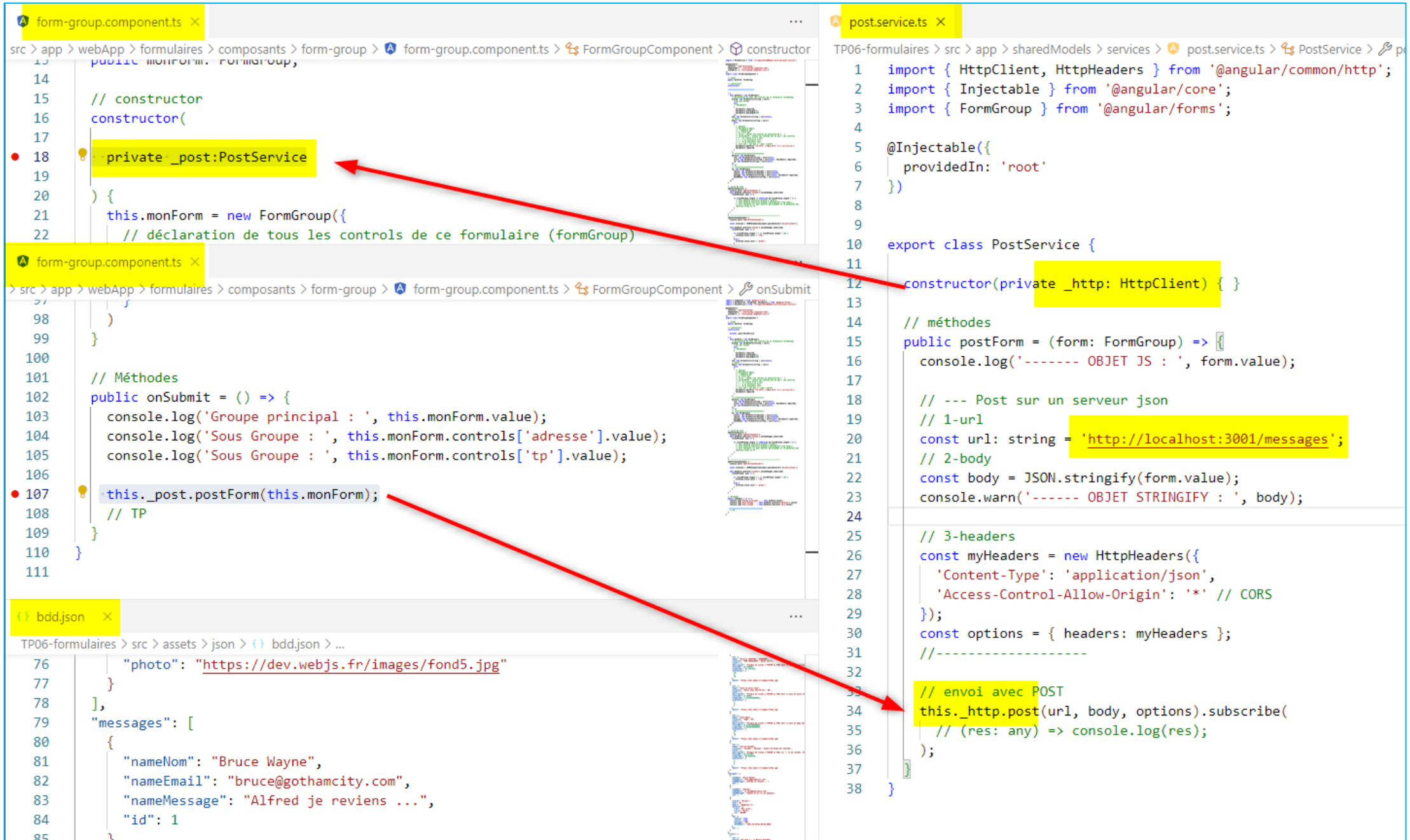
La Class formGroup

```
        // + : 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
```

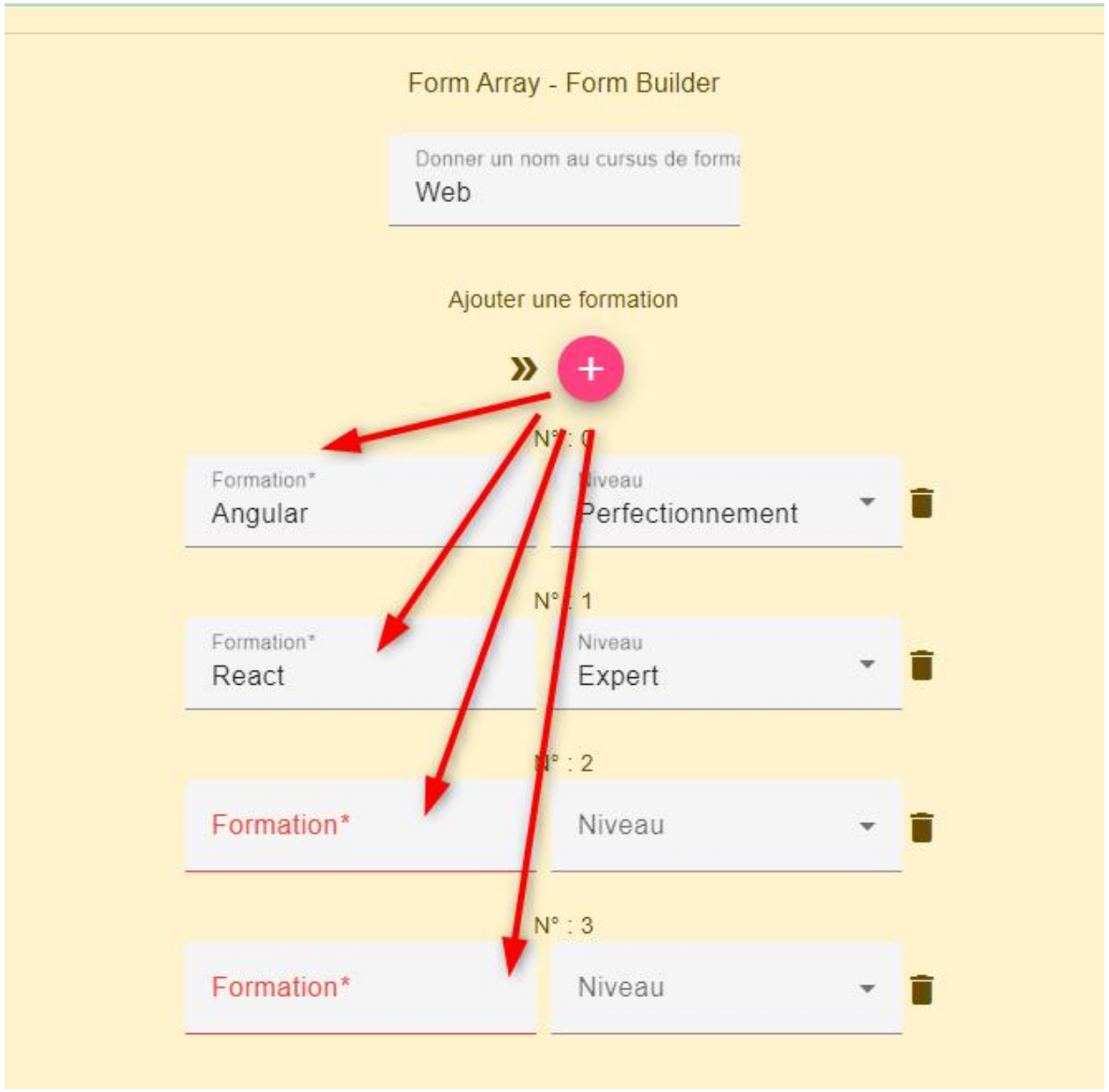
```
1 Issue: 1

[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



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>
```