

Plan du cours



- Objectifs
 - Angular
 - Installation d'Angular CLI
 - Création d'un nouveau projet
 - Structure d'un projet Angular
 - Lancer le serveur
 - Compréhension, création et affichage d'un composant
 - Propriétés de composant et databinding
-

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

Compréhension, création & affichage d'un composant



Compréhension, création & affichage d'un composant

Qu'est-ce qu'un composant dans Angular ?

- Les composants sont les blocs de construction de l'interface utilisateur dans Angular
- Ils regroupent la logique, la structure et les styles d'une partie de l'application

Compréhension, création & affichage d'un composant

Architecture d'un composant Angular

- Template : représente la structure HTML du composant
- Classe : contient la logique du composant en TypeScript
- Styles : définissent l'apparence du composant

Compréhension, création & affichage d'un composant

Étapes pour créer un composant Angular

1. Utilisation de la commande Angular CLI pour générer un nouveau composant
2. Exploration des fichiers générés : fichier de template, fichier de classe, fichiers de styles
3. Personnalisation du template, ajout de contenu HTML
4. Implémentation de la logique du composant dans la classe associée

Compréhension, création & affichage d'un composant

Étapes pour créer un composant Angular la CLI d'angular
tapez la commande :

```
ng generate component nom-du-composant
```

ou la version courte:

```
ng g c nom-du-composant
```

Compréhension, création & affichage d'un composant

```
diokel@diokel-T440s:~/cours/Angular/code/myapp$ ng g c product
CREATE src/app/product/product.component.css (0 bytes)
CREATE src/app/product/product.component.html (22 bytes)
CREATE src/app/product/product.component.ts (206 bytes)
UPDATE src/app/app.module.ts (487 bytes)
diokel@diokel-T440s:~/cours/Angular/code/myapp$
```


Compréhension, création & affichage d'un composant

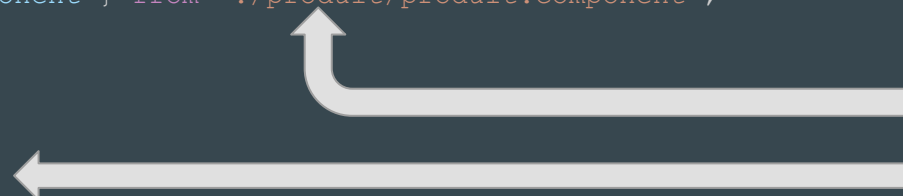
```
import { Component, OnInit, Input } from '@angular/core';
```

```
@Component({  
  selector: 'app-produit',  
  templateUrl: './produit.component.html',  
  styleUrls: ['./produit.component.css']  
})  
export class ProduitComponent {  
}
```

```
<p>Composant Produit</p>
```

Compréhension, création & affichage d'un composant

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import { ProduitComponent } from './produit/produit.component';
@NgModule({
  declarations: [
    AppComponent,
    ProduitComponent,
  ],
  imports: [
    BrowserModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```



importation et déclaration
dans app.module.ts

Compréhension, création & affichage d'un composant

une fois le composant créé on pourra l'afficher grâce à son nom de sélecteur défini dans le fichier 'ts'

```
@Component({  
  selector: 'app-mon-composant',  
  templateUrl: './mon-composant.component.html',  
  styleUrls: ['./mon-composant.component.css']  
})
```



nom de sélecteur du composant

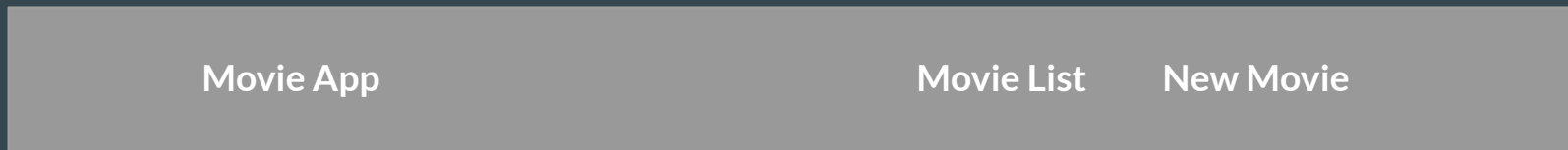
on va l'injecter dans le fichier app.component.html

```
<app-mon-composant></app-mon-composant>
```

cela a pour effet d'englober le composant en question dans le composant principale.

Compréhension, création & affichage d'un composant

Exercice : créez un composant que vous allez nommer “navBar” et qui correspondrait à un menu de navigation il doit être identique à la figure suivante :



Afficher le composant

Propriétés de composant et databinding



Propriétés de composant et databinding

Les propriétés de composant et le databinding sont des concepts clés dans Angular qui permettent de communiquer et de transmettre des données entre les composants et les templates.

Propriétés de composant et databinding

Propriétés de composant :

- Les propriétés de composant sont des variables déclarées dans la classe du composant.
- Elles représentent l'état et les données spécifiques à un composant.
- Les propriétés peuvent être définies avec des types de données spécifiques et initialisées avec des valeurs par défaut.
- Les propriétés sont utilisées pour stocker et manipuler des données dans le composant.

Propriétés de composant et databinding

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

@Component ({
  selector: 'app-mon-composant',
  templateUrl: './mon-composant.component.html' ,
  styleUrls: ['./mon-composant.component.css' ]
})

export class ProduitComponent implements OnInit{
  titre!: string;
  description!: string;
  img!: string;
  ngOnInit(): void {
    this.titre = "Mon composant";
    this.description = "blablater ici...";
    this.img = "https://t3.ftcdn.net/jpg/05/76/35/90/240_F_576359075_icHqsmA0QaGO8uqazn6djjCKZNPhtUon.jpg" ;
  }
}
```


Propriétés de composant et databinding

Le databinding permet de lier les données entre les propriétés du composant et les éléments du template HTML. Il existe trois types de databinding dans Angular :

1. Interpolation : Utilisé pour afficher des valeurs dans le template en utilisant la syntaxe `{{ expression }}`. Par exemple : `{{ title }}`.
2. Property binding : Utilisé pour lier une propriété d'un élément HTML à une propriété du composant à l'aide de la syntaxe `[prop]="expression"`. Par exemple : `<input [value]="name">`.
3. Event binding : Utilisé pour réagir aux événements déclenchés par l'utilisateur en liant une méthode du composant à un événement HTML à l'aide de la syntaxe `(event)="expression"`. Par exemple : `<button (click)="mafonction()">`.

Propriétés de composant et databinding

```
<div>  
  <h1>{{titre}}</h1>  
  <img [src]="urlImage" alt="image">  
  <p>{{description}}</p>  
</div>
```

Propriétés de composant et databinding

Exercice : créez un composant movie avec les propriétés suivantes : title, description, likes, urlImage il doit être identique à la figure ci-dessous.



Oppenheimer

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

27 likes

Écouteur d'événement dans un composant



Écouteur d'événement dans un composant

Dans Angular, vous pouvez ajouter un écouteur d'événement à un composant en utilisant la syntaxe d'event binding. Cela vous permet de réagir aux événements déclenchés par l'utilisateur ou d'autres composants et d'exécuter une logique spécifique en réponse à ces événements.

Voici comment ajouter un écouteur d'événement dans un composant Angular :

Écouteur d'événement dans un composant

Dans le template HTML du composant, ajoutez l'écouteur d'événement à l'élément HTML approprié en utilisant la syntaxe `(event)="handleEvent($event)"` Par exemple, pour écouter un événement de clic sur un bouton.

```
<div>
  <h1>{{titre}}</h1>
  <img [src]="urlImage" alt="image">
  <p>{{description}}</p>
  <button (click)="handleClick()">Cliquez ici</button>
</div>
```

Écouteur d'événement dans un composant

Dans la classe du composant, définissez la méthode `handleEvent()` correspondante pour gérer l'événement. Cette méthode sera appelée chaque fois que l'événement est déclenché.

```
handleClick() {  
    console.log('Le bouton a été cliqué !');  
}
```

Écouteur d'événement dans un composant

Exercice : Complétez le composant précédent en y ajoutant un bouton cliquable figure suivante.



Oppenheimer

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



27 likes

Le click sur le bouton doit
incrémenter le nombre de
like de 1.

Propriétés personnalisées



Propriétés personnalisées

En Angular, vous pouvez définir des propriétés personnalisées (custom properties) pour les composants en utilisant les décorateurs.

Ces décorateurs permettent de créer des liaisons de données personnalisées pour les composants, ce qui facilite la communication entre les composants parents et enfants.

Propriétés personnalisées

```
export class Produit {  
  titre: string;  
  description: string;  
  urlImage: string;  
  nbclick: number;  
  click: boolean;  
  
  constructor(titre: string, description: string, urlImage: string, nbclick: number, click: boolean){  
    this.titre = "Mon composant";  
    this.description = "blablater ici...";  
    this.urlImage  
    "https://t3.ftcdn.net/jpg/05/76/35/90/240_F_576359075_icHqsmA0QaGO8uqazn6djJCKZNPhtUon.jpg"  
    this.nbclick = 1;  
    this.click = false;  
  }  
}
```

Propriétés personnalisées

```
export class Produit {  
  constructor (  
    public titre: string,  
    public description: string,  
    public urlImage: string,  
    public nbclick: number,  
    public click: boolean  
  ) {}  
}
```

Propriétés personnalisées

```
import { Component, OnInit, Input } from '@angular/core';
import { Produit } from '../models/produit.model';
@Component({
  selector: 'app-produit',
  templateUrl: './produit.component.html',
  styleUrls: ['./produit.component.css']
})
export class ProduitComponent implements OnInit {
  @Input() produit!: Produit;
  ngOnInit(): void {}
  handleClick() {
    if(this.produit.click) {
      this.produit.nbclick--;
      this.produit.click = false;
    }else{
      this.produit.nbclick++;
      this.produit.click = true;
    }
  }
}
```

dans la classe du
composant produit

Propriétés personnalisées

```
<div>  
  <h1>{{produit.titre}}</h1>  
  <p>{{produit.description}}</p>  
  <img [src]="produit.urlImage"/>  
  <p>{{produit.nbclick}}</p>  
  <button (click)="handleClick()">click</button>  
</div>
```

dans le template du
composant produit

Propriétés personnalisées

```
import { Component, OnDestroy, OnInit } from '@angular/core';
import { Produit } from '../models/produit.model';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent implements OnInit {
  title = 'Composant Principale';
  prod!: Produit;

  ngOnInit(): void {
    this.prod = new Produit(
      "Mon composant",
      "blablater ici...",
      "https://t3.ftcdn.net/jpg/05/76/35/90/240_F_576359075_icHqsmA0QaGO8uqazn6djjCKZNPhtUon.jpg",
      1,
      false
    );
  }
}
```

dans la classe du
composant principale
app

Propriétés personnalisées

...

```
<app-produit [produit]="prod"></app-produit>
```

...

dans le template du
composant produit

Propriétés personnalisées

Exercice : Créer une propriété personnalisée “Movie” puis créer de nouveaux films de sorte à avoir l’affichage suivant :

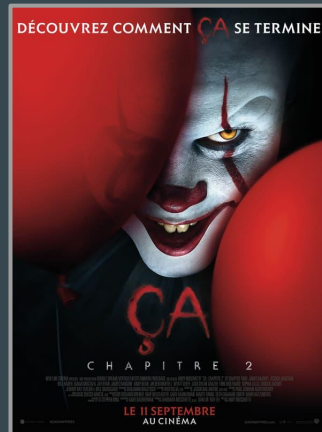


Oppenheimer

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



27 likes



Chapitre 2

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



54 likes

Affichage Conditionnels



Affichage Conditionnels

En Angular, vous pouvez réaliser un affichage conditionnel en utilisant les directives `ngIf` et `ngSwitch`. Ces directives vous permettent de contrôler dynamiquement la présence ou l'absence d'éléments dans le template en fonction d'une condition.

Affichage Conditionnels

```
<p *ngIf="produit.price">{{produit.price}} €</p>
```

```
<div [ngSwitch]="role">  
  <p *ngSwitchCase="'admin'">Bienvenue, administrateur !</p>  
  <p *ngSwitchCase="'user'">Bienvenue, utilisateur !</p>  
  <p *ngSwitchDefault>Connectez-vous pour afficher un message de  
bienvenue.</p>  
</div>
```

Affichage d'une liste d'éléments



Affichage d'une liste d'éléments

La directive **ngFor** est une directive intégrée d'Angular qui permet d'itérer sur une liste ou un tableau d'éléments dans le template et de générer dynamiquement des éléments répétés.

Affichage d'une liste d'éléments

```
export class AppComponent implements OnInit{
  produits!: Produit[];
  ngOnInit(): void {
    this.produits = [
      new Produit(
        "Produit 1",
        "blablater ici...",
        "https://t3.ftcdn.net/jpg/05/76/35/90/240_F_576359075_icHqsmA0QaGO8uqazn6djCKZNPhtUon.jpg"
      ),
      new Produit(
        "Produit 2",
        "blablater ici...",
        "https://cdn.pixabay.com/photo/2023/07/02/18/49/cup-8102791_960_720.jpg"
      ),
      new Produit(
        "Produit 2",
        "blablater ici...",
        "https://cdn.pixabay.com/photo/2023/06/29/10/33/lion-8096155_960_720.png"
      ),
    ];
  }
}
```

dans la classe du
composant principale
app

Affichage d'une liste d'éléments

```
<app-produit *ngFor="let produit of produits" [produit]="produit"></app-produit>
```

La directive ngFor se comporte comme une boucle dans ce cas ci elle va utiliser le tableau produits et itérer sur chacun de ses éléments et pour chaque élément va afficher un composant produit avec les valeurs de l'élément en question.

Propriétés personnalisées

Exercice : En utilisant la directive ngFor au lieu de répéter le composant movie dans le composant app modifier le code de votre application de sorte à avoir le même affichage que précédemment.

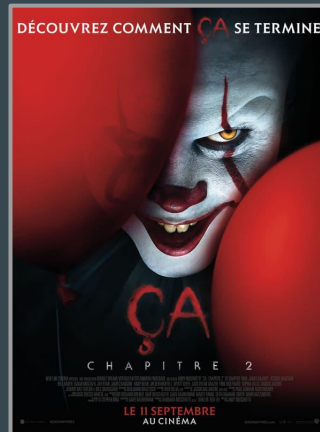


Oppenheimer

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...



54 likes

Liens utiles

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