CHAPITRE 08 : INTRODUCTION A ANGULAR

1. Introduction

Angular est un framework open-source développé par Google qui permet de construire des applications web dynamiques et interactives. Il est basé sur TypeScript et suit le principe de programmation orientée composant. Angular offre un ensemble robuste d'outils pour le développement d'applications web modernes. Caractéristiques des Composants Angular:

a. Architecture basée sur les composants : Angular encourage le développement d'applications modulaires en utilisant des composants réutilisables. Chaque composant gère une partie de l'interface utilisateur et de la logique de l'application.

b. TypeScript : Angular est écrit en TypeScript, un sur-ensemble de JavaScript qui ajoute des fonctionnalités telles que le typage statique, ce qui améliore la maintenance du code et permet de détecter plus facilement les erreurs.

c. Binding de données : Angular offre une liaison de données bidirectionnelle, ce qui signifie que les modifications effectuées dans l'interface utilisateur sont automatiquement reflétées dans le modèle et **vice versa.**

d. Directives : Les directives sont utilisées pour étendre le HTML avec de nouvelles fonctionnalités personnalisées, ce qui permet de créer des applications interactives et dynamiques.

e. Services et injection de dépendances : Angular encourage l'utilisation de services pour partager la logique métier entre différents composants. L'injection de dépendances est utilisée pour fournir des instances de services aux composants qui en ont besoin.

f. Routing : Angular fournit un système de routage pour créer des applications à navigation multiple-pages, permettant de définir des routes et d'afficher différents composants en fonction de l'URL.

g. Pipes : Les pipes sont des outils de transformation de données pour formater et afficher des données dans les templates de manière propre et efficace.

h. Observables et RxJS : Angular utilise la programmation réactive avec RxJS pour gérer les flux de données asynchrones, ce qui facilite le traitement des événements et des requêtes HTTP.

i. Testing : Angular est conçu pour être testable et fournit des outils de test intégrés pour réaliser des tests unitaires et de bout en bout.

2- Anatomies d'une application Angular

L'anatomie d'une application Angular comprend plusieurs éléments clés qui travaillent ensemble pour créer une application web robuste et maintenable. les principaux composants de base d'une application Angular :

- **Modules :**

- Les modules sont des conteneurs logiques qui regroupent des éléments de code liés tels que des composants, des directives, des pipes et des services. Chaque application Angular contient au moins un module racine appelé AppModule.

- **Composants :**

- Les composants sont les éléments de base de l'interface utilisateur dans Angular. Ils encapsulent la logique et le comportement de l'interface utilisateur d'une partie spécifique de l'application. Chaque composant est associé à un template HTML et peut avoir une logique définie dans une classe TypeScript.

- **Services :**

- Les services sont des classes qui encapsulent la logique métier et les fonctionnalités réutilisables. Ils sont utilisés pour effectuer des opérations telles que les appels HTTP, la gestion de l'état de l'application et la logique métier.

- **Directives :**

- Les directives sont des instructions dans le DOM qui modifient le comportement ou l'apparence d'un élément HTML. Angular propose des directives intégrées telles que ngIf, ngFor et ngClass, ainsi que la possibilité de créer des directives personnalisées.

- **Pipes** :

- Les pipes sont des outils permettant de formater et de transformer des données dans les templates Angular. Ils sont utilisés pour afficher les données de manière conditionnelle, filtrer les données, formater des chaînes et bien plus encore

- **Routing :**

- Le système de routage d'Angular permet de naviguer entre les différentes vues de l'application en fonction des URL. Il permet de définir des routes et d'associer des composants à ces routes.

- **Observables et RxJS :**

- Angular utilise RxJS, une bibliothèque JavaScript pour la programmation réactive, afin de gérer les flux de données asynchrones. Les observables sont largement utilisés pour gérer les événements et les appels HTTP.

- **Templates HTML :**

- Les templates HTML définissent la structure visuelle de l'interface utilisateur de l'application. Angular utilise la liaison de données pour synchroniser les données du composant avec le template HTML.

- **Environnements :**

- Les fichiers d'environnement permettent de gérer les configurations spécifiques à l'environnement de développement, de production ou d'autres environnements.

3- Affichage d'attribut ou de méthodes

En Angular, pour afficher des attributs ou appeler des méthodes d'un composant dans le template HTML, on utilise la liaison de données et les interpolations. Voici comment vous pouvez afficher des attributs et appeler des méthodes dans un template Angular :

❖ Affichage d'attributs :

Pour afficher des attributs d'un composant dans le template HTML, vous pouvez utiliser des interpolations {{ }}. Par exemple, si vous avez un attribut nom défini dans votre composant, vous pouvez l'afficher comme suit :

```
<p>Nom: {{ nom }}</p>
```

❖ Appel de méthodes :

Pour appeler des méthodes d'un composant dans le template HTML, vous pouvez les appeler directement en utilisant les interpolations {{ }}. Par exemple, si vous avez une méthode getAge() définie dans votre composant, vous pouvez l'appeler comme suit :

```
<p>Âge: {{ getAge() }}</p>
```

Exemple complet :

Voici un exemple complet pour afficher un attribut et appeler une méthode dans un composant Angular:

```
// composant.ts

export class MonComposant {
  nom: string = 'Alice';
  age: number = 30;
  getAge(): number {
    return this.age;
  }
}

<!-- template.html -->

<div>
  <p>Nom: {{ nom }}</p>
  <p>Âge: {{ getAge() }}</p>
</div>
```

Dans cet exemple, le nom est directement affiché et la méthode getAge() est appelée pour afficher l'âge dans le template HTML.

En utilisant la liaison de données et les interpolations, Angular facilite l'affichage des données et l'appel des méthodes d'un composant dans le template, offrant ainsi une interaction dynamique entre le composant et l'interface utilisateur. Si vous avez besoin de plus d'informations ou d'exemples spécifiques, n'hésitez pas à demander.

4- Les directives Angular

Ah, les directives Angular ! Elles sont vraiment intéressantes et puissantes pour étendre les fonctionnalités du HTML dans une application Angular. Les directives permettent de manipuler le DOM et d'ajouter des comportements dynamiques aux éléments HTML. Voici une petite plongée dans le monde des directives Angular :

Types de directives Angular :

1. **Directives Structurelles** :

- Les directives structurelles comme `*ngIf`, `*ngFor`, et `*ngSwitch` permettent de modifier la structure du DOM en fonction des conditions définies dans le composant.

Exemple d'utilisation de *ngIf pour afficher un élément conditionnellement :

```html
<div *ngIf="isLoggedIn">Contenu visible pour les utilisateurs connectés.</div>
```

2. **Directives d'Attribut** :

- Les directives d'attribut modifient l'apparence ou le comportement des éléments DOM. Par exemple, la directive `ngClass` permet de dynamiquement ajouter ou supprimer des classes CSS.

Exemple d'utilisation de ngClass pour appliquer des classes conditionnellement :

```html
<div [ngClass]="{ 'classe1': condition1, 'classe2': condition2 }">Élément avec classes conditionnelles.</div>
```

3. **Directives de Liaison d'Événement** :

- Les directives de liaison d'événement permettent de réagir aux événements DOM tels que les clics, les survols, etc. Par exemple, `ngClick` pour gérer les clics.

Exemple d'utilisation de ngClick pour gérer un événement de clic :

```html <button (click)="onClick()">Cliquez ici</button>

4. **Création de Directives Personnalisées** :

- Angular permet également de créer des directives personnalisées pour répondre à des besoins spécifiques de l'application.

Exemple de création d'une directive personnalisée pour mettre en surbrillance un élément :

```typescript
@Directive({
  selector: '[appMiseEnSurbrillance]'
})
export class MiseEnSurbrillanceDirective {
  constructor(private el: ElementRef) {
  el.nativeElement.style.backgroundColor = 'yellow';
  }
                    }
                    ```

Utilisation de la directive dans le template :

```html
<p appMiseEnSurbrillance>Texte mis en surbrillance</p>

Les directives jouent un rôle crucial dans le développement d'applications Angular en permettant de rendre le code plus réutilisable, modulaire et maintenable. Elles

offrent une manière élégante d'ajouter de la logique personnalisée aux éléments HTML, améliorant ainsi l'interactivité et la flexibilité des applications. Si vous avez besoin de plus de détails ou d'exemples spécifiques sur les directives Angular, n'hésitez pas à demander !.

## 5- Comment installer t configurer angular

Installer et configurer Angular est assez simple une fois que vous avez Node.js et npm installés sur votre système. Voici une procédure générale pour installer Angular :

Étapes pour installer Angular :

- Installer Node.js : Angular nécessite Node.js, donc si vous ne l'avez pas déjà, vous pouvez le télécharger et l'installer depuis nodejs.org (https://nodejs.org/).

- Vérifier l'installation de Node.js et npm : Après l'installation, vous pouvez vérifier si Node.js et npm sont installés correctement en exécutant les commandes suivantes dans votre terminal :

  ```bash
  node -v
  npm -v
  ```

- . Installer Angular CLI : Angular CLI est un outil de ligne de commande officiel pour travailler avec des projets Angular. Vous pouvez l'installer en utilisant npm :

  ```bash
  npm install -g @angular/cli
  ```

- Créer un nouveau projet Angular : Une fois Angular CLI installé, vous pouvez créer un nouveau projet Angular en exécutant la commande suivante :

  ```bash
  ng new nom-du-projet
  ```

- Lancer l'application Angular : Naviguez vers le répertoire du projet créé et lancez votre application en utilisant la commande :

  ```bash
  cd nom-du-projet
  ng serve
  ```

- Accéder à l'application : Une fois que la compilation est terminée, vous pouvez accéder à votre application Angular dans un navigateur en ouvrant http://localhost:4200/.### Configuration d'Angular :Une fois que vous avez installé Angular, vous pouvez personnaliser la configuration de votre projet en modifiant divers fichiers de configuration. Voici quelques points clés à considérer pour la configuration :

- Environnements : Utilisez les fichiers d'environnement pour gérer les paramètres spécifiques à chaque environnement, tels que les URL d'API ou les clés d'authentification.

- Angular.json : Ce fichier de configuration contient des informations sur la structure de votre projet Angular, les ressources utilisées, les options de compilation, etc.

- tsconfig.json : Le fichier de configuration TypeScript définit la configuration du compilateur TypeScript pour votre projet Angular.

- angular-cli.json / angular.json : Ces fichiers contiennent diverses configurations pour Angular CLI et votre projet, telles que le chemin des fichiers sources, les règles de linting, etc.En ajustant ces fichiers de configuration selon vos besoins, vous pouvez personnaliser votre projet Angular pour répondre aux exigences spécifiques de votre application.

6- différents composants angular

Dans Angular, les composants jouent un rôle essentiel dans la construction d'une application web. Ils sont responsables de la logique et de l'interface utilisateur d'une partie spécifique de l'application. Voici un aperçu des principaux types de composants que l'on peut rencontrer dans une application Angular :

- **Types de Composants Angular :**

1. Composants Racine (App Module) :

- Le composant racine, souvent nommé AppModule, est le point d'entrée de l'application Angular. Il encapsule tous les autres composants, directives, services et modules de l'application.

2. Composants Applicatifs :

- Les composants applicatifs sont des composants qui représentent des fonctionnalités spécifiques de l'application. Chaque fonctionnalité peut être encapsulée dans un composant distinct pour une meilleure organisation.

3. Composants de Vue :

- Ces composants sont responsables de la présentation visuelle de l'application. Ils affichent des données et interagissent avec l'utilisateur. Exemple : un composant affichant une liste de produits.

4. Composants de Container :

- Les composants de conteneur agissent comme des éléments qui encapsulent d'autres composants. Ils sont utilisés pour structurer la disposition de l'application et pour gérer la communication entre différents composants enfants.

5. Composants Partagés :

- Les composants partagés sont des composants réutilisables à travers différentes parties de l'application. Ils contiennent des fonctionnalités communes telles que des barres de navigation, des boutons ou des cartes.

6. Composants de Routage (Route Components) :

- Ces composants sont utilisés pour définir les différentes vues et les routes de navigation de l'application. Chaque route peut être associée à un composant spécifique à afficher.

7. Composants de Service :

- Bien qu'ils ne soient pas des composants visuels, les composants de service sont des classes qui contiennent la logique métier partagée entre différents composants. Ils sont injectés à travers l'application pour fournir des fonctionnalités communes.

8. Composants Dynamiques :

- Les composants dynamiques sont des composants chargés dynamiquement à l'exécution. Ils permettent de créer des interfaces utilisateurs flexibles et d'intégrer des éléments dynamiques dans l'application.

En structurant votre application Angular avec différents types de composants, vous pouvez séparer la logique applicative, améliorer la réutilisabilité du code et rendre votre application plus modulaire et maintenable. Chaque type de composant a un rôle précis à jouer dans la construction d'une application web robuste et bien organisée. Si vous souhaitez en savoir plus sur un type de composant spécifique ou si vous avez d'autres questions sur les composants Angular, n'hésitez pas à demander !