ORSYS formation

Formation ANGULAR

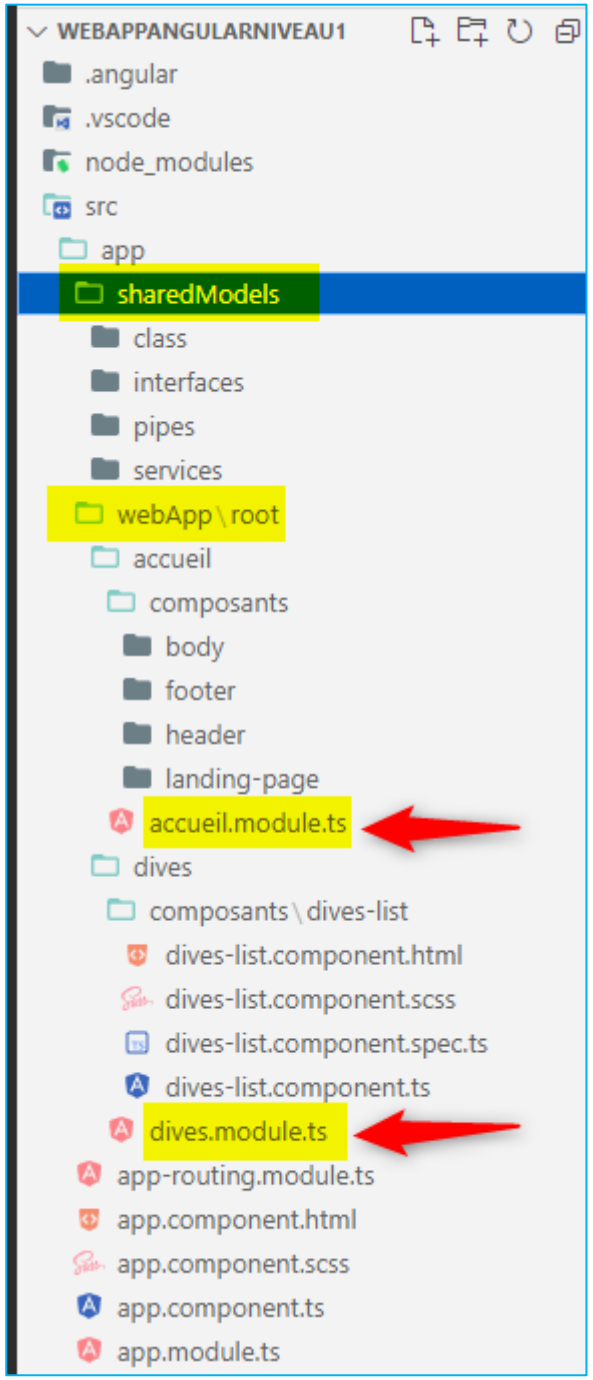
Chapitre 2

Animé par Michel BOCCIOLESI

Table des matières

Mise en place du TP fil rouge

- Création d'un nouveau projet – avec le routage et SCSS
- Installation des dépendances Bootstrap et bootstrap-icons
  https://www.npmjs.com/package/bootstrap
  https://www.npmjs.com/package/bootstrap-icons
- Installation de la « dev » dépendance json-server
  https://www.npmjs.com/package/json-server
- Bonnes pratiques « structure et architecture » de projet
- injection de dépendances…



L'injection de dépendances

1er exemple : le service

dives-list.component.ts

bApp > root > dives > composants > dives-list > dives-list.component.ts > DivesListComponent

```ts
// @ViewChild('btnPlus') eltBtn!:ElementRef;
@ViewChildren('btnPlus') eltBtnCollection!: QueryList

// 2- const
// l'injection de dépendances permet de lier un servi

constructor(
  private _service: DivesService,
  private _title: Title,
  private _meta: Meta
) {
  console.warn('Constructor');
  this.dives = [];
  this._title.setTitle('Liste des Plongées SEO ....')
  this._meta.addTags([
    { name: 'description', content: 'Texte de descrip
    { name: 'author', content: 'MB CREATION WEB' }
  ]);
}

// ----------------------------------------------------
// 3- lifeCycle
// ----------------------------------------------------
ngOnInit(): void {
  // Chargement des datas
  console.warn('NgOnInit');
  this._service.getDives()

    // .subscribe(
    //   // next seulement ...
    //   (datas:Dives[]) => {
    //     this.dives = datas;
    //   }
    // );

    .subscribe({
      next: // Call Back OK
        (datas: Dives[]) => {
          this.dives = datas;
        },
      error:
        e => console.log(e)
      ,
      complete:
        () => console.log('Observer Complete')
    }
```

dives.service.ts

TP03-services-http > src > app > sharedModels > services > dives.service.ts > DivesService > getDives

```ts
import { Injectable } from '@angular/core';
import { Dives } from '../class/dives';
import { HttpClient } from '@angular/common/http';
import { Observable, tap , map} from 'rxjs';

@Injectable({
  providedIn: 'root'
})

export class DivesService {

  // 1- props
  private dives: Dives[];

  // 2- const
  constructor(private _http: HttpClient) {
    this.dives = [];
  }

  // 3- Méthodes
  public getDives = (): Observable<Dives[]> => {

    const API_URL = 'http://localhost:3001/dives';
    // const API_URL = 'https://dev.webjs.fr/dives.json';

    return this._http.get<Dives[]>(API_URL).pipe(
      // le pipe permet d'enchainer les opérateurs RXJS
      // 1 opérateur RXJS est une fonction qui modifie
      // ou transforme la séquence(stream ou flux) de valeurs émises par l'observable

      tap(
        // opérateur de débug => console
        (response: any) => {
          console.log('------ Réponse depuis le service : ', response);
        }
      ),
      map(
        (datas:Dives[]) => {
          return datas.filter(
            (data:Dives) => {
              // return data.id===5
              return data.id >=1
            }
          )
        }
      )
```

RXJS – 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 le gestion des requêtes asynchrones.

RXJS et les OBSERVABLES

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

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

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

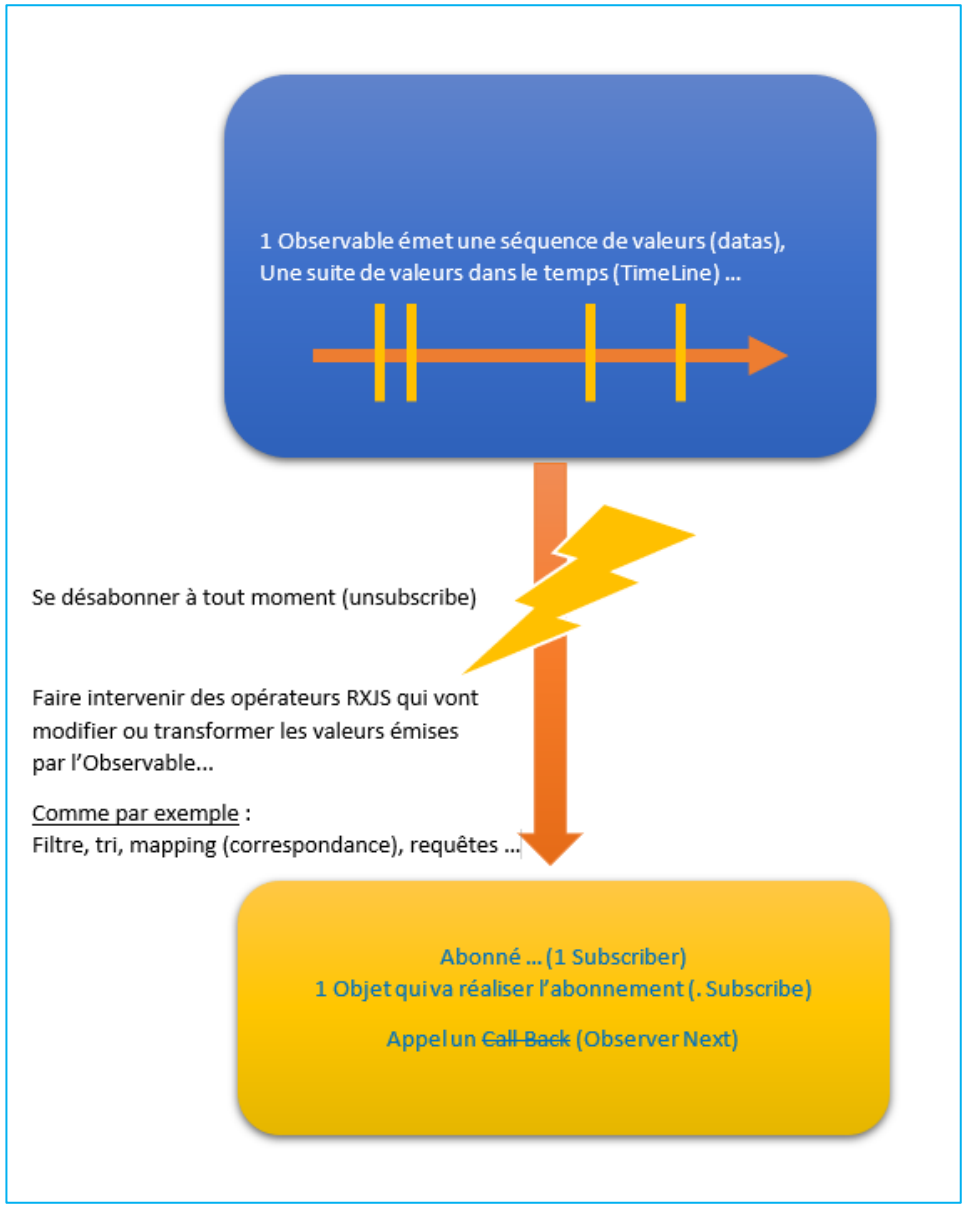
Opérateurs : des fonctions tap, map, merge, filter qui vont pouvoir modifier/transformer ce flux de valeurs émises par l'observable

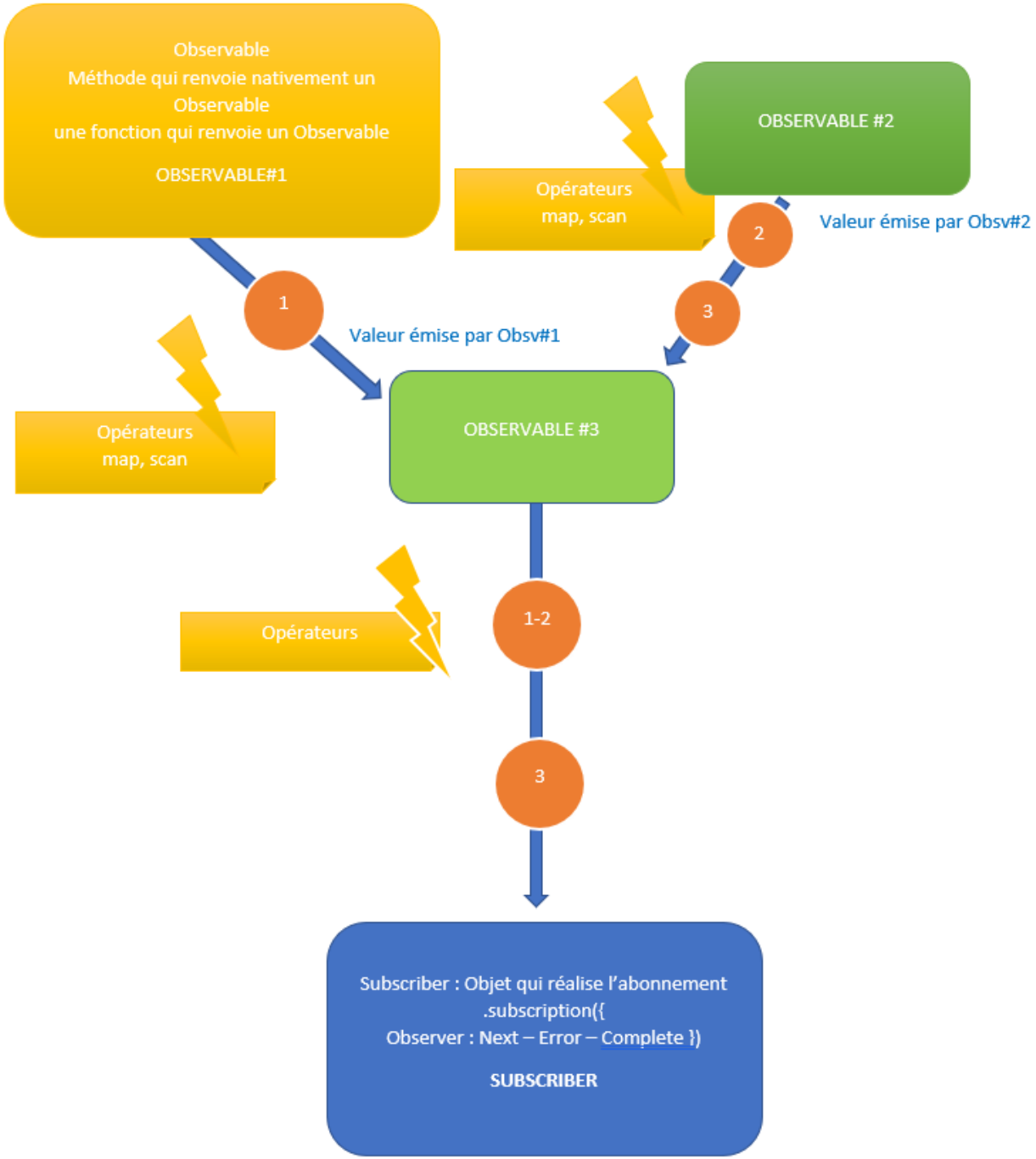
Dans le subscribe => Les observers (fcts de call back) qui se définissent dans le subscribe : Next - Error – Complete

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

Les séquences de valeurs émises sont facilement annulables !

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





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

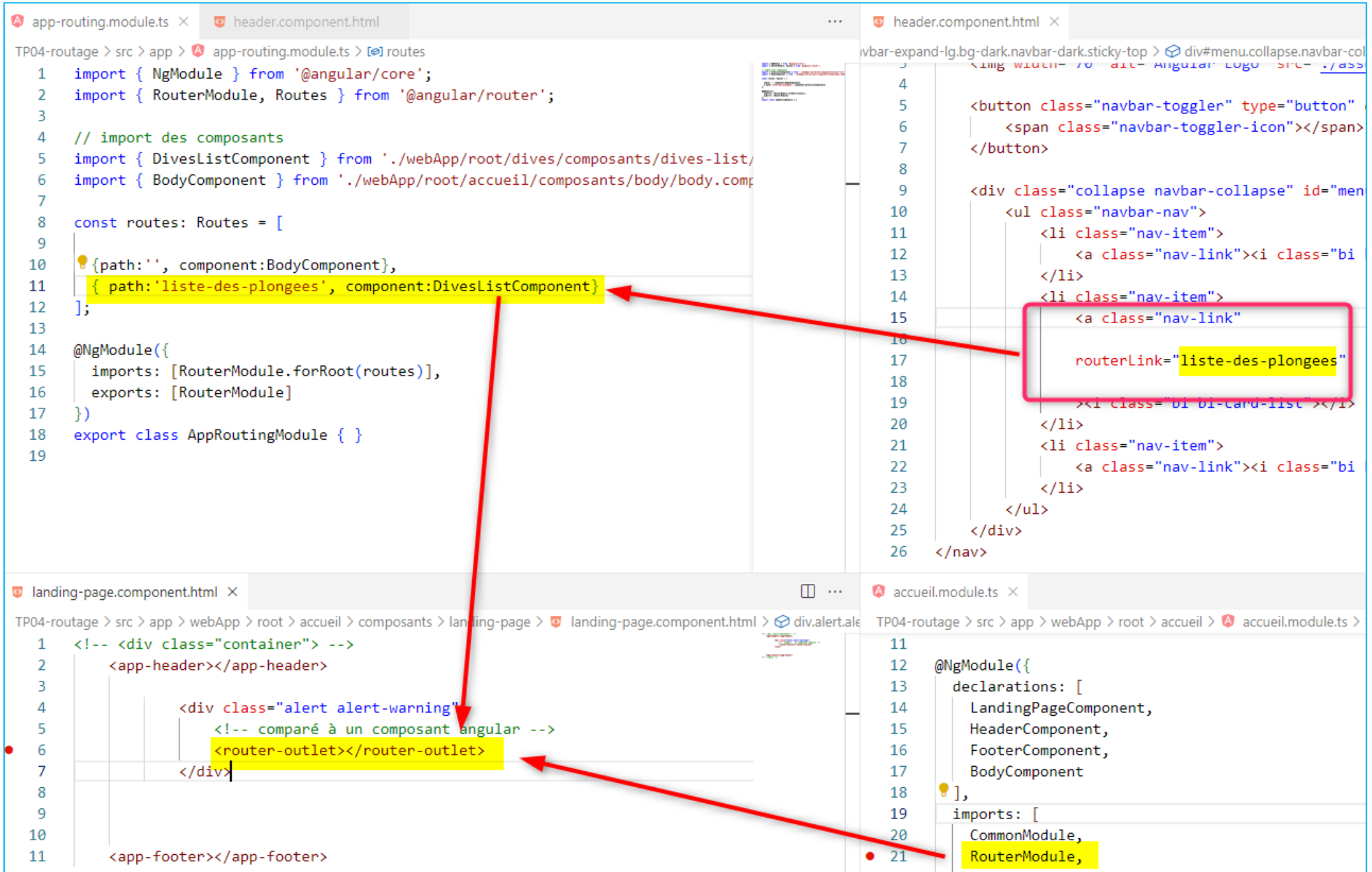
Routage

Routing : Concepts de base

Un path (URL ou fragment d'URL, l'URI) est associé à un composant dans le fichier « app-routing.module.ts »

La vue du composant est chargée dans le route-outlet, lorsque ce path est invoqué.

Routing : Concepts de base



app-routing.module.ts

```ts
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';

// import des composants
import { DivesListComponent } from './webApp/root/dives/composants/dives-list/
import { BodyComponent } from './webApp/root/accueil/composants/body/body.comp

const routes: Routes = [

{path:'', component:BodyComponent},
{ path:'liste-des-plongees', component:DivesListComponent}
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

header.component.html

```html
    <button class="navbar-toggler" type="button"
        <span class="navbar-toggler-icon"></span>
    </button>

    <div class="collapse navbar-collapse" id="men
        <ul class="navbar-nav">
            <li class="nav-item">
                <a class="nav-link"><i class="bi
            </li>
            <li class="nav-item">
                <a class="nav-link"

                routerLink="liste-des-plongees"

                ></i>
            </li>
            <li class="nav-item">
                <a class="nav-link"><i class="bi
            </li>
        </ul>
    </div>
</nav>
```

landing-page.component.html

```html
<!-- <div class="container"> -->
    <app-header></app-header>

        <div class="alert alert-warning"
            <!-- comparé à un composant angular -->
            <router-outlet></router-outlet>
        </div>



    <app-footer></app-footer>
```

accueil.module.ts

```ts
@NgModule({
  declarations: [
    LandingPageComponent,
    HeaderComponent,
    FooterComponent,
    BodyComponent
  ],
  imports: [
    CommonModule,
    RouterModule,
```

Routing : Concepts de base

app-routing.module.ts

TP04-routage > src > app > app-routing.module.ts > routes

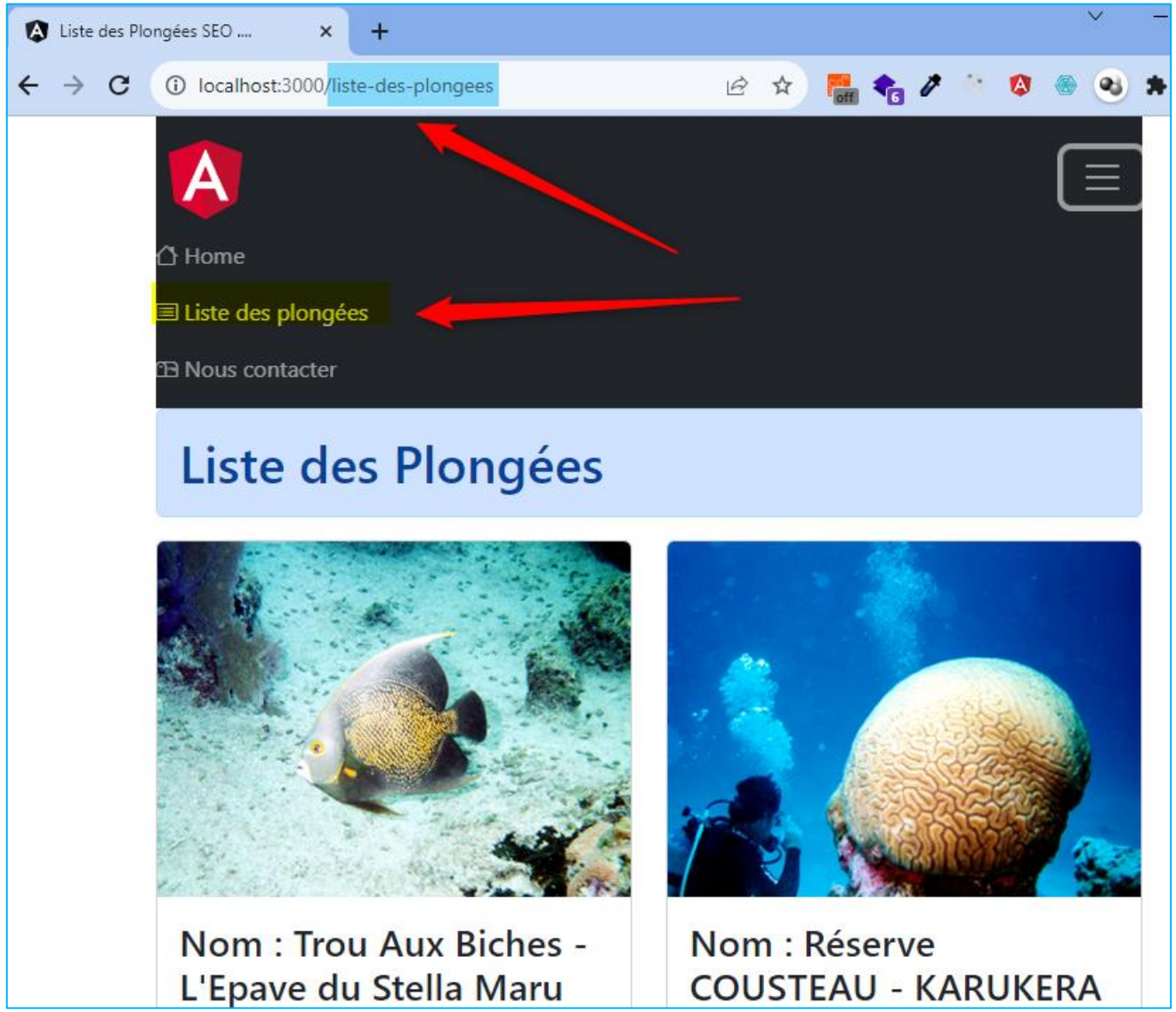
```ts
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { BodyComponent } from './webApp/root/accueil/composants/body/body.component';
import { DivesListComponent } from './webApp/root/dives/composants/dives-list/dives-list.component';
import { Page404Component } from './sharedModels/composants/page404.component';
import { ContactsComponent } from './webApp/root/form/composants/contacts/contacts.component';
import { DivesDetailComponent } from './webApp/root/dives/composants/dives-detail/dives-detail.component';

const routes: Routes = [
  { path: '', component: BodyComponent },
  // domain.tld/ (page de démarrage)
  // { path:'accueil', component:BodyComponent}, // domain.tld/accueil
  { path: 'accueil', redirectTo: '', pathMatch: 'full' },
  // { path:'nouvelle-url', component:DivesListComponent},
  // { path:'ancienne-url-seo-déjà-référencée', redirectTo:'nouvelle-url', pathMatch:'full'},
  { path: 'contactez-nous', component: ContactsComponent },
  { path: 'liste-des-plongees', component: DivesListComponent },
  { path: 'liste-des-plongees/:id', component: DivesDetailComponent },

  //  Gestion des erreurs : toujours en dernier !!
  // { path:'**', redirectTo:'', pathMatch:'full'}
  { path: '**', component: Page404Component }
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

header.component.html

TP04-routage > src > app > webApp > root > accueil > composants > header > header.component.html > nav.nav

```html
<nav class="navbar navbar-expand-lg bg-dark navbar-dark sticky-top">
    <img width="70" alt="Angular Logo" src="./assets/images/angular.png">

    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" dat
        <span class="navbar-toggler-icon"></span>
    </button>

    <div class="collapse navbar-collapse" id="menu">
        <ul class="navbar-nav">
            <li class="nav-item">
                <a class="nav-link"
                routerLink="accueil"
                ><i class="bi bi-house"></i> Home</a>
            </li>

            <li class="nav-item">
                <!-- <a class="nav-link"
                [routerLink]="['liste-des-plongees']"
                ><i class="bi bi-card-list"></i> Liste des plongées</a> -->
                <a class="nav-link"
                routerLink="liste-des-plongees"
                ><i class="bi bi-card-list"></i> Liste des plongées</a>
            </li>

            <li class="nav-item">
                <a class="nav-link"
                routerLink="contactez-nous"
                ><i class="bi bi-mailbox"></i> Nous contacter</a>
            </li>
        </ul>
    </div>
</nav>
```

Routing : Concepts de base



Routing : Concepts de base

landing-page.component.html

src > app > webApp > root > accueil > composants > landing-page > landing-page.component.html

```
<div style="min-height:100vh;" class="alert alert-warning"
    <app-header></app-header>
    <!-- <app-body></app-body>     -->
    <!-- <app-dives-list></app-dives-list> -->

    <!-- router-outlet primary -->
    <div class="alert alert-primary">

        <router-outlet></router-outlet>
    </div>

    <!-- router-outlet autres -->
    <div class="alert alert-success">

        <router-outlet name="outlet-2"></router-outlet>
    </div>


</div>
<app-footer></app-footer>
```

app-routing.module.ts

src > app > app-routing.module.ts > routes > canActivate

```
domain.tld/accueil
// { path:'accueil' , redirectTo:'',
pathMatch:'full'},
// { path:'nouvelle-url',
component:DivesListComponent},
// { path:'ancienne-url-seo-déjà-référencée',
redirectTo:'nouvelle-url', pathMatch:'full'},
{ path: 'liste-des-plongees', component:
DivesListComponent },
{ path: 'liste-des-plongees/:id', component:
DiveDetailComponent },

{ path: 'routing-angular', component:
CompRoutingOutletComponent, outlet: 'outlet-2',
canActivate: [RouteGuardService]  },

// 1ère manière de faire : charger un composant depuis
un module qui a son propre sytème de routage forChild
// {
//   path: 'mon-compte-client',
//   component: ClientLandingPageComponent,
//   children: clientRoutes
// },

// 2nde manière oprimisée : lazy loading avec ES2015
(promesses-import)
// {
//   path: 'mon-compte-client',
//   component: ClientLandingPageComponent,
//   loadChildren:
//     ():any => import('./webApp/compte-client/
compte-client.module').then(
```

header.component.html

src > app > webApp > root > accueil > composants > header > header.component.html

```
                Liste des plongées
            </a>
        </li>

        <li class="nav-item">
            <a class="nav-link" routerLink="mon-compt
        </li>

        <li class="nav-item">
            <a class="nav-link" [routerLink]="[{
            outlets : {
                'outlet-2':'routing-angular'
            }
        }]"

        [skipLocationChange]="true"><i class="bi bi-m
        </li>

        <li class="nav-item">
            <a class="nav-link"><i class="bi bi-mailb
        </li>
    </ul>
</div>
</nav>
```

Routing : Concepts avancés – QueryParams - Router


```
{ path: 'liste-des-plongees/:id', component: DivesDetailComponent },
```

Routing : Concepts avancés – QueryParams - Router

EXPLORATEUR

ÉDITEURS OUVERTS

GROUPE 1

dives-list.component.html TP04-rout...

GROUPE 2

dives-detail.component.ts TP04-routa...

FORMATION-ANGULAR-2023-06

- accueil.module.ts
- dives
  - composants
    - dives-detail
      - dives-detail.component.html
      - dives-detail.component.scss
      - dives-detail.component.ts
    - dives-list
      - dives-list.component.html
      - dives-list.component.scss
      - dives-list.component.ts
  - dives.module.ts
- form
- app-routing.module.ts
- app.component.html
- app.component.scss
- app.component.ts
- app.module.ts
- assets
- favicon.ico
- index.html
- main.ts
- styles.scss
- .editorconfig
- .gitignore

dives-list.component.html

TP04-routage > src > app > webApp > root > dives > composants > dives-list > dives-list.component.html > h1.alert.alert-prima

```html
<h1 class="alert alert-primary">Liste des Plongées</h1>

<div class="row">

    <div class="col-md-6" *ngFor="let dive of dives">

        <div class="card">
            <!-- récupérer chaque image -->
            <img src="{{dive.photo}}" class="card-img-top">
            <div class="card-body">
                <h3 class="card-title">Nom : {{dive.name}} </h3>
                <p class="card-text">Description : {{dive.description}}</p>

                <!-- <a
                [routerLink]="[dive.id]"
                [queryParams]="[dive.name]"
                > -->
                <!-- on interprète un des params de l'objet -->
                <a
                [routerLink]="[dive.id]"
                [queryParams]="dive"
                [skipLocationChange]="true"

                >
                <!--  on passe tout l'objet -->
                    <button class="btn btn-primary" #btnPlus>En Savoir Plus<
                </a>
            </div>
        </div>
    </div>
</div>
```

dives-detail.component.ts

TP04-routage > src > app > webApp > root > dives > composants > dives-detail > dives-detail.compone

```ts
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute, Router } from '@angular/router';
import { Dives } from 'src/app/sharedModels/class/dives';

@Component({
  selector: 'app-dives-detail',
  templateUrl: './dives-detail.component.html',
  styleUrls: ['./dives-detail.component.scss']
})
export class DivesDetailComponent implements OnInit {
  // 1- props
  public title: string;
  public dive:Dives;

  // 2- const
  constructor(
    private _routeActive: ActivatedRoute,
    private _router:Router
  ) {
    this.title = '';
    this.dive =  new Dives();
  }

  // 3- lifeCycle
  ngOnInit(): void {

    const id = this._routeActive.snapshot.params['id'];
    console.log(id);
    this.title = `Plongée N° ${id}`;
    // re-consommer le service ?
    // ré-exploiter les datas stockées en storage ou en indexedDB
    // Evidemment utiliser les fonctions de routage d'Angular ...

```

Routing : Concepts avancés – QueryParams - Router

dives-list.component.html

```
<h1 class="alert alert-primary">Liste des Plongées</h1>

<div class="row">

    <div class="col-md-6" *ngFor="let dive of dives">

        <div class="card">
            <!-- récupérer chaque image -->
            <img src="{{dive.photo}}" class="card-img-top">
            <div class="card-body">
                <h3 class="card-title">Nom : {{dive.name}} </h3>
                <p class="card-text">Description : {{dive.description}}</

                <!-- [routerLink]="[dive.id]" -->
                <!-- Ecriture simplifiée du routerLink => routerLink="{{di
                <!-- routerLink="details/{{dive.id}}"  -->
                <!-- en chemin relatif -->
                 <button

                routerLink="/liste-des-plongees/details/{{dive.id}}"
                [queryParams]="dive"
                [skipLocationChange]="true"

                class="btn btn-primary" #btnPlus>En Savoir Plus</button>

            </div>
```

dives-details.component.ts

```
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute } from '@angular/router';

@Component({
  selector: 'app-dives-details',
  templateUrl: './dives-details.component.html',
  styleUrls: ['./dives-details.component.scss']
})
export class DivesDetailsComponent implements OnInit {

  // -------------------------
  constructor(
    private _routeActive:ActivatedRoute
  )
  { }
  // -------------------------
  ngOnInit() {
    const id=this._routeActive.snapshot.params['param'];
    console.clear;
    console.log('ID récupéré :' , id);
    // TP : passer l'ID à la Vue
    // ---------
    this._routeActive.queryParams
    .subscribe(
      (params:any) => {
        console.log('QueryParams : ', params);
        // TP : passer l'objet récupéréré à la vue
      }
    )
```

dives-details.component.html

```
<h1>Détail de la plongée</
h1>
<!-- TP afficher l'objet
passé par routage -->
```

app-routing.module.ts

```
const routes: Routes = [

  { path: '', component: BodyComponent },   // domain.tld/ (page de démar
  { path: 'liste-des-plongees', component: DivesListComponent },

  // { path:'accueil', component:BodyComponent}, // domain.tld/accueil

  // { path:'nouvelle-url', component:DivesListComponent},
  // { path:'ancienne-url-seo-déjà-référencée', redirectTo:'nouvelle-url'

  { path: 'liste-des-plongees/details/:param', component: DivesDetailsCom

  { path:'**', component:Page404Component}
```

bdd.json

```
{
    "dives": [
        {
            "id": 1,
            "name": "Trou Aux Biches - L'Epave du Stella Maru",
            "location": "Ile Maurice - Trou aux Biches - Nord/Ouest",
            "level": 2,
            "description": "Plongée de niveau 2 FFESSM ou PADI au arge de
            "latitude": -20.0363,
            "longitude": 57.544700000000034,
            "evaluation": [
```

dives.module.ts 1

```
import { NgModule } from '@
import { CommonModule } fro
import { DivesListComponent
import { HttpClientModule }
import { FormsModule } from
import { RouterModule } fro
import { DivesDetailsCompon


@NgModule({
  declarations: [
```

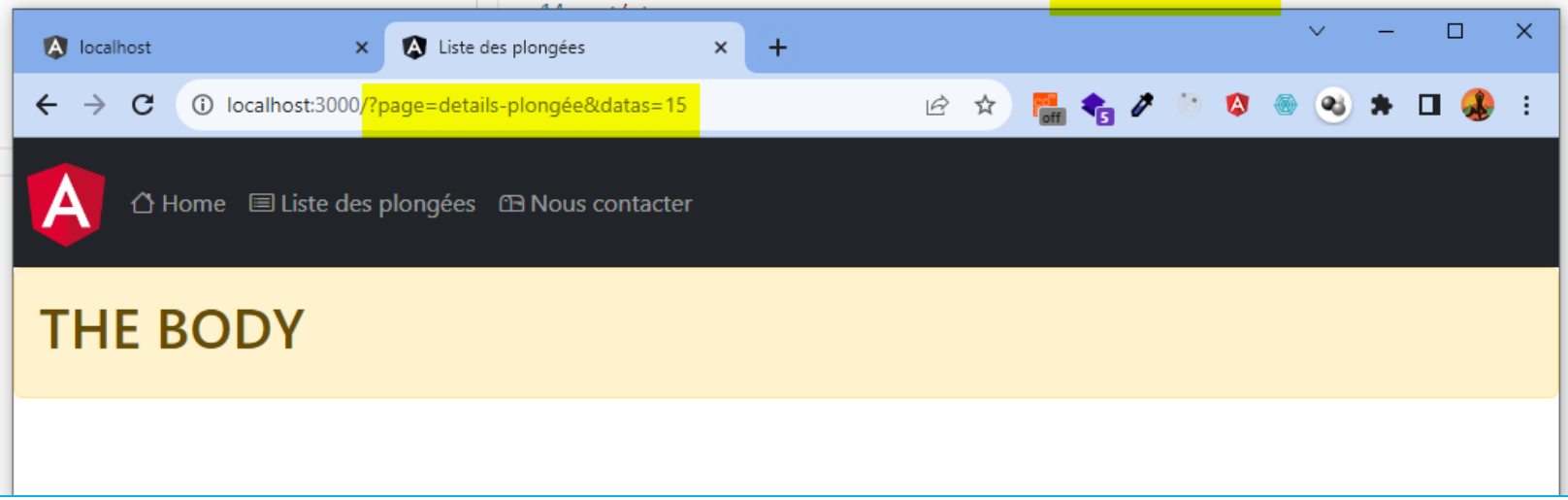
Routing : Concepts avancés – QueryParams - Router

dives-details.component.ts

```
ngOnInit() {
  const id = this._routeActive.snapshot.params['param'];
  console.clear;
  console.log('ID récupéré :', id);
  // TP : passer l'ID à la Vue
  // ---------
  this._routeActive.queryParams
    .subscribe(
      (params: any) => {
        console.log('QueryParams : ', params);
        // TP : passer l'objet récupéréré à la vue
        this.dive = params;
      }
    )
}
// -- Méthodes
public goBack = () => {
  this._router.navigateByUrl('liste-des-plongees');
}

public goHome = () => {
  this._router.navigate(
    [''],
    {
      queryParams: {
        page: 'details-plongée',
        datas: 15
      }
    }
  )
```

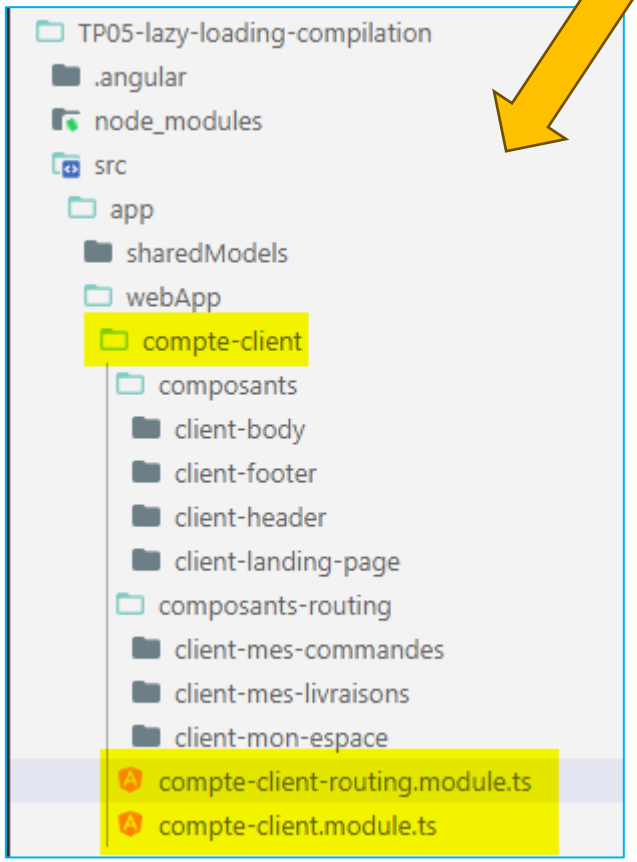
dives-details.component.html

```
<h1>Détail de la plongée N° {{dive.id}}</h1>
<!-- TP afficher l'objet passé par routage -->

<!-- Single way binding -->
<!-- TS => HTML : [directive] et {{string interpolation}} -->
<!-- HTML => TS : (event du DOM) par ex : (click)-->
<p>
    <button class="btn btn-warning" (click)="goBack()">Go Back</button>

    <button class="btn btn-danger" routerLink="/liste-des-plongees">RouterLink</button>
</p>
<p>
    <button class="btn btn-success" (click)="goHome()">Go Home</button>
```

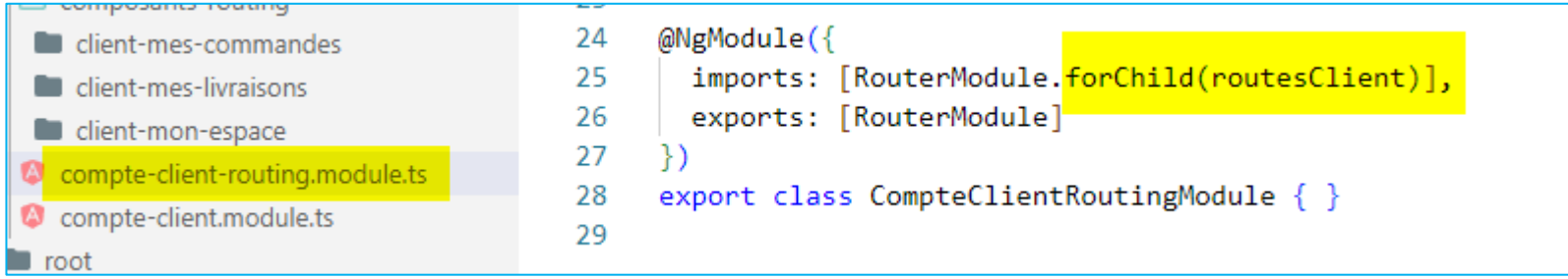


Routing : Concepts avancés – Le Lazy Loading

TP : déployer une architecture de module et composants « compte-client » comme ci-dessous

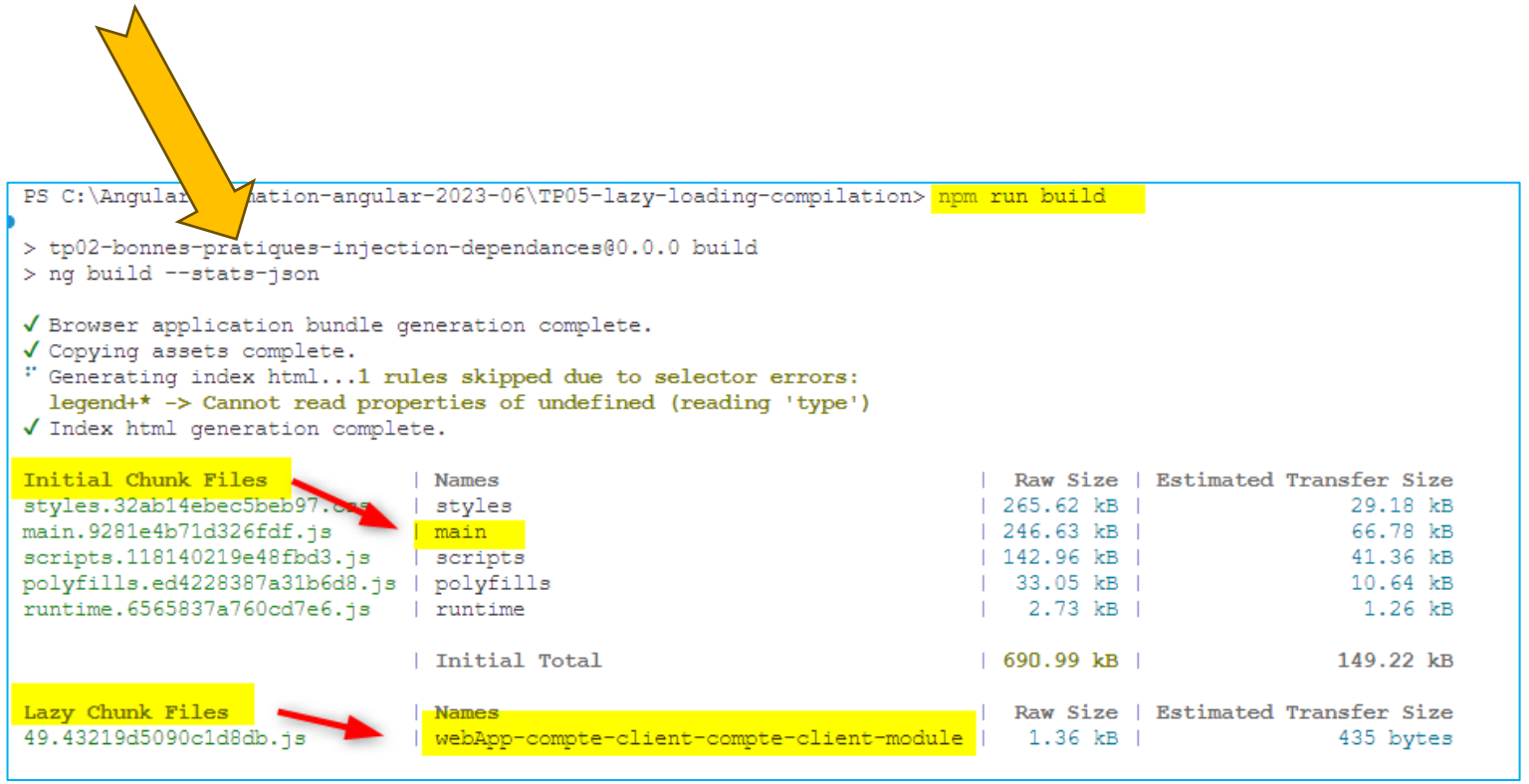


Créer le module compte-client avec l'option « --routing »
Cette option crée un module de routage spécifique forChild

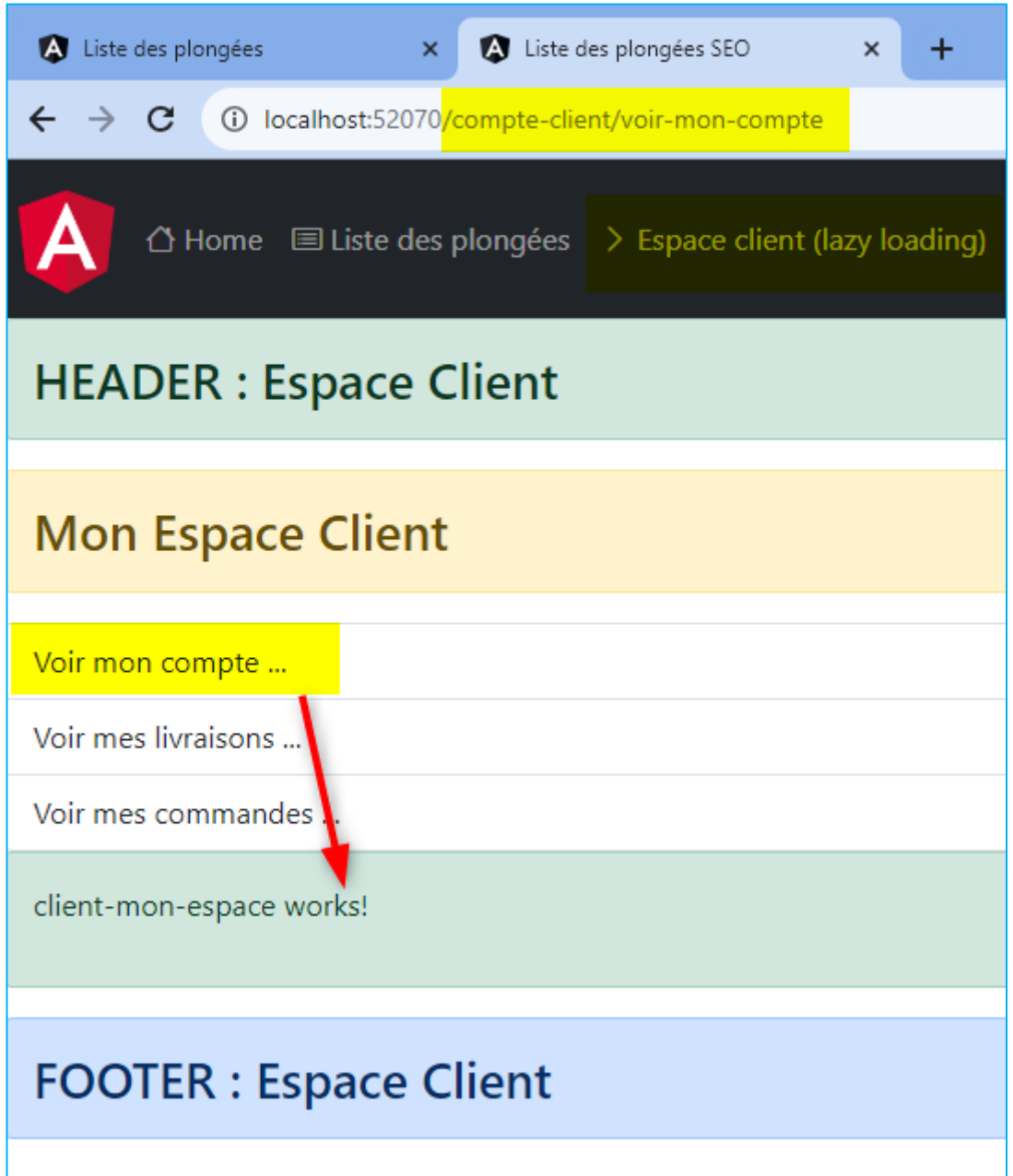


Routing : Concepts avancés – Le Lazy Loading

Le Lazy Loading permet de charger tout un module (compilé dans un fichier extrait du main.js), seulement quand l'utilisateur sollicitera ce module par une action de navigation.
Les fichiers de build (compilation) sont donc dégroupés et le projet final « buildé » est optimisé.

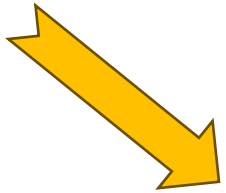


Routing : Concepts avancés – Le Lazy Loading



Routing : Concepts avancés – Le Lazy Loading

Afin de bien analyser ces customisations « tuning » de compilation, mise en place d'une stratégie de lecture de statistiques avec le « webpack-bundle-analyzer »
https://www.npmjs.com/package/webpack-bundle-analyzer

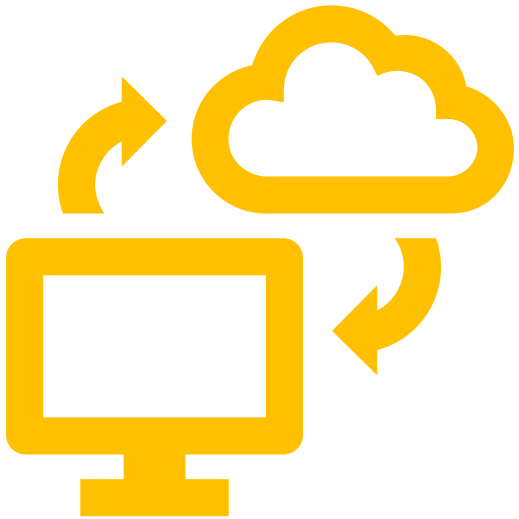


package.json

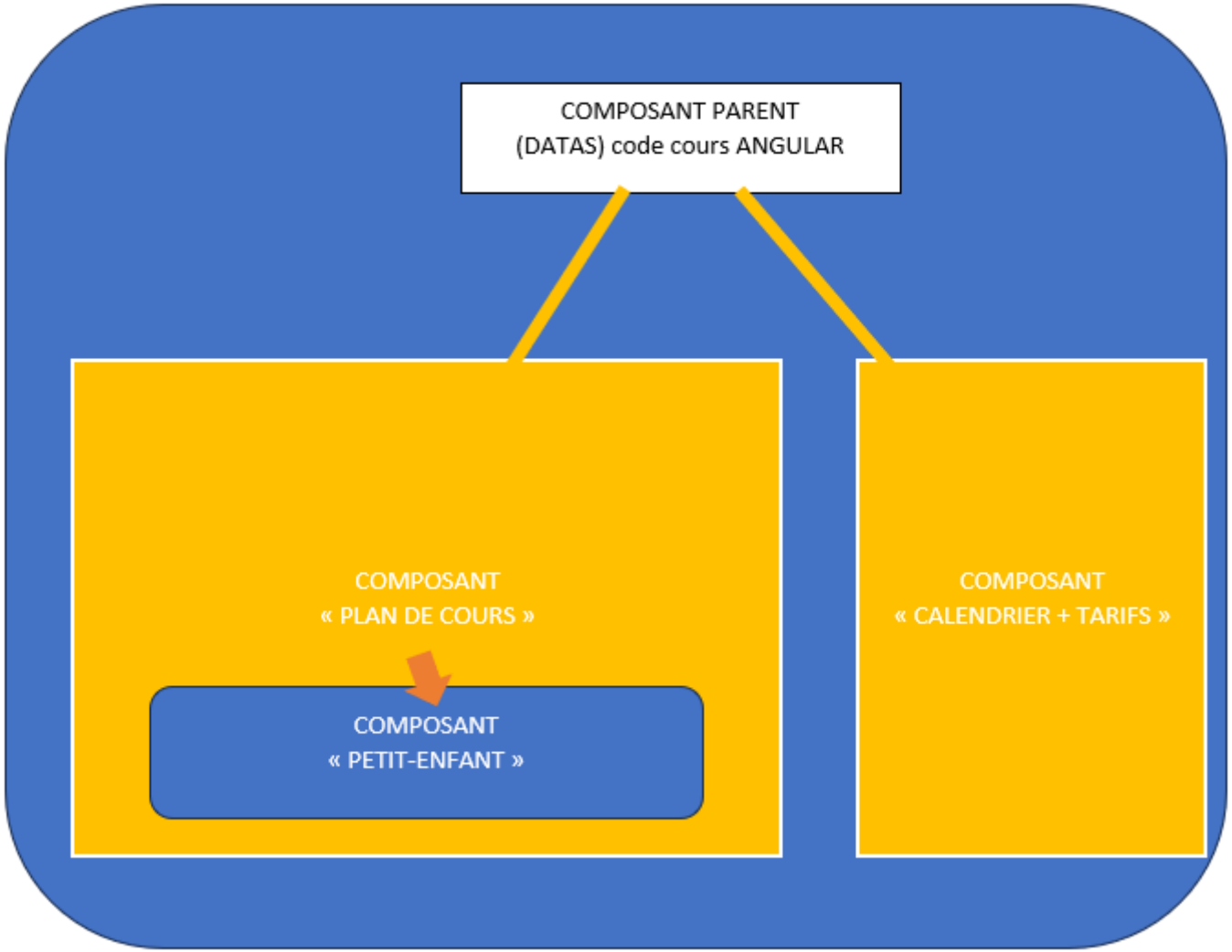
P05-lazy-loading-compilation > package.json > ...

```
5     "ng": "ng",
6     "start": "ng serve --open --port=3000",
7     "build": "ng build --stats-json",
8     "watch": "ng build --watch --configuration development --stats-json",
9     "test": "ng test",
10    "json-server": "json-server --watch src/assets/json/dives.json -p 3001",
11    "analyze":"webpack-bundle-analyzer dist/stats.json"
12  },
```

Input - Output

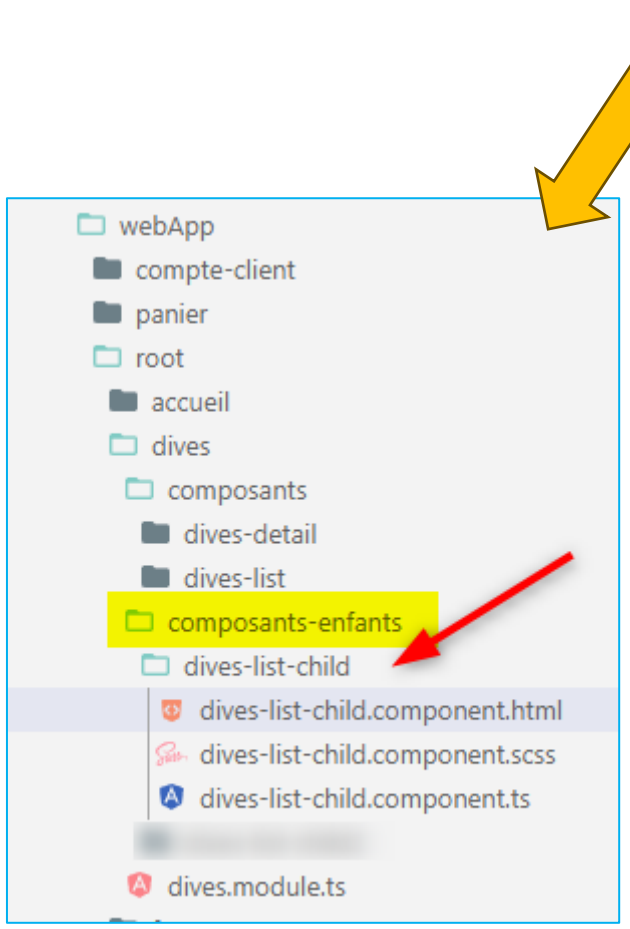


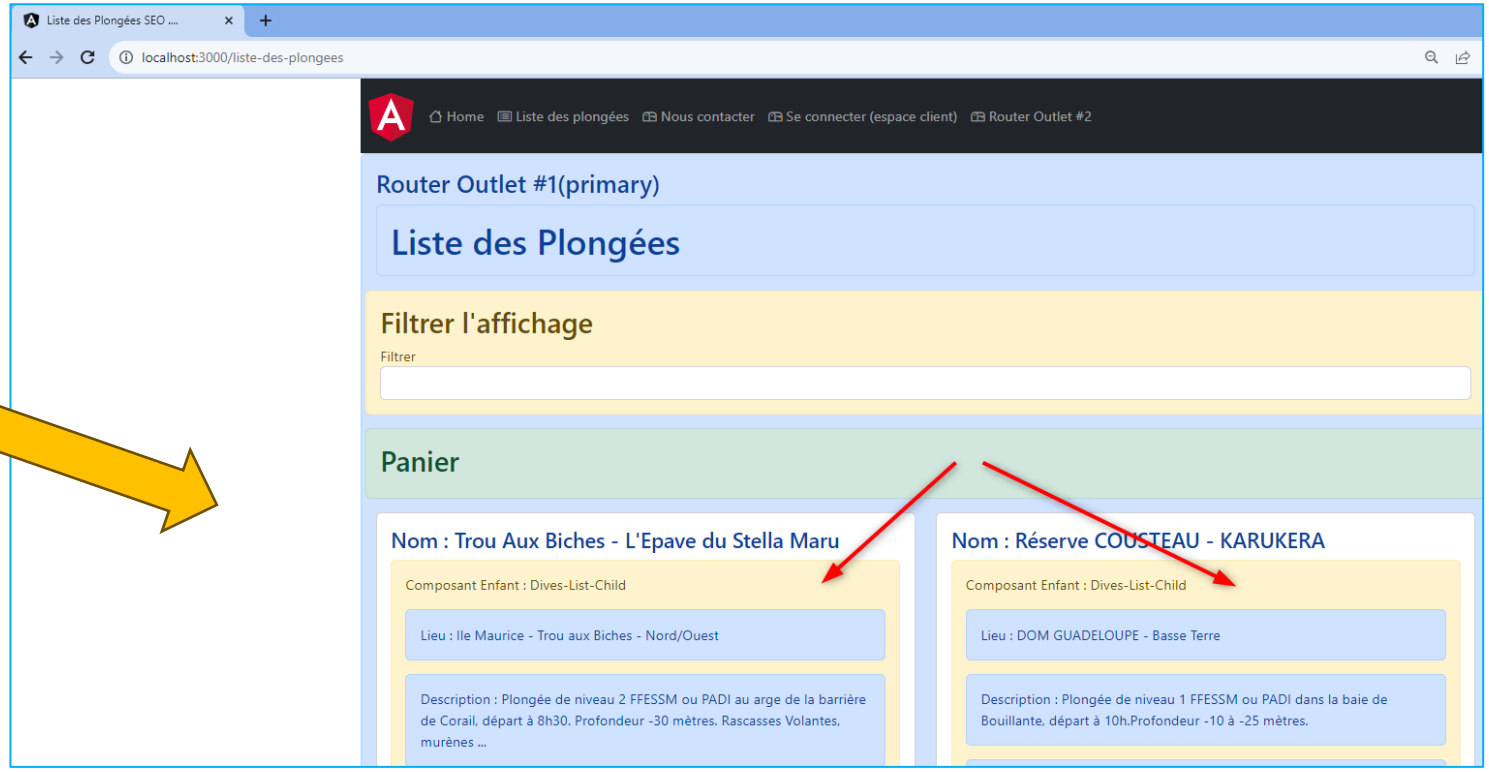
Communication : composants parents et enfants



Communication : composants parents et enfants

TP : déployer une architecture de composants dans la partie « dives » comme illustré ci-dessous





Communication :
composants parents et enfants

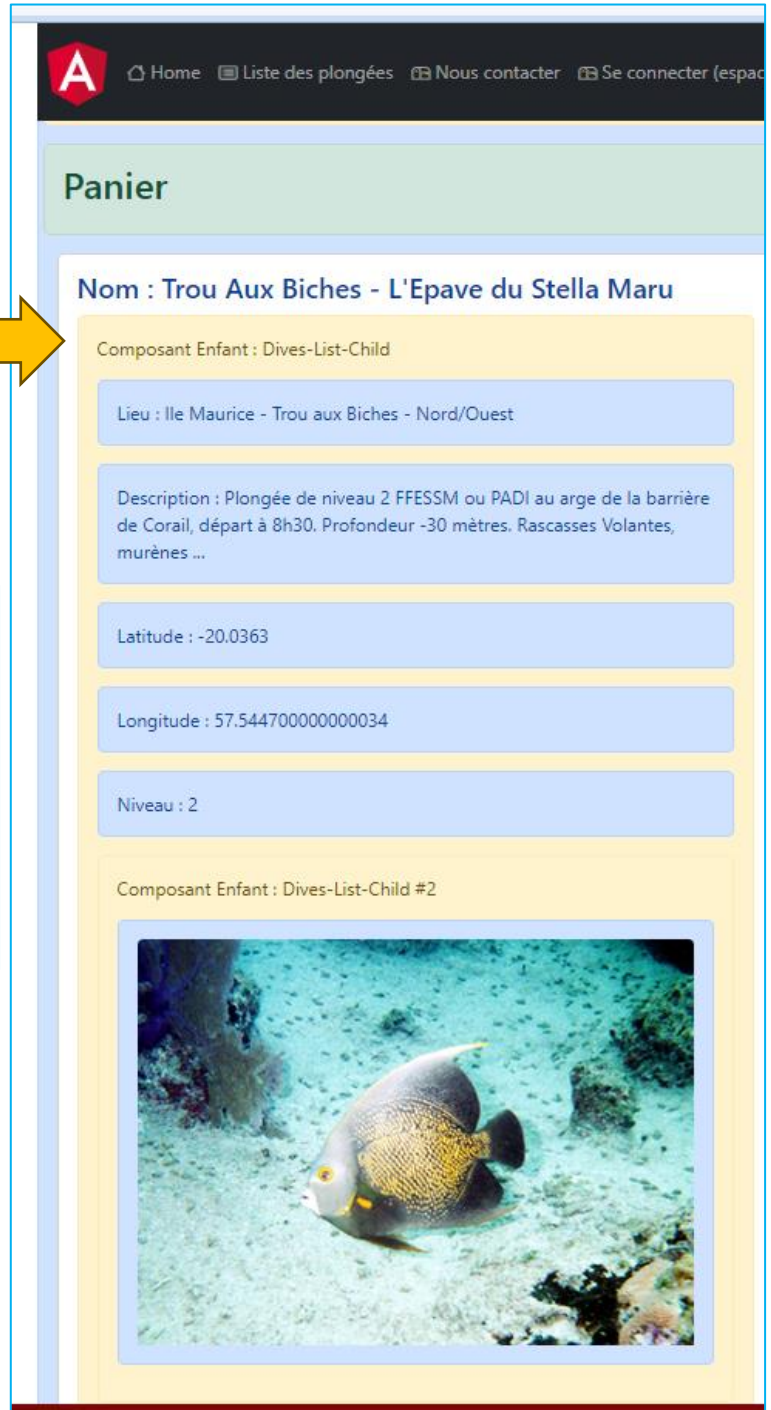
TP : Chaque composant enfant généré par la boucle *ngFor est représenté dans une classe alert-warning.

Pour faire passer une donnés de la vue HTML du composant parent vers le TS du composant enfant, nous utilisons les @Input() qui sont toujours définis dans le composant enfant.

Cet input est utilisé
par le parent comme une directive

```
<!-- Composant Enfant -->
<div class="alert alert-warning">

    <p>Composant Enfant : Dives-List-Child #2</p>
    <app-dives-list-child2
    [inputDive]="inputDive"

    ></app-dives-list-child2>

</div>
<!-- Composant Enfant -->
```

```
9   export class DivesListChildComponent {
10
11  // les @input et les @output permettent d'établir la communication entre comp parents et enfant(s)
12  // ils se définissent toujours dans l'enfant !
13
14  // 1- props
15  // décorateur   nom_du_décorateur:type
16  // le nom du décorateur (devient) une directive dans le parent
17  @Input() inputDive!:Dives;
18  @Input() inputInfos!:string;
19  // ----------------------
20  @Output() outputEventDive:EventEmitter<any>;
21
```



Communication : composants parents et enfants

dives-list.component.ts

```ts
// 4- méthodes
public diveSelected = (e: any) => {
  console.log('depuis le parent $event => ',e);

  let isChecked=e.paramIsChecked;
  let diveSelect=e.paramDive;

  console.log(`Plongée : ${diveSelect.name} - Etat : ${isChecked}`

  if (isChecked) {
    this.panier.push(diveSelect);
    console.table(this.panier);
  } else {
    let keyPanier = this.panier.indexOf(diveSelect);
    if (keyPanier >= 0) {
      this.panier.splice(keyPanier, 1);
      console.table(this.panier);
    }
  }

}
```

dives-list-child.component.ts

```ts
export class DivesListChildComponent {

// les @input et les @output permettent d'établir la co
// ils se définissent toujours dans l'enfant !

// 1- props
// décorateur   nom_du_décorateur:type
// le nom du décorateur (devient) une directive dans le
@Input() inputDive!:Dives;
@Input() inputInfos!:string;
// ----------------------
@Output() outputEventDive:EventEmitter<any>;

// 2- const
constructor(){
  this.outputEventDive =  new EventEmitter();
}

// 3- méthodes
public choixDive = (e:any) => {
  console.clear();
  console.log(e.target.checked);
```

dives-list-child2.component.ts

```ts
import { Component, Input } from '@angular/core';
import { Dives } from 'src/app/sharedModels/class

@Component({
  selector: 'app-dives-list-child2',
  templateUrl: './dives-list-child2.component.html
  styleUrls: ['./dives-list-child2.component.scss
})
export class DivesListChild2Component {

  @Input() inputDive!:Dives;

}
```

dives-list.component.html

```html
<div class="card-body">
    <h4 class="card-title">Nom : {{dive.name}} </h

    <!-- Composant Enfant -->
    <div class="alert alert-warning">
        <p>Composant Enfant : Dives-List-Child</p>
        <app-dives-list-child
        [inputDive]="dive"
        [inputInfos]="infos"
        (outputEventDive)="diveSelected($event)"
        ></app-dives-list-child>

    </div>
    <!-- Composant Enfant -->
```

dives-list-child.component.html

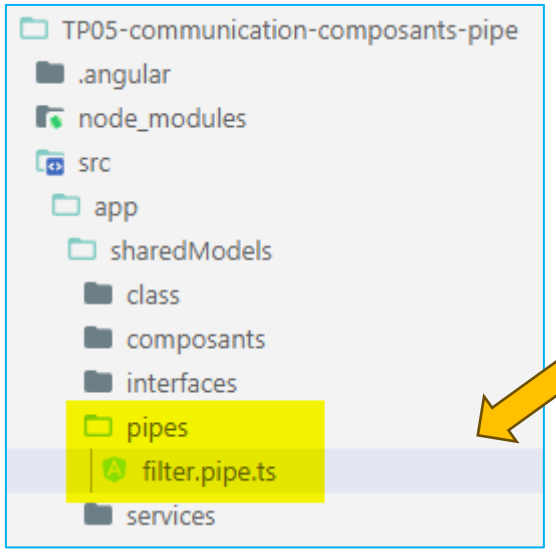
```html
    <p>Composant Enfant : Dives-List-Child #2</p>
    <app-dives-list-child2
    [inputDive]="inputDive"

    ></app-dives-list-child2>

</div>
<!-- Composant Enfant -->


<p class="alert alert-primary">
    <input type="checkbox" (change)="choixDive($event)">
</p>
<!-- $event est l'objet global evenement -->
```

dives-list-child2.component.html

```html
<p class="alert alert-primary">
    <img src="{{inputDive.photo}}" class="card-img

</p>
```

Pipes Angular

Les PIPES Angular



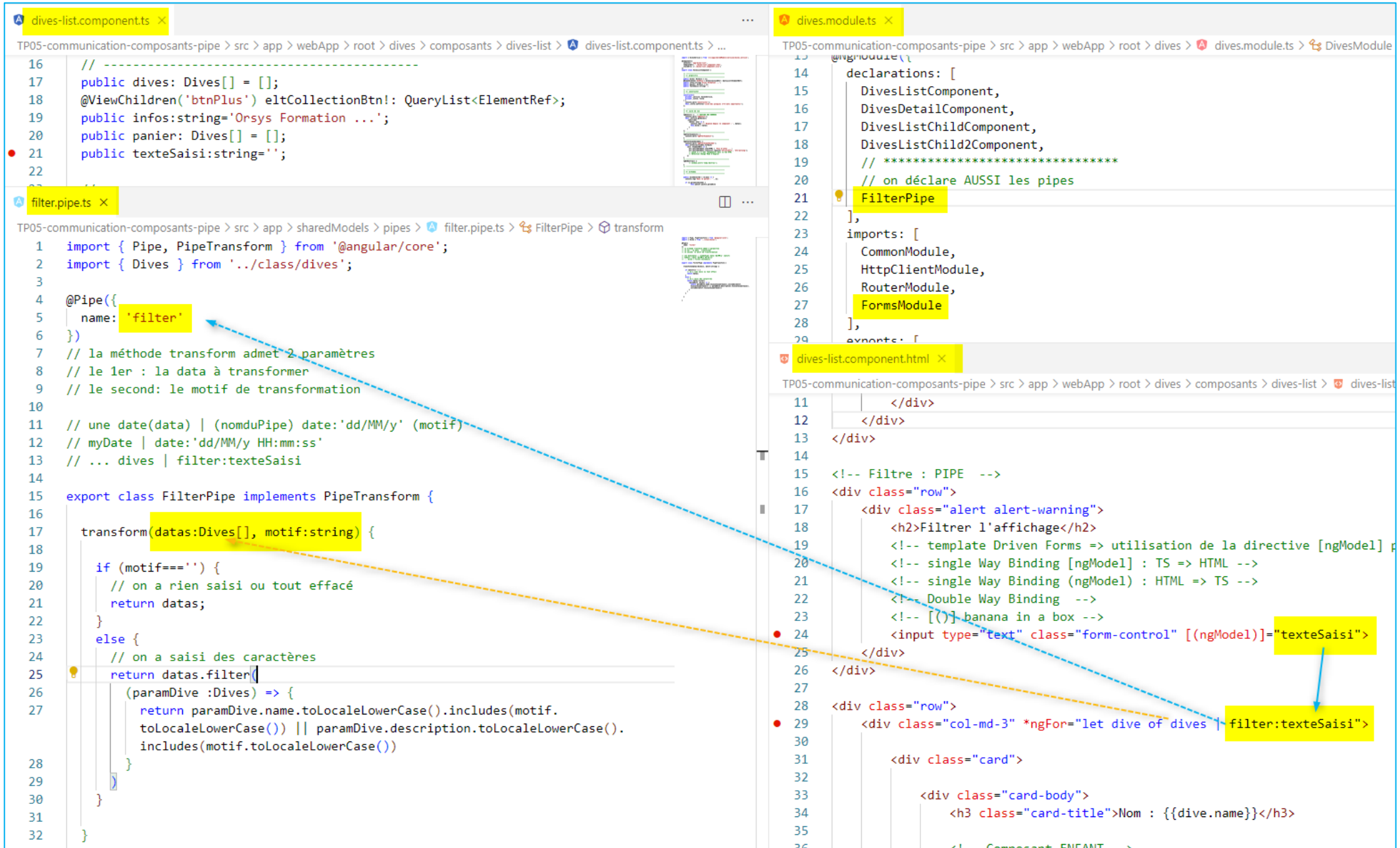
Créer ses propres Pipes avec la commande :

ng g(enerate) p(ipe)

Ce pipe va permettre de filtrer l'affichage des datas en fonction des caractères saisis.

Exemple : Agay

Les PIPES Angular



dives-list.component.ts

TP05-communication-composants-pipe > src > app > webApp > root > dives > composants > dives-list > dives-list.component.ts > ...

```
16    // ----------------------------------------
17    public dives: Dives[] = [];
18    @ViewChildren('btnPlus') eltCollectionBtn!: QueryList<ElementRef>;
19    public infos:string='Orsys Formation ...';
20    public panier: Dives[] = [];
21    public texteSaisi:string='';
22
```

filter.pipe.ts

TP05-communication-composants-pipe > src > app > sharedModels > pipes > filter.pipe.ts > FilterPipe > transform

```
1   import { Pipe, PipeTransform } from '@angular/core';
2   import { Dives } from '../class/dives';
3
4   @Pipe({
5     name: 'filter'
6   })
7   // la méthode transform admet 2 paramètres
8   // le 1er : la data à transformer
9   // le second: le motif de transformation
10
11  // une date(data) | (nomduPipe) date:'dd/MM/y' (motif)
12  // myDate | date:'dd/MM/y HH:mm:ss'
13  // ... dives | filter:texteSaisi
14
15  export class FilterPipe implements PipeTransform {
16
17    transform(datas:Dives[], motif:string) {
18
19      if (motif==='') {
20        // on a rien saisi ou tout effacé
21        return datas;
22      }
23      else {
24        // on a saisi des caractères
25        return datas.filter(
26          (paramDive :Dives) => {
27            return paramDive.name.toLocaleLowerCase().includes(motif.
              toLocaleLowerCase()) || paramDive.description.toLocaleLowerCase().
              includes(motif.toLocaleLowerCase())
28          }
29        )
30      }
31
32    }
```

dives.module.ts

TP05-communication-composants-pipe > src > app > webApp > root > dives > dives.module.ts > DivesModule

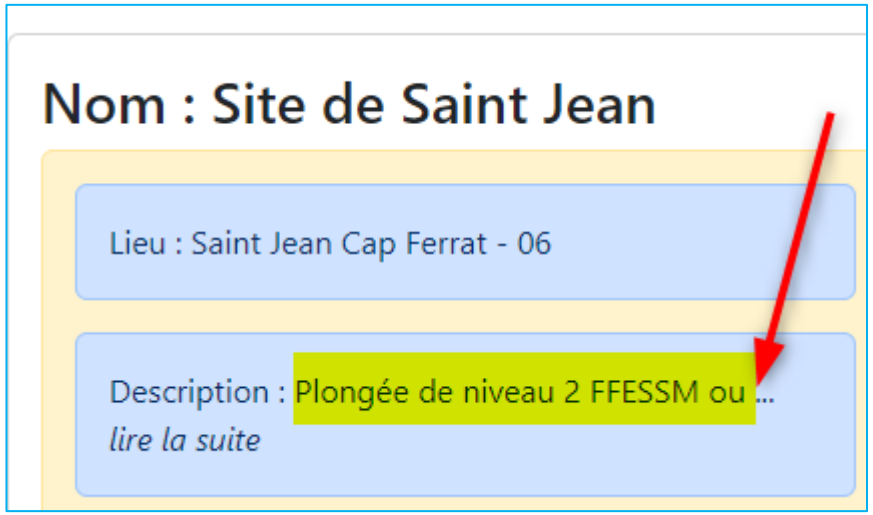
```
14    declarations: [
15      DivesListComponent,
16      DivesDetailComponent,
17      DivesListChildComponent,
18      DivesListChild2Component,
19      // ********************************
20      // on déclare AUSSI les pipes
21      FilterPipe
22    ],
23    imports: [
24      CommonModule,
25      HttpClientModule,
26      RouterModule,
27      FormsModule
28    ],
```

dives-list.component.html

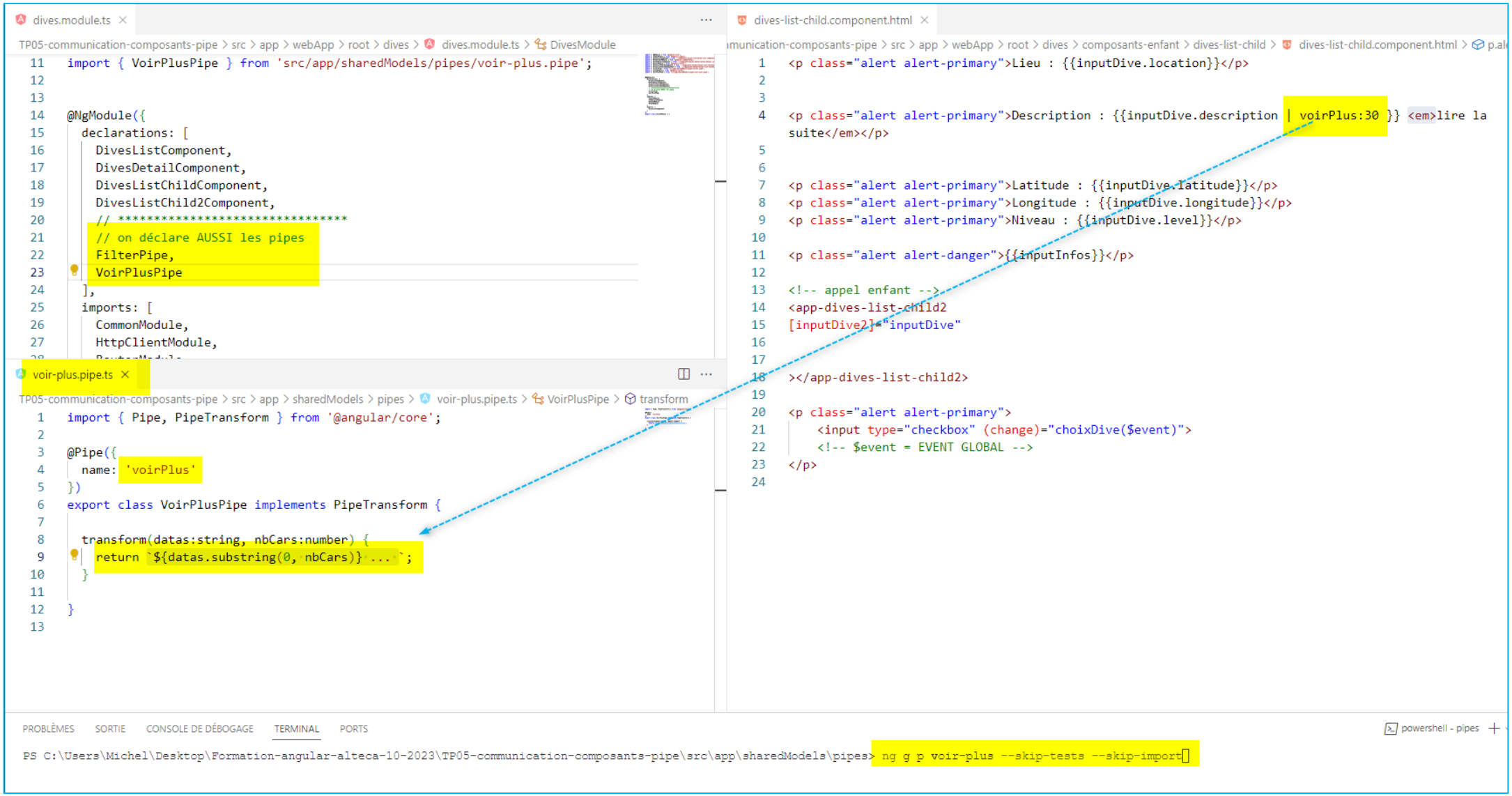
TP05-communication-composants-pipe > src > app > webApp > root > dives > composants > dives-list > dives-list

```
11          </div>
12        </div>
13    </div>
14
15    <!-- Filtre : PIPE  -->
16    <div class="row">
17        <div class="alert alert-warning">
18            <h2>Filtrer l'affichage</h2>
19            <!-- template Driven Forms => utilisation de la directive [ngModel] p
20            <!-- single Way Binding [ngModel] : TS => HTML -->
21            <!-- single Way Binding (ngModel) : HTML => TS -->
22            <!-- Double Way Binding  -->
23            <!-- [()] banana in a box -->
24            <input type="text" class="form-control" [(ngModel)]="texteSaisi">
25        </div>
26    </div>
27
28    <div class="row">
29        <div class="col-md-3" *ngFor="let dive of dives | filter:texteSaisi">
30
31            <div class="card">
32
33                <div class="card-body">
34                    <h3 class="card-title">Nom : {{dive.name}}</h3>
35
```

TP : Créer un pipe qui affiche seulement les 20 premiers caractères de la description avec la méthode javascript substring()



TP : Créer un pipe qui affiche seulement les 20 premiers caractères de la description avec la méthode javascript substring()



dives.module.ts

```
import { VoirPlusPipe } from 'src/app/sharedModels/pipes/voir-plus.pipe';


@NgModule({
  declarations: [
    DivesListComponent,
    DivesDetailComponent,
    DivesListChildComponent,
    DivesListChild2Component,
    // ********************************
    // on déclare AUSSI les pipes
    FilterPipe,
    VoirPlusPipe
  ],
  imports: [
    CommonModule,
    HttpClientModule,
```

voir-plus.pipe.ts

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'voirPlus'
})
export class VoirPlusPipe implements PipeTransform {

  transform(datas:string, nbCars:number) {
    return `${datas.substring(0, nbCars)} ...`;
  }

}
```

dives-list-child.component.html

```
<p class="alert alert-primary">Lieu : {{inputDive.location}}</p>


<p class="alert alert-primary">Description : {{inputDive.description | voirPlus:30 }} <em>lire la suite</em></p>


<p class="alert alert-primary">Latitude : {{inputDive.latitude}}</p>
<p class="alert alert-primary">Longitude : {{inputDive.longitude}}</p>
<p class="alert alert-primary">Niveau : {{inputDive.level}}</p>

<p class="alert alert-danger">{{inputInfos}}</p>

<!-- appel enfant -->
<app-dives-list-child2
[inputDive2]="inputDive"


></app-dives-list-child2>

<p class="alert alert-primary">
    <input type="checkbox" (change)="choixDive($event)">
    <!-- $event = EVENT GLOBAL -->
</p>
```

PROBLÈMES SORTIE CONSOLE DE DÉBOGAGE TERMINAL PORTS

```
PS C:\Users\Michel\Desktop\Formation-angular-alteca-10-2023\TP05-communication-composants-pipe\src\app\sharedModels\pipes> ng g p voir-plus --skip-tests --skip-import
```