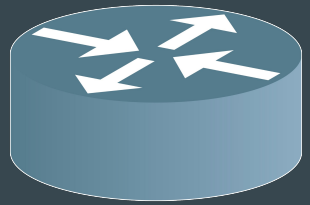
Plan du cours

- Écouteur d’événement dans un composant
- Propriétés personnalisées
- Affichage Conditionnels
- Affichage d’une liste d'éléments
- Configuration du routage
- Création d’un service
- Les formulaires

Configuration du routage



Configuration du routage

La configuration du routage dans Angular permet de définir les différentes routes de l'application et d'associer chaque route à un composant spécifique. Cela permet à l'application de naviguer entre les différentes vues et d'afficher les composants correspondants.

Configuration du routage

Pour configurer le routage dans Angular créez un nouveau fichier dans le dossier app on va le nommer “routing.module.ts”.
dedans on va déclarer l’ensemble des routes de l’application ensuite on va importer le RouterModule en utilisant la méthode for Root pour spécifier le tableau contenant l’ensemble des route puis l’exporter.
ensuit dans les balise a on va utiliser l’attribut routerLink pour préciser la route vers laquelle rediriger.

Configuration du routage

```
import { NgModule } from "@angular/core";
import { RouterModule, Routes } from "@angular/router";
import { HomeComponent } from "./home/home.component";
import { ListeProduitComponent } from "./liste-produit/liste-produit.component";
const routes: Routes = [
    { path: '', component: HomeComponent },
    { path: 'liste', component: ListeProduitComponent }
]
@NgModule({
    imports: [
        RouterModule.forRoot(routes)
    ],
    exports: [
        RouterModule
    ]
})
export class RoutingModule{}
```

Configuration du routage

```
<div class="nav">
    <a routerLink="">Home</a>
    <a routerLink="liste">Liste produit</a>
    <a routerLink="new">Nouveau produit</a>
</div>
```

Configuration du routage

Exercice: toujours dans l'application movie créez un composant Home et un composant NewMovie. configurer le routage de sorte que si on click sur "Movie App" que le composant Home soit affiché, si on click sur "Movie List" que le composant ListeMovie soit affiché et si on click sur "New Movie" ce soit le composant NewMovie qui soit affiché.

Configuration du routage (home)

Configuration du routage (home)

Movie App | Movie List | New Movie

Liste des films

Oppenheimer

Lorem ipsum, dolor sit amet consectetur adipisicing elit. Sapiente ut evenie...

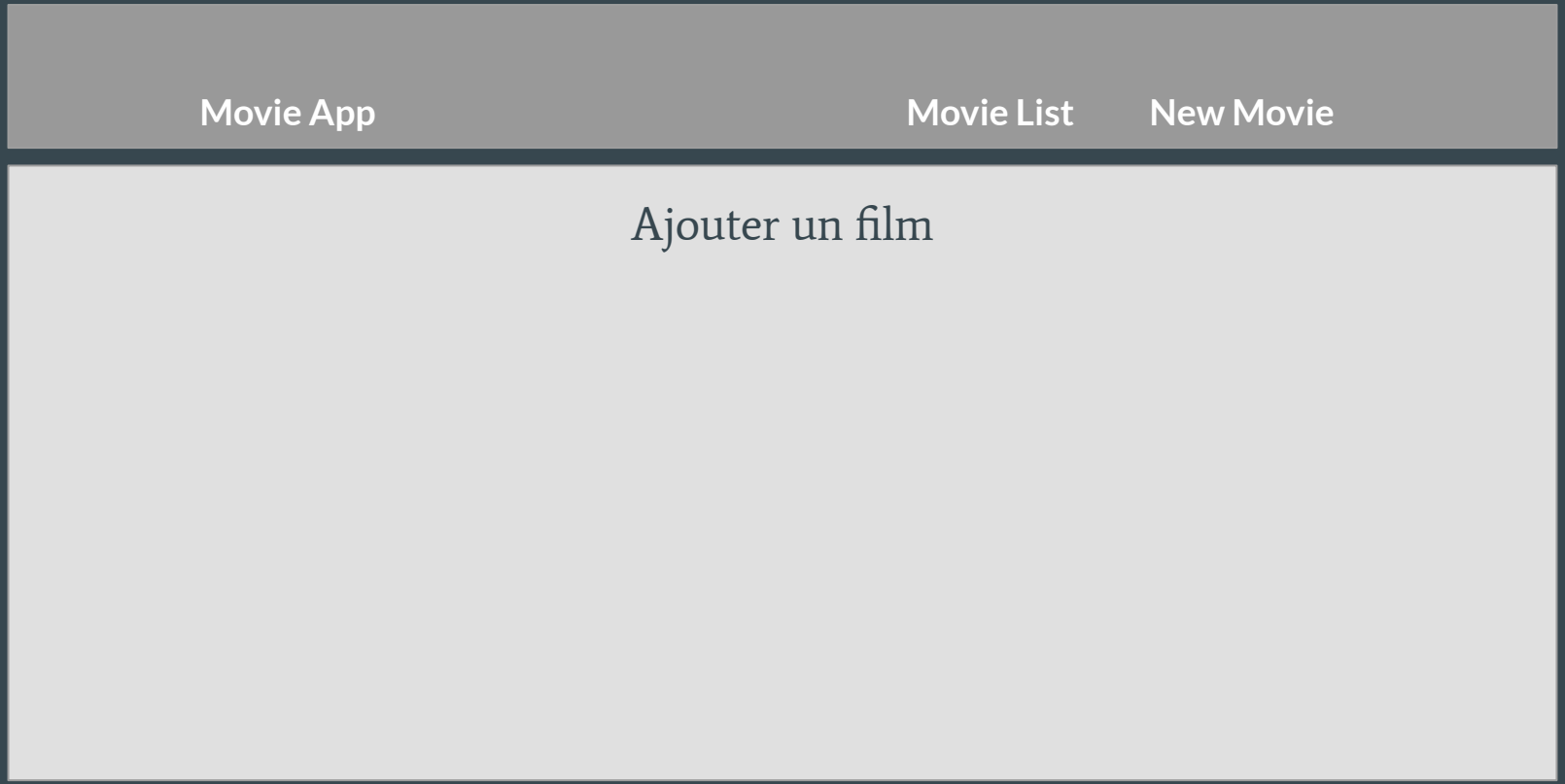
27 likes

Chapitre 2

Lorem ipsum, dolor sit amet consectetur adipisicing elit. Sapiente ut evenie...

27 likes

Configuration du routage (NewMovie)



Création d’un service

Création d’un service

Dans Angular, les services sont des classes qui fournissent des fonctionnalités spécifiques et peuvent être injectés dans d'autres composants, services ou modules de l'application. Les services sont utilisés pour centraliser la logique métier, la communication avec les API, la gestion des données, etc. Ils permettent de rendre le code plus modulaire, réutilisable et facilement testable.
pour créer un service il suffit de créer un fichier dans le dossier app et d’y mettre le code correspondant à vos besoins.

Création d'un service

```
import { Injectable } from '@angular/core';
import { Produit } from './models/produit.model';

@Injectable({
 providedIn: 'root'
})
export class ProduitService {
 produits!: Produit[];

 constructor() {
   this.initProduit();
 }
}
```

Création d’un service

Vous pouvez injecter le service dans d'autres composants, services ou modules en utilisant l'injection de dépendances d'Angular.

Pour injecter le service, importez-le dans le composant ou le service où vous souhaitez l'utiliser, puis déclarez-le dans le constructeur.

Création d'un service

```
import { Component } from '@angular/core';
import { Produit } from '../models/produit.model';
import { ProduitService } from '../produit.service';
@Component({
 selector: 'app-liste-produit',
 templateUrl: './liste-produit.component.html',
 styleUrls: ['./liste-produit.component.css']
})
export class ListeProduitComponent {
 constructor(private produitService: ProduitService){
   this.produits = produitService.produits;
 }
 produits!: Produit[];
}
```

importation du service

Injection du service

Création d’un service

Exercice : créez un service nommé “movieService” modifiez le code du fichier “Movie List” de sorte qu’il utilise le service “movieService” afin d’initialiser la liste des movies.

importez le service, injectez le dans le constructeur puis utilisez le pour initialiser la liste des movies.

Les formulaires

Les formulaires

Pour créer un composant Angular qui affiche un formulaire avec des champs vous pouvez suivre les étapes suivantes :

1. généré le composant : ng generate component FormComponent
2. Définir le formulaire
3. Définissez les propriétés du composant
4. Importez le module FormsModule dans app.module.ts

Les formulaires

```
<div class="new">
    <form (ngSubmit)="submitForm()">
        <label for="title">Titre</label> <br>
        <input type="text" id="title" name="title" [(ngModel)]="title" required> <br>
        <label for="description">Description</label> <br>
        <textarea id="description" name="description" [(ngModel)]="description"
required></textarea> <br>
        <label for="url">URL</label> <br>
        <input type="text" id="url" name="url" [(ngModel)]="url" required> <br>
        <button type="submit">Envoyer</button>
    </form>
</div>
```

Les formulaires

```
import { Component } from '@angular/core';
import { Produit } from '../models/produit.model';
import { ProduitService } from '../produit.service';
@Component({
 selector: 'app-new',
 templateUrl: './new.component.html',
 styleUrls: ['./new.component.css']
})
export class NewComponent {
 constructor(private produitService: ProduitService){}
 title!: string;
 description!: string;
 url!: string;
 ...
}
```

Les formulaires

```
import { FormsModule } from '@angular/forms';
...
@NgModule({
 declarations: [
   ...
   NewComponent
 ],
 imports: [
   FormsModule
 ],
 providers: [],
 bootstrap: [AppComponent]
})
export class AppModule { }
```

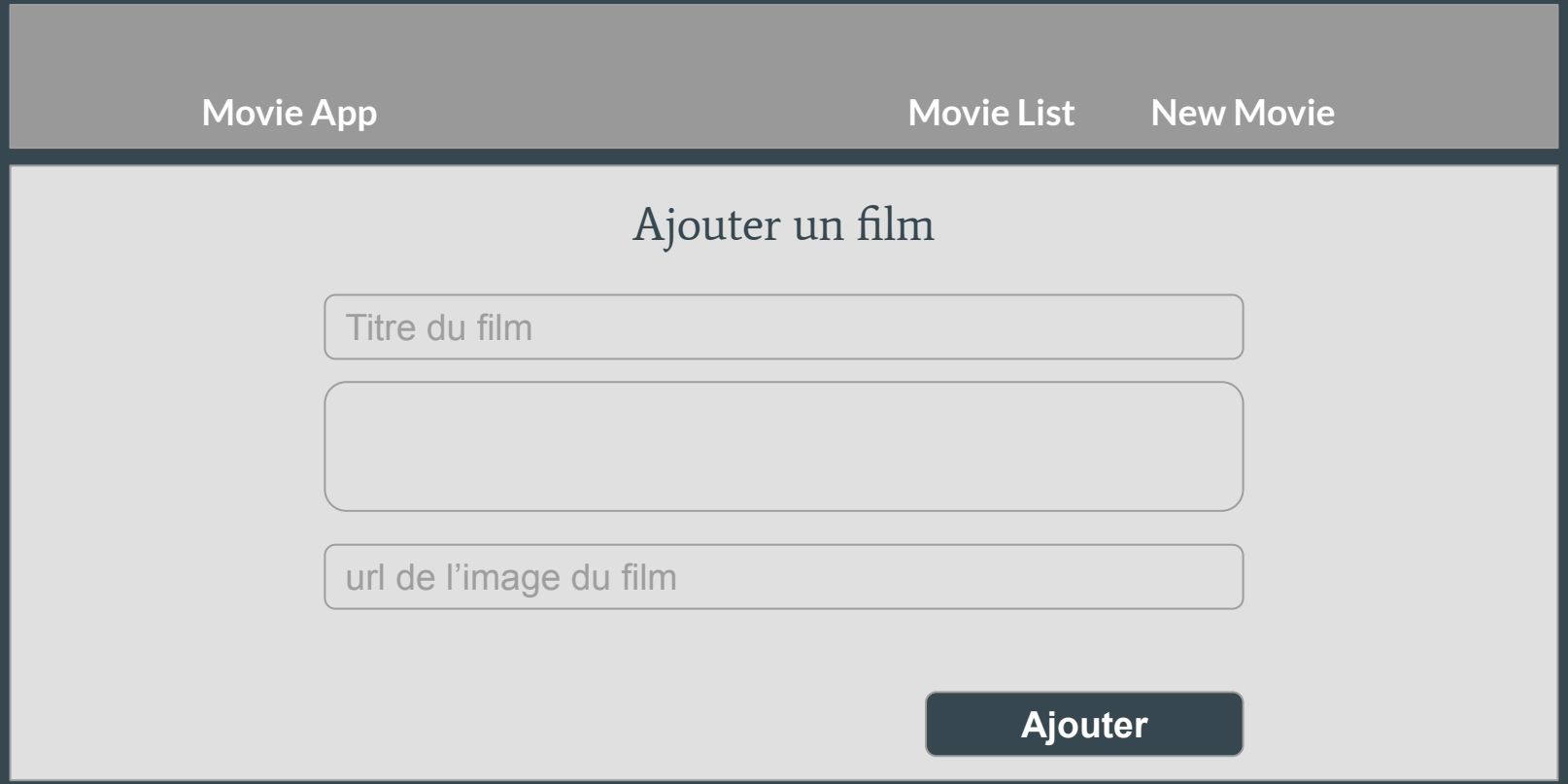
Les formulaires

```
import { Component } from '@angular/core';
import { Produit } from '../models/produit.model';
import { ProduitService } from '../produit.service';
...
export class NewComponent {
 submitForm() {
   this.produit = new Produit(
     this.title,
     this.description,
     this.url,
     0,
     false,
   );
   this.produitService.addProduct(this.produit);
 }
}
```

Création du formulaire

Exercice : avec le composant NewMovie créé précédemment écrire le code du composant de de sorte à avoir un visuel identique au slide suivant mais aussi d'implémenter la logique de sorte que si on click sur le bouton Ajouter que ça ajoute un nouveau film dans le tableau movies qui se trouve dans le service.

Configuration du routage (NewMovie)



Styles inlines & styles dynamiques

Styles inlines & styles dynamiques

Dans Angular, vous pouvez utiliser les directives ngClass et ngStyle pour appliquer des styles de manière dynamique à des éléments.

exemple:

```
<h1 [ngStyle]="{'background-color': 'red'}">{{produit.titre}}</h1>

<h1 [ngStyle]="{color: couleur}">{{produit.titre}}</h1>

<h1 [ngClass]="{maclass: produit.click}">{{produit.titre}}</h1>
```

stylisation du bouton favoris

Exercice :

modifier le code du composant movie de manière à ce que si on clique sur le coeur le bouton devient rose et si on re-clique dessus il reprend sa couleur normale (comme les likes sur youtube)

Les Pipes

Les Pipes

Dans Angular, les pipes sont des fonctionnalités qui permettent de transformer et de formater les données affichées dans les templates. Les pipes sont utilisés pour effectuer des opérations telles que le formatage de dates, la conversion de chaînes de caractères, etc.

Les Pipes

`{{ valeur | uppercase }}` : Ce pipe transforme le texte en majuscules.

Exemple : `{{ 'hello' | uppercase }}` affichera "HELLO".

`{{ valeur | lowercase }}` : Ce pipe transforme le texte en minuscules.

Exemple : `{{ 'WORLD' | lowercase }}` affichera "world".

`{{ valeur | titlecase }}` : Ce pipe met la premiere lettre de chaque mot en majuscule.

Exemple : `{{ 'salut le monde' | lowercase }}` affichera "Salut Le Monde".

`{{ valeur | currency }}` : Ce pipe formate un nombre en une représentation monétaire.

Exemple : `{{ 123.45 | currency:'EUR' }}` affichera "€123.45".

`{{ valeur | date }}` : Ce pipe formate une date.

Exemple : `{{ myDate | date:'short' }}` affichera une représentation courte de la date.

Passage de parametre a une route

Passage de parametre à une route

Avec angular vous pouvez passer des paramètres dans une route. Le but du passage de paramètres à une route est de permettre une communication entre les composants qui interagissent au sein de l'application. Lorsque vous naviguez vers une route avec des paramètres, vous pouvez passer des données telles que des identifiants.

Pour faire passer des paramètres a un route suivre les étapes qui suivent :

Passage de parametre à une route

- Créer le composant vers lequel rediriger si il n'existe pas déjà
- Définir la route dans le module de routage

```
const routes: Routes = [
   { path: 'produit/:id', component: DetailProduitComponent},
   ...
]
```

- Importer le module Router, l'injecter dans le constructeur, créer le lien

```
<h1 (click)="viewDetails()">{{produit.titre}}</h1>
import { Router } from '@angular/router';
export class ProduitComponent {
      constructor(private router: Router){}
      viewDetails(){
         this.router.navigateByUrl(`produit/${this.produit.id}`);
      }
}
```

Passage de parametre à une route

- Importer le module ActivatedRoute et de l'injecter dans le constructeur

```
import { ActivatedRoute } from '@angular/router';
export class DetailProduitComponent {
    constructor(private route: ActivatedRoute){}
}
```

- Récupérer le paramètre dans le composant vers lequel on a redirigé l'internaute

```
idProduit = +this.route.snapshot.params['id'];
```

Liens utils

- https://angular.io/
- https://angular.io/guide/styleguide