

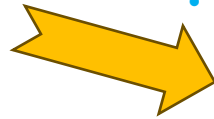
Formation ANGULAR

Chapitre 3

Animé par Michel BOCCIOLESI

Table des matières

- **Intérêt du Framework Angular**
Framework à base de modules et de web-components - webpack
Architecture à base de web-components
- **« Nouveautés » EcmaScript 2015+**
La révolution Javascript
- **Le Framework Angular - Historique**
Angular JS à Angular 2+
Les différentes versions Angular 2 à 16
Les versions marquantes
- **Documenter son projet – fichiers MD**
- **Les composants :**
TS – HTML – CSS
Binding – décorateurs
composants parents-enfants
directives - pipes
- **Les modules :**
« structurels » - architecture de projet
« fonctionnels » - partie spécifique de l'application (compte-client, panier)
- **Mise en place du projet « fil rouge » de la formation**
Application des acquis
- **Injection de dépendances (service Web ou autres)**
- **Les cycles de vie du composant**
constructor – ngOnInit – la gestion des datas
- **Le routage Angular « navigation et liens »**
Concepts de base et avancés
- **Communication entre composants parents et enfants**
@Input() @Output()
- **Les formulaires Angular**
Reactive Forms vs Template
- **La programmation RXJS et les observables**
- **Test Unitaires**
Introduction à Karma et Jasmine



Les Formulaires



Les formulaires Angular (avec Matériel Design)

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

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

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

Material Components CDK Guides

Button

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

OVERVIEW API EXAMPLES

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

Basic buttons

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

Les formulaires Angular (avec Matériel Design)

```
ng add @angular/material
```

```
ng add @angular/localize
```

```
ng add @angular/pwa
```

Les formulaires Angular (avec Matériel Design)

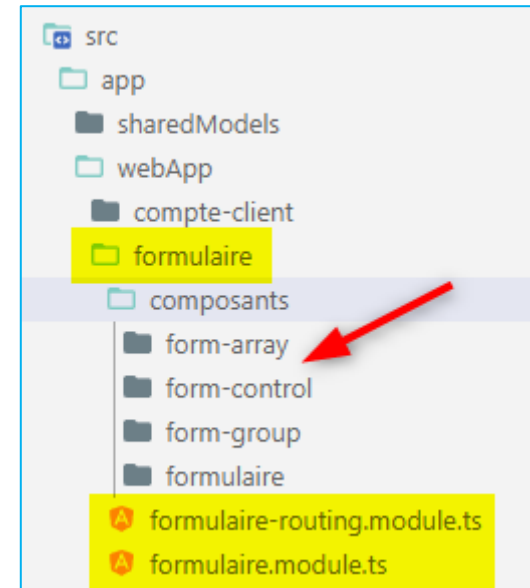
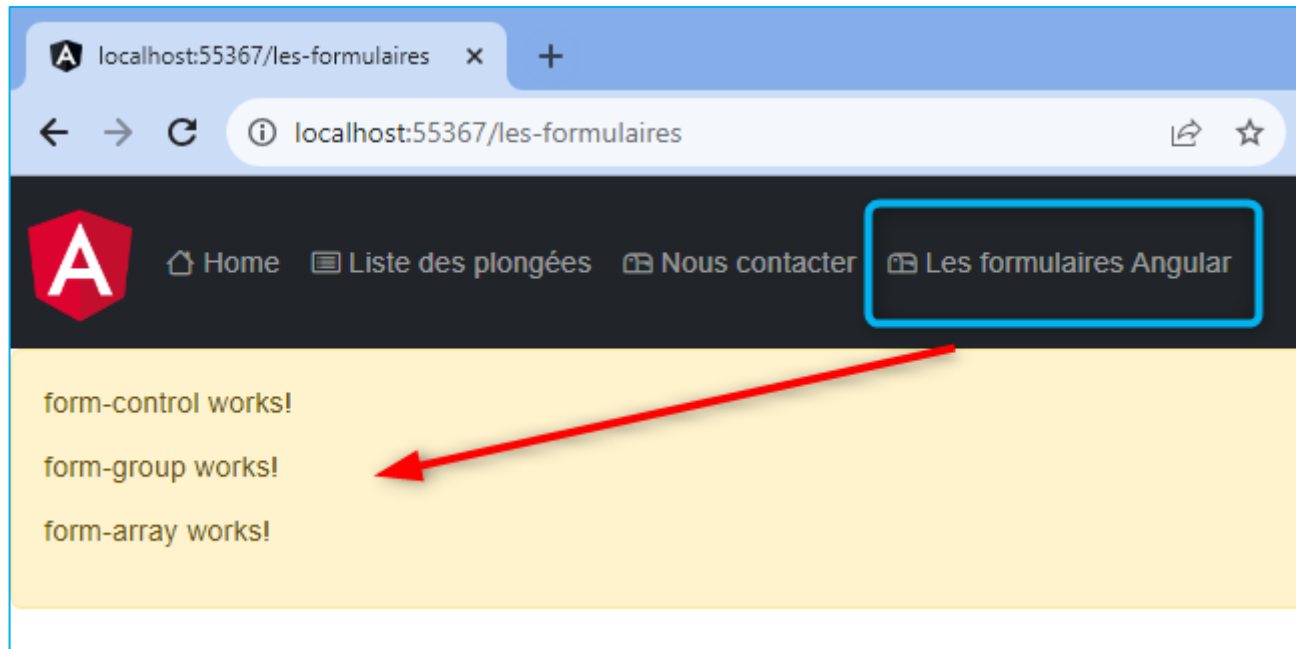
<https://dev.webjs.fr/1-Angular/ressources-formation/material.module.ts>

<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 (avec Matériel Design)

The screenshot shows an IDE with three open files in the main editor:

- formulaires.component.html**: Contains an alert warning.

```
1 <div class="alert alert-warning">
2   <app-form-control></app-form-control>
3   <app-form-group></app-form-group>
4   <app-form-array></app-form-array>
5 </div>
6
```
- header.component.html**: Contains navigation links. A red arrow points to the link 'Les formulaires Angular'.

```
16 <i class="bi bi-house"></i> Home</a>
17 </li>
18 <li class="nav-item">
19   <a class="nav-link"
20
21     routerLink="liste-des-plongees"
22
23   ><i class="bi bi-card-list"></i> Liste des plongées</a>
24 </li>
25 <li class="nav-item">
26   <a class="nav-link"
27     routerLink="contactez-nous"
28   ><i class="bi bi-mailbox"></i> Nous contacter</a>
29 </li>
30
31 <li class="nav-item">
32   <a class="nav-link"
33     routerLink="les-formulaires"
34   ><i class="bi bi-mailbox"></i> Les formulaires Angular</a>
35 </li>
36 </ul>
37 </div>
38 </nav>
```
- accueil.module.ts**: Shows the module configuration. A green arrow points to the 'FormulairesModule' import.

```
6 import { BodyComponent } from './composants/body/body.component';
7 import { DivesModule } from '../dives/dives.module';
8 import { RouterModule } from '@angular/router';
9 import { FormulairesModule } from '../formulaires/formulaires.module';
10
11 @NgModule({
12   declarations: [
13     LandingPageComponent,
14     HeaderComponent,
15     FooterComponent,
16     BodyComponent
17   ],
18   imports: [
19     CommonModule,
20     RouterModule,
21     // nos propres modules
22     DivesModule,
23     FormulairesModule
24   ],
25   exports: [
26     LandingPageComponent,
27     // HeaderComponent
28   ]
29 })
30 export class AccueilModule {}
```
- app-routing.module.ts**: Shows the route definitions. A red arrow points to the 'les-formulaires' route.

```
4 import { DivesListComponent } from './webApp/root/dives/composants/dives-list';
5 import { BodyComponent } from './webApp/root/composants/body/body.component';
6 import { ContactsComponent } from './webApp/root/contacts/composants/contacts';
7 import { Page404Component } from './sharedModels/composants/page404/page404.component';
8 import { DivesDetailComponent } from './webApp/root/dives/composants/dives-detail';
9 import { FormulairesComponent } from './webApp/formulaires/composants/formulaires';
10
11 const routes: Routes = [
12   { path: '', component: BodyComponent }, // domain.tld
13   { path: 'accueil', component: BodyComponent },
14   { path: 'liste-des-plongees', component: DivesListComponent },
15   { path: 'contactez-nous', component: ContactsComponent },
16   { path: 'liste-des-plongees/details/:param', component: DivesDetailComponent },
17   { path: 'les-formulaires', component: FormulairesComponent },
18
19   // page404
20   { path: '**', redirectTo: '', pathMatch: 'full' },
21   { path: '**', component: Page404Component },
22 ]
```

Les formulaires Angular

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

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

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

La Class form-control

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

Les Forms groups

 Votre prénom :*

Zoé

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

 Votre nom :

 Votre email :*

z.angular@tech.lead

 Saisir votre rue :

 Saisir votre ville :*

New York

 Saisir votre cp :

Allez à la suite ...

Une autre partie du formulaire à compléter ...

 submit ...

form-group.component.html

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

```

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

```

La class form-group

La Class formGroup

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

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

TP :

1- récupérer le HTML : <https://dev.webjs.fr/1-Angular/ressources-formation/form-group.component.zip>

2- Mettre en place la validation TS

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

"json-server": "json-server --watch src/assets/json/bdd.json -p 3001"

Base de données JSON : <https://dev.webjs.fr/1-Angular/ressources-formation/json.zip>

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

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

```
rxjs.component.ts  form-group.component.html  ...  rxjs.component.ts  form-group.component.ts  x
P06-formulaires > src > app > webApp > formulaires > composants > form-group > form-group.component.html > div.alert.alert-su
61      <mat-label>Saisir votre cp :</mat-label>
62      <input matInput type="text" FormControlName="cp">
63    </mat-form-field>
64    <p></p>
65  </div>
66 </div>
67
68
69  <div FormGroupName="tp" class="alert alert-danger">
70
71    <p></p>
72    <mat-label>Votre choix</mat-label>
73    <mat-icon>question_mark</mat-icon>&nbsp;
74    <mat-checkbox name="name_choix1" FormControlName="choix1"> Choix #1<
75    <mat-checkbox name="name_choix2" FormControlName="choix2"> Choix #2<
76    <p></p>
77
78    <mat-icon>settings</mat-icon>&nbsp;
79    <mat-form-field>
80      <mat-label>Votre message :</mat-label>
81      <textarea matInput FormControlName="message"></textarea>
82    </mat-form-field>
83
84    <p></p>
85    <mat-icon>calendar_today</mat-icon>&nbsp;
86    <mat-form-field appearance="fill">
87      <mat-label>Choisir une date</mat-label>
88      <input matInput [matDatepicker]="date" FormControlName="dateDebut
89
90      <mat-datepicker-toggle matSuffix [for]="date"></mat-datepicker-t
91      <mat-datepicker #date></mat-datepicker>
92    </mat-form-field>
93    <p></p>
94  </div>
95
96
97  <div [hidden]="!monForm.controls['adresse'].valid">
98
99    Allez à la suite ...
100    <p>Une autre partie du formulaire à compléter ...</p>
101
TP06-formulaires > src > app > webApp > formulaires > composants > form-group > form-group.component.ts
48
49  ]
50  ),
51  // *****
52  adresse: new FormGroup({
53    rue: new FormControl<string | null>(null),
54    ville: new FormControl<string | null>(null, Validators.required),
55    cp: new FormControl<string | null>(null),
56  }),
57  // TP
58  // *****
59  tp: new FormGroup({
60    choix1: new FormControl<boolean | null>(true),
61    choix2: new FormControl<boolean | null>(false),
62    message: new FormControl<string | null>(null, Validators.required),
63    dateDebut: new FormControl<string | null>(null),
64  })
65  })
66  }
67
68  // cycle de vies
69  ngAfterViewInit() {
formulaires.module.ts  x
TP06-formulaires > src > app > webApp > formulaires > formulaires.module.ts > ...
1  import { NgModule } from '@angular/core';
2  import { CommonModule } from '@angular/common';
3  import { FormulairesComponent } from './composants/formulaires/formulaires.component';
4  import { FormControlComponent } from './composants/form-control/form-control.component';
5  import { FormGroupComponent } from './composants/form-group/form-group.component';
6  import { FormArrayComponent } from './composants/form-array/form-array.component';
7
8  // ---- Matériel Design -----
9  import { MatIconModule } from '@angular/material/icon';
10 import { MatInputModule } from '@angular/material/input';
11 import { MatButtonModule } from '@angular/material/button';
12 import { MatDatepickerModule } from '@angular/material/datepicker';
13 import { MatNativeDateModule } from '@angular/material/core';
14 import { MAT_DATE_LOCALE } from '@angular/material/core';
15 import { MatCheckboxModule } from '@angular/material/checkbox';
16 // ---- Matériel Design -----
```

La Class formGroup

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

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

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

1 Issue: 1

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

Angular is running in development mode. [core.mjs:25499](#)

Group (form) principal : [form-group.component.ts:61](#)

- ▼ {prenom: 'mbo', nom: 'gfggh', email: 'mbo@free.fr', adresse: {...}}
- ▼ adresse:
 - cp: "3"
 - rue: "2"
 - ville: "2"
 - ▶ [[Prototype]]: Object
- ▶ [[Prototype]]: Object

Sous-Group (form) adresse : [form-group.component.ts:62](#)

- ▼ {rue: '2', ville: '2', cp: '3'}
 - cp: "3"
 - rue: "2"
 - ville: "2"
 - ▶ [[Prototype]]: Object

localhost:3000/formulaires

localhost:3000/for...

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

submit ...

```
{ "prenom": "Michel", "nom": "B", "email": "mbo@free.fr",  
  "adresse": { "rue": "des Lilas", "ville": "Nice", "cp": "06200" },  
  "tp": { "choix1": true, "choix2": true, "message": "OK",  
    "dateDebut": "2023-10-19T22:00:00.000Z" } }
```

THE FOOTER

Elements Console Sources

3 Issues

----- OBJET JS : [post.service.ts:16](#)

```
{prenom: 'Michel', nom: 'B', email: 'mbo@free.fr', adresse:  
  {...}, tp: {...}}  
  > adresse: {rue: 'des Lilas', ville: 'Nice', cp: '06200'}  
    email: "mbo@free.fr"  
    nom: "B"  
    prenom: "Michel"  
  > tp: {choix1: true, choix2: true, message: 'OK', dateDebut: Fr  
    > [[Prototype]]: Object
```

----- OBJET STRINGIFY : [post.service.ts:23](#)

```
{ "prenom": "Michel", "nom": "B", "email": "mbo@free.fr", "adresse":  
  { "rue": "des Lilas", "ville": "Nice", "cp": "06200" }, "tp":  
  { "choix1": true, "choix2": true, "message": "OK", "dateDebut": "2023-10-  
    19T22:00:00.000Z" } }
```

localhost:3001/messages

localhost:3001/mes...

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

```

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

```

```

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

```

```

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

```

```

post.service.ts x
TP06-formulaires > src > app > sharedModels > services > post.service.ts > PostService > post
1 import { HttpClient, HttpHeaders } from '@angular/common/http';
2 import { Injectable } from '@angular/core';
3 import { FormGroup } from '@angular/forms';
4
5 @Injectable({
6     providedIn: 'root'
7 })
8
9
10 export class PostService {
11
12     constructor(private _http: HttpClient) {
13
14     }
15
16 // méthodes
17 public postForm = (form: FormGroup) => {
18     console.log('----- OBJET JS : ', form.value);
19
20     // --- Post sur un serveur json
21     // 1-url
22     const url: string = 'http://localhost:3001/messages';
23     // 2-body
24     const body = JSON.stringify(form.value);
25     console.warn('----- OBJET STRINGIFY : ', body);
26
27     // 3-headers
28     const myHeaders = new HttpHeaders({
29         'Content-Type': 'application/json',
30         'Access-Control-Allow-Origin': '*' // CORS
31     });
32     const options = { headers: myHeaders };
33     //-----
34
35     // envoi avec POST
36     this._http.post(url, body, options).subscribe(
37         (res: any) => console.log(res);
38     );
39 }

```

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>
```

Les Tests Unitaires



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

app.component.ts ×

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

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

app.component.html ×

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

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

app.component.spec.ts ×

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

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

2nd exemple : Injection de services

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

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

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

2nd exemple : Injection de services

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

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

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

2nd exemple : Injection de services

```
warn.service.ts
TP07-tests-unitaires > src > app > services > warn.service.ts > WarnService > logErreur
1 import { Injectable } from '@angular/core';
2
```

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

```
operations.service.spec.ts
TP07-tests-unitaires > src > app > services > operations.service.spec.ts > describe('Test Injection de dépendances') callback > it('Ajouter 2 nombres OK ?')
1 import { OperationsService } from './operations.service';
2 import { WarnService } from './warn.service';
3
4 // 1- Création de l'objet de Test
5 describe('Test Injection de dépendances', () => {
6
7   // 2- liste des spécifications
8
9   // it (xit - fit)
10  it('Ajouter 2 nombres OK ?', () => {
11
12    // 5- Mocker WarnService (Jasmine Spies)
13    const MockWarn = jasmine.createSpyObj('WarnService', ['warnMessage', 'logErreur']);
14    const operations = new OperationsService(MockWarn);
15    const resultat = operations.ajouter(10, 5);
16
17    // 3- expectation => ce que l'on attend
18    expect(resultat).withContext('--- Erreur Expectation ---').toBe(15);
19    // 7- ajout d'une autre expectation
20    expect(MockWarn.warnMessage).toHaveBeenCalledTimes(1);
21  });
22
23  // 4- duplication du premier IT
24
25 }
```

PROBLÈMES SORTIE CONSOLE DE DÉBOGAGE TERMINAL PORTS

node - TP07-tests-unitaires

```
==== 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

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

operations.service.ts

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

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

operations.service.spec.ts

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

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

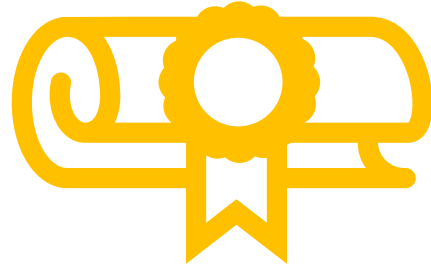
2nd exemple : Injection de services

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

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

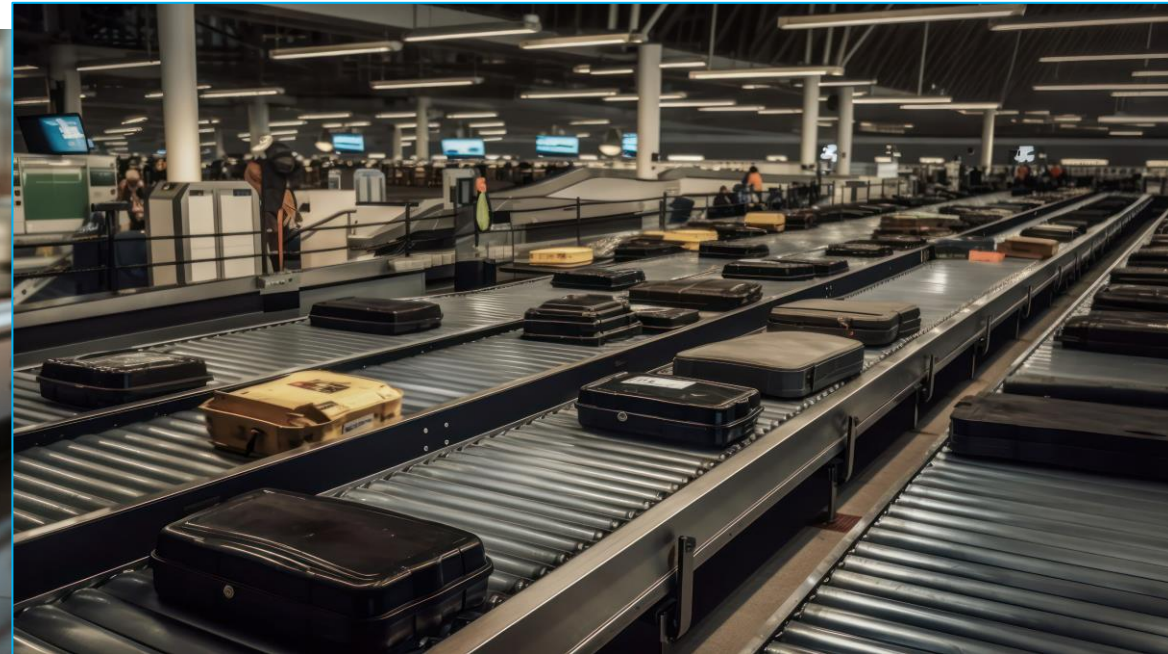
operations.service.spec.ts ×
TP07-tests-unitaires > src > app > services > operations.service.spec.ts > describe('Test Injection de dépendances') callb
1 import { TestBed } from '@angular/core/testing';
2 import { OperationsService } from './operations.service';
3 import { WarnService } from './warn.service';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

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)

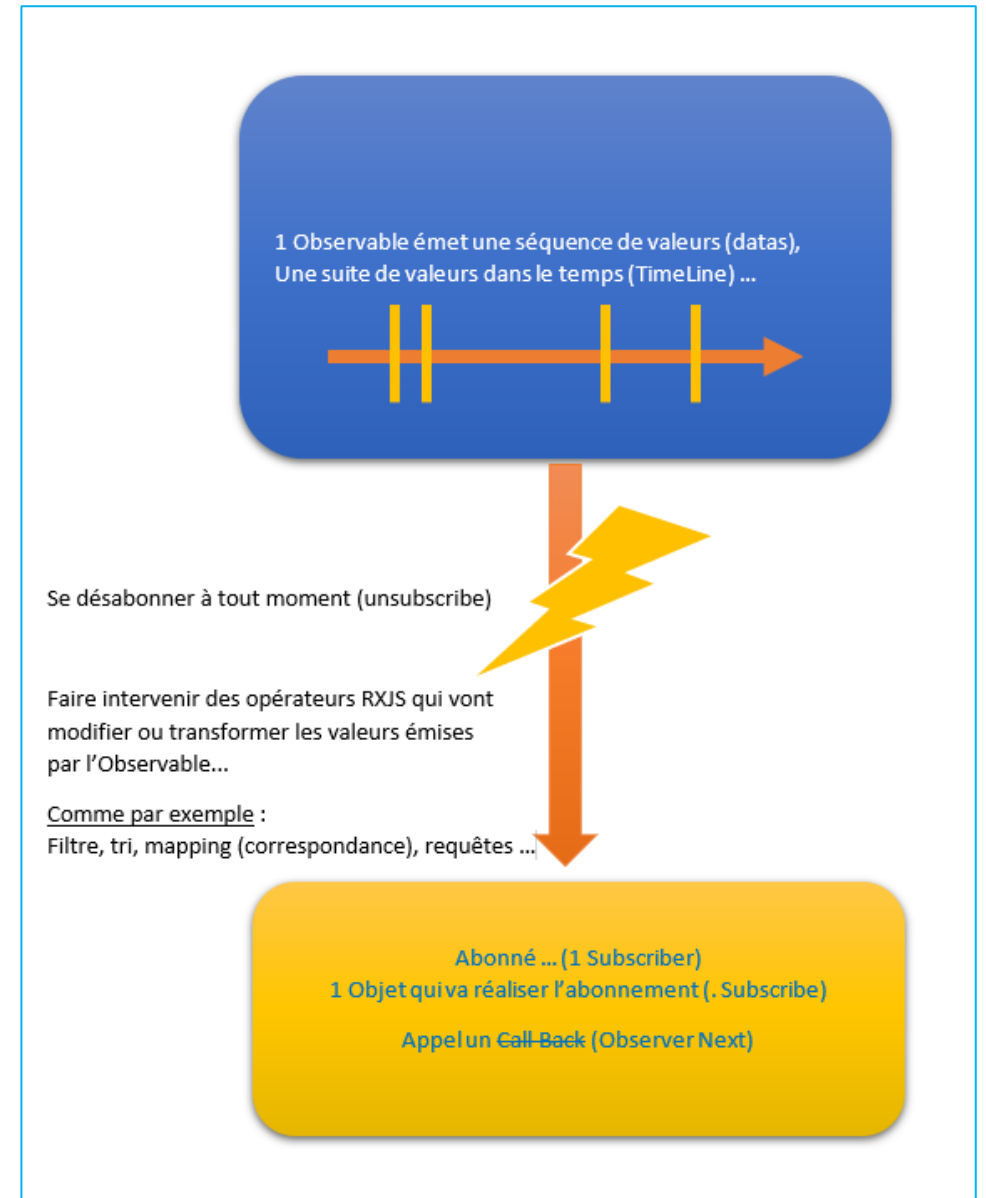
Opérateurs : des fonctions tap, map, merge, filter qui vont pouvoir modifier/transformer ce flux de valeurs émises par l'observable

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 !

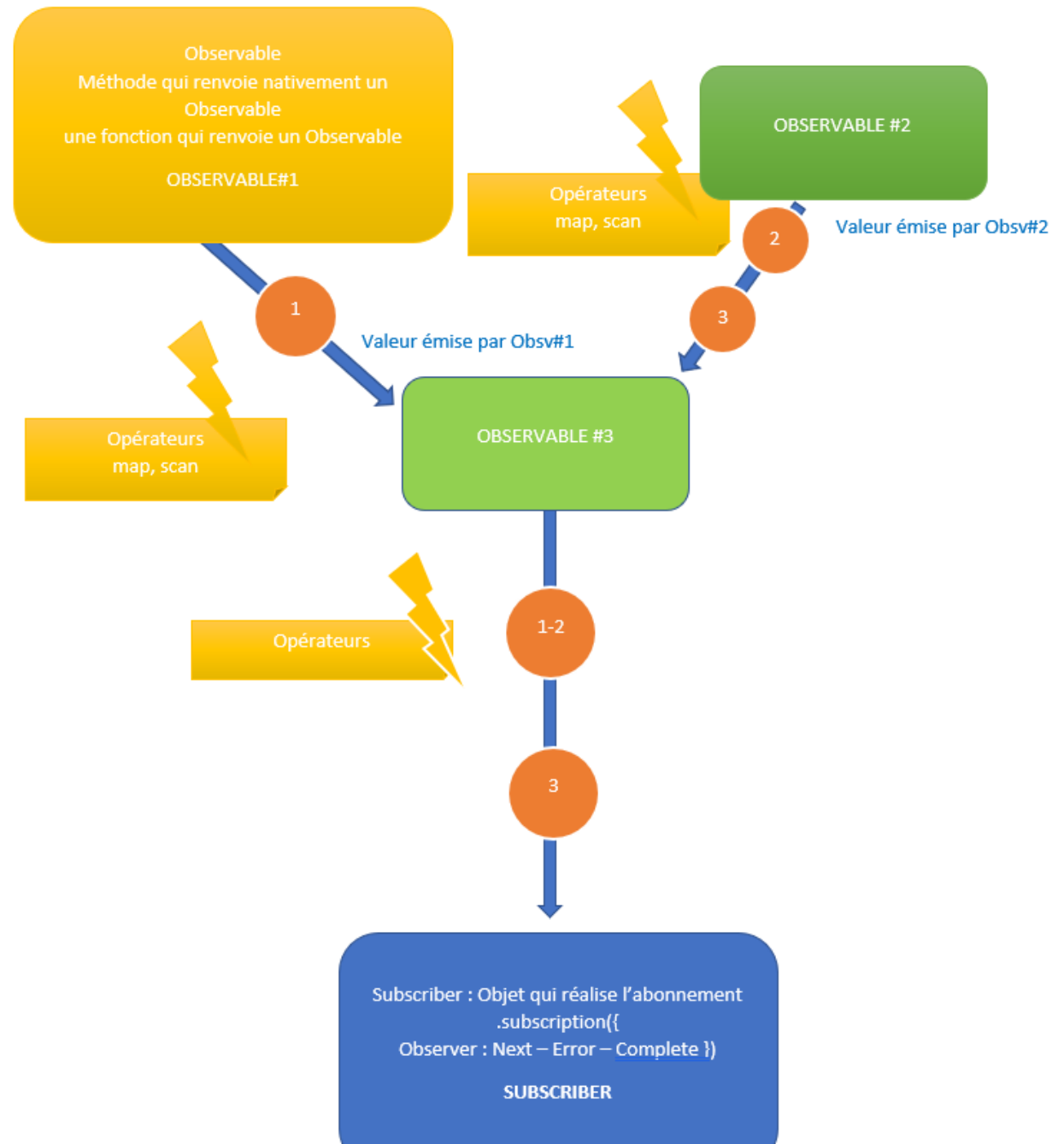
Les séquences de valeurs émises sont facilement annulables !

On peut facilement les recomposer ou les recombinaison 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 !



RXJS et les OBSERVABLES

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

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

```
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: two code editors on the left and a browser preview on the right.

Left Editor (body.component.html):

```
TP08-rxjs-compil-prod > src > app > webApp > root > accueil > composants > body > body.component.html
1 <div class="alert alert-primary">
2
3 <h1>Formation Angular ORSYS</h1>
4 <h2>Home Page</h2>
5 <button class="btn btn-raised" color="warn">Material Design</button>
6
7
8 <div class="alert alert-warning">
9
10 </div>
11
12
```

Middle Editor (rxjs.component.ts):

```
TP08-rxjs-compil-prod > src > app > webApp > root > accueil > composants > 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 Editor (rxjs.component.html):

```
TP08-rxjs-compil-prod > src > app > webApp > root > accueil > composants > rxjs.component.html
1 <button class="btn btn-success" #plus><i class="fa fa-plus"></i>
2 </button>
3 <button class="btn btn-danger" #moins><i class="fa fa-minus"></i>
4 </button>
5 <p></p>
6
7 <p class="alert alert-danger" #panier></p>
8 <p></p>
9
```

Browser Preview (localhost:3000):

The browser shows the rendered application. It features a dark blue header with a white 'A' logo and the text "Formation Angular ORSYS". Below the header is a light blue section with the text "Home Page" and a red button labeled "Material Design". The main content area has a yellow background and contains two shopping cart icons (one green, one red) and a pink rectangular placeholder.

```

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

```

```

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

const moins$ = fromEvent(this.eltMoins.nativeElement, 'click')
  .pipe(
    startWith(0),
    map(
      () => { return 1 }
    ),
    scan(
      (accu: number, nelleValeur: number) => { return accu + nelleValeur }
    )
  )

// .subscribe(
//   (valObs) => this.eltPanier.nativeElement.innerHTML = `<strong>${valObs}</strong> article(s) en panier`
// );

```

```

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

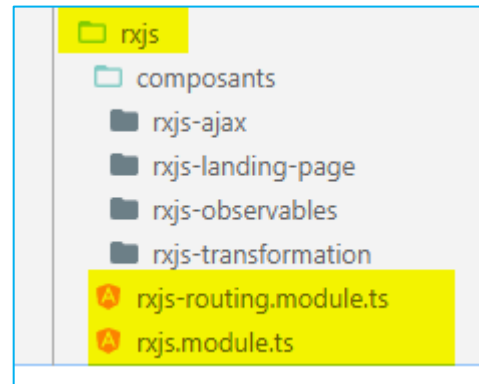
combineLatest([plus$, moins$])
  .pipe(
    map(
      ([valObs1, valObs2]) => {
        return valObs1 - valObs2;
      }
    )
  )
  .subscribe(
    (valObs) => {
      this.eltPanier.nativeElement.innerHTML = `<strong>${valObs}</strong>
      article(s) en panier`;

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

```

RXJS et les OBSERVABLES

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



```

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

<p></p>

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

```

```

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

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

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

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

```

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

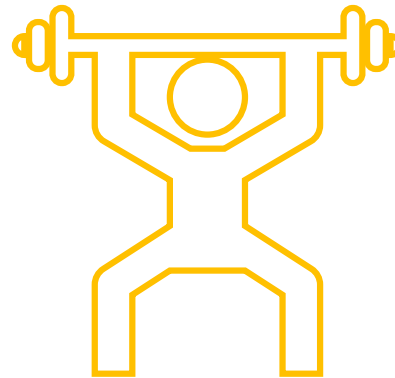
```

```

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

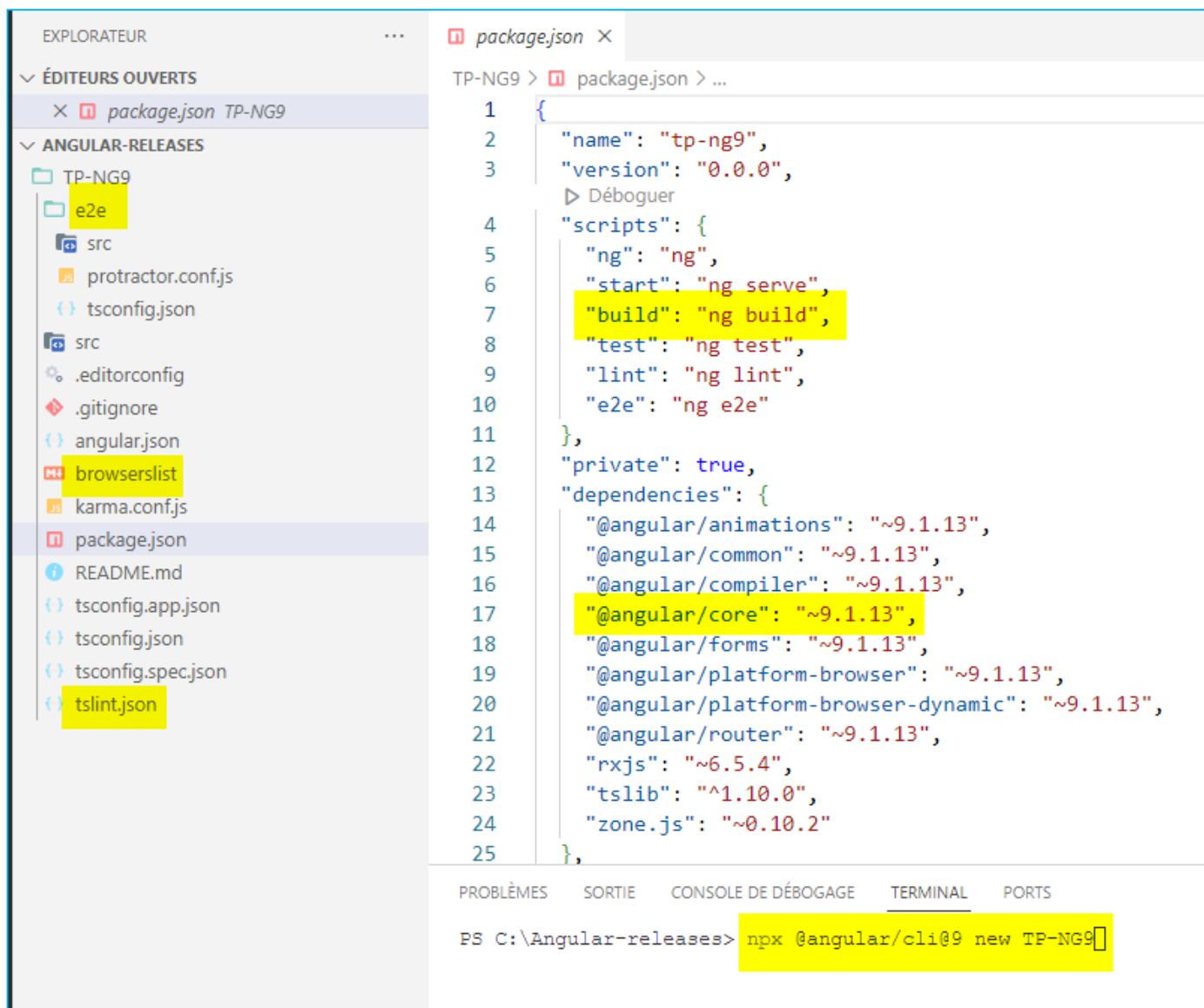
```

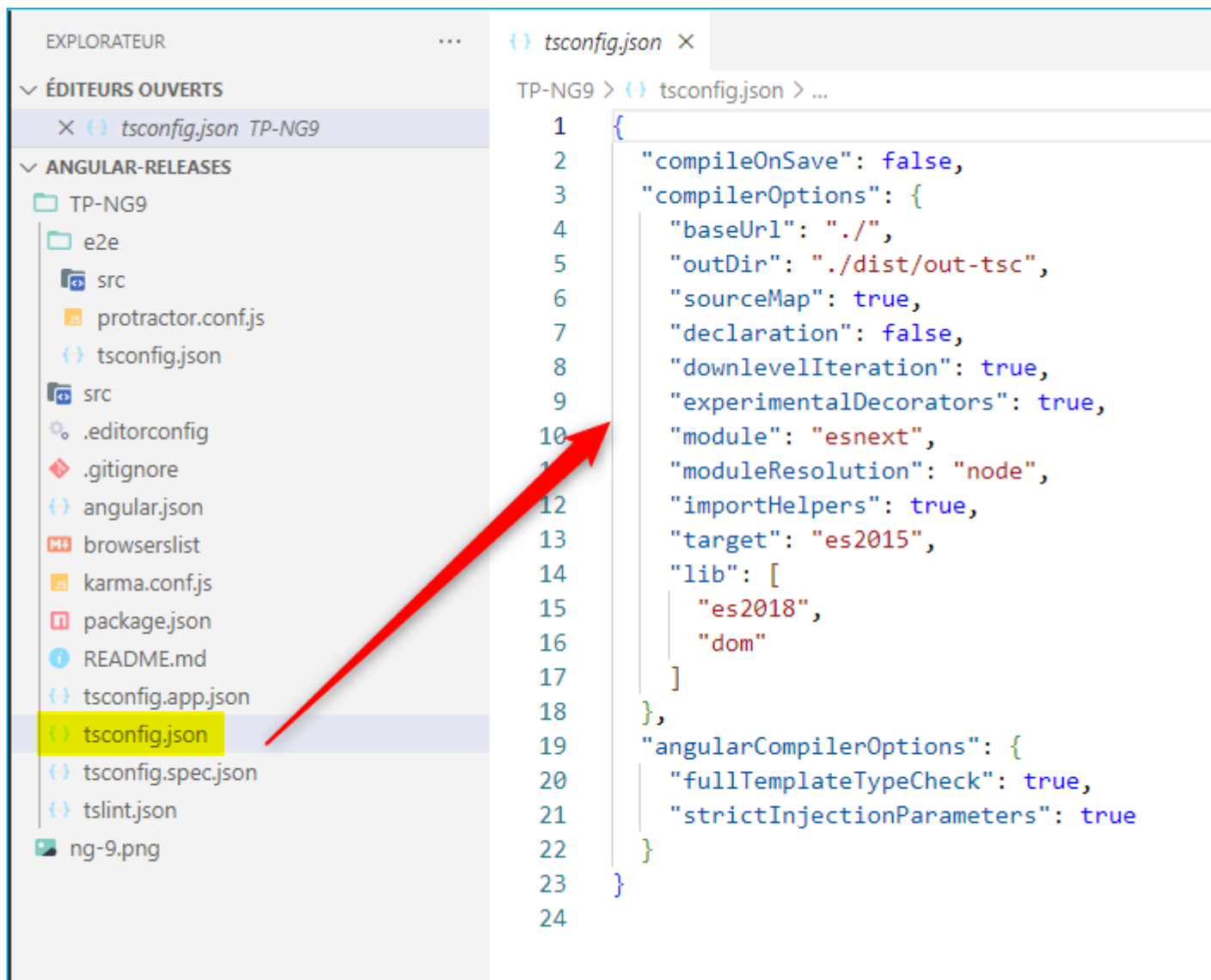
Les versions marquantes !



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, debug) AOT
activé par défaut avec ng serve => remontée des erreurs en mode dev ...
 - 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>





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 Matériel

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 > ...

```
1 {
2   "name": "tp-ng10",
3   "version": "0.0.0",
4   "scripts": {
5     "ng": "ng",
6     "start": "ng serve",
7     "build": "ng build",
8     "test": "ng test",
9     "lint": "ng lint",
10    "e2e": "ng e2e"
11  },
12  "private": true,
13  "dependencies": {
14    "@angular/animations": "~10.2.4",
15    "@angular/common": "~10.2.4",
16    "@angular/compiler": "~10.2.4",
17    "@angular/core": "~10.2.4",
18    "@angular/forms": "~10.2.4",
19    "@angular/platform-browser": "~10.2.4",
20    "@angular/platform-browser-dynamic": "~10.2.4",
21    "@angular/router": "~10.2.4",
22    "rxjs": "~6.6.0",
23    "tslib": "^2.0.0",
24    "zone.js": "~0.10.2"
25  }
```

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

```
PROBLÈMES  SORTIE  CONSOLE DE DÉBOGAGE  TERMINAL  PORTS

PS C:\Angular-releases> npx @angular/cli@11 new TP-NG11
Need to install the following packages:
  @angular/cli@11.1.2
Ok to proceed? (y) y
npm WARN deprecated @npmcli/move-file@1.1.2: This functionality has been moved to @npmcli/fs
npm WARN deprecated har-validator@5.1.5: this library is no longer supported
npm WARN deprecated source-map-codec@1.4.8: Please use @jridgewell/source-map-codec instead
npm WARN deprecated @npmcli/ci-detect@1.4.0: this package has been deprecated, use `ci-info` instead
npm WARN deprecated uuid@3.4.0: Please upgrade to version 7 or higher. Older versions may use Math.random() i
/math-random for details.
npm WARN deprecated uuid@3.4.0: Please upgrade to version 7 or higher. Older versions may use Math.random() i
/math-random for details.
npm WARN deprecated request@2.88.2: request has been deprecated, see https://github.com/request/request/issues/
npm WARN deprecated @schematics/update@0.1101.2: This was an internal-only Angular package up through Angular v
pendency.
? Do you want to enforce stricter type checking and stricter bundle budgets in the workspace?
  This setting helps improve maintainability and catch bugs ahead of time.
  For more information, see https://angular.io/strict (y/N) ☐
```

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 > ...

```
1 {
2   "name": "tp-ng12",
3   "version": "0.0.0",
4   "scripts": {
5     "ng": "ng",
6     "start": "ng serve",
7     "build": "ng build",
8     "watch": "ng build --watch --configuration development",
9     "test": "ng test"
10  },
11  "private": true,
12  "dependencies": {
13    "@angular/animations": "~12.2.0",
14    "@angular/common": "~12.2.0",
15    "@angular/compiler": "~12.2.0",
16    "@angular/core": "~12.2.0",
17    "@angular/forms": "~12.2.0",
18    "@angular/platform-browser": "~12.2.0",
19    "@angular/platform-browser-dynamic": "~12.2.0",
20    "@angular/router": "~12.2.0",
21    "rxjs": "~6.6.0",
22    "tslib": "^2.3.0",
23    "zone.js": "~0.11.4"
24  }
25 }
```

PROBLÈMES

SORTIE

CONSOLE DE DÉBOGAGE

TERMINAL

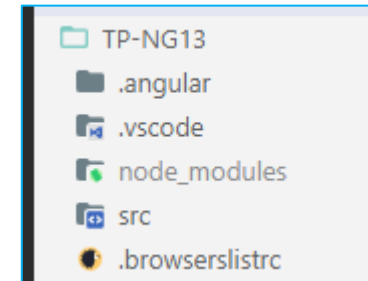
PORTS

PS C:\Angular-releases> npx @angular/cli@12 new TP-NG12

Les versions marquantes Angular

- Version 13 (3 novembre 2021)

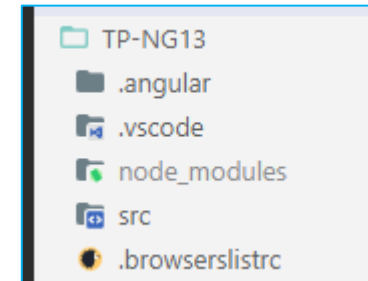
- Arrivée du cache .angular => accélération des temps de compilation
- démarrage en mode dév plus rapide
- Travail continu d'optimisation de IVY et ViewEngine



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 `{}`

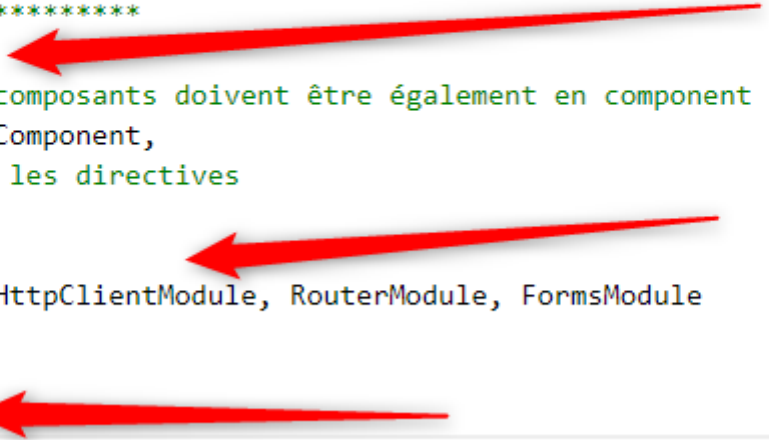


Les versions marquantes Angular

- 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

GRUPE 1

- main.ts src

GRUPE 2

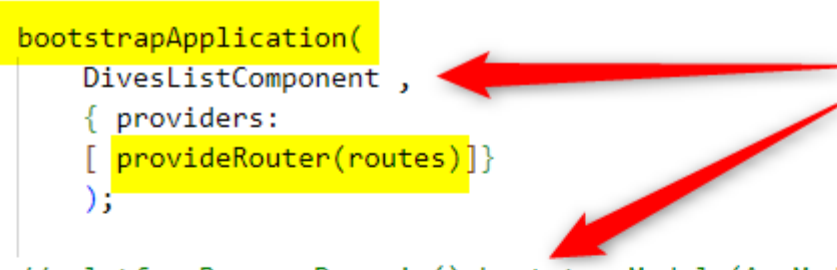
- divs-list.component.ts src\app\web...

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/divs-
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
```



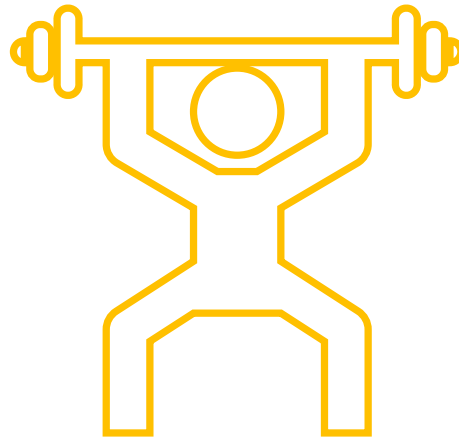
Les versions marquantes Angular

- 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

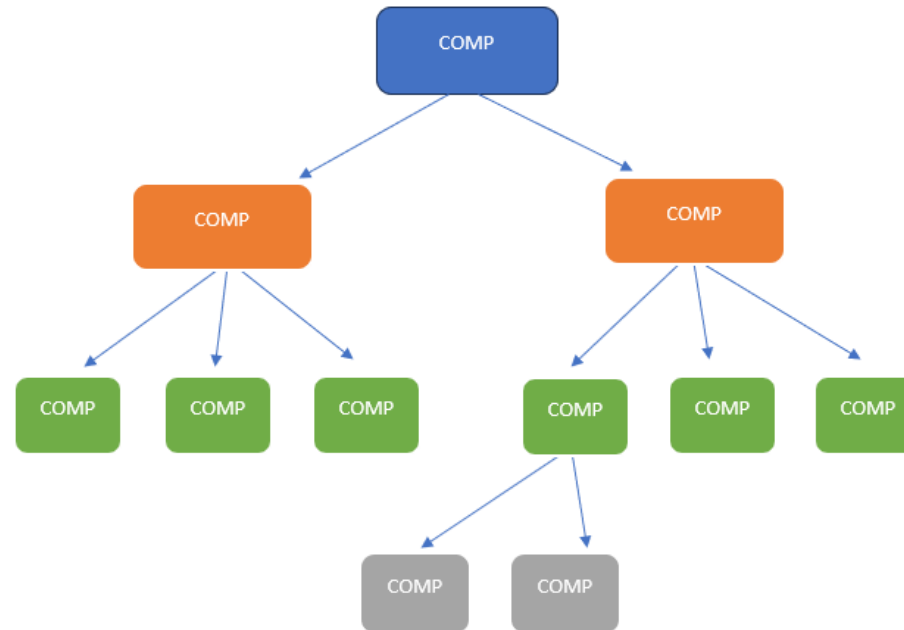
Lorsque des propriétés d'entrée (**Input**) ou des événements d'entrée sont **déclenchés**, Angular met à jour **l'arbre des composants pour détecter d'éventuels changements** à apporter aux affichages des Vues.

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

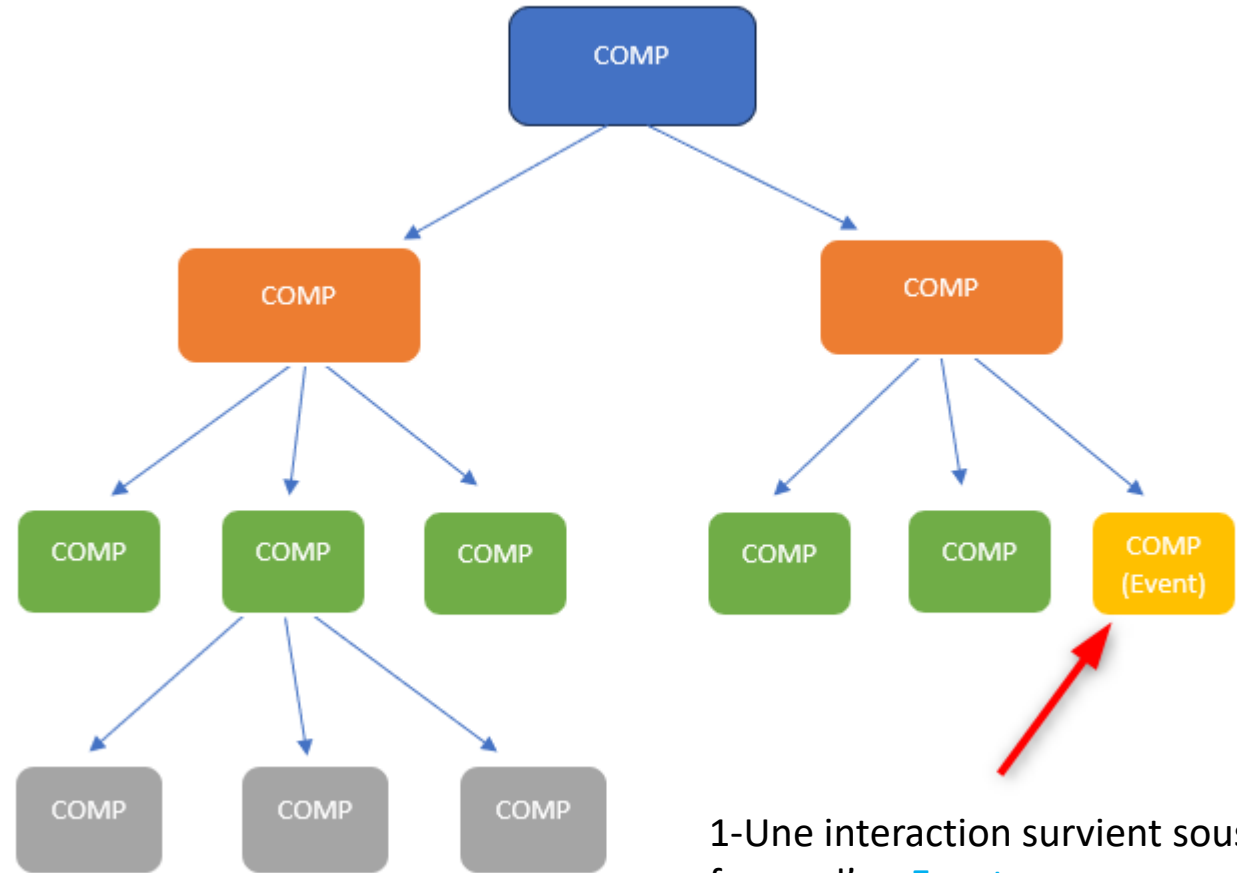
Comment Angular met à jour l'affichage des composants imbriqués par des Inputs, dans une stratégie composants parents-enfants



Le Detection Change Mode par défaut

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

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



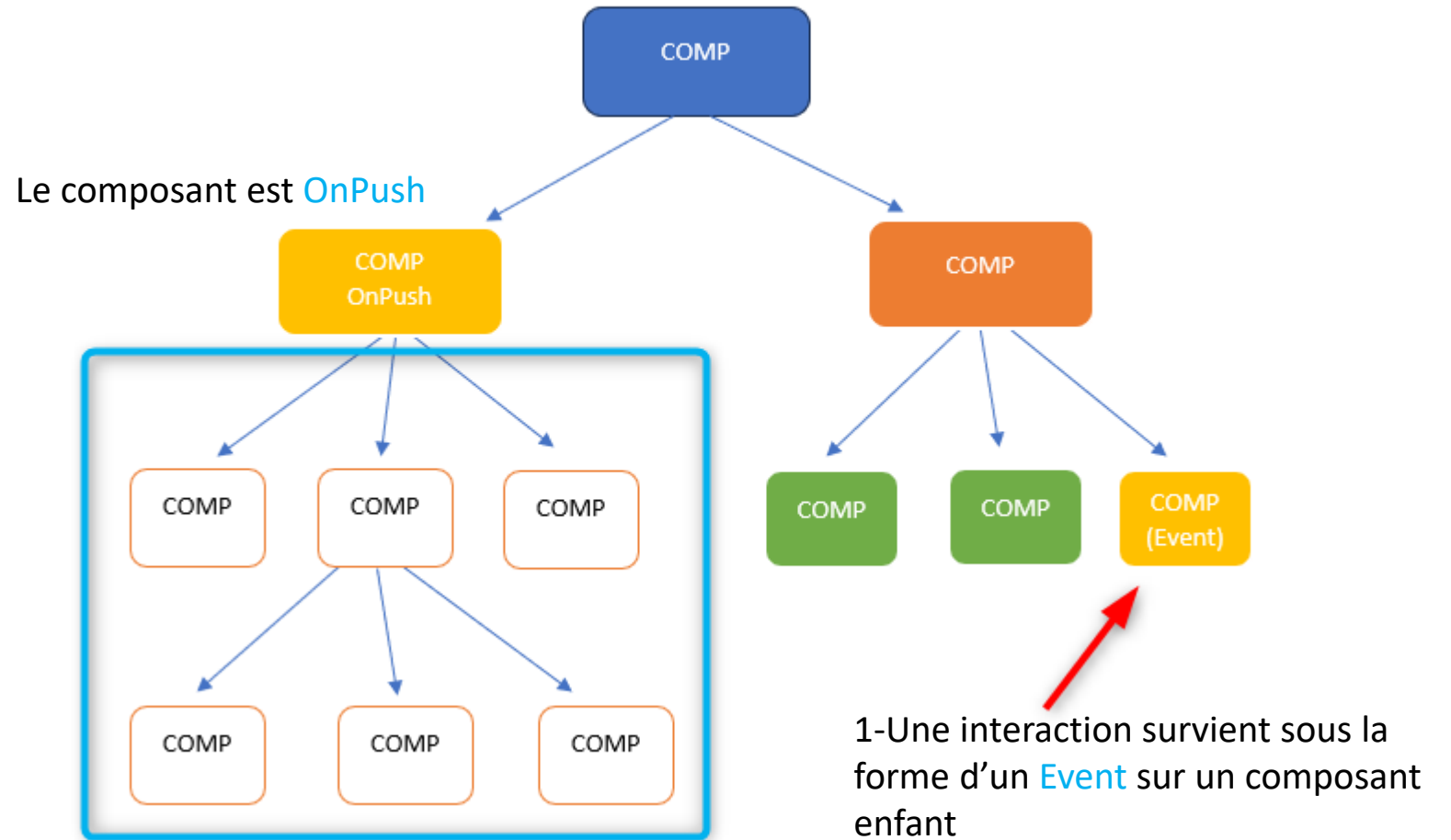
1-Une interaction survient sous la forme d'un **Event** sur un composant enfant

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...**



La class ChangeDetectorRef

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.



```
films-details.component.ts x
webApp2 > src > app > webApp > root > films > composants > films-details > films-details.component.ts > FilmsDe

4  @Component({
5    selector: 'app-films-details',
6    templateUrl: './films-details.component.html',
7    styleUrls: ['./films-details.component.scss'],
8    changeDetection: ChangeDetectionStrategy.OnPush
9  })
10 export class FilmsDetailsComponent {
11
12   @Input() childFilm: Films;
13   // @Input() set childInfos(value: string) { // console.log(value); };
14   @Output() outputEvent: EventEmitter<any>;
15
16   // -----
17   constructor(
18     private _cd: ChangeDetectorRef
19   ) {
20     this.childFilm = new Films;
21     this.outputEvent = new EventEmitter<any>;
22   }
```

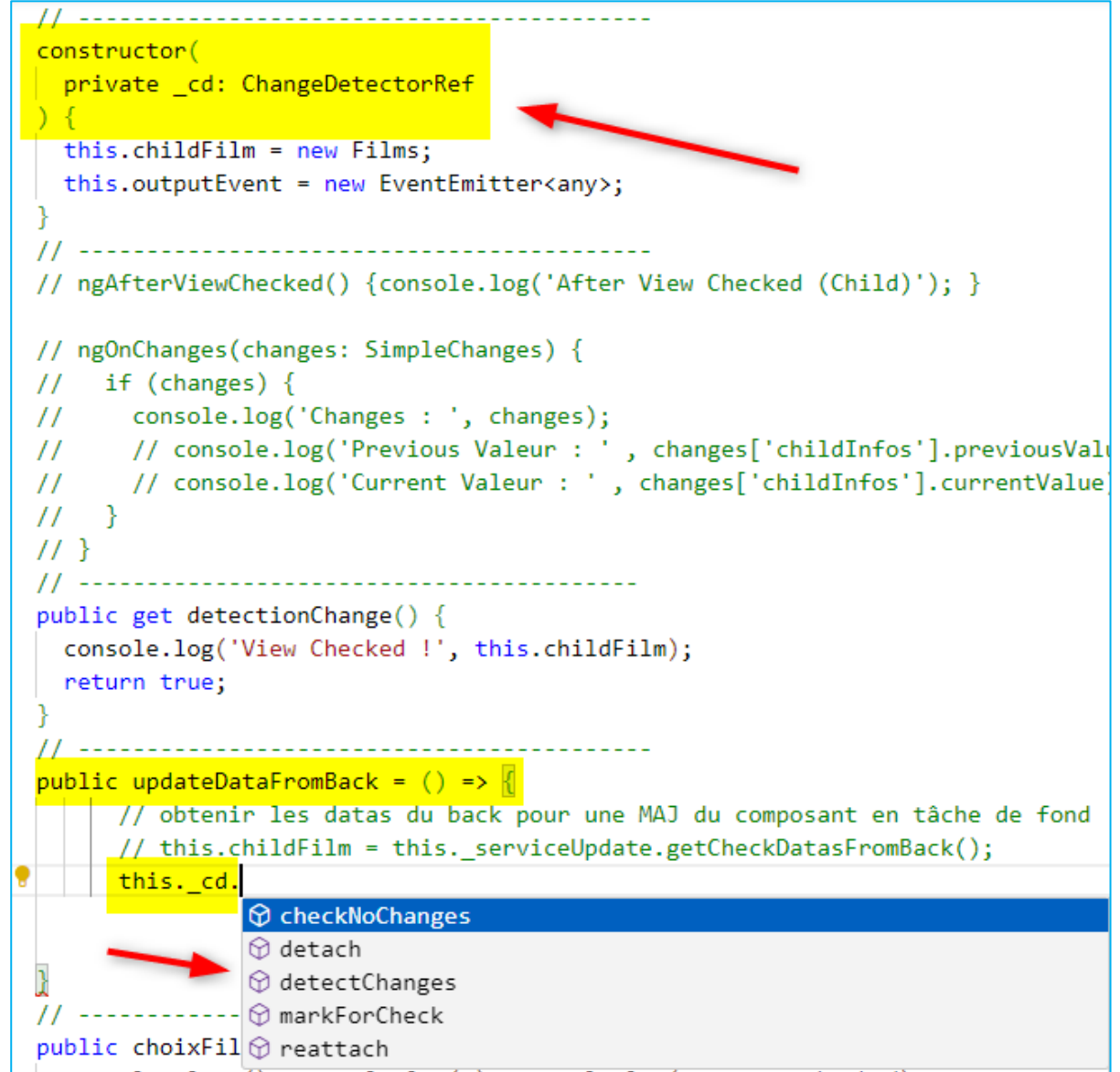
The image shows a code editor window titled 'films-details.component.ts'. The code defines a Vue component 'FilmsDetailsComponent' using the '@Component' decorator. Line 8, 'changeDetection: ChangeDetectionStrategy.OnPush', is highlighted in yellow, with a red arrow pointing to it from the right. The component class has an '@Input()' for 'childFilm' and an '@Output()' for 'outputEvent'. The 'constructor' (lines 17-22) is also highlighted in yellow, with a red arrow pointing to it from the right. Inside the constructor, a private property '_cd' of type 'ChangeDetectorRef' is declared on line 18.

La class ChangeDetectorRef

« **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**

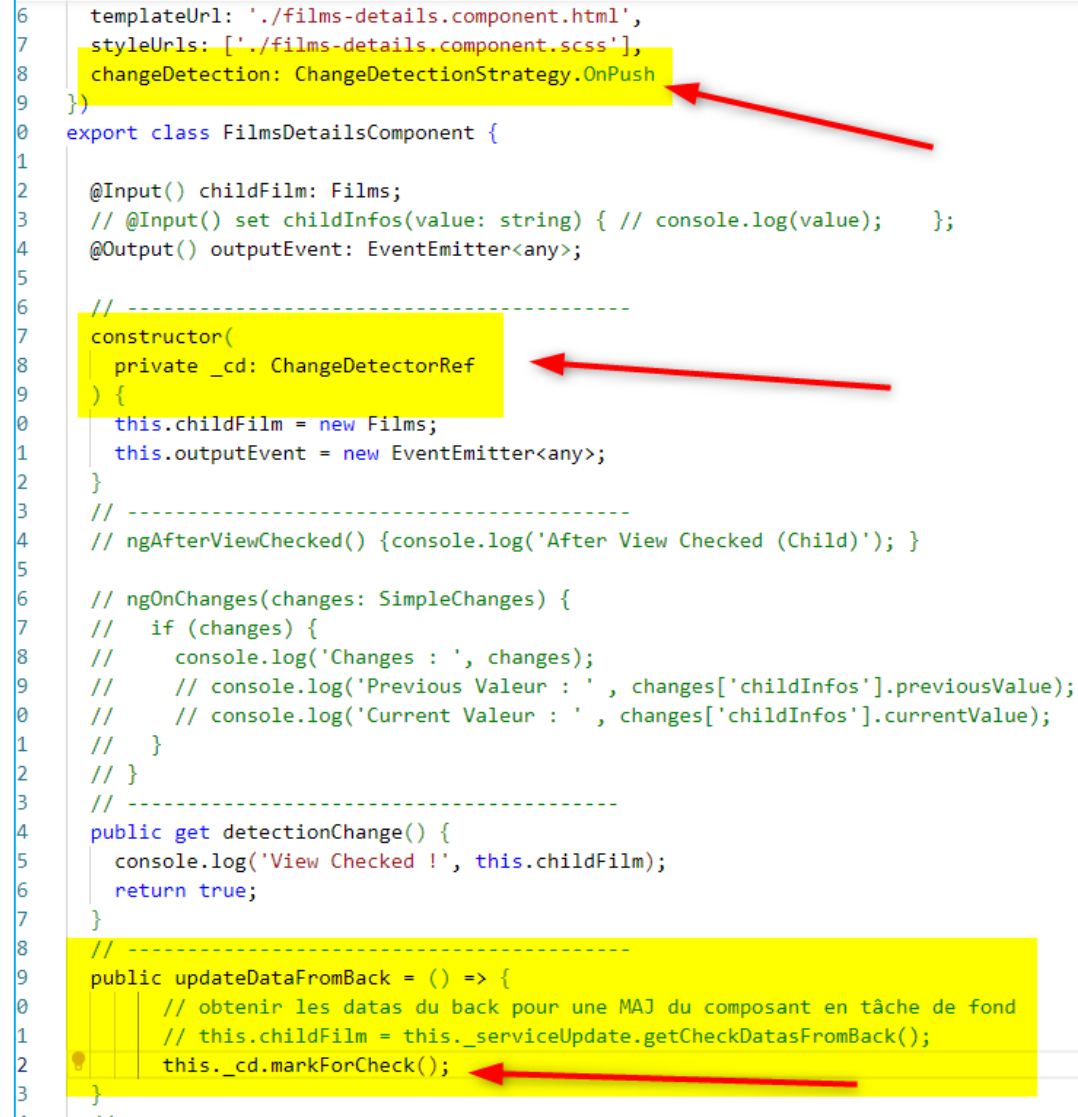
```
// -----  
public updateDataFromBack = () => {  
    // obtenir les datas du back pour un MAJ du composant en tâche de fond  
    // this.childFilm = this._serviceUpdate.getCheckDatasFromBack();  
    this._cd.markForCheck();  
}
```

```
// -----  
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'].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.  
    // -----  
    public choixFil
```



La class ChangeDetectorRef

```
6   templateUrl: './films-details.component.html',
7   styleUrls: ['./films-details.component.scss'],
8   changeDetection: ChangeDetectionStrategy.OnPush
9 })
10 export class FilmsDetailsComponent {
11
12   @Input() childFilm: Films;
13   // @Input() set childInfos(value: string) { // console.log(value);   };
14   @Output() outputEvent: EventEmitter<any>;
15
16   // -----
17   constructor(
18     private _cd: ChangeDetectorRef
19   ) {
20     this.childFilm = new Films;
21     this.outputEvent = new EventEmitter<any>;
22   }
23   // -----
24   // ngAfterViewChecked() {console.log('After View Checked (Child)'); }
25
26   // ngOnChanges(changes: SimpleChanges) {
27   //   if (changes) {
28   //     console.log('Changes : ', changes);
29   //     // console.log('Previous Valeur : ' , changes['childInfos'].previousValue);
30   //     // console.log('Current Valeur : ' , changes['childInfos'].currentValue);
31   //   }
32   // }
33   // -----
34   public get detectionChange() {
35     console.log('View Checked !', this.childFilm);
36     return true;
37   }
38   // -----
39   public updateDataFromBack = () => {
40     // obtenir les datas du back pour une MAJ du composant en tâche de fond
41     // this.childFilm = this._serviceUpdate.getCheckDatasFromBack();
42     this._cd.markForCheck();
43   }
44 }
```



Le Signal

Meilleures **performances** en termes d'exécution !

Le signal réduit la **complexité du parcours de l'arbre des composants parents-enfants**

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

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

Le signaux peuvent interagir avec RXJS et apporter de la **complémentarité** !

On peut passer facilement d'un Observable à un signal et vice –versa

toSignal() toObservable()

Le Signal

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

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

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

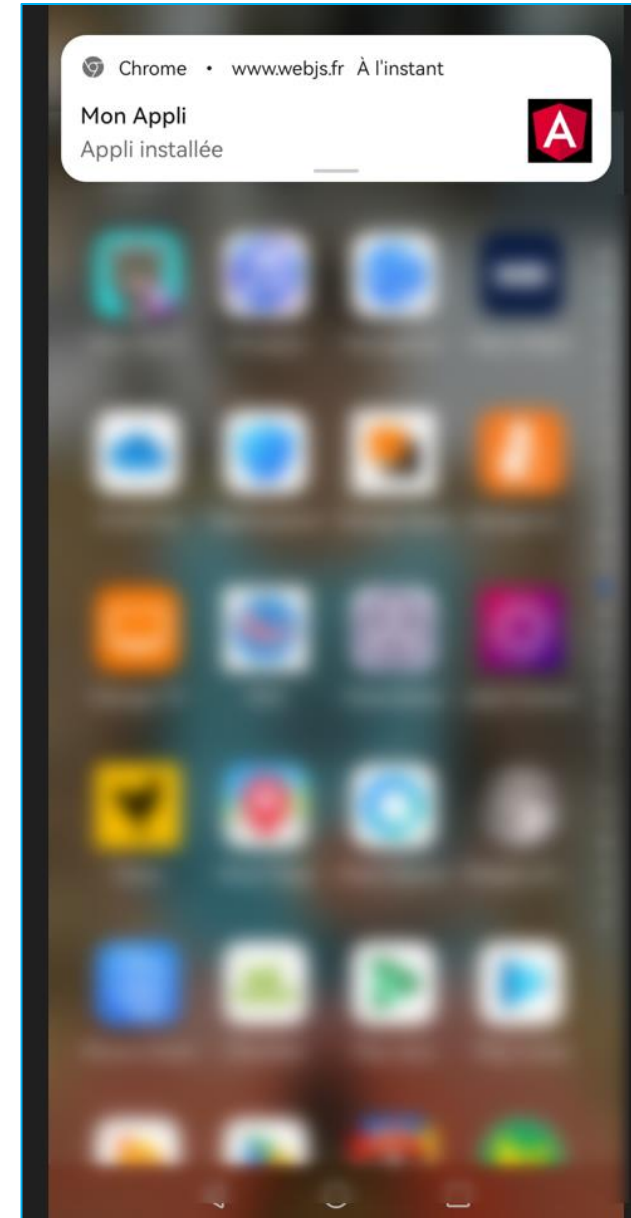
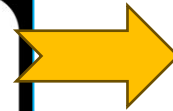
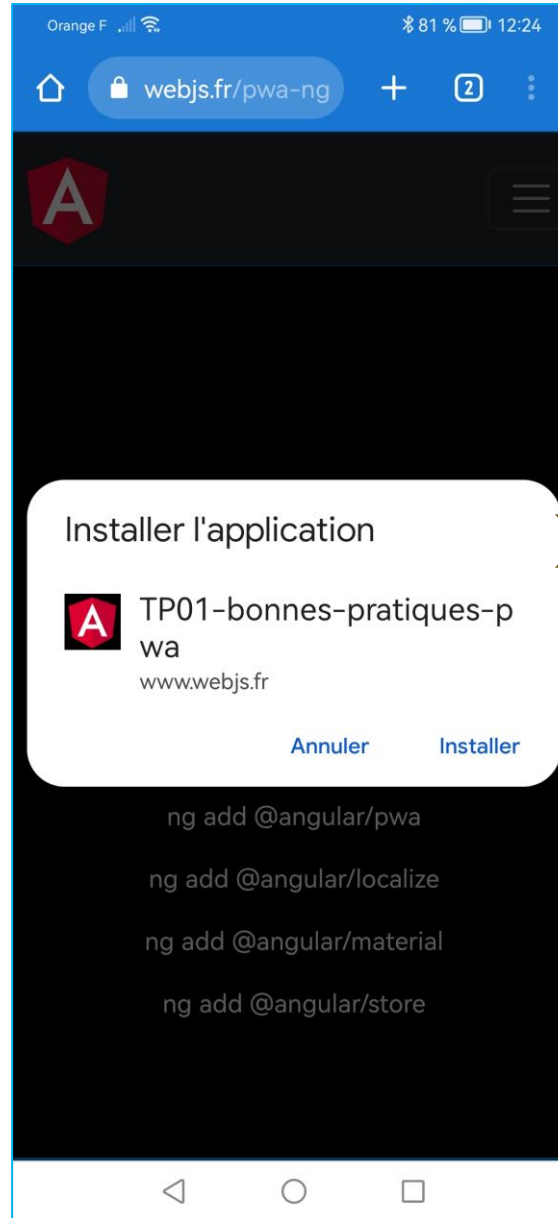
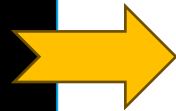
PWA – les Progressive Web Apps

Mise en situation :

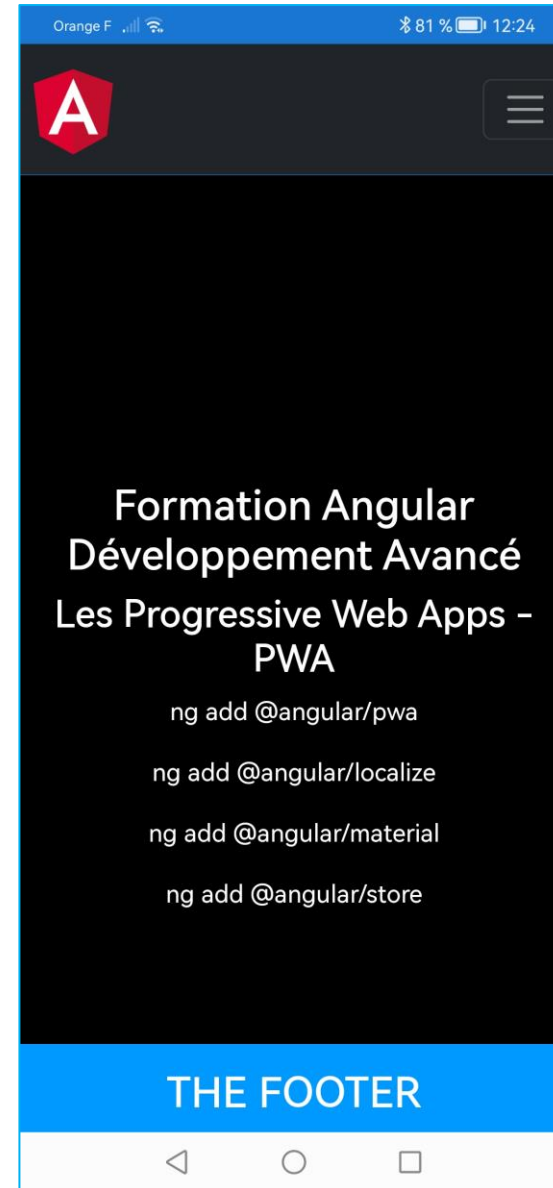
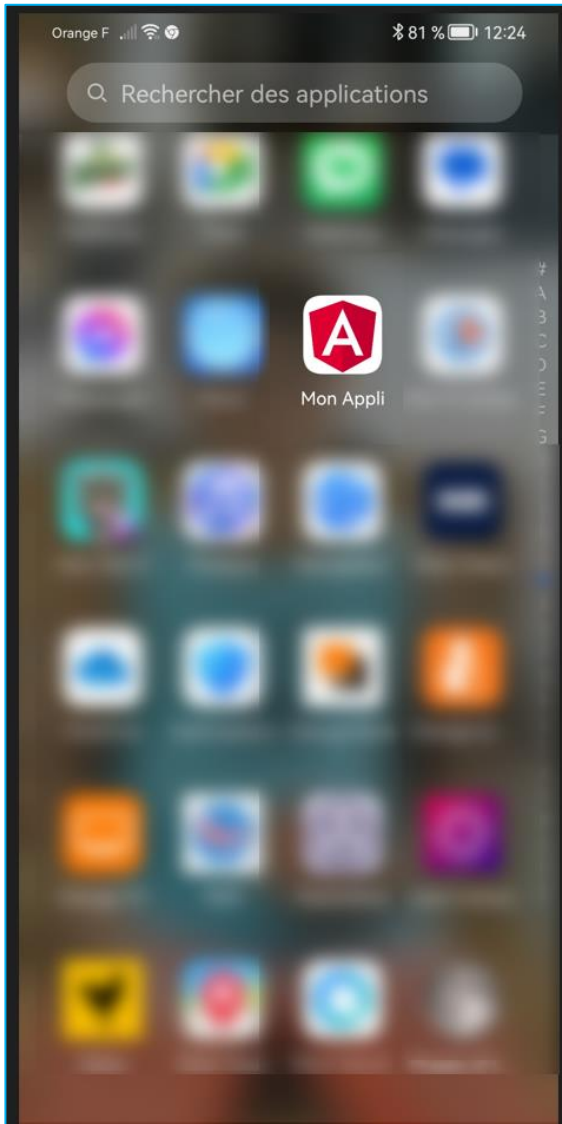
consulter cette appli depuis l'URL : <https://webjs.fr/pwa-ng/>

ng add @angular/pwa

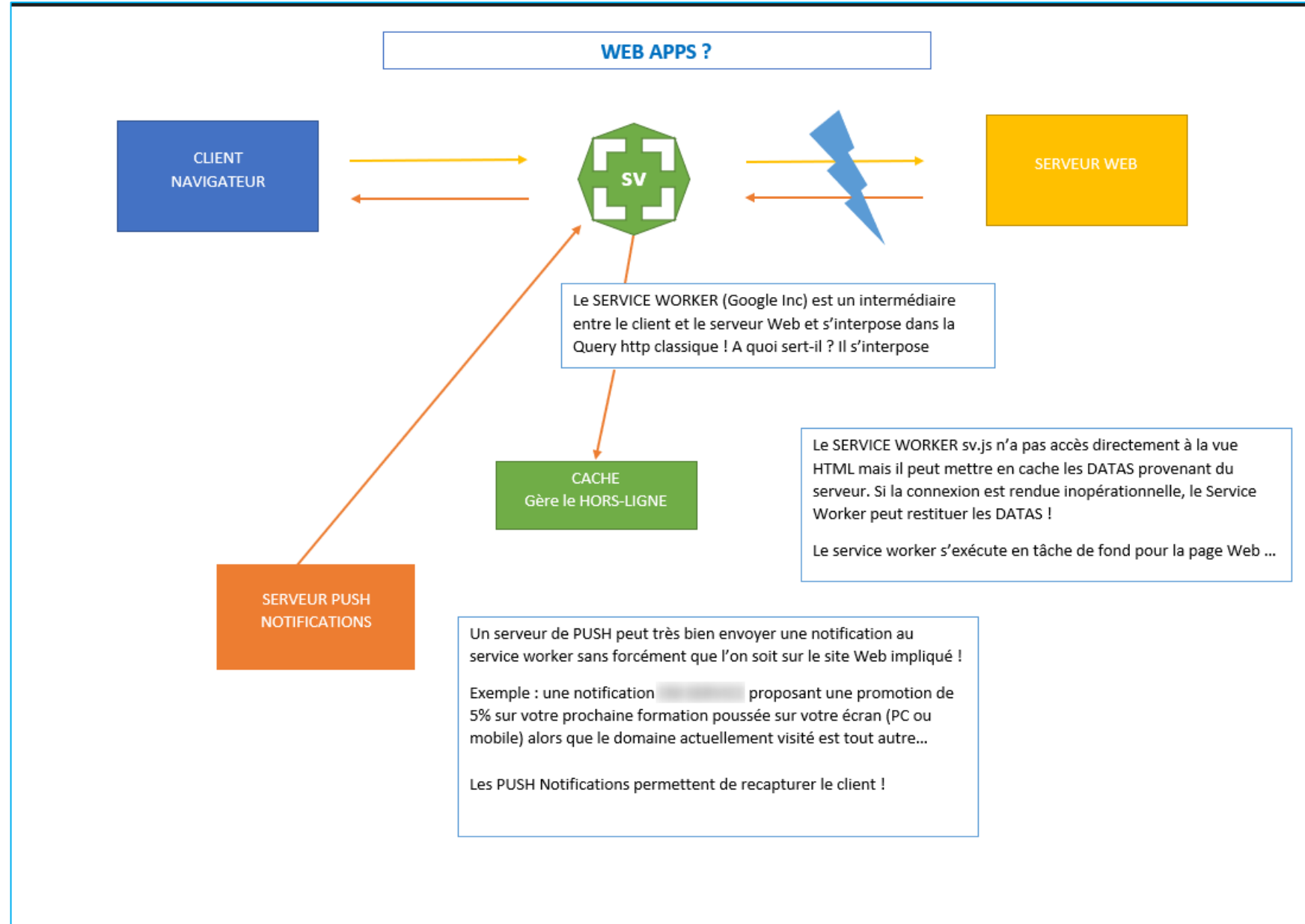
PWA – les Progressive Web Apps



PWA – les Progressive Web Apps



PWA – les Progressive Web Apps



PWA – les Progressive Web Apps

The screenshot shows the Chrome DevTools Application panel for a Progressive Web App (PWA) at `webjs.fr/pwa-ng/`. The interface is divided into several sections:

- Application:** Contains a tree view on the left with `Manifest`, `Service workers` (highlighted with a red arrow), and `Storage`.
- Storage:** Contains a tree view with `Local storage`, `Session storage`, `IndexedDB` (highlighted with a red arrow), `Web SQL`, `Cookies`, `Private state tokens`, `Interest groups`, `Shared storage`, and `Cache storage`.
- Background services:** A list of background services including `Back/forward cache`, `Background fetch`, `Background sync`, `Bounce tracking mitigations`, `Notifications`, `Payment handler`, `Periodic background sync`, `Push messaging`, and `Reporting API`. This section is highlighted with a red box.
- Service workers:** The main panel shows the service worker for `https://webjs.fr/pwa-ng/`. It includes a `Source` field with `ngsw-worker.js` (highlighted with a red arrow), a `Received` timestamp of `04/10/2023 17:44:53`, and a `Status` of `#3396 activated and is running` with a `stop` link. Below this are controls for `Push` (with a text input `Test push message from DevTools.` and a `Push` button), `Sync` (with a text input `pwa` and a `Sync` button), and `Periodic Sync` (with a text input `periodic` and a `Periodic Sync` button).
- Update Cycle:** A table showing the update cycle for the service worker.
- Service workers from other origins:** A section at the bottom with a link `See all registrations`.

Version	Update Activity	Timeline
▶ #3396	Install	
▶ #3396	Wait	
▶ #3396	Activate	