**Mémo Angular**

Concepts

**Node.js** est une bibliothèque et un environnement d’exécution multi-plateforme pour exécuter des applications JavaScript en dehors du navigateur. Il s’agit d’un outil gratuit et open source utilisé pour créer des applications JavaScript côté serveur.

**Node.JS est un outil :**

- utile pour créer des applications de mise en réseau côté serveur rapides et évolutives

- est un langage idéal pour développer des projets de petite taille

**AngularJS** est un framework de développement d’applications Web open source.

**AngularJS est :**

- mieux adapté pour créer des applications Web côté client.

- est un langage idéal pour créer des applications Web hautement interactives.

Installation

**Pour installer Angular, cela nécessite node et npm**

pour installer NodeJS (choisissez la LTS) : <https://nodejs.org/en/download>

vérifier dans un terminal :

node --version

npm –version

Désinstaller angular : npm uninstall -g @angular/cli

Installer la dernière version d’Angular : **npm install -g @angular/cli**

Installer une version spécifique d’angular : npm install -g @angular/cli@13.3.11

Commandes Angular CLI

**ng version** Afficher la version installée

**ng help** Afficher les commandes de ng

**ng new** nomprojet Créer un nouveau projet Angular

(se mettre dans le repertoire souhaité):

add angular routine: yes

choisir SCSS

**ng serve** Commande dans le dossier projet, Construit et sert votre application

**ng generate component** component-name Créé un nouveau composant

**ng directive | pipe | service | class | guard | interface | enum | module** | **test**

**ng build** Build le projet → dossier dist à la racine du projet

**package. json**

est véritablement le coeur de tout projet utilisant node. Il permet de définir :

- les metadatas d'un projet,

- d'installer toutes ses dépendances,

- d'exécuter les différents scripts npm.

"@angular/animations": "**~**13.3.0" ~ : permet d’indiquer de prendre la dernière version 13.x.x sans passer à la 14

**angular.json**

La configuration d’Angular CLI. Permet de définir plusieurs valeurs par défaut et de configurer également les fichiers inclus lors de la création de votre projet.

Directives Angular

**\*ngFor** permet de boucler sur un array et d'injecter les éléments dans le DOM.

**\*ngIf** permet de réaliser une condition sur une expression booléenne.

Code

**Une page « Accueil » contenant un titre, une image et un paragraphe qui apparaît/disparaît à l’appui d’un bouton « + ».**

app.component.html

<app-accueil-paragraph></app-accueil-paragraph>  
<router-outlet></router-outlet>

xxxxxx.component.hmtl

<!-- titre et image de la page -->  
<h1>titre de ma page accueil</h1>  
<img src="assets/zen.jpg" width=30% height=30% alt="zen">  
<p></p>  
  
<!-- bouton + qui appelle la methode affichage sur clic -->  
<button type="button" class="btn" (click)="affichage();">+</button>  
<p></p>  
  
<!-- texte affiche suivant condition (true/false) -->  
<div *\*ngIf*="condition;then ***contenu***"></div>  
<ng-template #***contenu***>texte paragraphe</ng-template>

xxxxxx.component.ts

// declaration condition  
public condition: any;  
  
// methode affichage : alterne condition true/false  
affichage = () : void => {  
 this.condition = !this.condition;  
};

**Une page avec 2 composants qui affiche une liste d’artistes (nom, image).**

**La liste est initiée et bouclée dans un composant : article-liste.components**

**Le nom et l’image sont affichés dans un composant fille : article-fiche.components**

article-liste.components.html

<h3>Liste des artistes</h3>  
<!-- boucle de balayage de la liste et envoi au composant enfant (artiste-fiche) -->  
<div *\*ngFor*="let ***artiste*** of artistes">  
 <app-artiste-fiche [artiste]="***artiste***"></app-artiste-fiche>  
</div>

article-liste.components.ts

export class ArtisteListeComponent implements OnInit {  
  
 // initialisation liste artistes  
 artistes : Artiste[] = [  
 {  
 nom : "Gauguin",  
 urlImage : "assets/gauguin.jpg"  
 },{  
 nom : "Monet",  
 urlImage : "assets/monet.jpg"  
 }, {  
 nom :"Van Gogh",  
 urlImage :"assets/vangogh.jpg"  
 }  
 ];

Artiste.ts

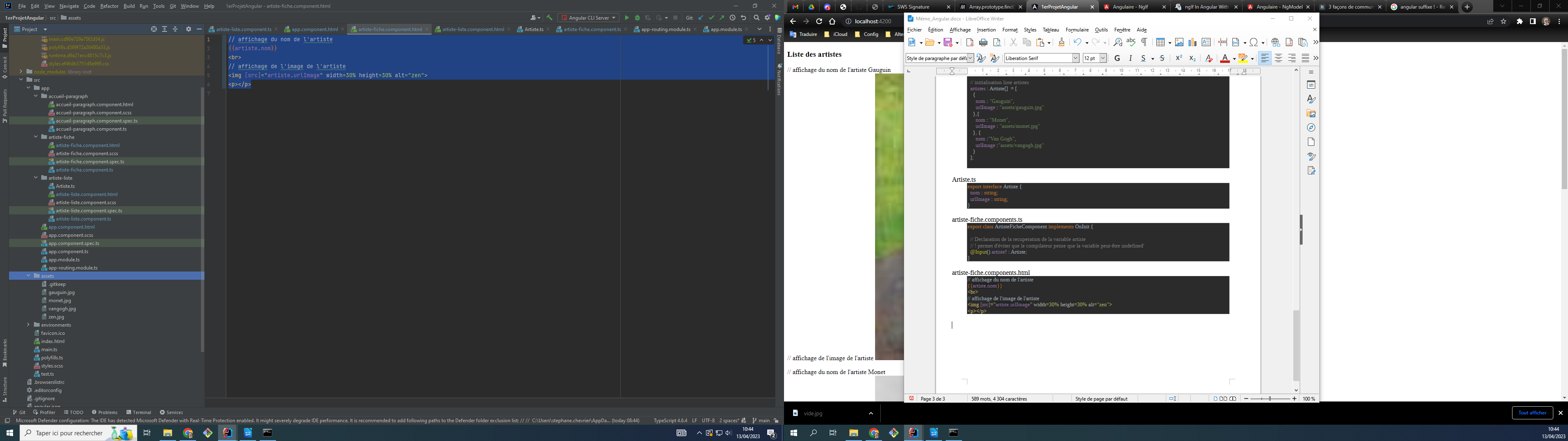
export interface Artiste {  
 nom : string;  
 urlImage : string;  
}

artiste-fiche.components.ts

export class ArtisteFicheComponent implements OnInit {  
  
 // Declaration de la recuperation de la variable artiste  
 // ! permet d'éviter que le compilateur pense que la variable peut-être undefined'  
 @Input() artiste! : Artiste;  
}

Attention dans ce cas il s’agit d’un objet. Si on modifie la valeur de l’objet, Angular ne va pas mettre à jour le composant enfant car la reference de l’objet de change pas. Ce probleme ne va pas se passer quand il s’agit d’une chaine ou d’un nombre.

artiste-fiche.components.html

<!-- affichage du nom de l'artiste -->  
{{artiste.nom}}  
<br>  
<!-- affichage de l'image de l'artiste -->  
<img [src]="artiste.urlImage" width=30% height=30% alt="zen">  
<p></p>

dossier assets (convention angular du dossier où sont stockés les fichiers)

**Une barre de navigation permettant de passer d’une page à l’autre grâce au routing Angular .**

App-routing.module.ts

import { AccueilParagraphComponent} from "./accueil-paragraph/accueil-paragraph.component";  
import { ArtisteListeComponent} from "./artiste-liste/artiste-liste.component";  
  
const routes: Routes = [  
 { path: 'Accueil', component: AccueilParagraphComponent},  
 { path: 'Artistes', component: ArtisteListeComponent}  
];

bar-nav.component.html

<ul class="nav navbar-nav">  
 <li>  
 <a routerLink="Accueil">Accueil</a>  
 </li>  
 <li>  
 <a routerLink="Artistes">Liste Artistes</a>  
 </li>  
</ul>

bar-nav.component.scss

\* {  
 font-family: 'Lucida Sans';  
}  
.nav{  
 display:flex;  
 justify-content: center;  
 background-color: antiquewhite;  
 padding: 20px;  
 margin: 0;  
 color: white;  
}  
li {  
 margin-right: 20px;  
 margin-left: 20px;  
 list-style: none;  
}

**Un bouton « supprimer » à côté de chaque artiste qui permet de supprimer l’artiste de la liste des artistes.**

Artiste-fiche.component.html

<!-- bouton + qui appelle la methode suppressionArtiste sur clic -->  
<button type="button" class="btn" (click)="suppressionArtiste()">supprimer cet artiste</button>

Artiste-fiche.component.ts

// Declaration de l'evenement qui permet d'envoyer l'Artiste du composant enfant (celui-ci) au parent  
@Output() artisteSup : ***EventEmitter***<Artiste> = new ***EventEmitter***<Artiste>();  
  
// Methode d'envoi de l'artiste au composant parent  
public suppressionArtiste() : void {  
 this.artisteSup.emit(this.artiste);  
}

Artiste-liste.component.html

<h3>Liste des artistes</h3>  
<!-- boucle de balayage de la liste et -->  
<div *\*ngFor*="let ***artiste*** of artistes">  
 <!-- envoi au composant enfant (artiste-fiche) -->  
 <!-- et recuperation de l'evenement envoye par l'enfant (artiste a suprimer) -->  
 <app-artiste-fiche [artiste]="***artiste***" (artisteSup)="artisteSup(***$event***)"></app-artiste-fiche>  
</div>

Artiste-liste.component.ts

// Methode de suppression de la liste d'artistes le l'artiste en argument  
artisteSup(artiste:Artiste) : void {  
 this.artistes = this.artistes.filter((artiste1 : Artiste): boolean => artiste1.nom != artiste.nom);  
}

**Le nouvel artiste est ajouté à la liste des artistes au clic d’un bouton « ajouter » en utilisant les reactive forms Angular.**

Artiste-form.component.html

<!-- formulaire de saisie -->  
<div>  
 Name: <input [formControl]="nomArtiste">  
 <label *\*ngIf*="nomArtiste.invalid" [ngClass] = "'error'"> Le nom requiert au moins 1 carractère </label>  
</div>  
  
<!-- bouton enregistrer la saisie -->  
<button type="button" class="btn" (click)="setArtiste()">enregistrer la saisie</button>  
<p></p>

Artiste-form.component.ts

nomArtiste : FormControl = new FormControl('', [Validators.*required*, Validators.*minLength*(1)]);  
  
// Declaration de l'evenement qui permet d'envoyer l'Artiste du composant enfant (celui-ci) au parent  
@Output() artistePlus : ***EventEmitter***<string> = new ***EventEmitter***<string>();  
  
setArtiste() : void {  
 ***console***.log(this.nomArtiste.value)  
 this.artistePlus.emit(this.nomArtiste.value);  
}

artiste-liste.component.html

<!-- affichage du formulaire pour rajouter un artiste -->  
<!-- texte affiche suivant condition (true/false) -->  
<div *\*ngIf*="conditionAjouterArtiste">  
 <app-artiste-form (artistePlus)="ArtistePlus(***$event***)"></app-artiste-form>  
</div>

artiste-liste.component.ts

// Methode pour afficher le formulaire quand on veut rajouter un artiste  
ajouterArtiste() : void {  
 this.conditionAjouterArtiste = !this.conditionAjouterArtiste;  
}  
  
// Methode pour ajouter un element a la liste Artistes  
ArtistePlus(artisteNom : string) : void {  
 // Efface le formulaire  
 this.conditionAjouterArtiste = false;  
 // Rajoute l'element artiste nom  
 let artisteNew : Artiste = {nom:artisteNom,urlImage:""} ;  
 this.artistes.push(artisteNew);  
 ***console***.log(this.artistes)  
}