



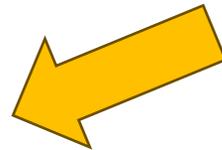
Formation ANGULAR

Chapitre 2

Animé par Michel BOCCIOLESI

Table des matières

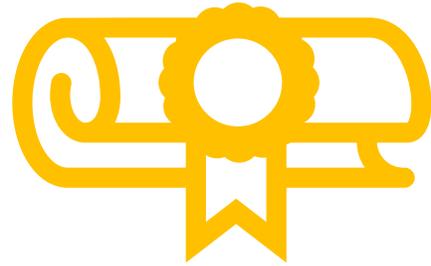
- « Nouveautés - Rappels » EcmaScript 2015+
La révolution Javascript (*rappels*)
- Documenter son projet – fichiers MD
- PWA – Progressive Web App
Comment une appli Web devient « progressivement » une appli Mobile ?
- I18N : L'Internationalisation
Comment traduire notre projet Angular ?
- Le routage avancé et le Lazy Loading
Concepts avancés de routage
Optimisation de la compilation
- La programmation RXJS et les observables
- Les formulaires Angular
Reactive Forms vs Template Driven Form
- Tests Unitaires
Karma & Jasmine
- Eco-système back FireBase
Base de données back-end en temps réel
- NGRX – la notion de Store dans Angular
aNGular Redux rXjs
- Le Framework Angular - Historique
Les versions marquantes 9 – 16
- Le « Detection Change Mode » Angular
Introduction au « signal NG16 – NG17 »



[Récupérer les sources du projet de formation ici](#)

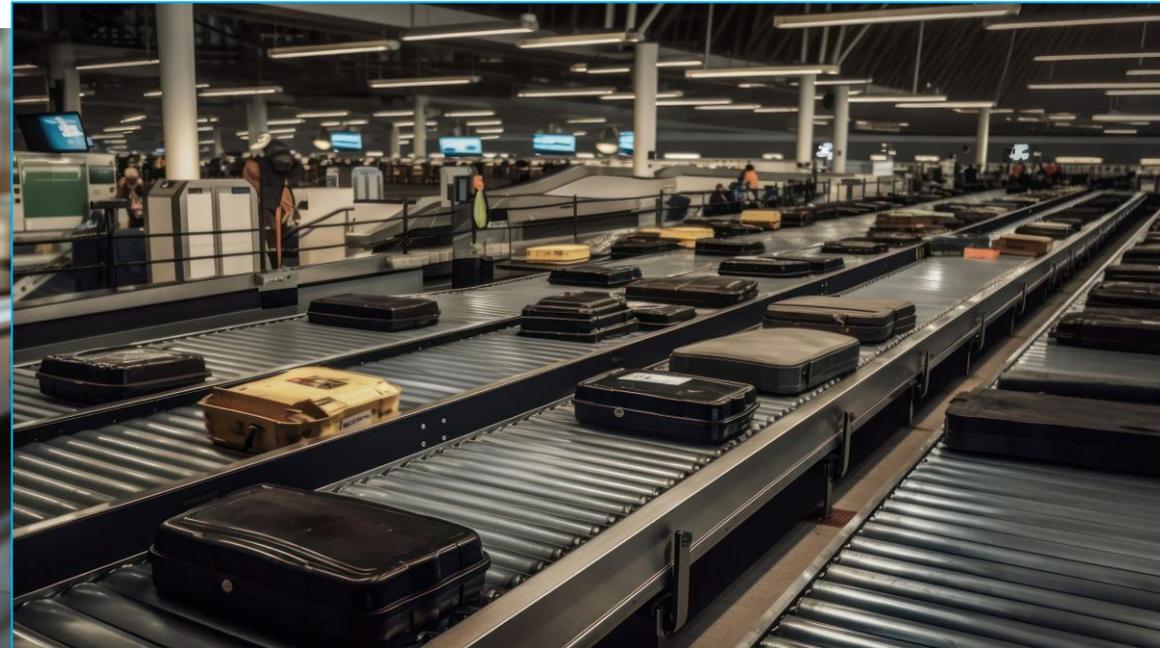
Reactive eXtensions for JavaScript

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)

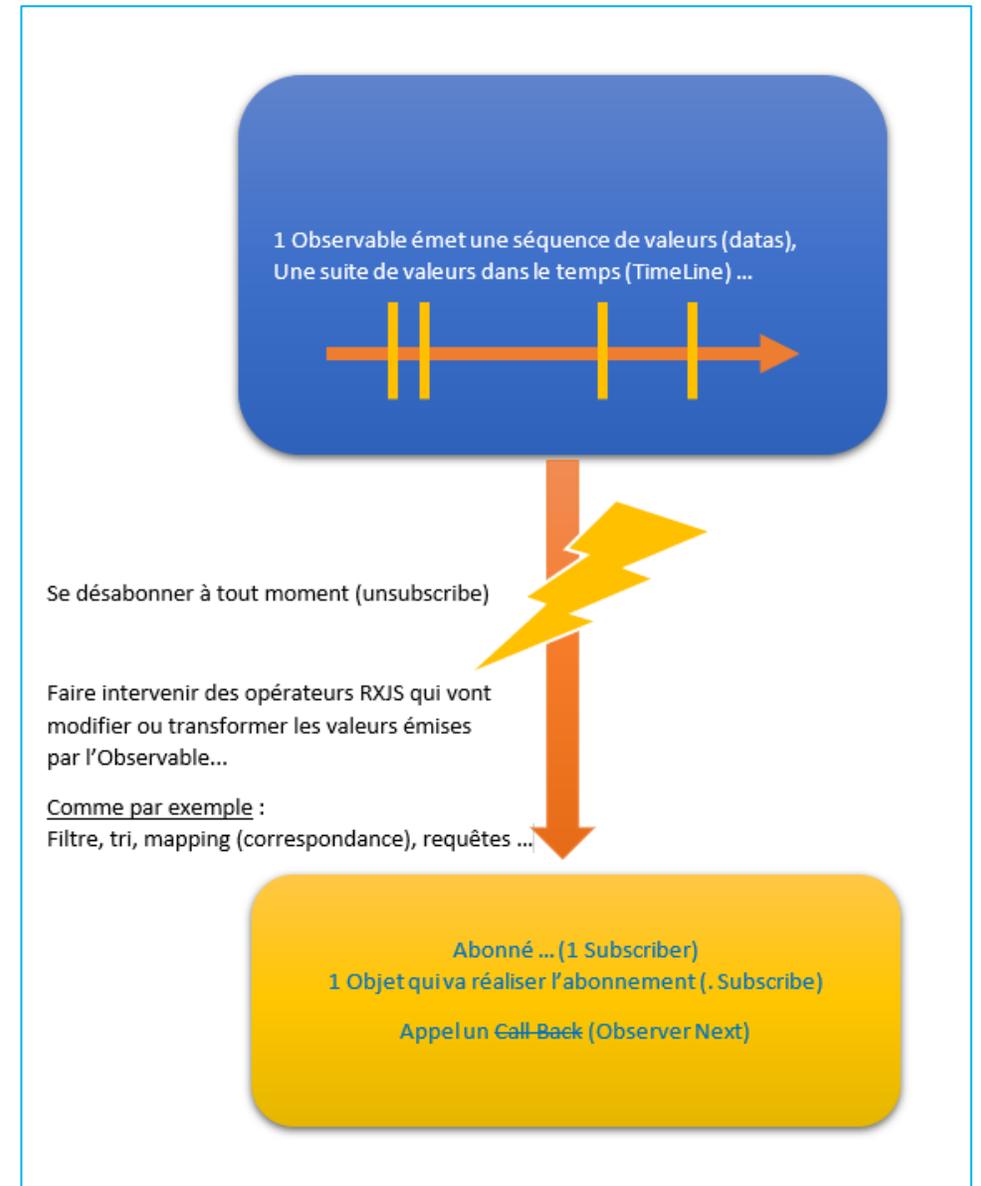
Opérateurs : des fonctions tap, map, merge, filter qui vont pouvoir modifier/transformer ce flux de valeurs émises par l'observable

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** !

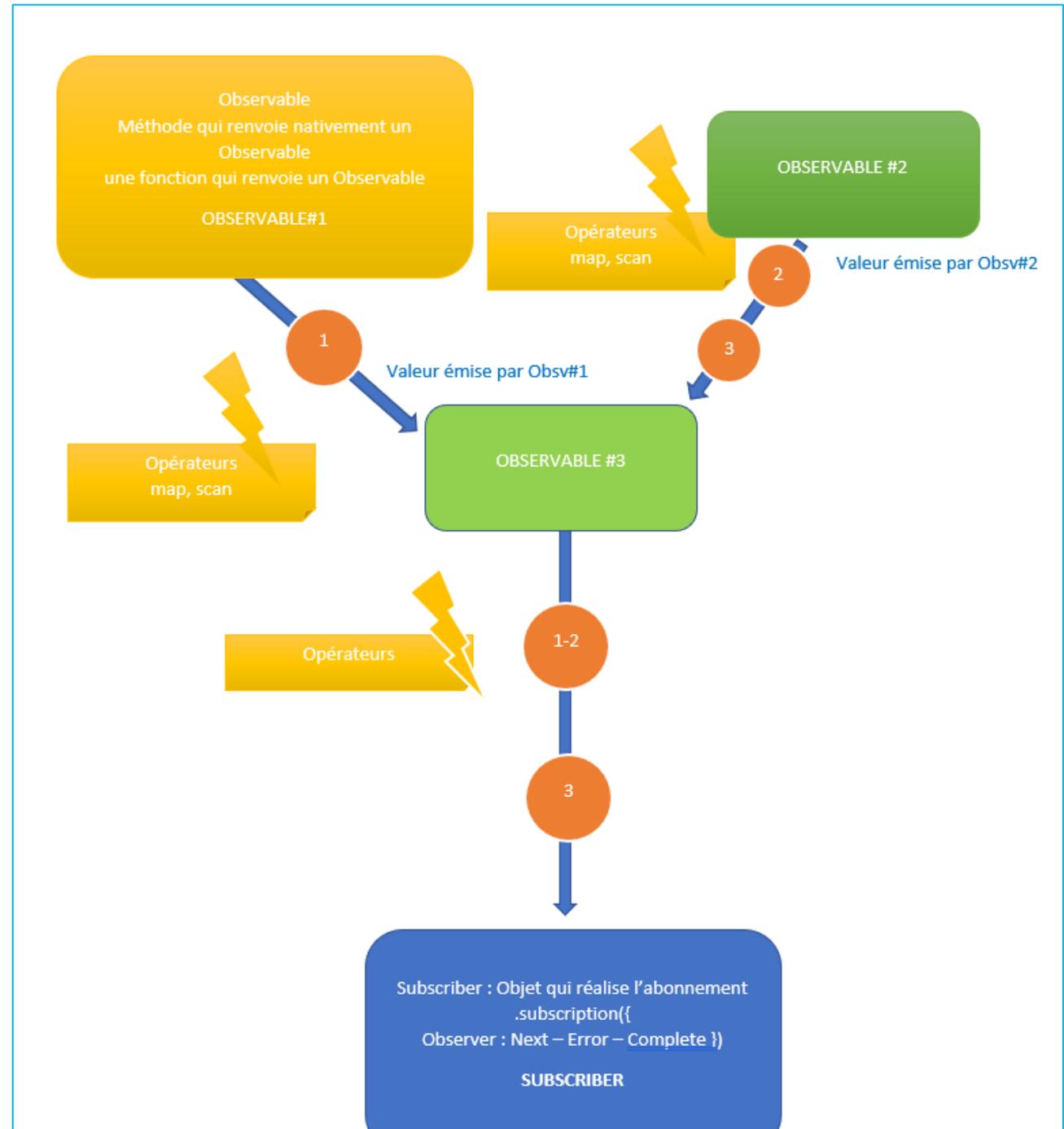
Les séquences de valeurs émises sont facilement **annulables** !

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

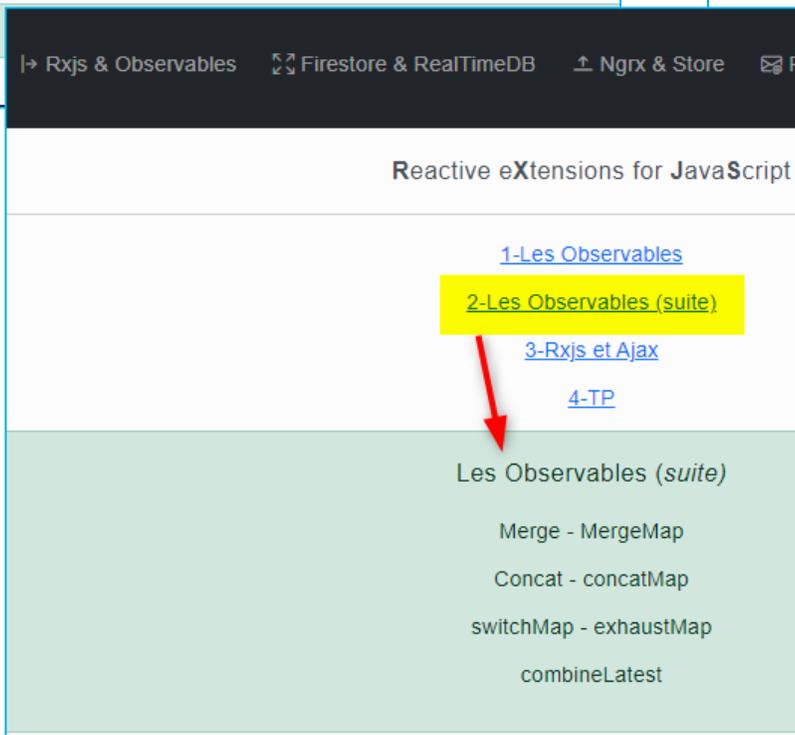
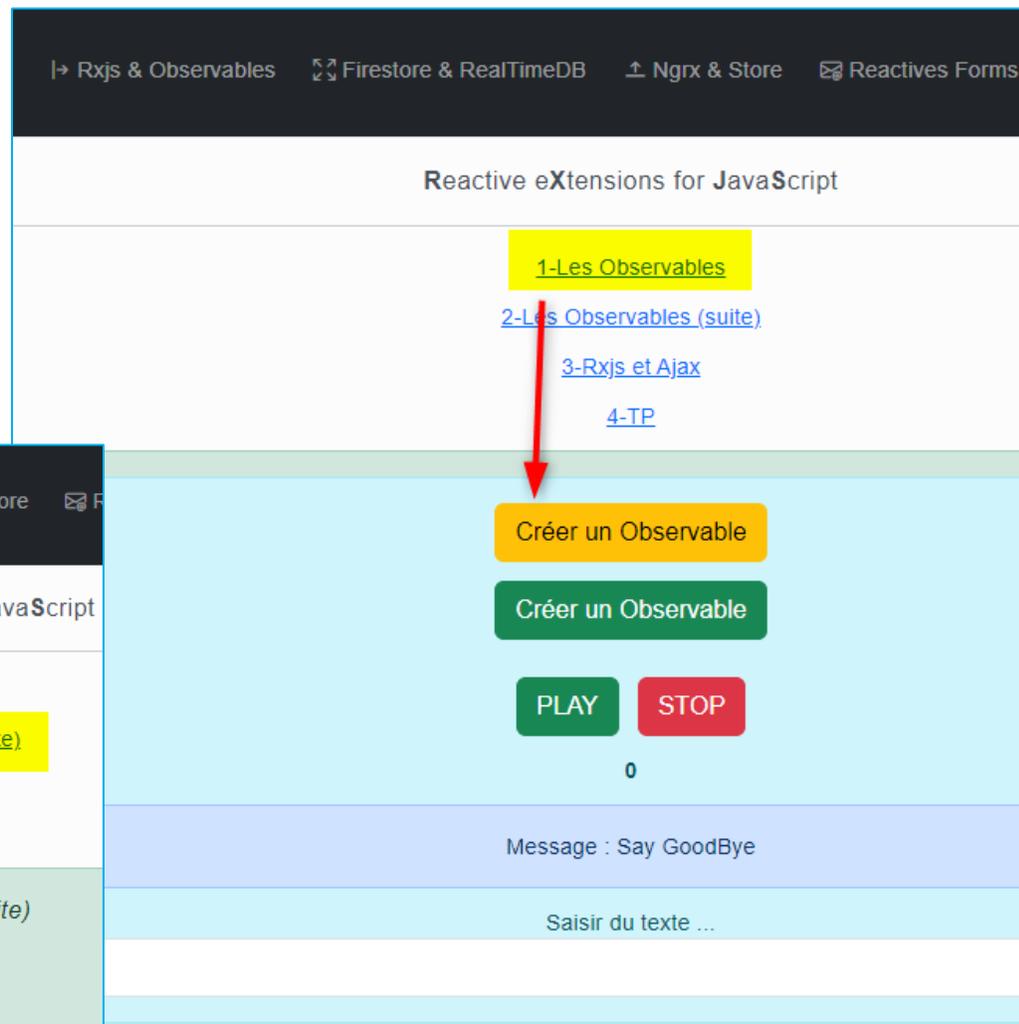
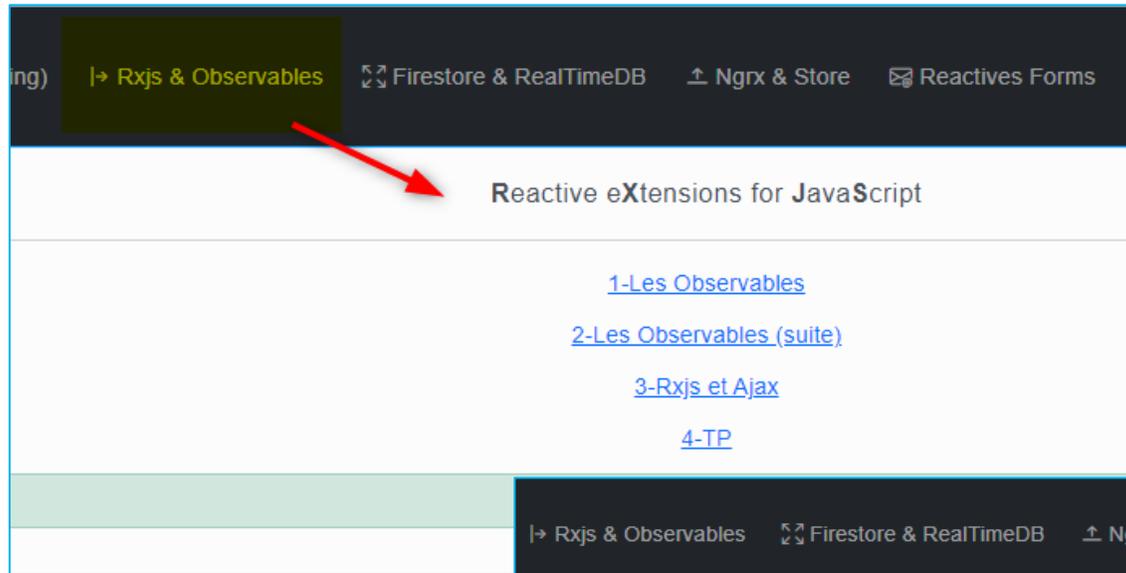


RXJS et les OBSERVABLES

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



TP : déployer l'arborescence en Lazy Loading RXJS



Créer un Observable

Créer un Observable

PLAY

STOP

0

Message : Say GoodBye

Saisir du texte ...

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>
  &nbsp;
  <!-- <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>
```

Reactive eXtensions for JavaScript

- [1-Les Observables](#)
- [2-Les Observables \(suite\)](#)
- [3-Rxjs et Ajax](#)
- [4-TP](#)

Créer un Observable

NG16

Créer un Observable

NG 16 : 4 jours.

VUE : 4 jours.

PLAY STOP

0

Message : Say GoodBye

Saisir du texte ...

Exercice #1



observables.component.ts

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

this.subscription = formation$
  .pipe(
    first()
  ).subscribe({
    // la notion d'Observer : next | error | complete
    next:
      (formation: string) => {
        console.log(formation);
        const eltLi = <HTMLElement>document.createElement('li');
        eltLi.innerHTML = formation;
        eltLi.className = 'list-group-item';
        this.eltU11.nativeElement.appendChild(eltLi);
      },
    // error
    error:
      (e) => {
        console.error(e);
      },
    // complete
    complete:
      () => {
        console.warn('Complete');
      }
  });
}
```

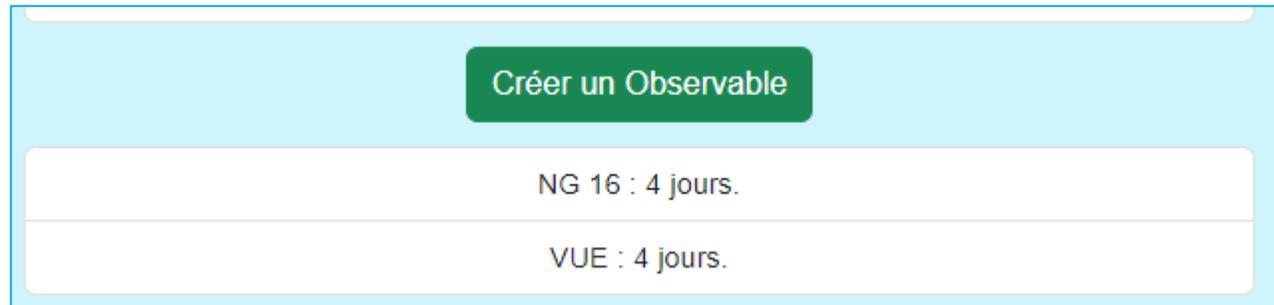
Exercice #1



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



Exercice #2



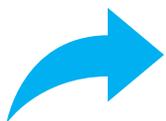
Observables-suite.component

```
ngOnInit() {  
  // Merge : une simple émission en parallèle  
  // -----  
  const observableA$ = interval(1000)  
    .pipe(take(3), map(  
      (valObs) => `ObsA ${valObs}`  
    ));  
  const observableB$ = interval(1000)  
    .pipe(take(3), map(  
      (valObs) => `ObsB ${valObs}`  
    ));  
  
  merge(observableA$, observableB$)  
    // émission en parallèle  
    .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 être interprété  
  // -----  
  
  concat(observableA$, observableB$)  
    // émission après complete  
    .subscribe(  
      (valObs) => console.warn('concat', valObs)  
    );  
}
```

Les fonctions de regroupement et de recombinaison de séquences

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

Exercice #2



Observables-suite.component

Les fonctions de regroupement et de recombinaison de séquences

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

```
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}
```

Exercice #2



Observables-suite.component

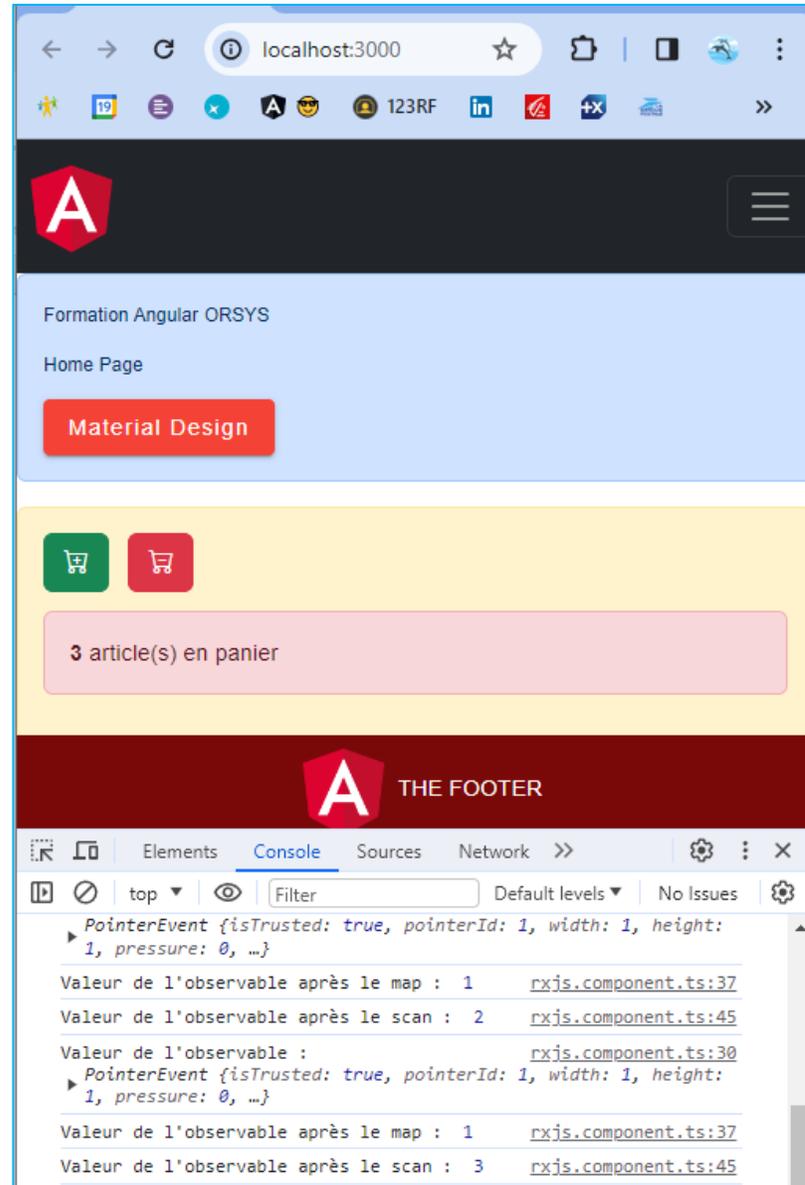
Les fonctions de regroupement et de recombinaison de séquences

```
▶ 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)



The screenshot displays a web browser window at localhost:3000. The page features a dark header with a red 'A' logo and a hamburger menu. Below the header, the text 'Formation Angular ORSYS' and 'Home Page' is visible, followed by a red 'Material Design' button. A yellow section contains two shopping cart icons (green and red) and a pink notification box stating '3 article(s) en panier'. The footer is dark red with the 'A' logo and the text 'THE FOOTER'.

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

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

RXJS et les OBSERVABLES

The image shows a development environment with three main components: a code editor on the left, a code editor in the middle, and a browser on the right.

Left Code Editor (body.component.html):

```
1 <div class="alert alert-primary">
2
3 <h3>Formation Angular ORSYS</h3>
4 <h2>Home Page</h2>
5 <button mat-raised-button color="warn">Material Design</button>
6
7
8 <div class="alert alert-warning">
9
10 <app-rxjs></app-rxjs>
11
12
```

Middle Code Editor (rxjs.component.ts):

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

Bottom Code Editor (rxjs.component.html):

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

Browser (localhost:3000):

The browser displays the rendered application. It features a blue header with a white 'A' logo and the text "Formation Angular ORSYS". Below the header is a "Home Page" section with a red "Material Design" button. A yellow sidebar contains two shopping cart icons (one green, one red) and a pink search bar.

```
// -----  
// 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, nouvelleValeur: number) => { return accu + nouvelleValeur }  
    ),  
    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 = `${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 = `${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 = `${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)

Comptes (Comptes) Compte Client(Lazy Loading) Rxjs & Observables Firestore & RealTimeDB Ngrx & Store Reactives Forms Unit Tests

Reactive eXtensions for JavaScript

[1-Les Observables](#)
[2-Les Observables \(suite\)](#)
[3-Rxjs et Ajax](#)
[4-TP](#)

[getdatas](#)

Titre	Description
Episode I : La Menace Fantôme	La République Galactique est en pleine ébullition. La taxation des routes commerciales reliant les systèmes éloignés provoque la cupide Fédération du Commerce et ses redoutables vaisseaux de guerre imposent un blocus à la petite planète Naboo.Face à ce Congrès de la République s'enlise dans des débats sans fin, le Chancelier Suprême charge en secret deux Chevaliers Jedi, gardiens de la galaxie, de résoudre le conflit ...
Episode II : L'Attaque des Clones	L'agitation règne au Sénat Galactique. Des milliers de systèmes solaires ont annoncé leur intention de quitter la République. Confronté par le Comte Dooku, les Chevaliers Jedi, en nombre limité, ont du mal à maintenir la paix et l'ordre dans la galaxie. La sénator Palpatine, revient au Sénat Galactique participer à un vote crucial sur la création d'une Armée de la République pour aider les Jedi à combattre les séparatistes.
Episode III : La Revanche des Sith	C'est la guerre ! La République croule sous les attaques de l'impitoyable Sith, le Comte Dooku. Il y a des héros dans les deux camps. Le général audace stupéfiante, le Général Grievous, diabolique chef droïde, est entré dans la capitale pour enlever le Chancelier Palpatine, Alors que l'Armée Séparatiste Droïde tente de quitter la capitale assiégée avec son précieux otage, deux Chevaliers Jedi mènent une mission pour secourir le Chancelier captif...

TP (Requêtes Ajax : forkJoin)

```
ajax.component.html x rxjs.module.ts
webApp-corrigé > src > app > webApp > formation > rxjs > composants-routing > ajax > ajax.component.html >
1 <button class="btn btn-primary" (click)="getDatas()">getdatas</button>
2
3 <table mat-table [dataSource]="films" class="mat-elevation-z8">
4
5   <ng-container matColumnDef="title">
6     <th mat-header-cell *matHeaderCellDef> Titre </th>
7     <td mat-cell *matCellDef="let element"> {{element.title}} </td>
8   </ng-container>
9
10
11   <ng-container matColumnDef="desc">
12     <th mat-header-cell *matHeaderCellDef> Description </th>
13     <td mat-cell *matCellDef="let element"> {{element.desc}} </td>
14   </ng-container>
15
16
17   <tr mat-header-row *matHeaderRowDef="displayedColumns"></tr>
18   <tr mat-row *matRowDef="let row; columns: displayedColumns;"></tr>
19
20 </table>
```

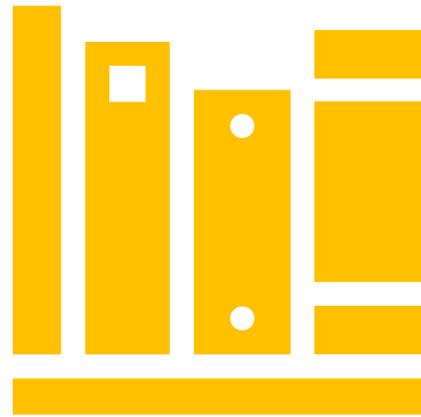
```
// -----
public getDatas = () => {

  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 Matériel Design)

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

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

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

Material Components CDK Guides

Button

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

OVERVIEW API EXAMPLES

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

Basic buttons

Basic	Basic	Primary	Accent	Warn	Disabled	Link
Raised	Basic	Primary	Accent	Warn	Disabled	Link
Stroked	Basic	Primary	Accent	Warn	Disabled	Link
Flat	Basic	Primary	Accent	Warn	Disabled	Link
Icon	⋮	🏠	☰	❤	📄	

Les formulaires Angular (avec Matériel Design)

`ng add @angular/material`

ng add @angular/localize

ng add @angular/pwa

Les formulaires Angular (avec Matériel Design)

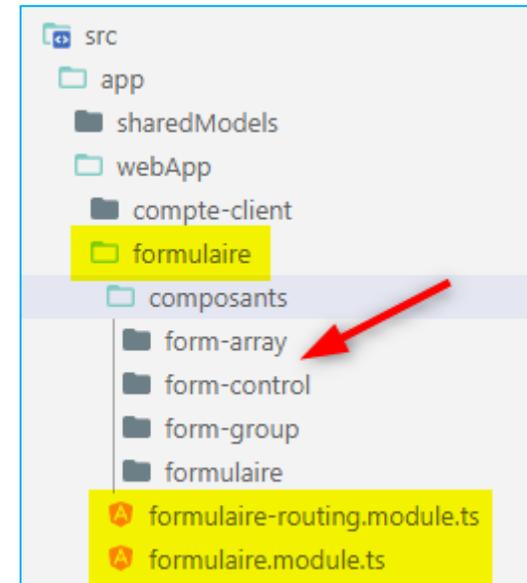
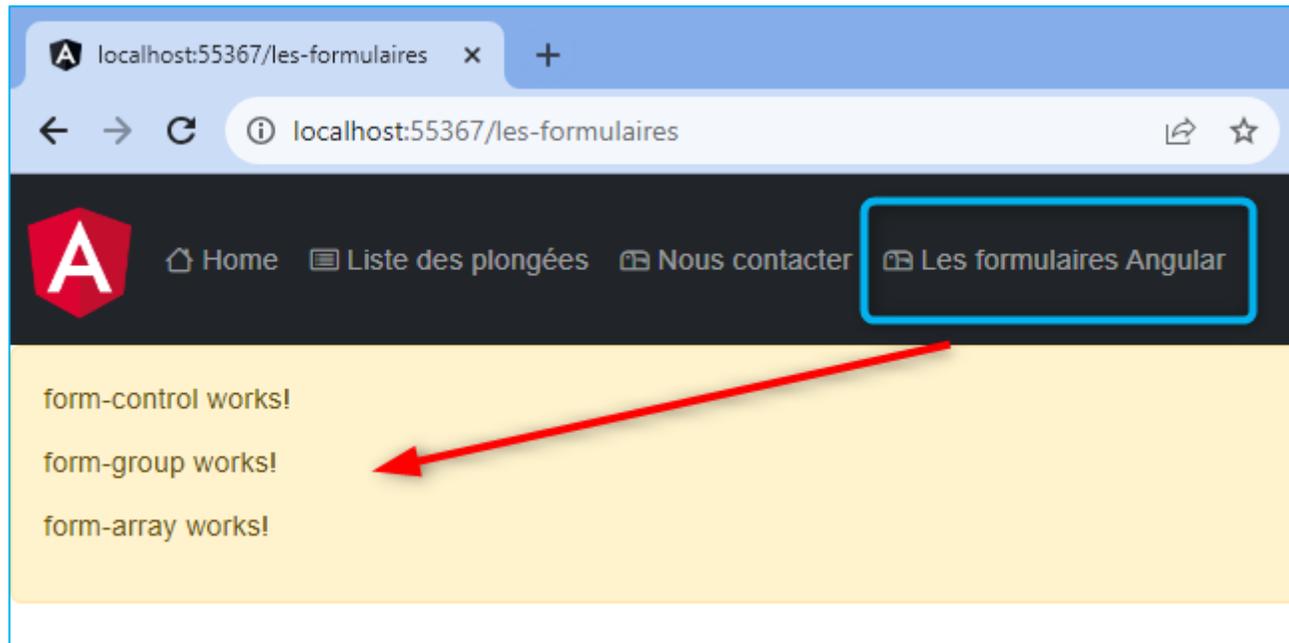
<https://dev.webjs.fr/1-Angular/ressources-formation/material.module.ts>

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

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

Ressources en ligne (*corrigé en ligne*) :

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



Les formulaires Angular

- 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 FormControl



Votre prénom :*

- Touched : false

- Pristine : true

- Dirty : false

- Valid : false

Le champ prénom est requis !

⚙ Le champ est requis !

[La Class form-control](#)

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

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

La Class form-control

[Home](#)
[Liste des plongées](#)
[Nous contacter](#)
[Les Formulaires](#)

Les Forms groups

Votre prénom :*
 Zoé

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

Votre nom :

Votre email :*
 z.angular@tech.lead

Saisir votre rue :

Saisir votre ville :*
 New York

Saisir votre cp :

Allez à la suite ...
 Une autre partie du formulaire à compléter ...

```

form-group.component.html
TP06-formulaires > src > app > webApp > formulaires > composants > form-group > form-group.component.html > div.a
1 <div class="alert alert-success">
2
3 <h2>Les Forms groups</h2>
4 <form [formGroup]="monForm">
5   <!-- <form [formGroup]="monForm" (submit)="onSubmit()" -->
6
7   <mat-icon id="icone-prenom">face</mat-icon>&nbsp;
8   <mat-form-field>
9     <mat-label>Votre prénom :</mat-label>
10    <input type="text" matInput formControlName="prenom">
11  </mat-form-field>
12  <hr>
13
14  <div class="alert alert-warning">
15    - Prénom (value) : {{monForm.controls['prenom'].value}}
16    <br> - Prénom (Touched) : {{monForm.controls['prenom'].touched}}
17    <br> - Prénom (Pristine) : {{monForm.controls['prenom'].pristine}}
18    <br> - Prénom (Dirty) : {{monForm.controls['prenom'].dirty}}
19    <br> - Prénom (Valid) : {{monForm.controls['prenom'].valid}}
20  </div>
21  <hr>
22
23  <p></p>
24  <mat-icon>emoji_people</mat-icon>&nbsp;
25  <mat-form-field>
26    <mat-label>Votre nom :</mat-label>
27    <input type="text" matInput formControlName="nom">
28  </mat-form-field>
29  <p></p>
30
31  <p></p>
32  <mat-icon>email</mat-icon>&nbsp;
33  <mat-form-field>
34    <mat-label>Votre email :</mat-label>
35    <input type="text" matInput formControlName="email">
36  </mat-form-field>
37  <p></p>
38
39  <div formGroupName="adresse" class="alert alert-danger">
40    <div>
41      <mat-icon>traffic</mat-icon>&nbsp;
42      <mat-form-field>
43        <mat-label>Saisir votre rue :</mat-label>
44        <input matInput type="text" formControlName="rue">
45      </mat-form-field>
46    <p></p>
  
```

[La class form-group](#)

La Class FormGroup

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

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

TP :

1- récupérer le HTML : <https://dev.webjss.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.webjss.fr/1-Angular/ressources-formation/json.zip>

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

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

```
nxjs.component.ts  form-group.component.html  TP06-formulaires > src > app > webApp > formulaires > composants > form-group > form-group.component.html  TP06-formulaires > src > app > webApp > formulaires > composants > form-group > form-group.component.html  TP06-formulaires > src > app > webApp > formulaires > formulaires.module.ts > ...  
61     <mat-label>Saisir votre cp :</mat-label>  
62     <input matInput type="text" FormControlName="cp">  
63   </mat-form-field>  
64   <p></p>  
65 </div>  
66 </div>  
67  
68  
69 <div FormGroupName="tp" class="alert alert-danger">  
70  
71   <p></p>  
72   <mat-label>Votre choix</mat-label>  
73   <mat-icon>question_mark</mat-icon>&nbsp;nbsp;nbsp;  
74   <mat-checkbox name="name_choix1" FormControlName="choix1"> Choix #1</mat-checkbox>  
75   <mat-checkbox name="name_choix2" FormControlName="choix2"> Choix #2</mat-checkbox>  
76   <p></p>  
77  
78   <mat-icon>settings</mat-icon>&nbsp;nbsp;nbsp;  
79   <mat-form-field>  
80     <mat-label>Votre message :</mat-label>  
81     <textarea matInput FormControlName="message"></textarea>  
82   </mat-form-field>  
83  
84   <p></p>  
85   <mat-icon>calendar_today</mat-icon>&nbsp;nbsp;nbsp;  
86   <mat-form-field appearance="fill">  
87     <mat-label>Choisir une date</mat-label>  
88     <input matInput [matDatepicker]="date" FormControlName="dateDebut">  
89  
90     <mat-datepicker-toggle matSuffix [for]="date"></mat-datepicker-toggle>  
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
```

```
48  
49   ],  
50 ),  
51 // *****  
52 adresse: new FormGroup({  
53   rue: new FormControl<string | null>(null),  
54   ville: new FormControl<string | null>(null, Validators.required),  
55   cp: new FormControl<string | null>(null),  
56 }),  
57 // TP  
58 // *****  
59 tp: new FormGroup({  
60   choix1: new FormControl<boolean | null>(true),  
61   choix2: new FormControl<boolean | null>(false),  
62   message: new FormControl<string | null>(null, Validators.required),  
63   dateDebut: new FormControl<string | null>(null),  
64 })  
65 }  
66 }  
67  
68 // cycle de vies  
69 ngAfterViewInit() {  
70  
71 }  
72 }  
73 }  
74 }  
75 }  
76 }  
77 }  
78 }  
79 }  
80 }  
81 }  
82 }  
83 }  
84 }  
85 }  
86 }  
87 }  
88 }  
89 }  
90 }  
91 }  
92 }  
93 }  
94 }  
95 }  
96 }  
97 }  
98 }  
99 }  
100 }  
101 }
```

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

La Class formGroup

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

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

```
50     <p></p>
51   </div>
52   <div>
53     <mat-icon>tr
54     <mat-form-fie
55     <mat-label
56     <input ma
57   </mat-form-fi
58
59   <p></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: 'gfg', email: 'mbo@free.fr', adresse: {...}}

- ▼ adresse: [form-group.component.ts:62](#)
 - cp: "3"
 - rue: "2"
 - ville: "2"
 - ▶ [[Prototype]]: Object
- ▶ [[Prototype]]: Object

Sous-Group (form) adresse : [form-group.component.ts:62](#)
▼ {rue: '2', ville: '2', cp: '3'} 1

- cp: "3"
- rue: "2"
- ville: "2"
- ▶ [[Prototype]]: Object

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" } }
```

```
----- 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"}}
```

```
[
  {
    "nameNom": "Bruce Wayne",
    "nameEmail": "bruce@gothamcity.com",
    "nameMessage": "Alfred je reviens ...",
    "id": 1
  },
  {
    "nameNom": "Ahsoka",
    "nameEmail": "a.tano@hyperspace.com",
    "nameMessage": "Sabine tu as été ma padawan",
    "id": 1
  },
  {
    "prenom": "Michel",
    "nom": "B",
    "email": "mbo@free.fr",
    "adresse": {
      "rue": "des Lilas",
      "ville": "Nice",
      "cp": "06200"
    },
    "tp": {
      "choix1": true,
      "choix2": true,
      "message": "OK",
      "dateDebut": "2023-10-19T22:00:00.000Z"
    },
    "id": 2
  }
]
```

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

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

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

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

La Class formArray

Récupérer les ressources en ligne :

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

Form Array - Form Builder

Donner un nom au cursus de formation
Web

Ajouter une formation

» +

N° : 0
Formation* Angular Niveau Perfectionnement

N° : 1
Formation* React Niveau Expert

N° : 2
Formation* Niveau

N° : 3
Formation* Niveau



La Class formArray

Récupérer les ressources en ligne :

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

```
export class FormArrayComponent {  
  
  public cursus: FormGroup;  
  // -----  
  constructor(private _fb: FormBuilder) {  
  
    this.cursus = this._fb.group({  
      nomCursus: '',  
      // définition du 2nd champ qui va être un tableau(ensemble) de formulaire  
      formationsArray: this._fb.array([])  
    });  
  }  
  // un getter qui va nous retourner le formationsArray  
  get getFormations(): FormArray {  
    return this.cursus.get('formationsArray') as FormArray;  
  }  
  // ----- Méthodes -----  
  public addFormation = () => {  
    this.getFormations.push(this.newFormation());  
  }  
  // méthode qui crée un nouveau formgroup  
  // avec la formation et le niveau  
  
  private newFormation = (): FormGroup<any> => {  
    return this._fb.group({ // définition du formGroup  
      formation: [null as string | null, Validators.required],  
      niveau: ''  
    });  
  }  
  // -----  
  public delFormation = (i: number) => {  
    this.getFormations.removeAt(i);  
  }  
}
```

```
<h3>Form Array - Form Builder</h3>  
  
<form [formGroup]="cursus">  
  <p>  
    <mat-form-field>  
      <mat-label>Donner un nom au cursus de formation :</mat-label>  
      <input matInput type="text" FormControlName="nomCursus">  
    </mat-form-field>  
  </p>  
  <p>Ajouter une formation </p>  
  <p>  
    <mat-icon>double_arrow</mat-icon>&nbsp;<br>  
    <button mat-mini-fab type="button" (click)="addFormation()">  
      <mat-icon>add</mat-icon>  
    </button>  
  </p>  
  
  <div formArrayName="formationsArray">  
    <div *ngFor="let formation of getFormations.controls; let i=index">  
      N° : {{i}}  
      <div [formGroupName]="i">  
  
        <mat-form-field>  
          <mat-label>Formation</mat-label>  
          <input matInput type="text" FormControlName="formation">  
        </mat-form-field>&nbsp;<br>  
  
        <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>
```