ORSYS formation

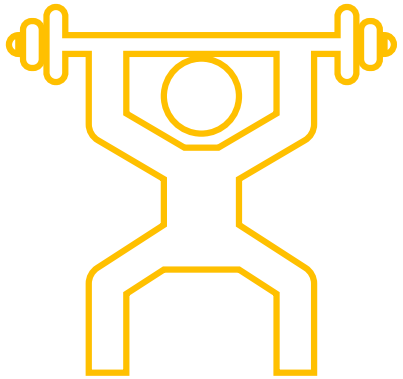
Formation ANGULAR Développement Avancé

Chapitre 4

Animé par Michel BOCCIOLESI

Table des matières

Les versions marquantes !



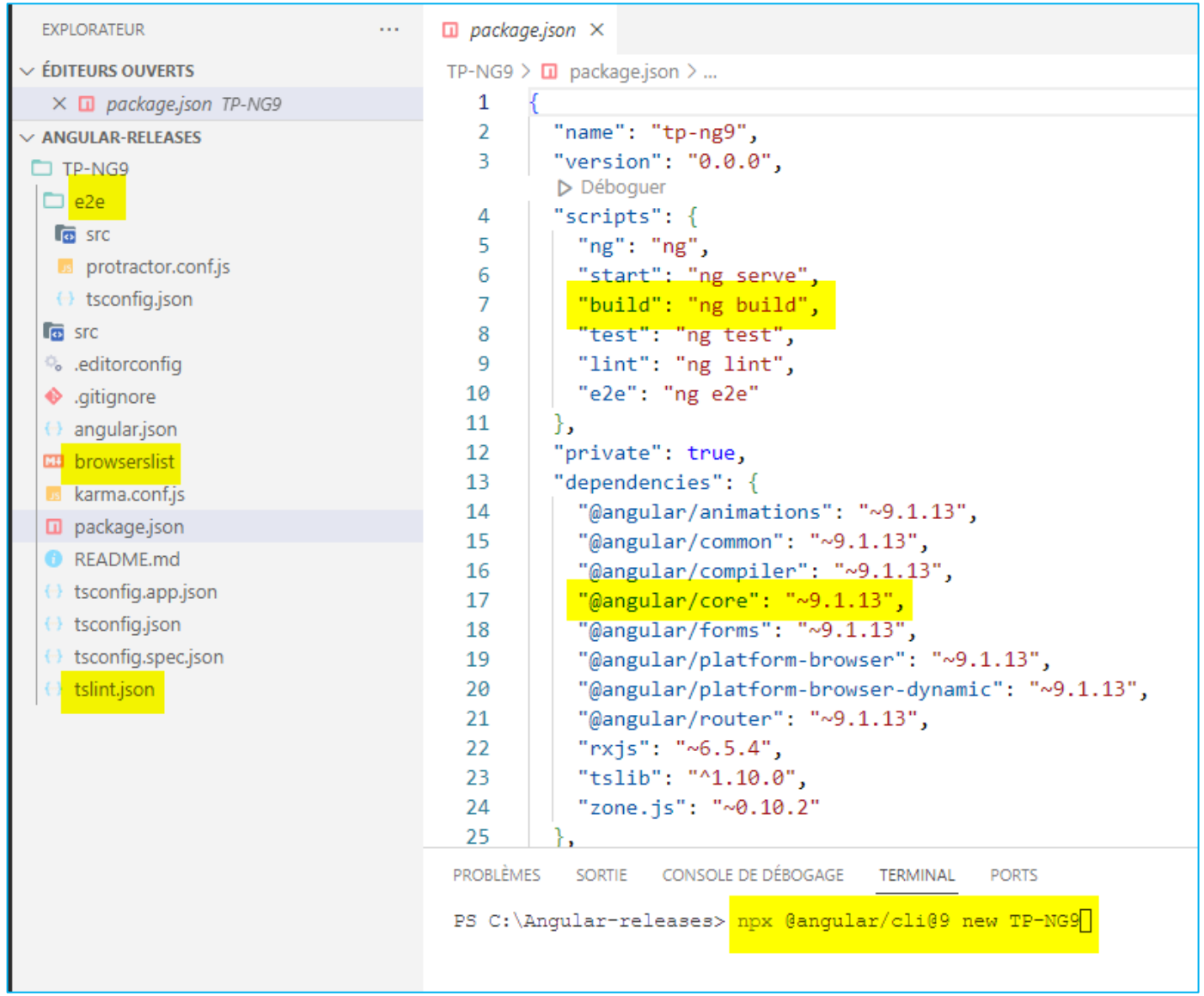
Les versions marquantes Angular

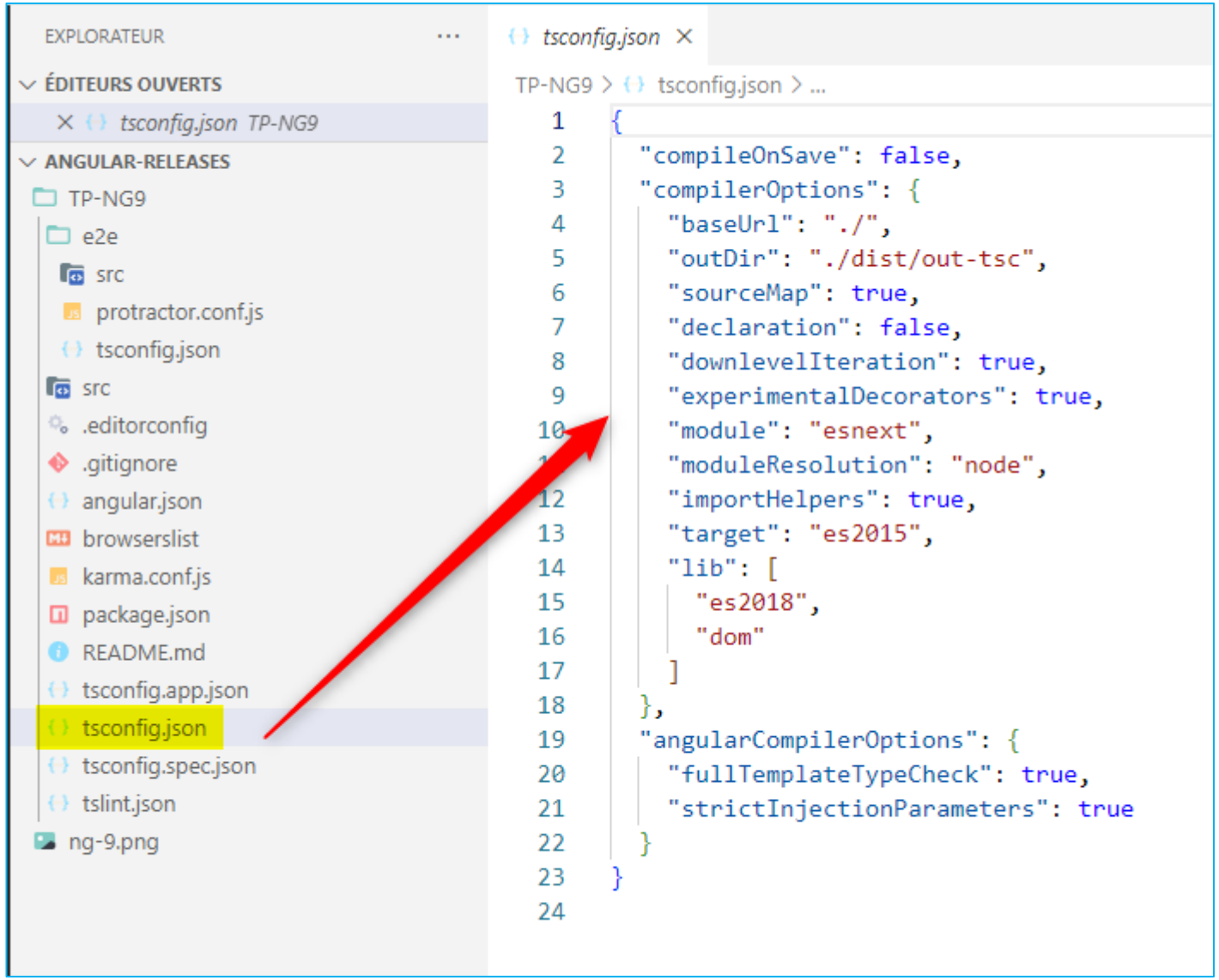
- Version 9 (6 février 2020)

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

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

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


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


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

Les versions marquantes Angular

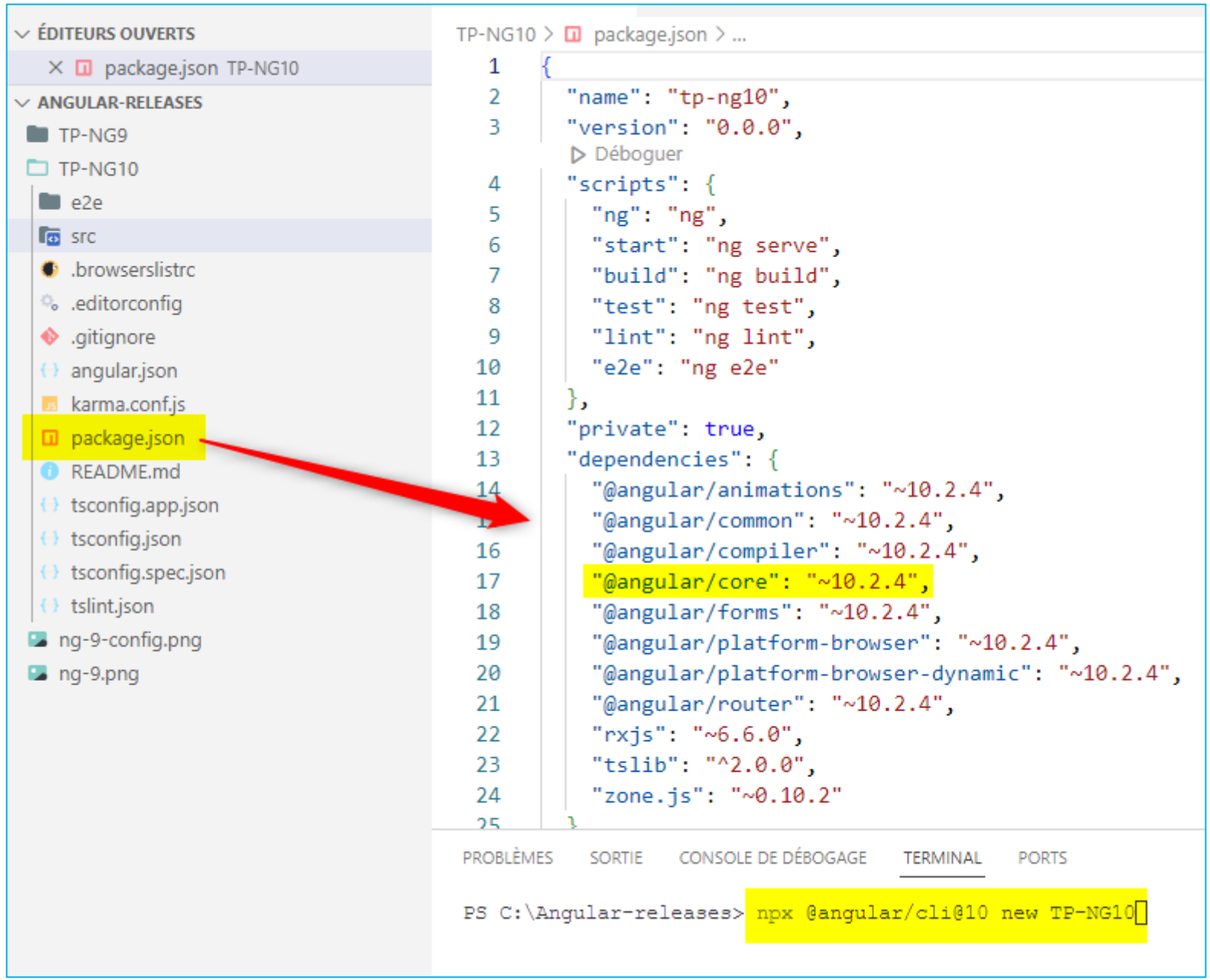
•Version 10 (24 juin 2020)

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

Rappels :

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

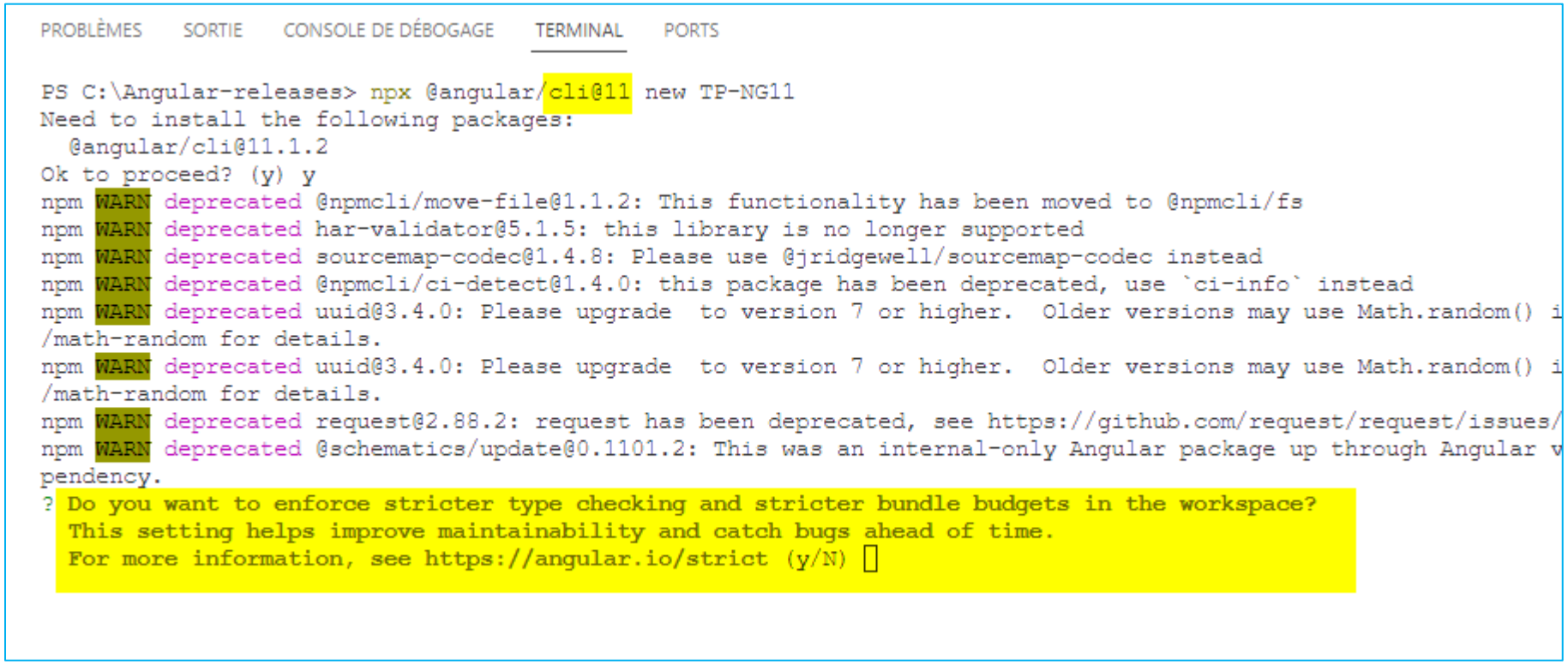
- nouveau datePicker dans Angular Matérial


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

Les versions marquantes Angular

•Version 11 (17 novembre 2020)

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



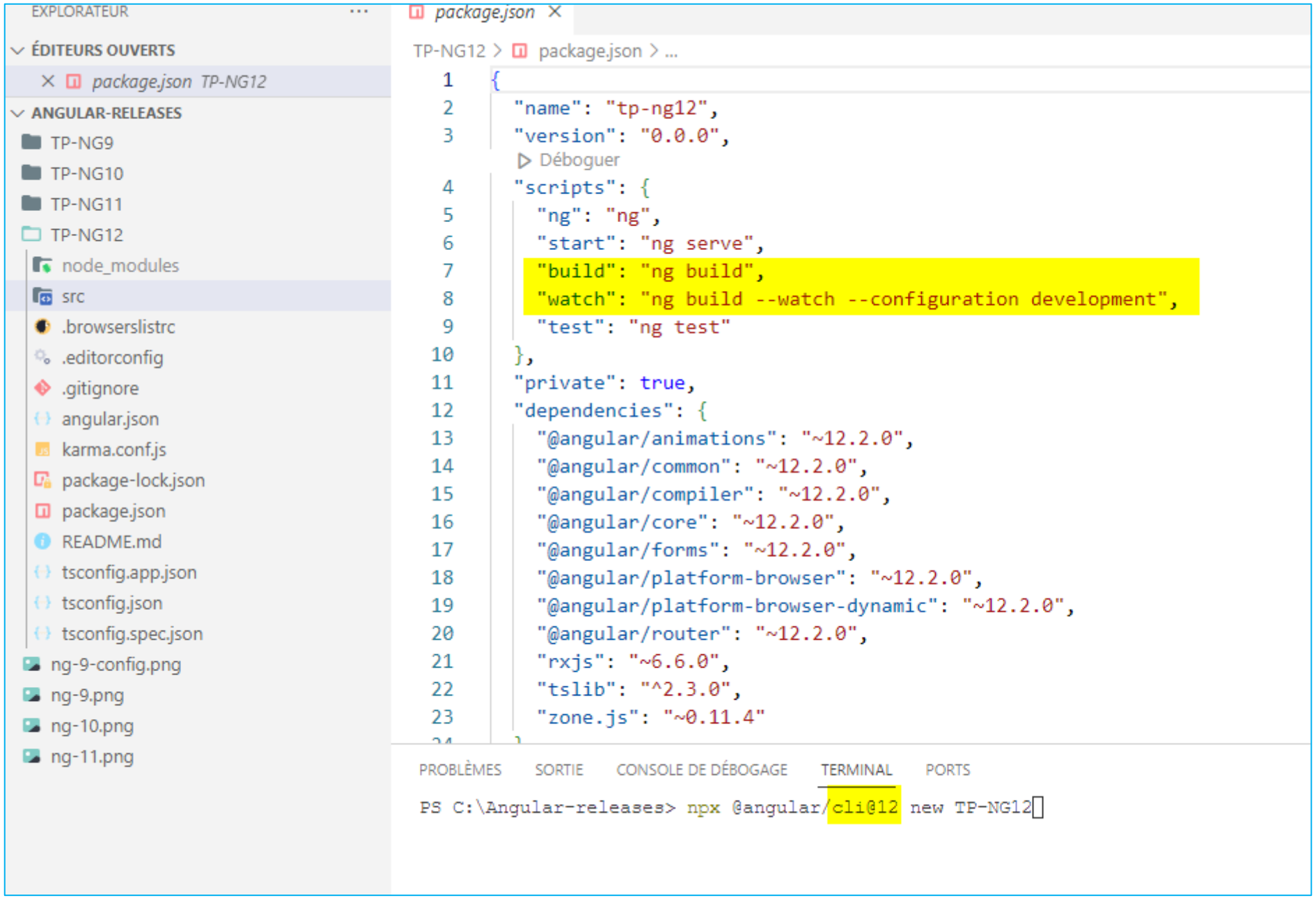
Les versions marquantes Angular

•Version 12 (19 mai 2021)

- Abandon progressif de la compatibilité et support IE

validé avec NG13

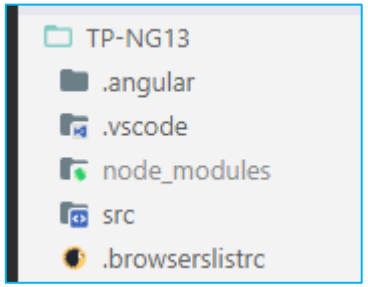
- Fin du support de e2e, protractor..
- TSlint déprécié depuis 2019 est remplacé par ESlint
- script watch qui remplace et améliore build
- build est désormais le mode prod


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

Les versions marquantes Angular

•Version 13 (3 novembre 2021)

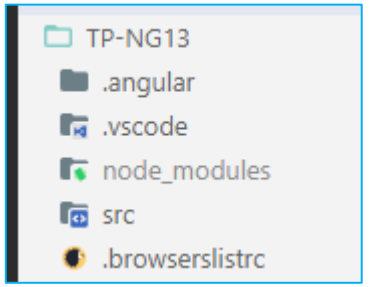
- Arrivée du cache .angular => accélération des temps de compilation
- démarrage en mode dév plus rapide
- Travail continuel d'optimisation de IVY et ViewEngine



Les versions marquantes Angular

•Version 14 (11 juin 2022)

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



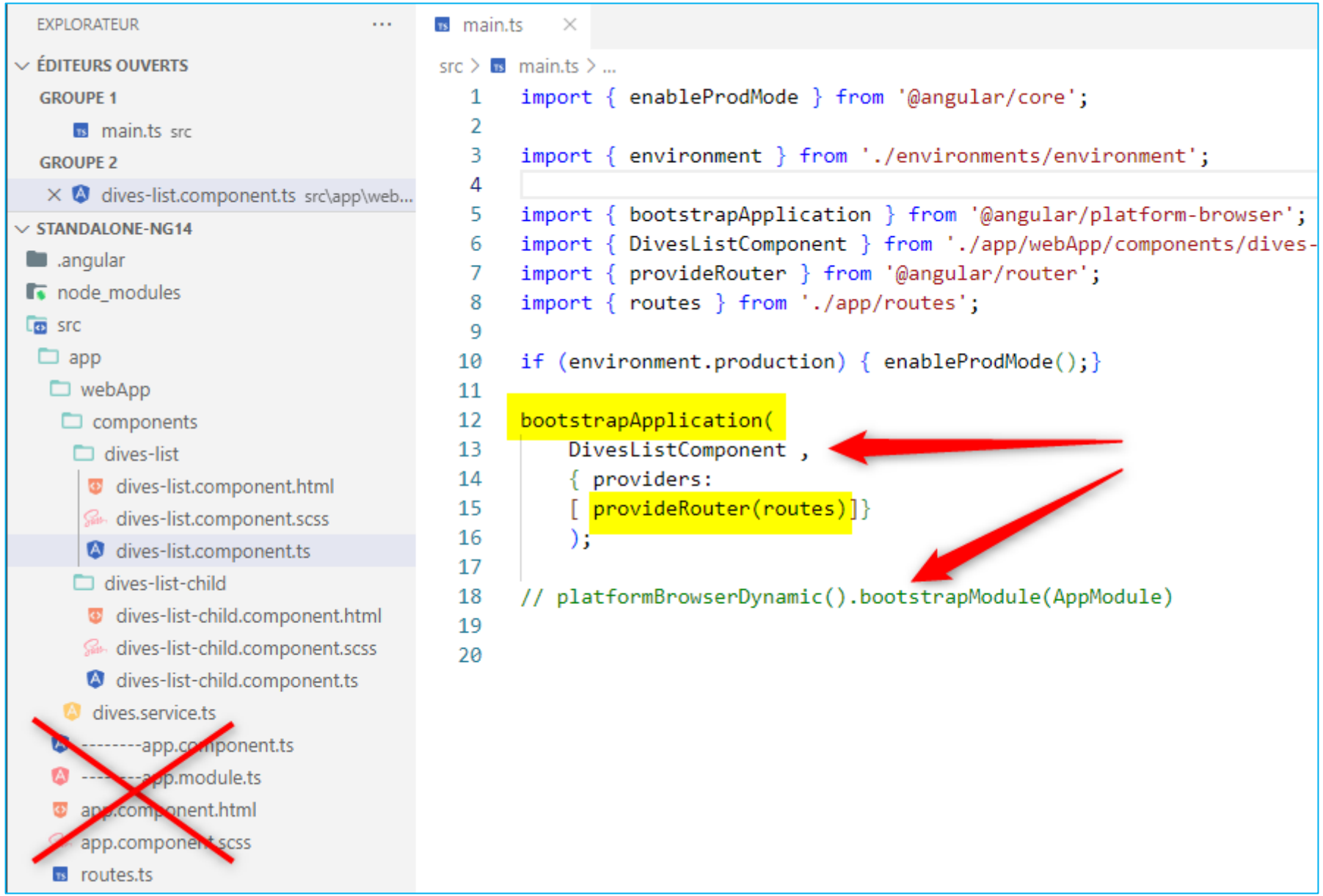
Les versions marquantes Angular

•Version 15 (16 novembre 2022)

- composants standalone

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

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


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

Les versions marquantes Angular

- Version 16 (12 mai 2023)

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

Angular 17 - La renaissance !

Angular 17

Version 17 (8 novembre 2023) https://angular.dev/

- Le control flow est totalement revu
*ngIf et *ngFor sont remplacés par @if et @for

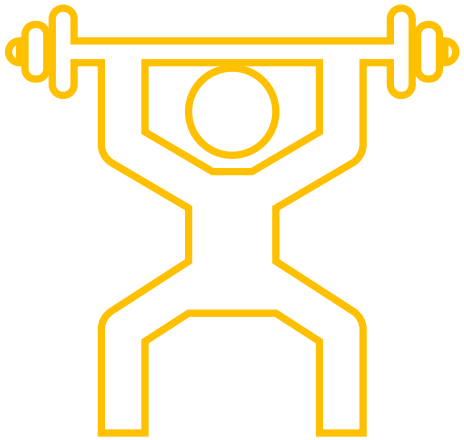
- Webpack est remplacé par ESBUILD (ng server et nb build iront 2 fois plus vite !

- SSR par défaut (Server Side Rendering = > SEO …)

- Lazy Loading (Vues différées) @defer @placeholder

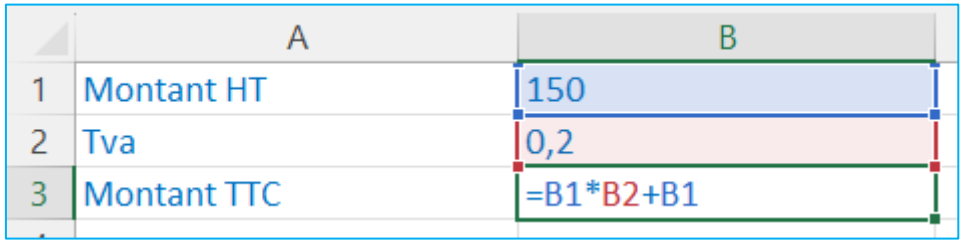
- Signal est abouti en tant qu'API de réactivité

Le Detection Change Mode
Nouveautés NG16 : Signal
Sucre Syntaxique ?
Réactivité ?
Rxjs ?



La notion de réactivité en Front End

Finalement quel Problème doit on résoudre ?
Tous les Frameworks modernes essaient de résoudre ce même problème



	A	B
1	Montant HT	150
2	Tva	0,2
3	Montant TTC	=B1*B2+B1

index.html × script.js ×

```
1  let montantHT=150;
2  const TVA=0.2;
3  let total=montantHT*TVA + montantHT;
4  console.log(total);
5
6  let montantHt=200;
7  console.log('Est ce réactif ? : ', total, '🤔😠');
8
```

Console ×

```
180
Est ce réactif ? :  180 🤔😠
```

La notion de réactivité en Front End

Comment implémenter cela dans nos applis JS ? (Google Sheets l'a déjà fait avec du « *fichier Excel* » en JS)

3 Méthodes pour implémenter la réactivité dans les Frameworks JS :

1- La méthode **Value Based** :

L'état actuel d'un composant est stockée dans une zone mémoire précise (store, composant.ts, local Storage, etc …)
le Framework utilise le mode « **Detection Change[1] »** ou **« *Dirty Checking*[2]** » pour savoir si la valeur a changé car il ne peut pas l'observer, il parcourt l'ensemble des composants pour vérifier les différents états et mettre à jour le DOM. La librairie actuelle* Zone.js fait ce travail de vérification.

Avantages : aucunes connaissances du « **core** » **Zone**.js n'est requise et c'est extrêmement simple à utiliser pour le dév. Le DOM est automatiquement MAJ.

Inconvénients: Performances 😬, des quantités de traitements sont faits pour mettre à jour les Vues HTML. Pour améliorer cela, il faut devenir expert en **Zone** et utiliser le **change detector** et le **onPush** !

1 : Pour Angular
2 : utilisé dans d'autres framework

La notion de réactivité en Front End

3 Méthodes pour implémenter la réactivité dans les Frameworks JS :

2- La méthode **Observable Based**:

Cette méthode est beaucoup plus performante que la « Value based » mais demande une solide investigation et des temps d'intégration et de compréhension conséquents.

La notion d'abonnement permet de mettre à jour le DOM uniquement lorsqu'on en aura besoin !

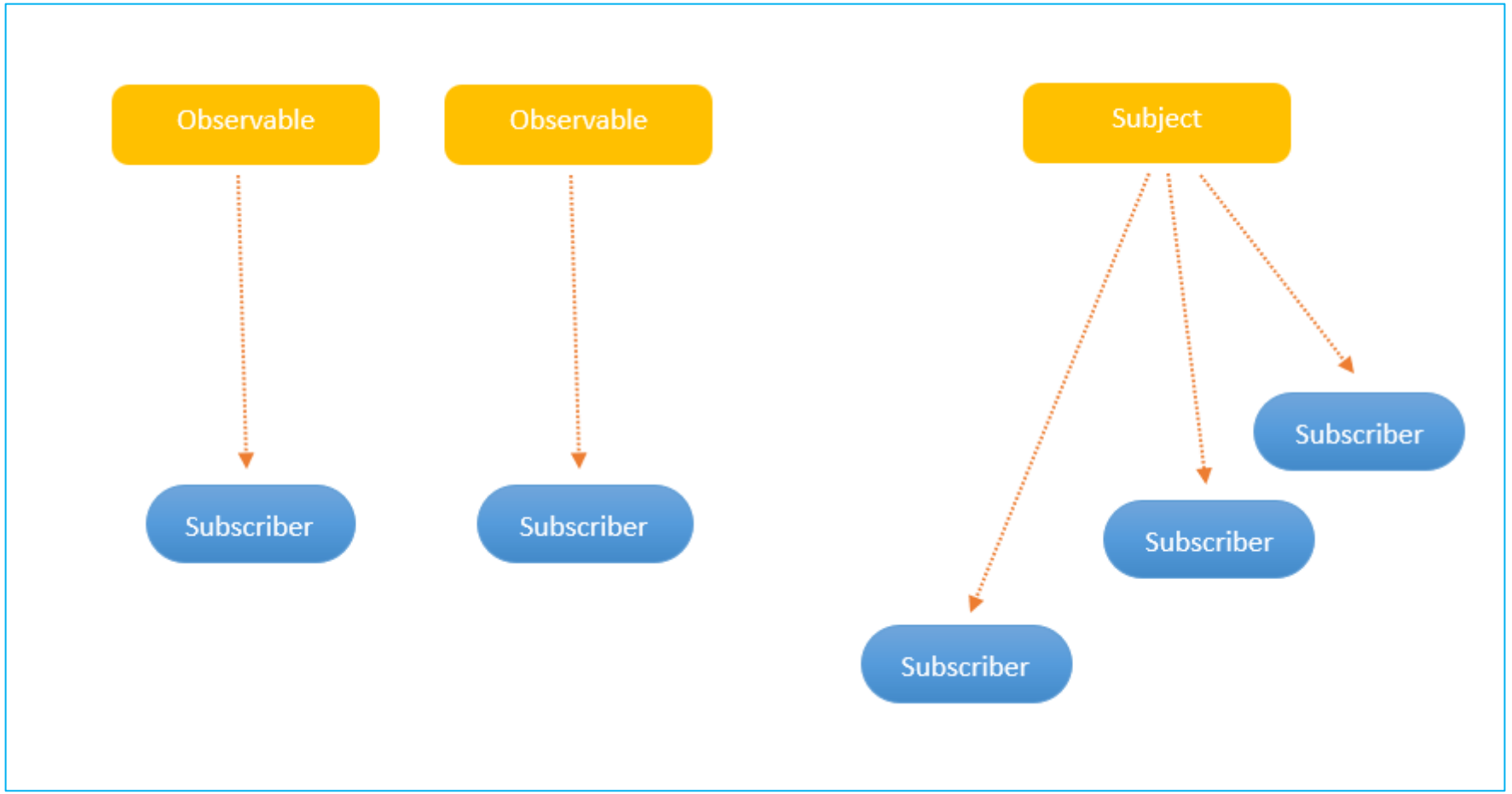
*Rappel :
1 « Observable » est un Objet qui émet une séquence de valeurs
1 « Observer » observe l'observable et réagit à l'arrivée de **nouvelles** valeurs

« Subject » et « Behaviour Subject (avec valeur initiale) » sont en même temps Observables et Observers :
Le « Subject » est multidiffusion par rapport à l'observable qui est monodiffusion

La notion de réactivité en Front End

Le « Subject » est multidiffusion par rapport à l'observable qui est monodiffusion, il faut user de complexité pour recevoir les valeurs du même Observable par plusieurs « Subscribers » 🙄

Plusieurs Subscriber peuvent s'attacher au Subject et recevoir les valeurs émises. 😊



Le Detection Change Mode Angular

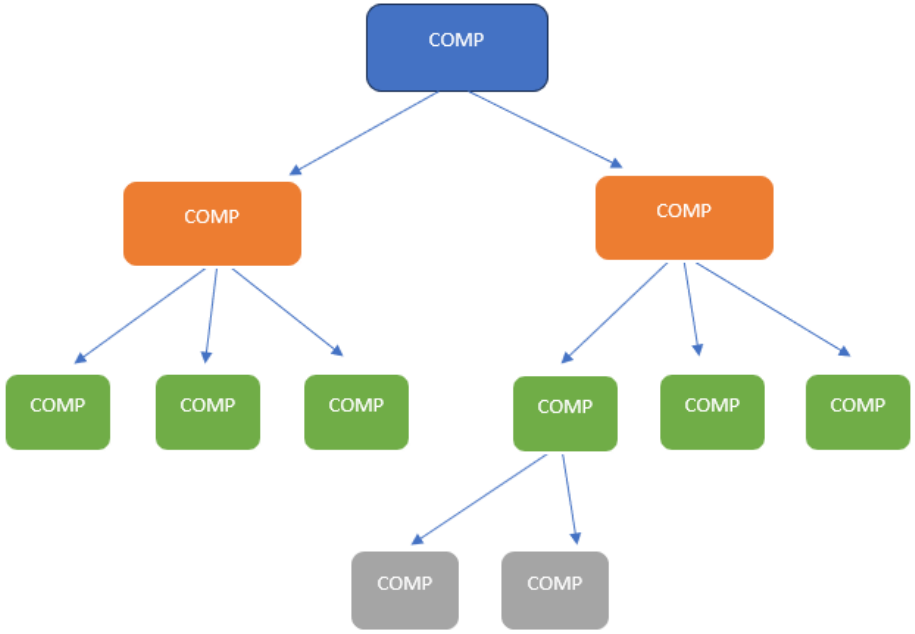
Lorsque des propriétés d'entrée (Input) ou des événements d'entrée sont déclenchés, Angular met à jour l'arbre des composants pour détecter d'éventuels changements à apporter aux affichages des Vues.

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

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

Le Detection Change Mode Angular

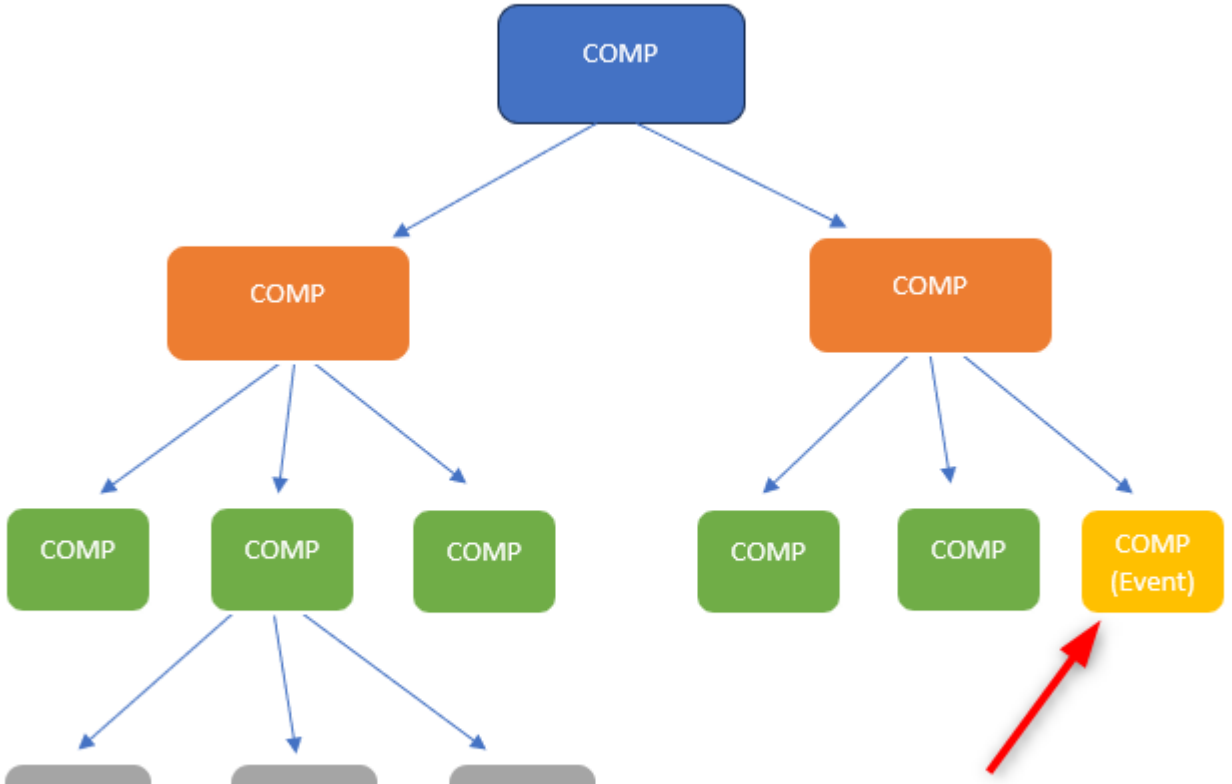
Comment Angular met à jour l'affichage des composants imbriqués par des Inputs,
dans une stratégie composants parents-enfants



Le Detection Change Mode par défaut

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

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



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

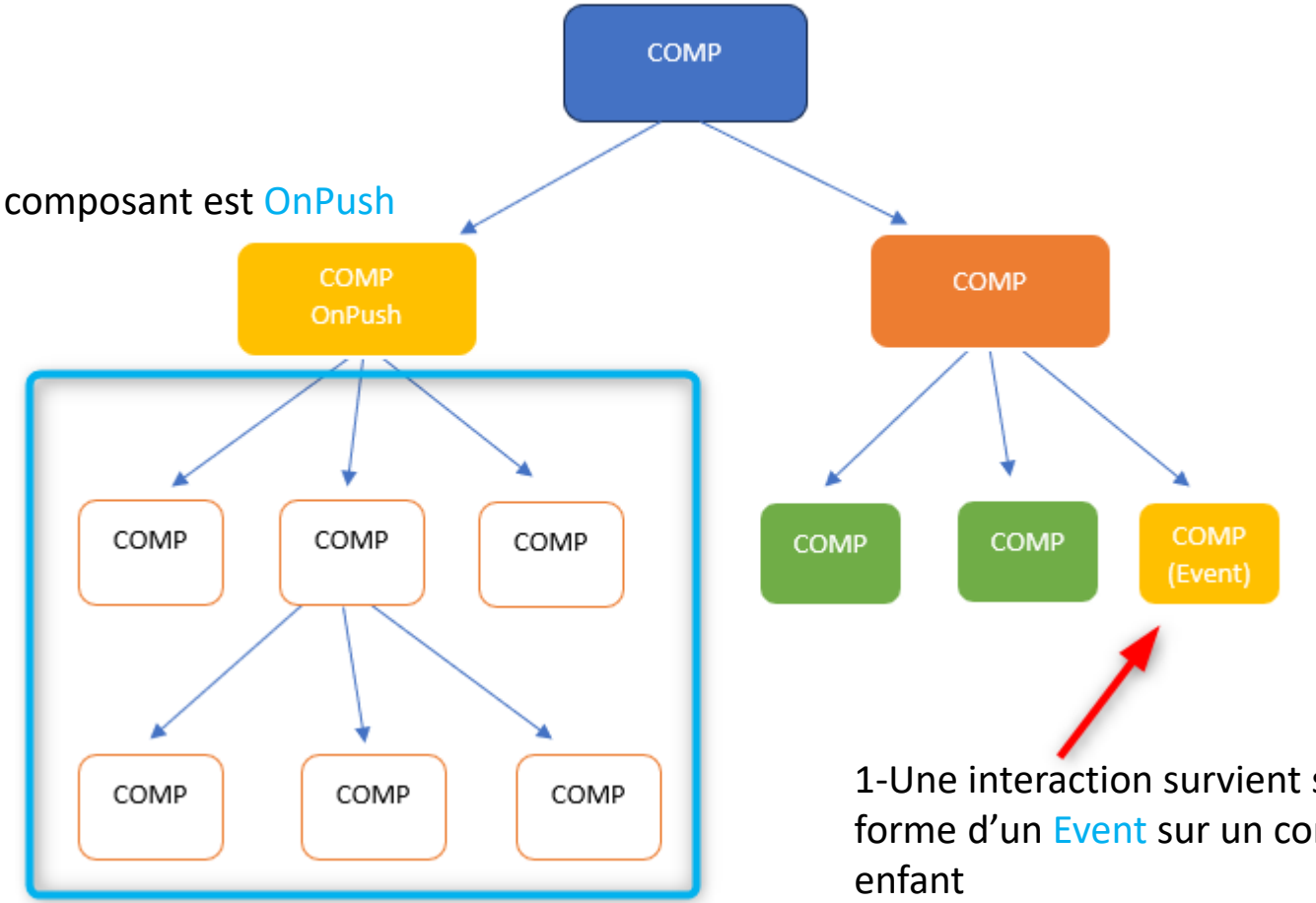
Le Detection Change Mode onPush

Angular limite le nombre de vérifications sur les enfants

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

Angular arrête son parcours…

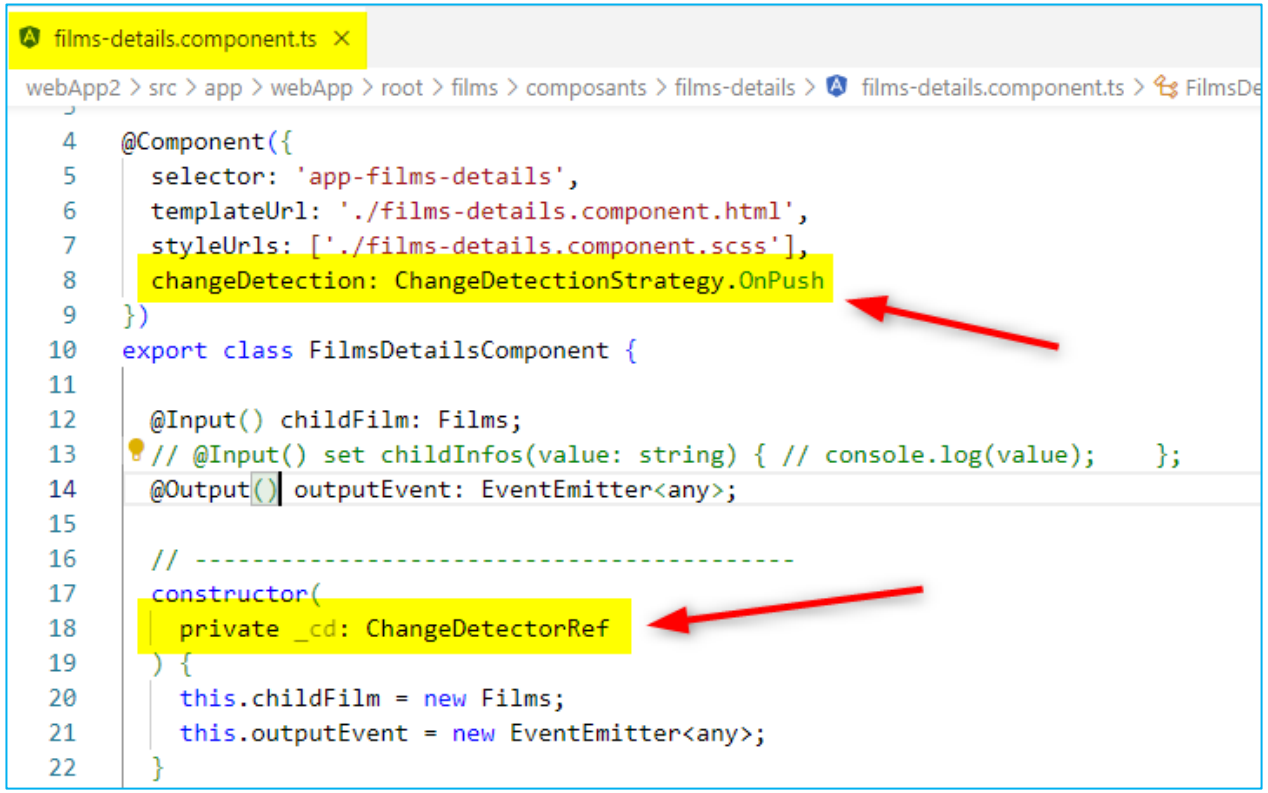
Le composant est OnPush



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

La class ChangeDetetctorRef

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



La class ChangeDetetctorRef

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

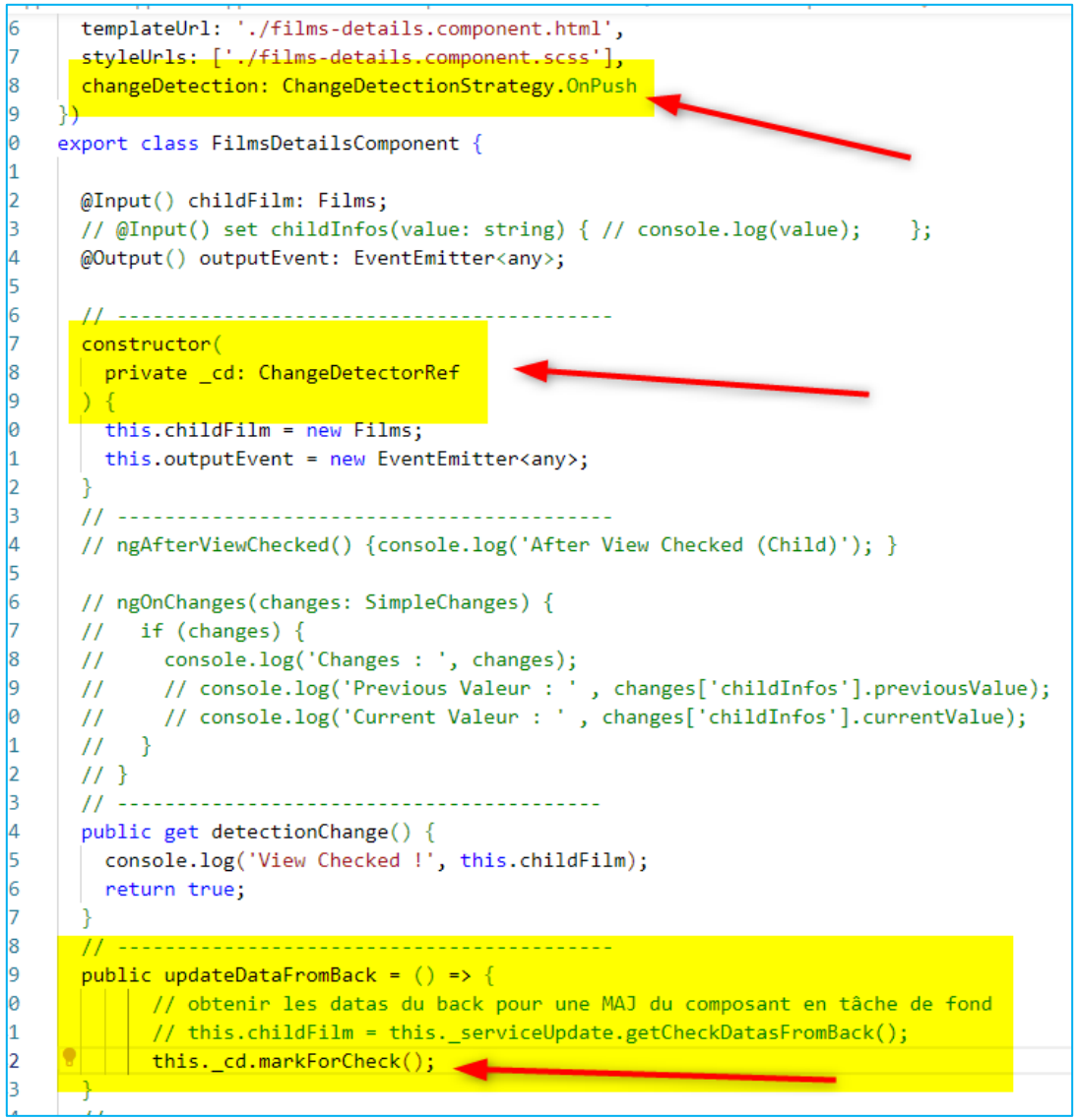
```
// ----------------------------------------
public updateDataFromBack = () => {
    // obtenir les datas du back pour ur
    // this.childFilm = this._serviceUpd
    this._cd.markForCheck();
}
```

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

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

checkNoChanges
detach
detectChanges
markForCheck
reattach

La class ChangeDetetctorRef



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

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

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

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

Angular 17 – Le Signal!

Le Signal

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

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

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

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

toSignal() toObservable()

Le Signal

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

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

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

Le Signal Angular

- La primitive Signal()
- Computed
- Méthode Set() Update() Mutate()

```
//signal

public signalNb = signal(3);
// la primitive signal = lecture ou en ecriture

// calculé à partir d'autre signaux
public infos = computed(
  () => `${this.signalNb()} étoiles attribuée(s) !`    )

//viewChildren
@ViewChild('plus') eltPlus!: ElementRef<HTMLElement>;
@ViewChild('moins') eltMoins!: ElementRef<HTMLElement>;

ngAfterViewInit() {
  // -------------------------
  fromEvent(this.eltPlus.nativeElement, 'click').pipe(
    tap( () => { this.signalNb.update( (valSignal) => valSignal + 1 )} )
  ).subscribe()
  // -------------------------
  fromEvent(this.eltMoins.nativeElement, 'click').pipe(
  tap( () => { this.signalNb.update( (valSignal) => valSignal - 1 ) } )
  ).subscribe()
  }
}
```