



Formation ANGULAR

Chapitre 2

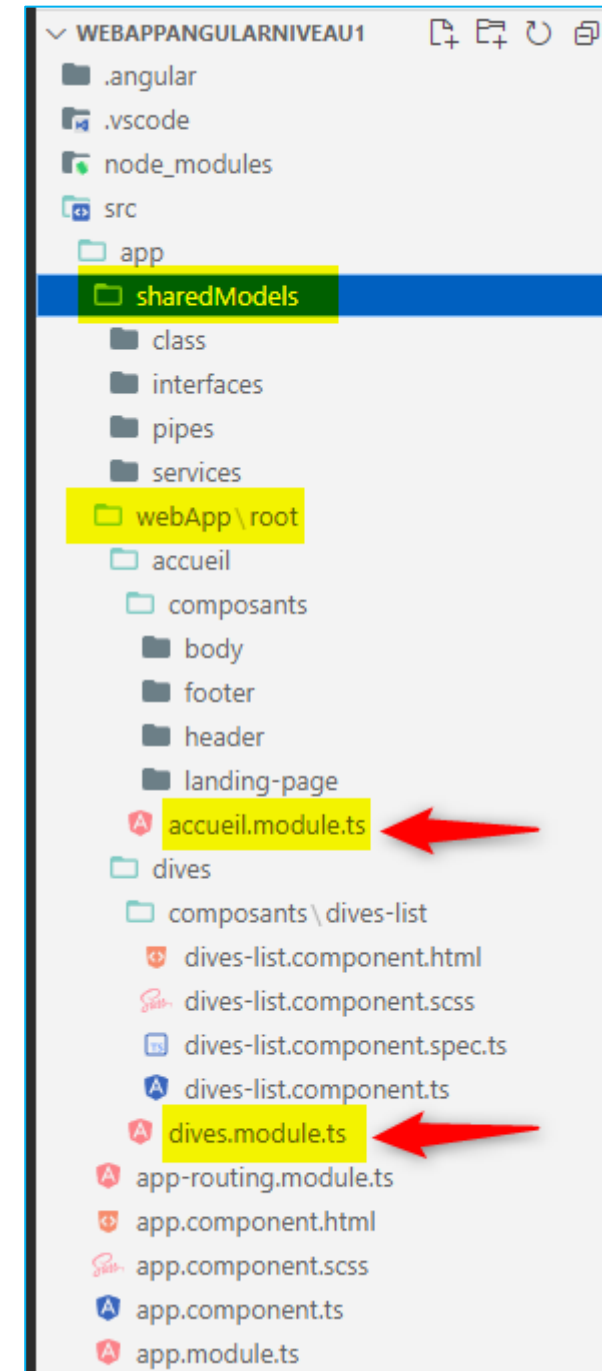
Animé par Michel BOCCIOLESI

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
- 

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

1^{er} exemple : le service

```
dives-list.component.ts x
dives-list > dives > composants > dives-list > dives-list.component.ts > DivesListComponent
16 // @ViewChild('btnPlus') eltBtn!: ElementRef;
17 @ViewChildren('btnPlus') eltBtnCollection!: QueryList<ElementRef>;
18
19 // 2- const
20 // 1'injection de dépendances permet de lier un service
21
22 constructor(
23     private _service: DivesService,
24     private _title: Title,
25     private _meta: Meta
26 ) {
27     console.warn('Constructor');
28     this.dives = [];
29     this._title.setTitle('Liste des Plongées SEO ....');
30     this._meta.addTags([
31         { name: 'description', content: 'Texte de description' },
32         { name: 'author', content: 'MB CREATION WEB' }
33     ]);
34 }
35
36 // -----
37 // 3- lifeCycle
38 // -----
39 ngOnInit(): void {
40     // Chargement des datas
41     console.warn('NgOnInit');
42     this._service.getDives().subscribe(
43         // .subscribe(
44         //     // next seulement ...
45         //     (datas: Dives[]) => {
46         //         this.dives = datas;
47         //     }
48         // );
49
50     .subscribe({
51         next: // Call Back OK
52         (datas: Dives[]) => {
53             this.dives = datas;
54         },
55         error:
56         e => console.log(e)
57     },
58     complete:
59     () => console.log('Observer Complete')
60 );
61 }

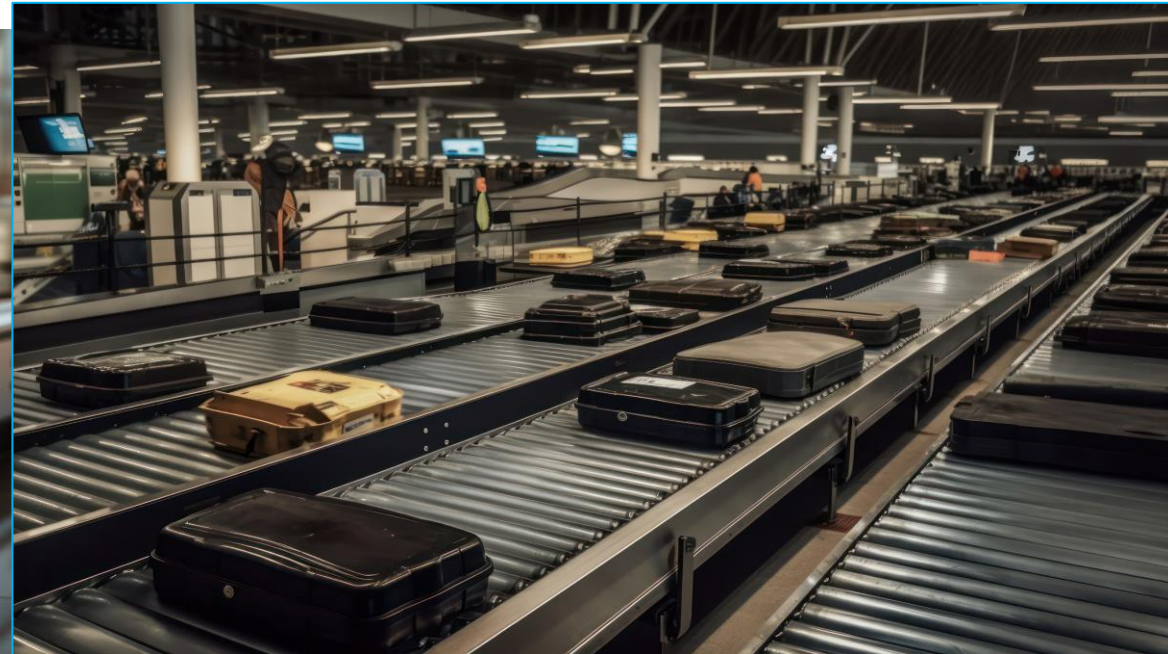
dives.service.ts x
TP03-services-http > src > app > sharedModels > services > dives.service.ts > DivesService > getDives
1 import { Injectable } from '@angular/core';
2 import { Dives } from '../class/dives';
3 import { HttpClient } from '@angular/common/http';
4 import { Observable, tap, map } from 'rxjs';
5
6 @Injectable({
7     providedIn: 'root'
8 })
9
10 export class DivesService {
11
12     // 1- props
13     private dives: Dives[];
14
15     // 2- const
16     constructor(private _http: HttpClient) {
17         this.dives = [];
18     }
19
20     // 3- Méthodes
21     public getDives = (): Observable<Dives[]> => {
22
23         const API_URL = 'http://localhost:3001/dives';
24         // const API_URL = 'https://dev.webjs.fr/dives.json';
25
26         return this._http.get<Dives[]>(API_URL).pipe(
27             // le pipe permet d'enchaîner les opérateurs RXJS
28             // 1 opérateur RXJS est une fonction qui modifie
29             // ou transforme la séquence(stream ou flux) de valeurs émises par l'observable
30
31             tap(
32                 // opérateur de debug => console
33                 (response: any) => {
34                     console.log('----- Réponse depuis le service : ', response);
35                 }
36             ),
37             map(
38                 (datas: Dives[]) => {
39                     return datas.filter(
40                         (data: Dives) => {
41                             // return data.id===5
42                             return data.id >= 1
43                         }
44                     );
45                 }
46             )
47         );
48     }
49 }
```

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 la 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)

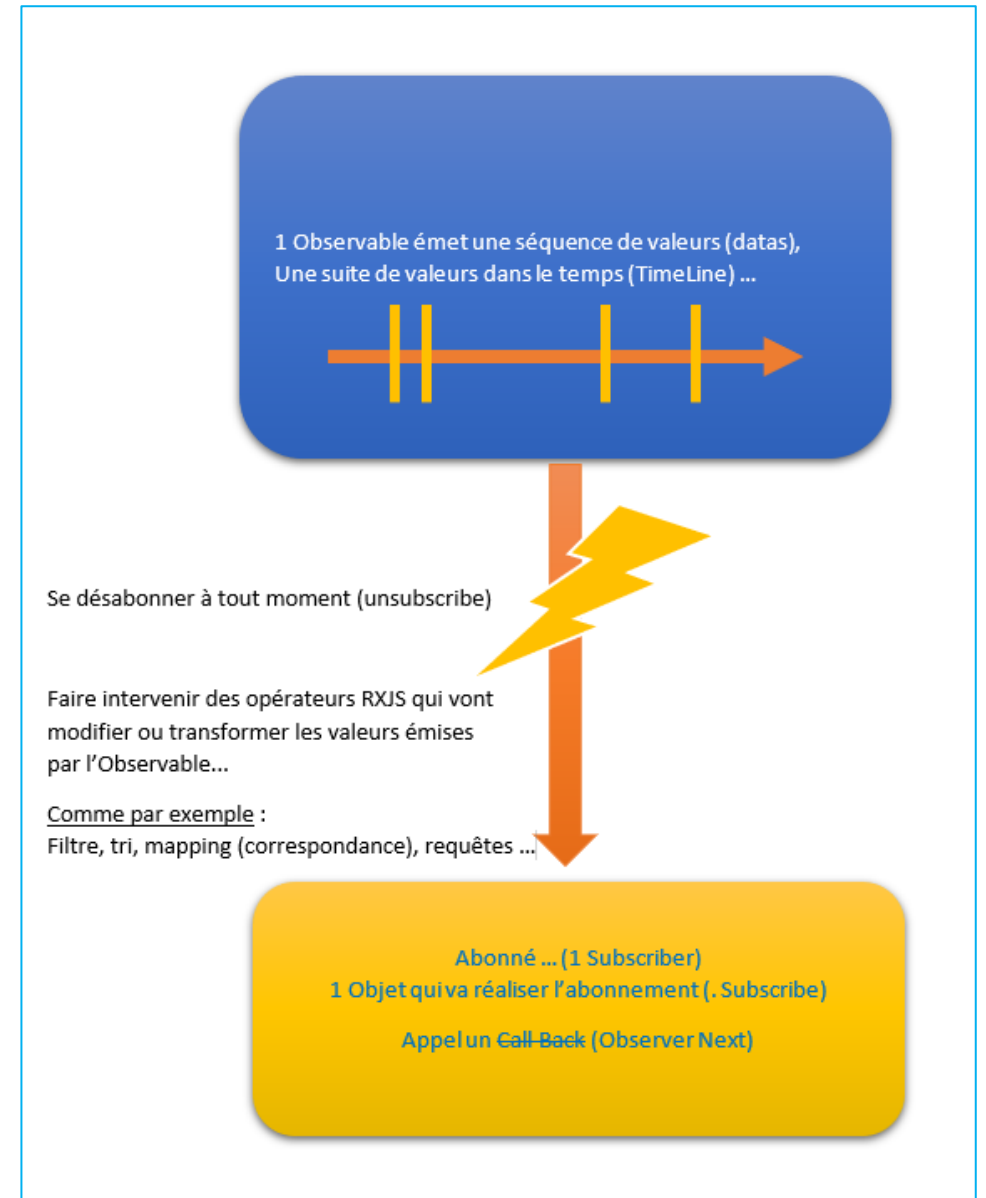
Opérateurs : des fonctions tap, map, merge, filter qui vont pouvoir modifier/transformer ce flux de valeurs émises par l'observable

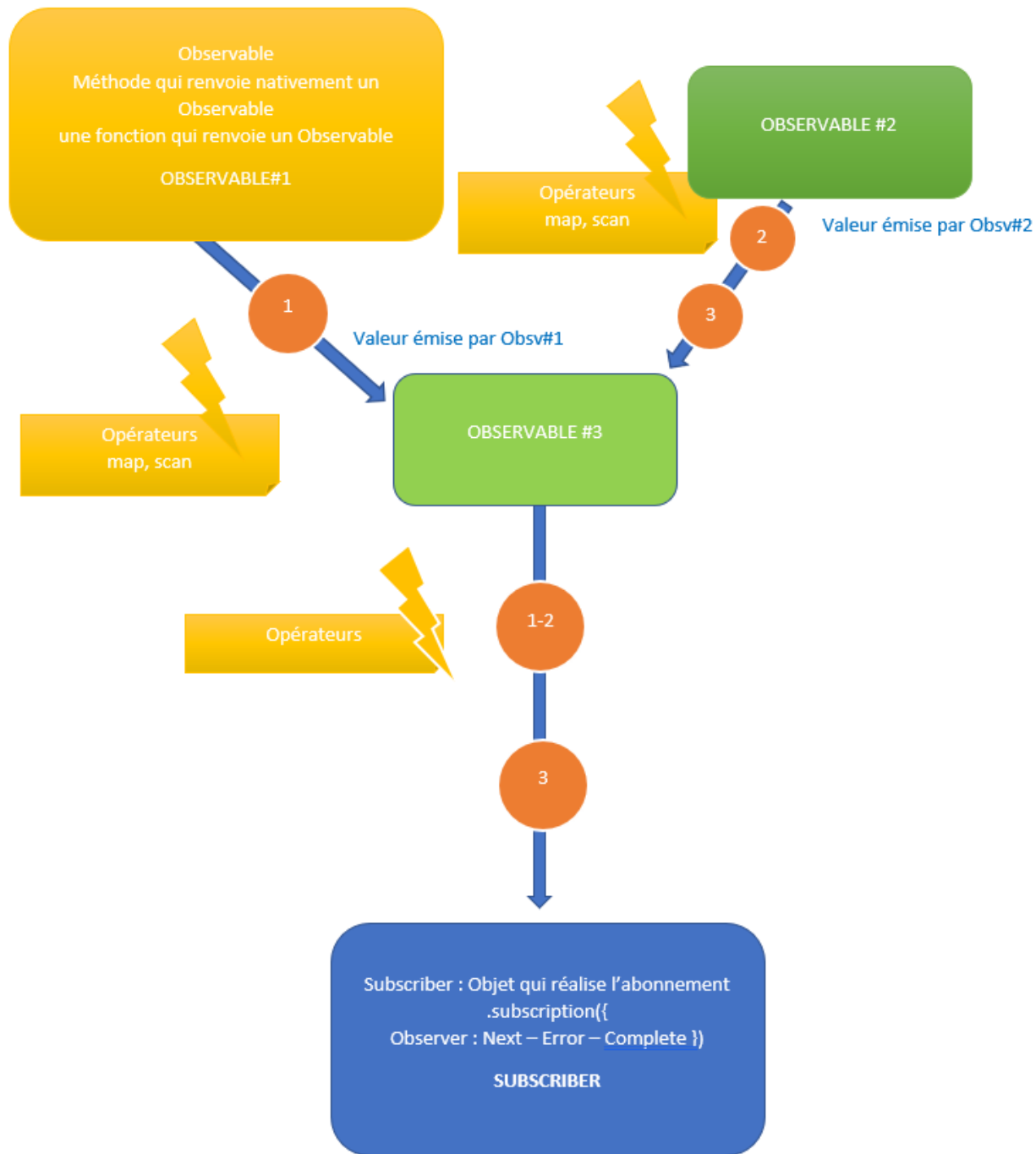
Dans le subscribe => Les observers (fcts de call back) qui se définissent dans le subscribe : Next - Error – Complete

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 !





On peut facilement les recomposer ou les recombinaer 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

The image shows a code editor with four files open, illustrating the basic concepts of Angular routing:

- app-routing.module.ts**: Defines the routes for the application. A route is defined for the path `'liste-des-plongees'` with the component `DivesListComponent`.

```
1 import { NgModule } from '@angular/core';
2 import { RouterModule, Routes } from '@angular/router';
3
4 // import des composants
5 import { DivesListComponent } from '../webApp/root/dives/composants/dives-list/';
6 import { BodyComponent } from '../webApp/root/accueil/composants/body/body.comp';
7
8 const routes: Routes = [
9   { path: '', component: BodyComponent },
10   { path: 'liste-des-plongees', component: DivesListComponent },
11 ];
12
13 @NgModule({
14   imports: [RouterModule.forRoot(routes)],
15   exports: [RouterModule]
16 })
17 export class AppRoutingModule { }
18
19
```
- header.component.html**: Contains the navigation menu. A link is defined with the `routerLink` attribute pointing to the `'liste-des-plongees'` route.

```
4 <div class="collapse navbar-collapse" id="navbar-collapse">
5   <ul class="navbar-nav">
6     <li class="nav-item">
7       <a class="nav-link" href="#>
8     </li>
9     <li class="nav-item">
10      <a class="nav-link" routerLink="liste-des-plongees">
11    </li>
12     <li class="nav-item">
13      <a class="nav-link" href="#>
14    </li>
15   </ul>
16 </div>
17 </nav>
18
```
- landing-page.component.html**: Contains the `<router-outlet>` directive, which is used to render the component specified in the route configuration.

```
1 <!-- <div class="container"> -->
2   <app-header></app-header>
3
4   <div class="alert alert-warning">
5     <!-- comparé à un composant angular -->
6     <router-outlet></router-outlet>
7   </div>
8
9   <app-footer></app-footer>
10
11
```
- accueil.module.ts**: Defines the module for the landing page, importing the `RouterModule` and the `DivesListComponent`.

```
11 @NgModule({
12   declarations: [
13     LandingPageComponent,
14     HeaderComponent,
15     FooterComponent,
16     BodyComponent
17   ],
18   imports: [
19     CommonModule,
20     RouterModule,
21
```

Routing : Concepts de base

app-routing.module.ts

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

```
1 import { NgModule } from '@angular/core';
2 import { RouterModule, Routes } from '@angular/router';
3 import { BodyComponent } from '../webApp/root/accueil/composants/body/body.component';
4 import { DivesListComponent } from '../webApp/root/dives/composants/dives-list/dives-list.component';
5 import { Page404Component } from '../sharedModels/composants/page404.component';
6 import { ContactsComponent } from '../webApp/root/form/composants/contacts/contacts.component';
7 import { DivesDetailComponent } from '../webApp/root/dives/composants/dives-detail/dives-detail.component';
8
9 const routes: Routes = [
10   { path: '', component: BodyComponent },
11   // domain.tld/ (page de démarrage)
12   // { path: 'accueil', component: BodyComponent }, // domain.tld/accueil
13   { path: 'accueil', redirectTo: '', pathMatch: 'full' },
14   // { path: 'nouvelle-url', component: DivesListComponent },
15   // { path: 'ancienne-url-seo-déjà-référencée',
16   //   redirectTo: 'nouvelle-url', pathMatch: 'full' },
17   { path: 'contactez-nous', component: ContactsComponent },
18   { path: 'liste-des-plongees', component: DivesListComponent },
19   { path: 'liste-des-plongees/:id', component: DivesDetailComponent },
20
21   // Gestion des erreurs : toujours en dernier !!
22   // { path: '**', redirectTo: '', pathMatch: 'full' }
23   { path: '**', component: Page404Component }
24 ];
25
26 @NgModule({
27   imports: [RouterModule.forRoot(routes)],
28   exports: [RouterModule]
29 })
30 export class AppRoutingModule { }
```

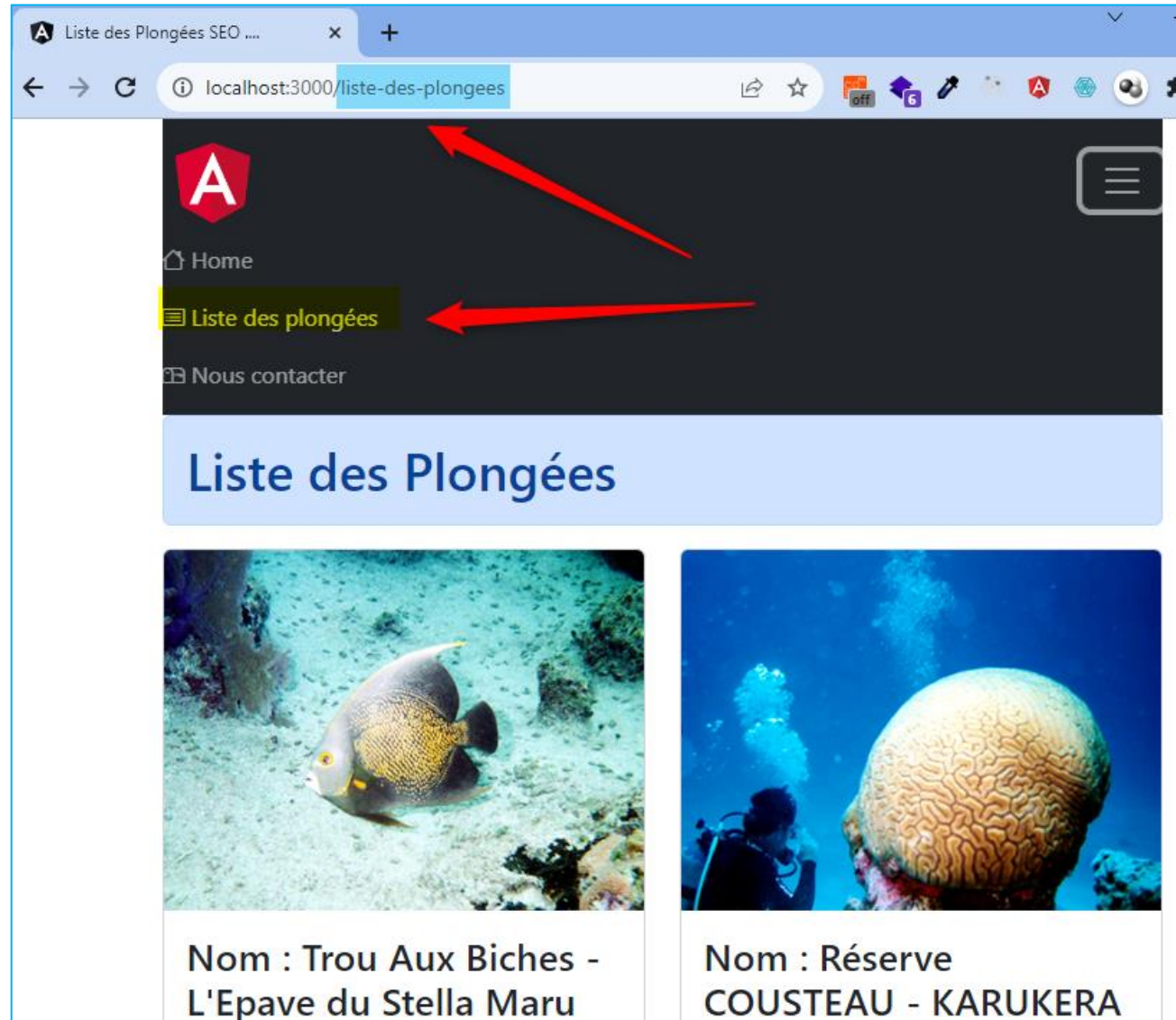
header.component.html

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

```
1 <nav class="navbar navbar-expand-lg bg-dark navbar-dark sticky-top">
2   
3
4   <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-
5     <span class="navbar-toggler-icon"></span>
6   </button>
7
8   <div class="collapse navbar-collapse" id="menu">
9     <ul class="navbar-nav">
10       <li class="nav-item">
11         <a class="nav-link"
12           routerLink="accueil"
13           ><i class="bi bi-house"></i> Home</a>
14       </li>
15
16       <li class="nav-item">
17         <!-- <a class="nav-link"
18           [routerLink]="['liste-des-plongees']"
19           ><i class="bi bi-card-list"></i> Liste des plongées</a> -->
20         <a class="nav-link"
21           routerLink="liste-des-plongees"
22           ><i class="bi bi-card-list"></i> Liste des plongées</a>
23       </li>
24
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     </ul>
31   </div>
32 </nav>
```

Routing : Concepts de base

A screenshot of a web browser displaying a web application. The browser's address bar shows the URL `localhost:3000/liste-des-plongees`. The application's navigation menu, located on the left, includes a red shield logo with a white 'A', a 'Home' link, a 'Liste des plongées' link (highlighted in yellow), and a 'Nous contacter' link. Two red arrows point from the 'Liste des plongées' link to the URL in the address bar and to the main content area. The main content area features a light blue header with the title 'Liste des Plongées'. Below this header, there are two cards. The first card displays an image of a fish and has the text 'Nom : Trou Aux Biches - L'Epave du Stella Maru'. The second card displays an image of a diver and a large brain coral and has the text 'Nom : Réserve COUSTEAU - KARUKERA'.



Localhost:3000/liste-des-plongees

Home

Liste des plongées

Nous contacter

Liste des Plongées



Nom : Trou Aux Biches - L'Epave du Stella Maru



Nom : Réserve COUSTEAU - KARUKERA

Routing : Concepts de base

landing-page.component.html ×

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

```
1 <div style="min-height:100vh;" class="alert alert-warning">
2   <app-header></app-header>
3   <!-- <app-body></app-body> -->
4   <!-- <app-dives-list></app-dives-list> -->
5
6   <!-- router-outlet primary -->
7   <div class="alert alert-primary">
8     <router-outlet></router-outlet>
9   </div>
10
11   <!-- router-outlet autres -->
12   <div class="alert alert-success">
13     <router-outlet name="outlet-2"></router-outlet>
14   </div>
15 </div>
16 <app-footer></app-footer>
```

↑

app-routing.module.ts ×

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

```
16 domain.tld/accueil
17 // { path:'accueil' , redirectTo:'',
18   pathMatch:'full'},
19 // { path:'nouvelle-url',
20   component:DivesListComponent},
21 // { path:'ancienne-url-seo-déjà-référencée',
22   redirectTo:'nouvelle-url', pathMatch:'full'},
23 { path: 'liste-des-plongees', component:
24   DivesListComponent },
25 { path: 'liste-des-plongees/:id', component:
26   DiveDetailComponent },
27
28 { path: 'routing-angular', component:
29   CompRoutingOutletComponent, outlet: 'outlet-2',
30   canActivate: [RouteGuardService] },
31
32 // 1ère manière de faire : charger un composant depuis
33 // un module qui a son propre système de routage forChild
34 // {
35 //   path: 'mon-compte-client',
36 //   component: ClientLandingPageComponent,
37 //   children: clientRoutes
38 // },
39
40 // 2nde manière optimisée : lazy loading avec ES2015
41 // (promesses-import)
42 // {
43 //   path: 'mon-compte-client',
44 //   component: ClientLandingPageComponent,
45 //   loadChildren:
46 //     ():any => import('./webApp/compte-client/
47 //     compte-client.module').then(
```

↑

header.component.html ×

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

```
16
17
18   </a>
19 </li>
20
21   <li class="nav-item">
22     <a class="nav-link" routerLink="mon-compt
23   </li>
24
25   <li class="nav-item">
26     <a class="nav-link" [routerLink]="[{
27       outlets : {
28         'outlet-2': 'routing-angular'
29       }]
30     ">
31     [skipLocationChange]="true"><i class="bi bi-m
32   </li>
33
34   <li class="nav-item">
35     <a class="nav-link"><i class="bi bi-mailb
36   </li>
37 </ul>
38 </div>
39 </nav>
```

↑

Routing : Concepts avancés – QueryParams - Router



Nom : Trou Aux Biches - L'Epave du Stella Maru

Description : Plongée de niveau 2 FFESSM ou PADI au arge de la barrière de Corail, départ à 8h30. Profondeur -30 mètres. Rascasses Volantes, murènes ...

[Plus d'infos](#)



Liste des Plongées SEO ...

localhost:3000/liste-des-plongees

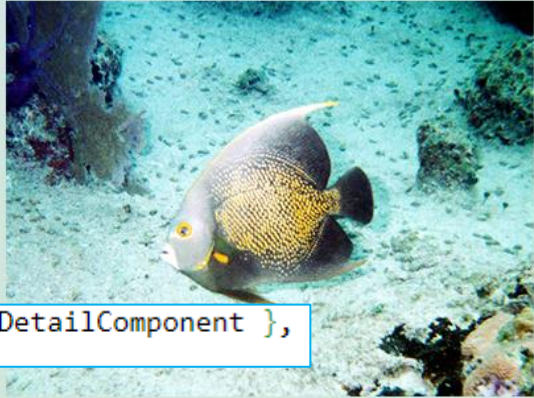
Plongée N° 1

Nom : Trou Aux Biches - L'Epave du Stella Maru

Description : Plongée de niveau 2 FFESSM ou PADI au arge de la barrière de Corail, départ à 8h30. Profondeur -30 mètres. Rascasses Volantes, murènes ...

Latitude : -20.0363

Longitude : 57.544700000000034



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

Routing : Concepts avancés – QueryParams - Router

The image shows a code editor with two files open. The left file, `dives-list.component.html`, contains the following HTML code:

```
1 <h1 class="alert alert-primary">Liste des Plongées</h1>
2
3 <div class="row">
4
5   <div class="col-md-6" *ngFor="let dive of dives">
6
7     <div class="card">
8       <!-- récupérer chaque image -->
9       
10      <div class="card-body">
11        <h3 class="card-title">Nom : {{dive.name}} </h3>
12        <p class="card-text">Description : {{dive.description}}</p>
13
14        <!-- <a
15          [routerLink]="[dive.id]"
16          [queryParams]="{dive.name}"
17        > -->
18        <!-- on interprète un des params de l'objet -->
19        <a
20          [routerLink]="[dive.id]"
21          [queryParams]="{dive}"
22          [skipLocationChange]="true"
23        >
24
25        <!-- on passe tout l'objet -->
26        <button class="btn btn-primary" #btnPlus>En Savoir Plus</button>
27      </a>
28    </div>
29  </div>
30 </div>
31 </div>
```

The right file, `dives-detail.component.ts`, contains the following TypeScript code:

```
1 import { Component, OnInit } from '@angular/core';
2 import { ActivatedRoute, Router } from '@angular/router';
3 import { Dives } from 'src/app/sharedModels/class/dives';
4
5 @Component({
6   selector: 'app-dives-detail',
7   templateUrl: './dives-detail.component.html',
8   styleUrls: ['./dives-detail.component.scss']
9 })
10 export class DivesDetailComponent implements OnInit {
11   // 1- props
12   public title: string;
13   public dive: Dives;
14
15   // 2- const
16   constructor(
17     private _routeActive: ActivatedRoute,
18     private _router: Router
19   ) {
20     this.title = '';
21     this.dive = new Dives();
22   }
23
24   // 3- lifeCycle
25   ngOnInit(): void {
26
27     const id = this._routeActive.snapshot.params['id'];
28     console.log(id);
29     this.title = `Plongée N° ${id}`;
30     // re-consommer le service ?
31     // ré-exploiter les datas stockées en storage ou en indexedDB
32     // Evidemment utiliser les fonctions de routage d'Angular ...
33   }
```

A red arrow points from the `[routerLink]` and `[queryParams]` attributes in the HTML file to the `ActivatedRoute` and `Router` imports in the TypeScript file.

Routing : Concepts avancés – QueryParams - Router

The image displays a code editor with five files open, illustrating the implementation of advanced routing concepts in Angular, specifically focusing on QueryParams and Router.

dives-list.component.html

```
1 <h1 class="alert alert-primary">Liste des Plongées</h1>
2
3 <div class="row">
4
5   <div class="col-md-6" *ngFor="let dive of dives">
6
7     <div class="card">
8       <!-- récupérer chaque image -->
9       
10      <div class="card-body">
11        <h3 class="card-title">Nom : {{dive.name}} </h3>
12        <p class="card-text">Description : {{dive.description}}</p>
13
14        <!-- [routerLink]="[dive.id]" -->
15        <!-- Ecriture simplifiée du routerLink => routerLink="{{dive.id}}<br>
16        <!-- routerLink="details/{{dive.id}}" -->
17        <!-- en chemin relatif -->
18        <button
19
20          routerLink="/liste-des-plongees/details/{{dive.id}}"
21          [queryParams]="{dive: dive.id}"
22          [skipLocationChange]="true"
23
24          class="btn btn-primary" #btnPlus>En Savoir Plus</button>
25
26      </div>
27    </div>
28  </div>
```

dives-details.component.ts

```
1 import { Component, OnInit } from '@angular/core';
2 import { ActivatedRoute } from '@angular/router';
3
4 @Component({
5   selector: 'app-dives-details',
6   templateUrl: './dives-details.component.html',
7   styleUrls: ['./dives-details.component.scss']
8 })
9 export class DivesDetailsComponent implements OnInit {
10
11   // -----
12   constructor(
13     private _routeActive: ActivatedRoute
14   ) {
15     // -----
16   }
17   ngOnInit() {
18     const id=this._routeActive.snapshot.params['param'];
19     console.clear();
20     console.log('ID récupéré : ', id);
21     // TP : passer l'ID à la Vue
22     // -----
23     this._routeActive.queryParams
24       .subscribe(
25         (params:any) => {
26           console.log('QueryParams : ', params);
27           // TP : passer l'objet récupéré à la vue
28         }
29       )
30   }
31 }
```

app-routing.module.ts

```
10 const routes: Routes = [
11
12   { path: '', component: BodyComponent }, // domain.tld/ (page de démarrage)
13   { path: 'liste-des-plongees', component: DivesListComponent },
14
15   // { path:'accueil', component:BodyComponent}, // domain.tld/accueil
16
17   // { path:'nouvelle-url', component:DivesListComponent},
18   // { path:'ancienne-url-seo-déjà-référencée', redirectTo:'nouvelle-url' },
19
20   { path: 'liste-des-plongees/details/:param', component: DivesDetailsComponent },
21
22   { path: '**', component: Page404Component}
23 ]
```

bdd.json

```
1 {
2   "dives": [
3     {
4       "id": 1,
5       "name": "Trou Aux Biches - L'Epave du Stella Maru",
6       "location": "Ile Maurice - Trou aux Biches - Nord/Ouest",
7       "level": 2,
8       "description": "Plongée de niveau 2 FFESSM ou PADI au arge de l'île",
9       "latitude": -20.0363,
10      "longitude": 57.544700000000034,
11      "evaluation": [
```

dives.module.ts

```
1 import { NgModule } from '@angular/core';
2 import { CommonModule } from '@angular/common';
3 import { DivesListComponent } from './dives-list.component';
4 import { HttpClientModule } from '@angular/common/http';
5 import { FormsModule } from './forms.module';
6 import { RouterModule } from '@angular/router';
7 import { DivesDetailsComponent } from './dives-details.component';
8
9
10
11 @NgModule({
12   declarations: [
```

Routing : Concepts avancés – QueryParams - Router

dives-details.component.ts

```
21 ngOnInit() {
22   const id = this._routeActive.snapshot.params['param'];
23   console.clear;
24   console.log('ID récupéré :', id);
25   // TP : passer l'ID à la Vue
26   // -----
27   this._routeActive.queryParams
28     .subscribe(
29       (params: any) => {
30         console.log('QueryParams : ', params);
31         // TP : passer l'objet récupéré à la vue
32         this.dive = params;
33       }
34     )
35 }
36 // -- Méthodes
37 public goBack = () => {
38   this._router.navigateByUri('liste-des-plongees');
39 }
40
41 public goHome = () => {
42   this._router.navigate(
43     [''],
44     {
45       queryParams: {
46         page: 'details-plongée',
47         datas: 15
48       }
49     }
50 )
51 }
```

dives-details.component.html

```
1 <h1>Détail de la plongée N° {{dive.id}}</h1>
2 <!-- TP afficher l'objet passé par routage -->
3
4 <!-- Single way binding -->
5 <!-- TS => HTML : [directive] et {{string interpolation}} -->
6 <!-- HTML => TS : (event du DOM) par ex : (click)-->
7 <p>
8   <button class="btn btn-warning" (click)="goBack()">Go Back</button>
9
10  <button class="btn btn-danger" routerLink="/liste-des-plongees">RouterLink</button>
11 </p>
12 <p>
13   <button class="btn btn-success" (click)="goHome()">Go Home</button>
14 </p>
```

localhost

Liste des plongées

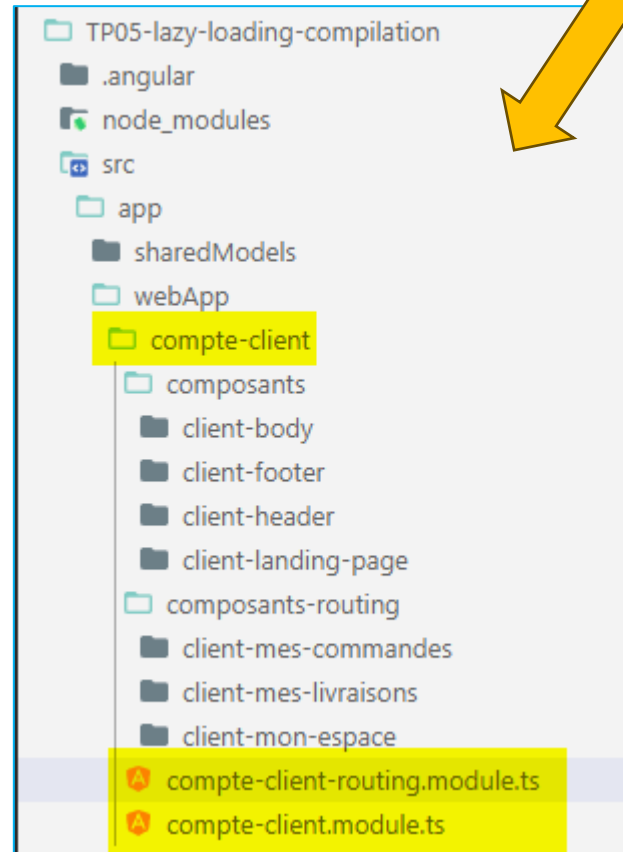
localhost:3000/?page=details-plongée&datas=15

Home Liste des plongées Nous contacter

THE BODY

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

```
composants-routing
├── client-mes-commandes
├── client-mes-livraisons
├── client-mon-espace
├── compte-client-routing.module.ts
├── compte-client.module.ts
└── root
```

```
24 @NgModule({
25   imports: [RouterModule.forChild(routesClient)],
26   exports: [RouterModule]
27 })
28 export class CompteClientRoutingModule { }
29
```


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.



```
PS C:\Angular\Injection-angular-2023-06\TP05-lazy-loading-compilation> npm run build

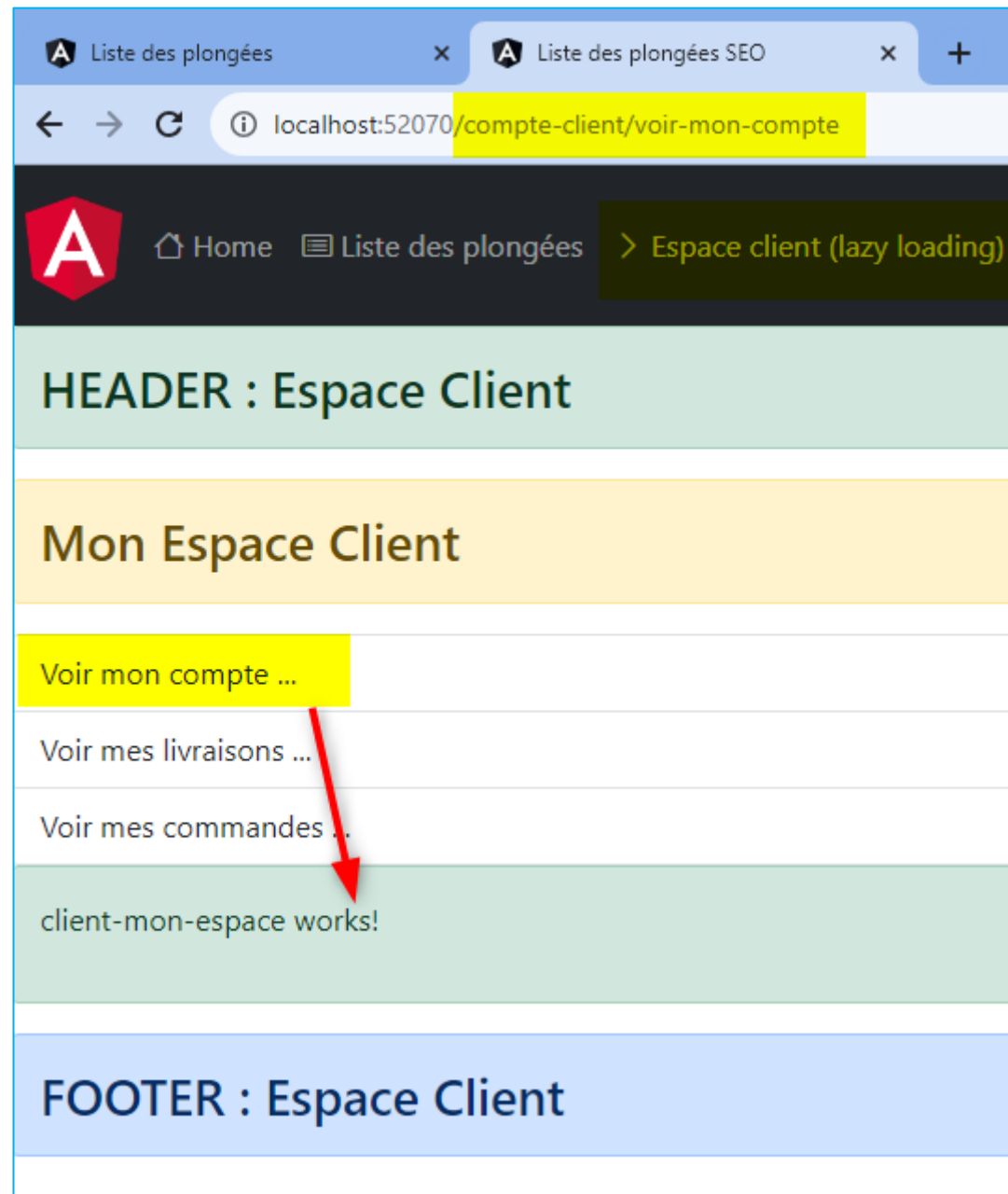
> tp02-bonnes-pratiques-injection-dependances@0.0.0 build
> ng build --stats-json

✓ Browser application bundle generation complete.
✓ Copying assets complete.
" Generating index html...1 rules skipped due to selector errors:
  legend+* -> Cannot read properties of undefined (reading 'type')
✓ Index html generation complete.
```

Initial Chunk Files	Names	Raw Size	Estimated Transfer Size
styles.32ab14ebec5beb97.css	styles	265.62 kB	29.18 kB
main.9281e4b71d326fdf.js	main	246.63 kB	66.78 kB
scripts.118140219e48fbd3.js	scripts	142.96 kB	41.36 kB
polyfills.ed4228387a31b6d8.js	polyfills	33.05 kB	10.64 kB
runtime.6565837a760cd7e6.js	runtime	2.73 kB	1.26 kB
	Initial Total	690.99 kB	149.22 kB

Lazy Chunk Files	Names	Raw Size	Estimated Transfer Size
49.43219d5090c1d8db.js	webApp-compte-client-compte-client-module	1.36 kB	435 bytes

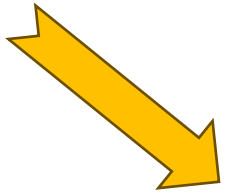
Routing : Concepts avancés – Le Lazy Loading



Routing : Concepts avancés – Le Lazy Loading

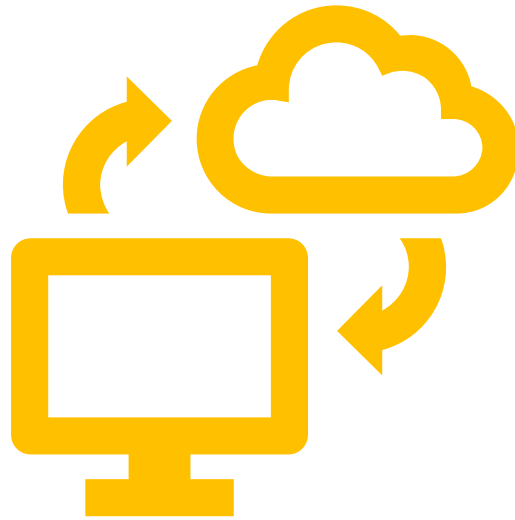
Afin de bien analyser ces customisations « tuning » de compilation, mise en place d'une stratégie de lecture de statistiques avec le « webpack-bundle-analyzer »

<https://www.npmjs.com/package/webpack-bundle-analyzer>

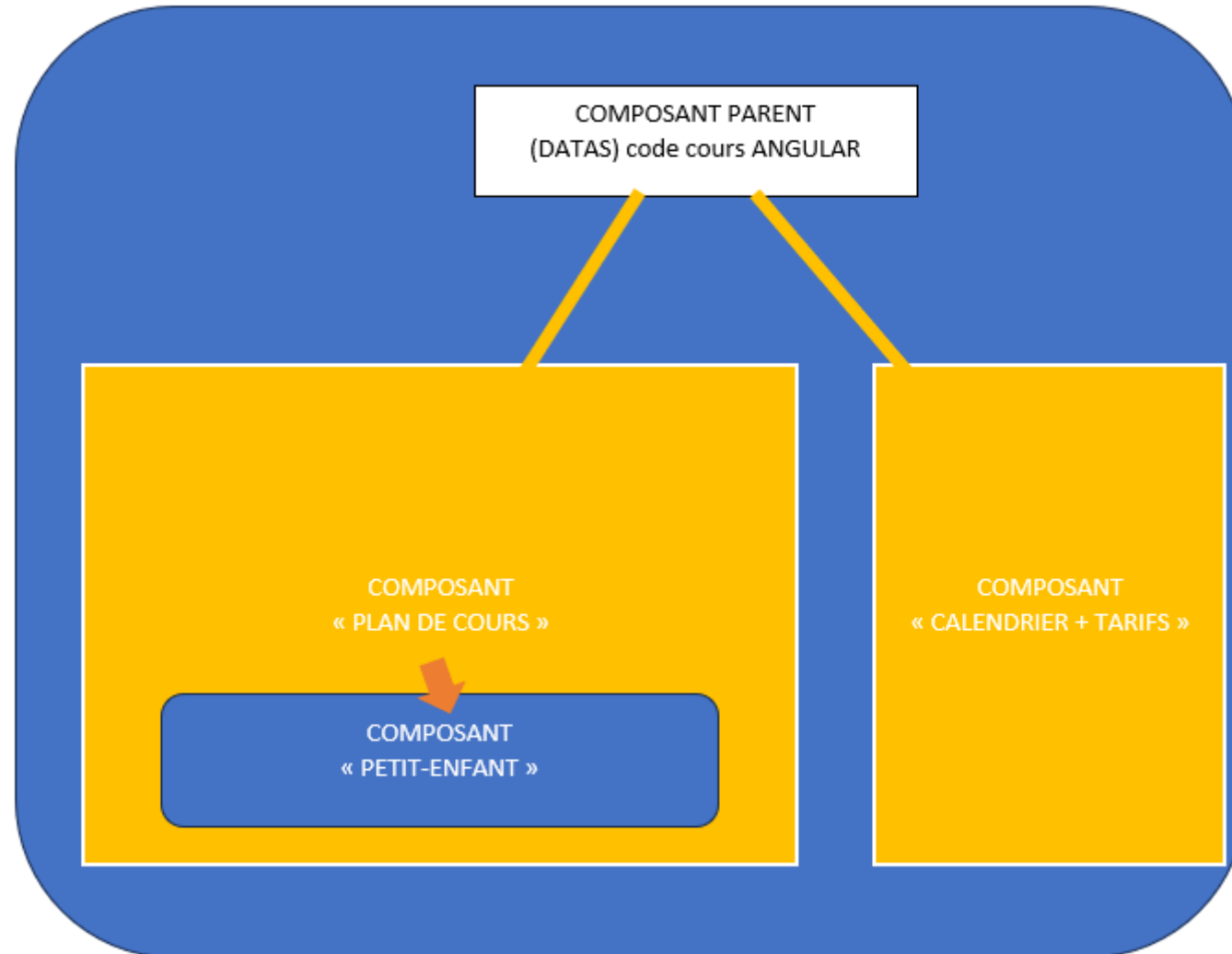


```
package.json ×
P05-lazy-loading-compilation > package.json > ...
5   "ng": "ng",
6   "start": "ng serve --open --port=3000",
7   "build": "ng build --stats-json",
8   "watch": "ng build --watch --configuration development --stats-json",
9   "test": "ng test",
10  "json-server": "json-server --watch src/assets/json/dives.json -p 3001",
11  "analyze": "webpack-bundle-analyzer dist/stats.json"
12  },
```

Input - Output

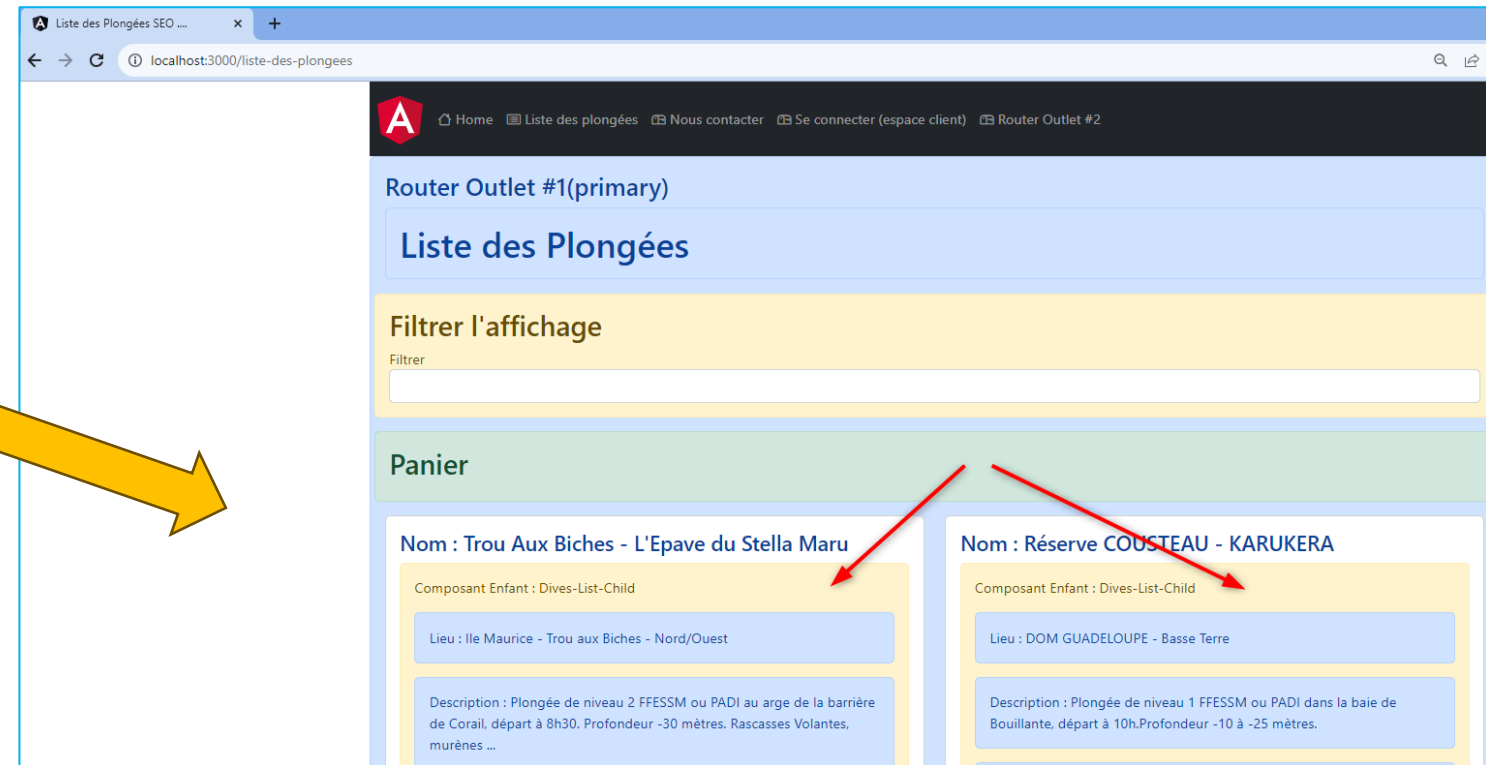
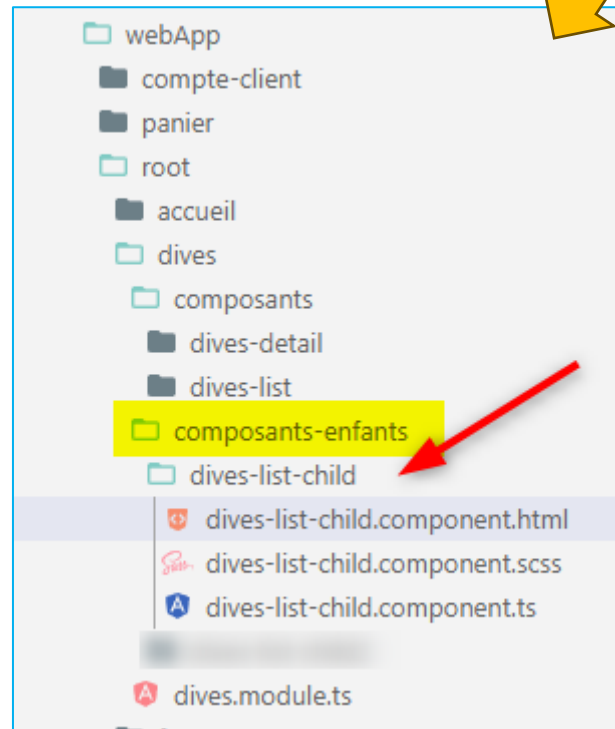


Communication : composants parents et enfants



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 données 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
```

 Home Liste des plongées Nous contacter Se connecter (espace)

Panier

Nom : Trou Aux Biches - L'Epave du Stella Maru

Composant Enfant : Dives-List-Child

Lieu : Ile Maurice - Trou aux Biches - Nord/Ouest

Description : Plongée de niveau 2 FFESSM ou PADI au arge de la barrière de Corail, départ à 8h30. Profondeur -30 mètres. Rascasses Volantes, murènes ...

Latitude : -20.0363

Longitude : 57.544700000000034

Niveau : 2

Composant Enfant : Dives-List-Child #2



Communication : composants parents et enfants

The image displays a code editor with four files open, illustrating the communication between parent and child components in an Angular application. Red arrows highlight the flow of data and events from the parent component to the child components.

dives-list.component.ts

```
114 // 4- méthodes
115 public diveSelected = (e: any) => {
116   console.log('depuis le parent $event => ',e);
117
118   let isChecked=e.paramIsChecked;
119   let diveSelect=e.paramDive;
120
121   console.log(`Plongée : ${diveSelect.name} - Etat : ${isChecked}`);
122
123   if (isChecked) {
124     this.panier.push(diveSelect);
125     console.table(this.panier);
126   } else {
127     let keyPanier = this.panier.indexOf(diveSelect);
128     if (keyPanier >= 0) {
129       this.panier.splice(keyPanier, 1);
130       console.table(this.panier);
131     }
132   }
133 }
134
135
136
```

dives-list-child.component.ts

```
9 export class DivesListChildComponent {
10
11   // les @input et les @output permettent d'établir la co
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
17   @Input() inputDive!:Dives;
18   @Input() inputInfos!:string;
19   // -----
20   @Output() outputEventDive:EventEmitter<any>;
21
22   // 2- const
23   constructor(){
24     this.outputEventDive = new EventEmitter();
25   }
26
27   // 3- méthodes
28   public choixDive = (e:any) => {
29     console.clear();
30     console.log(e.target.checked);
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56

```

dives-list-child2.component.ts

```
1 import { Component, Input } from '@angular/core';
2 import { Dives } from 'src/app/sharedModels/class';
3
4 @Component({
5   selector: 'app-dives-list-child2',
6   templateUrl: './dives-list-child2.component.html',
7   styleUrls: ['./dives-list-child2.component.scss'],
8 })
9 export class DivesListChild2Component {
10
11   @Input() inputDive!:Dives;
12
13 }
14
```

dives-list.component.html

```
41
42 <div class="card-body">
43   <h4 class="card-title">Nom : {{dive.name}} </h4>
44
45   <!-- Composant Enfant -->
46   <div class="alert alert-warning">
47     <p>Composant Enfant : Dives-List-Child</p>
48     <app-dives-list-child
49       [inputDive]="dive"
50       [inputInfos]="infos"
51       (outputEventDive)="diveSelected($event)"
52     ></app-dives-list-child>
53   </div>
54   <!-- Composant Enfant -->
55
56
```

dives-list-child.component.html

```
14
15 <p>Composant Enfant : Dives-List-Child #2</p>
16 <app-dives-list-child2
17   [inputDive]="inputDive"
18 ></app-dives-list-child2>
19
20 </div>
21 <!-- Composant Enfant -->
22
23
24
25 <p class="alert alert-primary">
26   <input type="checkbox" (change)="choixDive($event)">
27 </p>
28 <!-- $event est l'objet global evenement -->
29
30
```

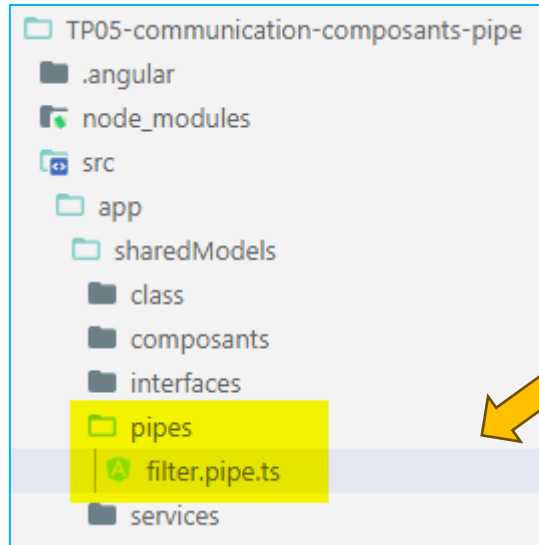
dives-list-child2.component.html

```
1 <p class="alert alert-primary">
2   
4
5
```

Pipes Angular

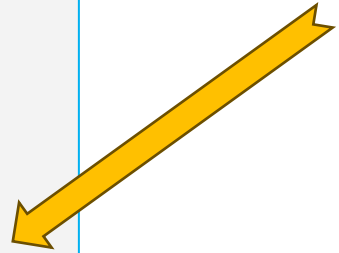


Les PIPES Angular



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

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

```
16 // -----
17 public dives: Dives[] = [];
18 @ViewChild('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.
28             toLocaleLowerCase()) || paramDive.description.toLocaleLowerCase().
29             includes(motif.toLocaleLowerCase())
30         }
31       )
32     }
33   }
34 }
```

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 ],
29 exports: [
```

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>Filtre 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
36       </div>
37     </div>
38   </div>
39 </div>
```



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

Nom : Site de Saint Jean

Lieu : Saint Jean Cap Ferrat - 06

Description : Plongée de niveau 2 FFESSM ou ...
lire la suite



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

```
divesModule.ts
TP05-communication-composants-pipe > src > app > webApp > root > dives > dives.module.ts > DivesModule
11 import { VoirPlusPipe } from 'src/app/sharedModels/pipes/voir-plus.pipe';
12
13
14 @NgModule({
15   declarations: [
16     DivesListComponent,
17     DivesDetailComponent,
18     DivesListChildComponent,
19     DivesListChild2Component,
20     // *****
21     // on déclare AUSSI les pipes
22     FilterPipe,
23     VoirPlusPipe
24   ],
25   imports: [
26     CommonModule,
27     HttpClientModule,
28   ]
29 })
30 export class DivesModule {}

voir-plus.pipe.ts
TP05-communication-composants-pipe > src > app > sharedModels > pipes > voir-plus.pipe.ts > VoirPlusPipe > transform
1 import { Pipe, PipeTransform } from '@angular/core';
2
3 @Pipe({
4   name: 'voirPlus'
5 })
6 export class VoirPlusPipe implements PipeTransform {
7
8   transform(datas:string, nbCars:number) {
9     return `${datas.substring(0, nbCars)} ...`;
10  }
11
12 }
13

divesModule.html
TP05-communication-composants-pipe > src > app > webApp > root > dives > dives-list-child.component.html > p.alert
1 <p class="alert alert-primary">Lieu : {{inputDive.location}}</p>
2
3
4 <p class="alert alert-primary">Description : {{inputDive.description | voirPlus:30}} <em>lire la
  suite</em></p>
5
6
7 <p class="alert alert-primary">Latitude : {{inputDive.latitude}}</p>
8 <p class="alert alert-primary">Longitude : {{inputDive.longitude}}</p>
9 <p class="alert alert-primary">Niveau : {{inputDive.level}}</p>
10
11 <p class="alert alert-danger">{{inputInfos}}</p>
12
13 <!-- appel enfant -->
14 <app-dives-list-child2
15   [inputDive2]="inputDive"
16 ></app-dives-list-child2>
17
18
19
20 <p class="alert alert-primary">
21   <input type="checkbox" (change)="choixDive($event)">
22   <!-- $event = EVENT GLOBAL -->
23 </p>
24
```

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