Ateliers Angular

TP 2 : Projet et Composants

TP 2 : Structure d'un Projet et Création de Composants

I. Objectif

Analyser la structure d'un projet Angular et apprendre à créer des projets et des composantes.

Objectifs Spécifiques :

1. Maitriser la structure d'un projet Angular et le rôle de chaque fichier.
2. Apprendre à créer des composants Angular et à les intégrer dans une application. Utiliser Angular CLI pour générer et organiser des composants.
3. Utiliser la liaison de données bidirectionnelle avec la directive ngModel.
4. Découvrir l'AppModule.
5. Apprendre l'importance de déclarer les composants dans l'AppModule

Ce TP comporte deux types d'activités :
Ateliers pratiques en salle : Ces exercices seront réalisés collectivement sous la direction de l'enseignant.
Ateliers guidés : Ces exercices sont accompagnés de solutions détaillées pour vous aider dans votre apprentissage. N'hésitez pas à solliciter votre enseignant à chaque étape si vous rencontrez des difficultés.

II. Ateliers

Activité 1 : Exploration de la structure d'un projet Angular

1. Sous VSC, Créez un nouveau projet avec Angular CLI et parcourez les dossiers et fichiers générés.
2. Pour chaque dossier et fichier principal, rédigez une brève description de son rôle. Concentrez-vous sur :
   - src/app……
   - src/assets……
   - src/environments……
   - src/main.ts……..
   - src/index.html……
   - angular.json……
   - package.json……
   - tsconfig.json……

Activité 2 : Génération et intégration de composants avec Angular CLI

1. **Génération d'un nouveau composant** : Utilisez la commande suivante pour générer un composant nommé utilisateur :

```
ng generate component utilisateur
```

2. **Modification du composant** : Dans utilisateur.component.ts, ajoutez une propriété nom initialisée avec votre nom.
    - Dans utilisateur.component.html, affichez la propriété nom à l'aide de la liaison de données interpolation ({{ nom }}).
3. **Intégration du composant dans l'application** : Dans app.component.html, ajoutez la balise de votre nouveau composant :

```
<app-utilisateur></app-utilisateur>
```

Activité 3 : Formulaire avec liaison de données bidirectionnelle

1. **Importation du FormsModule** :
    - Ouvrez app.module.ts.
    - Importez FormsModule depuis @angular/forms :
    - import { FormsModule } from '@angular/forms';
    - Ajoutez FormsModule au tableau des imports de @NgModule.
2. **Création du composant de formulaire** :
    - Générez un composant nommé profil : ng generate component profil
3. **Mise en place du formulaire** :
    - Dans profil.component.ts, déclarez une propriété utilisateur avec des sous-propriétés prenom et age.
    - Dans profil.component.html, créez un formulaire avec :
        - Un champ de saisie pour le prénom lié à utilisateur.prenom avec [(ngModel)].
        - Un champ de saisie pour l'âge lié à utilisateur.age avec [(ngModel)].
    - Affichez en temps réel les valeurs saisies en dessous du formulaire.
4. **Intégration du composant** :
    - Ajoutez <app-profil></app-profil> dans app.component.html.

Activité 4 : Comprendre le rôle du FormsModule

1. **Expérimentation sans FormsModule** :
    - Commentez ou supprimez l'importation de FormsModule dans app.module.ts.
    - Observez les erreurs générées dans l'application liées à l'utilisation de ngModel.
2. **Analyse** :
    - Expliquez pourquoi l'importation de FormsModule est nécessaire pour utiliser ngModel.
    - Décrivez le rôle de AppModule dans la gestion des modules et des dépendances de l'application.
3. **Restauration** :
    - Réintégrez FormsModule dans app.module.ts pour corriger les erreurs.

Activité 5 : Déclaration manuelle de composants

1. **Création d'un composant manuellement** :
    - Dans le dossier src/app/composants, créez un nouveau dossier adresse.
    - À l'intérieur, créez les fichiers suivants :
        - adresse.component.ts

- adresse.component.html
- adresse.component.css

2. **Code du composant** :
   - **adresse.component.ts** :

```
import { Component } from '@angular/core';
@Component({
  selector: 'app-adresse',
  templateUrl: './adresse.component.html',
  styleUrls: ['./adresse.component.css']
})
export class AdresseComponent {
  rue = '123 Rue Exemple';
}
```

   - **adresse.component.html** :

```
<p>Adresse : {{ rue }}</p>
```

3. **Non-déclaration du composant** :
   - N'ajoutez pas AdresseComponent dans les déclarations de app.module.ts.
   - Essayez d'utiliser <app-adresse></app-adresse> dans app.component.html et observez les erreurs.
4. **Déclaration du composant** :
   - Maintenant, ajoutez AdresseComponent au tableau declarations de @NgModule dans app.module.ts.
   - Vérifiez que l'erreur est résolue et que le composant s'affiche correctement.
5. **Conclusion** :
   - Expliquez pourquoi il est nécessaire de déclarer les composants dans AppModule.
   - Décrivez le rôle du tableau declarations dans @NgModule.

III. Atelier guidé

Commençons par créer un projet :

1) Dans Visual Studio Code, et sur le dossier racine de vos projets, créer un nouveau projet :

```
ng new ListeEtudiants --defaults
cd ListeEtudiants
ng serve -o
```

2) On se propose maintenant de modifier le titre du projet. C'est dans le fichier de la classe du composant principal `app.component.ts`.
3) Avec VSC, ouvrez le dossier ListeEtudiants puis le fichier app.component.ts (dans le dossier src/app). La propriété *title* représente le titre de l'application. Modifier la comme suit :

```
export class AppComponent {
  title = MyClass Will be Angular Heroes';
}
```

Ou un titre de votre choix.

4) Maintenant on va afficher cet attribut dans le fichier html. Ouvrez le fichier html (`app.component.html`) et supprimez tout le code HTML de ce fichier. Remplacez-le par la ligne suivante de HTML.

```
<h1>{{title}}</h1>
```

Vérifiez le résultat dans le navigateur.

Rappel : Le double accolades est la syntaxe de liaison d'interpolation d'Angular.

La plupart des applications s'efforcent de donner une apparence cohérente à l'ensemble de l'application. C'est le cas avec Angular.

5) Une feuille de style a été généré dans le dossier src, `styles.css`, vide pour le moment. C'est la feuille du style de tout le projet.

Ouvrez ce fichier et ajoutez le code ci-après.

```
/* Application-wide Styles */
h1 {
    color: #369;
    font-family: Arial, Helvetica, sans-serif;
    font-size: 250%;
  }
  h2, h3 {
    color: #444;
    font-family: Arial, Helvetica, sans-serif;
    font-weight: lighter;
  }
  body {
    margin: 2em;
  }
  body, input[type="text"], button {
    color: #333;
    font-family: Cambria, Georgia;
  }
  /* everywhere else */
  * {
    font-family: Arial, Helvetica, sans-serif;
  }
```

Sauvegarder et vous pouvez remarquer directement le changement sur la page

Défi :

- Ajouter l'attribut : `myname`, donnez-lui une mise en forme et affichez-le.

Créer un composant Etudiant

1) Dans le terminal de VSC, générez un nouveau composant nommé Etudiant avec :

Vous pouvez ouvrir un nouveau terminal et laisser le terminal précédant tourner le serveur

```
ng generate component etudiant
```

Angular crée un nouveau dossier, `src/app/etudiant/`, et génère 4 fichiers et met à jours un autre.

```
CREATE src/app/etudiant/etudiant.component.html
CREATE src/app/etudiant/etudiant.component.spec.ts
CREATE src/app/etudiant/etudiant.component.ts
CREATE src/app/etudiant/etudiant.component.css
```

2) 4 fichiers sont créés et un fichier est mis à jours. Donner le rôle de chaque fichier
   a. ……
   b. ……
   c. ……
   d. ……
   e. ……
3) Consulter le contenu de ces 4 fichiers. Et commenter leurs contenus.
4) Ouvrez le fichier `app/etudiant/etudiant.component.ts` et décrire son contenu.

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

@Component({
  selector: 'app-etudiant',
  standalone: true,
  imports: [],
  templateUrl: './etudiant.component.html',
  styleUrl: './etudiant.component.css'
})
export class EtudiantComponent {

}
```

Le décorateur @:'@Component' est une tag décoratrice qui spécifie les métadonnées d'Angular du composant. Le décorateur @Component a été utilisé ici pour indiquer que cette classe doit être interprétée comme un composant.
Remarquez les propriétés de métadonnées :
- *selector - le sélecteur d'élément CSS du composant*
- *standalone - attribut a ajouté dans notre composant pour le rendre autonome.*
- *templateUrl - l'emplacement du fichier modèle du composant.*
- *styleUrls - l'emplacement des styles CSS privés du composant.*

Ces éléments ont-été décrite dans la partie précédante.

Le selector, app-etudiant, correspond au nom de l'élément HTML qui identifie cette composant dans le modèle d'un composant parent.
Angular appelle ngOnInit() juste après la création d'un composant.
Remarquez qu'il faut Toujours exporter (export class) la classe composant de sorte que vous pouvez l'importer ailleurs … comme dans le AppModule.

5) Ajouter une variable `etudiant` à la classe `EtudiantComponent`(fichier `etudiant.component. ts`)

```ts
etudiant = 'Ali Ben Saleh';
```

```
<p>etudiant works!</p>
{{ etudiant }}
```

6) Pour afficher cet attribut, dans le fichier `etudiant.component.html.` Supprimez le texte par défaut généré par la CLI angulaire et remplacez-le par ceci :

```
{{ etudiant }}
```

Y-a-t-il un changement dans l'affichage dans l'application ?……
Pourquoi ?.....
……

7) Pour afficher le composant `etudiantComponent` on doit l'ajouter au modèle de l'application principale `app.component.html` avec la balise `<app-etudiant>`

```
<h1> {{title }} < / h1>
<app-etudiant> </app-etudiant>
```

Le navigateur doit afficher à la fois le titre de l'application et le nom de l'étudiant.

A ce stade il pourrait y avoir une erreur

'app-etudiant' is not a known element:

Une solution est d'ajouter l'import du composante au début de app.component.ts

```
import { EtudiantComponent } from './etudiant/etudiant.component';
```

et dans @**Component**({

```
  imports: [RouterOutlet, EtudiantComponent],
```

Créer et manipuler une Classe Etudiant

On veut utiliser plus qu'un attribut. Il est temps de manipuler une classe. On se propose dans cette partie d'enrichir la classe `Etudiant` avec d'autres attributs. Et même de créer une classe Etudiant.ts dans laquelle on stocker les attributs.

1. Créez un nouveau fichier `src/app/etudiant/TypeStudent.ts`. Développer la classe `Student` comme suit.

```
export interface Student {
  id: number;
  name: string;
  classe ?:string
}
```

2. Remarquez que dans TS, la déclaration d'une interface. Googlez.

   Chaque terme sert à quoi ?
   Export :……
   Interface:……
   Qu'elle est la différence entre ces trois déclarations ?

   a. `name : string`
   b. `name!: string`
   c. `name ? : string`

   ……
   ……
   ……

3. Revenez à la classe `etudiantComponent` (fichier `.ts`) et importez l'interface Etudiant.

```
import {Student} from './TypeStudent' ;
```

4. Changer la propriété de `etudiant` de la classe qu'elle soit un objet de la classe `Student`. Initialisez-le.

```
etudiant : Student={
  id:1,
  name:'Ali Ben Saleh'
};
```

   Le fichier de classe `(etudiant.component.ts)` devrait ressembler à ceci:

```
import { Component } from '@angular/core';
import {Student} from './TypeStudent' ;

@Component({
  selector: 'app-etudiant',
  standalone: true,
  imports: [],
  templateUrl: './etudiant.component.html',
  styleUrl: './etudiant.component.css'
})
export class EtudiantComponent {
  etudiant : Student={
    id:1,
    name:'Ali Ben Saleh'
  };
}
```

5. Si vous retourner maintenant à la page, vous remarquer qu'elle ne s'affiche plus correctement. Pourquoi ? Justifier ta réponse.

……

6) Pour afficher le nom de l'étudiant et son id, on adapte l'affichage dans le fichier `etudiant.component.html` comme suit :

```
<h2>{{etudiant.name}}</h2>
<div><strong> id: </strong> {{etudiant.id}}</div>
<div><strong> name: </strong> {{etudiant.name}}</div>
```

Que se passe dans le navigateur ? Expliquer ?

……

……

Défi :

- Ajouter un attribut *lastname* à la classe **Etudiant** et l'afficher
- Ajouter un attribut *average* à la classe **Etudiant** et utiliser une nouvelle présentation qui s'appuie sur les tables html .

7) Les pipes ('|') sont un bon moyen de formater des chaînes, des montants en devises, des dates et d'autres données d'affichage.

Dans le même fichier, ajouter une pipe ('|') devant le 2eme `etudiant.name`.

```
<h2>{{etudiant.name}} Details</h2>
<div><B>id: </B>{{etudiant.id}}</div>
<div><B>name: </B>{{etudiant.name | uppercase}}</div>
```

Le navigateur s'actualise et maintenant le nom de l'étudiant est affiché en majuscules.

a. Que fait le mot **uppercase**?
b. Angular est livré avec plusieurs tuyaux (pipes) intégrés, citer quelques-uns…..

Parmi les filtres prédéfinis :
uppercase: *convertit la valeur de la propriété en majuscule.*
lowercase: *convertit la valeur de la propriété en minuscules.*
percent: *Exprime la valeur décimale en pourcentage avec le signe%.*
currency: *convertie dans la devise spécifiée.*
date: *affiche la propriété sous la forme d'une chaîne de date.*

c. Pouvez-vous créer des pipes vous-mêmes ? Cherchez dans le net comment.

……

Defi

Utiliser les pipes percent, currency et date

Binding (liaisons)

On se propose dans la suite de donner à l'utilisateur la possibilité de modifier le nom d'un étudiant. Pour ce faire on va utiliser la zone de texte `<input>`.

La zone de texte doit à la fois *afficher* la propriété de `name` de l'`etudiant` et *mettre à jour* cette propriété à mesure que l'utilisateur tape du texte. Cela signifie que les données circulent de la classe de composant *vers l'écran* et de l'écran *vers la classe*.

Pour automatiser ce flux de données, on configure une liaison de données bidirectionnelle entre l'élément de formulaire `<input>` et la propriété `etudiant.name`.

1) Refactorisez la zone de détails dans le modèle `etudiantComponent` pour qu'elle ressemble à ceci le fichier (`src/app/etudiant/etudiant.component.html`)

```
<h2>{{etudiant.name}}</h2>
<div><strong> id: </strong> {{etudiant.id}}</div>
<div>
    <label><strong>name: </strong>
        <input [(ngModel)]="etudiant.name" placeholder="name"/>
    </label>
</div>
```

[(ngModel)] est la syntaxe de liaison de données bidirectionnelle d'Angular .

Ici, ngModule permet de lier la propriété `etudiant.name` à la zone de texte HTML afin que les données puissent circuler *dans les deux sens :* de la propriété `etudiant.name` à la zone de texte, et de la zone de texte au `etudiant.name`.

Que se passe dans le navigateur ? Expliquer pourquoi ?

……

……

Defi:

Ouvrez l'outil de développement du navigateur (avec la touche F12), quelle erreur est affichée ?

L'erreur doit être quelque chose comme ci-dessous.

```
Erreurs d'analyse du modèle :
Impossible de se lier à 'ngModel' car ce n'est pas une propriété connue de 'input'.
```

Bien que ngModel soit une directive angulare valide, elle n'est pas disponible par défaut. Il appartient au FormsModule facultatif et vous devez choisir de l'utiliser.

Angular a besoin de savoir comment les éléments de votre application s'emboîtent et quels sont les autres fichiers et bibliothèques requis par l'application. Ces informations sont appelées *métadonnées.*

Certaines des métadonnées se trouvent dans les décorateurs `@Component` que vous avez ajoutés à vos classes de composants. D'autres métadonnées critiques se trouvent dans les décorateurs `@NgModule`.

Le décorateur @NgModule le plus important annote la classe AppModule de niveau supérieur. La CLI d'Angular a généré une classe AppModule dans src/app/app.module.ts lors de la création du projet. C'est là que vous optez dans la FormsModule.

2) Pour résoudre ce problème, Ouvrez AppModule (`app.module.ts`) et importez le symbole `FormsModule` à partir de la bibliothèque @angular/forms.

```
import { FormsModule } from '@angular/forms'; // <-- NgModel est declaré ici
```

Ajoutez ensuite `FormsModule` au tableau des importations de métadonnées `@NgModule`, qui contient une liste de modules externes dont l'application a besoin.

```
  imports: [
    BrowserModule,
    FormsModule
  ],
```

Lorsque le navigateur s'actualise, l'application devrait fonctionner à nouveau. Vous pouvez modifier le nom de l'étudiant et voir les changements reflétés immédiatement dans le `<h2>` au-dessus de la zone de texte.

IV. Récapitulatif

1. Vous avez utilisé la CLI pour créer un deuxième `EtudiantComponent` .
2. Vous avez affiché le `EtudiantComponent` en l'ajoutant au shell `AppComponent` .
3. Vous avez appliqué le `UppercasePipe` pour mettre en forme le nom.
4. Vous avez utilisé la liaison de données bidirectionnelle avec la directive `ngModel` .
5. Vous avez découvert l' `AppModule`.
6. Vous avez importé le `FormsModule` dans l' `AppModule` afin `qu'Angular` reconnaisse et applique la directive `ngModel`.
7. Vous avez appris l'importance de déclarer les composants dans l' `AppModule`.

Revue du code

Voici les codes sources des fichiers présentés dans ce TP.

src/app/etudiant/etudiant.component.ts

```
import { Component, OnInit } from '@angular/core';
import { Etudiant} from './Etudiant' ;
@Component({
  selector: 'app-etudiant',
  templateUrl: './etudiant.component.html',
  styleUrls: ['./etudiant.component.css']
})
export class EtudiantComponent implements OnInit {

  etudiant : Etudiant={
    id:1,
    name:'Ali Ben Saleh'
  };
  constructor() {
  }
```

```
  ngOnInit(): void {
  }
}
```

src /app/ **etudiant / etudiant.component.html**

```
<h2>{{etudiant.name}}</h2>
<div><strong> id: </strong> {{etudiant.id}}</div>
<div>
    <label><strong>name: </strong>
        <input [(ngModel)]="etudiant.name" placeholder="name"/>
    </label>
</div>
```

src / app / app.module.ts

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { FormsModule } from '@angular/forms';

import { AppComponent } from './app.component';
import { EtudiantComponent } from './etudiant/etudiant.component';

@NgModule({
  declarations: [
    AppComponent,
    EtudiantComponent
  ],
  imports: [
    BrowserModule,
    FormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

src / app / app.component.ts

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = MyClass Will be Angular Heroes';
}
```

src /app/app.component.html

```
<h1>{{title}}</h1>
```

```
<app-etudiant> </app-etudiant>
```

src / app / etudiant.ts

```
export  class Etudiant {
    id   !: number;
    name !: string;
}
```

V. Validation des acquis

Question 1 : la Structure d'un Projet Angular

Veuillez répondre aux questions suivantes pour évaluer votre compréhension de la structure d'un projet Angular et du rôle de chaque fichier.

1. Quel fichier sert de point d'entrée principal pour démarrer une application Angular ?

a) index.html
b) main.ts
c) app.module.ts
d) angular.json

2. Dans quel répertoire se trouvent généralement les composants, services et autres éléments de l'application Angular ?

a) src/assets/
b) src/environments/
c) src/app/
d) src/

3. Quel est le rôle du fichier angular.json dans un projet Angular ?

a) Configurer le compilateur TypeScript.
b) Définir les configurations de construction et les options pour Angular CLI.
c) Gérer les dépendances npm du projet.
d) Spécifier les styles CSS globaux de l'application.

4. À quoi sert le fichier package.json dans un projet Angular ?

a) Il décrit les dépendances du projet et les scripts npm disponibles.
b) Il définit les configurations de routage de l'application.
c) Il spécifie les options de compilation pour TypeScript.
d) Il contient le code HTML principal de l'application.

5. Quel est le rôle du fichier tsconfig.json dans un projet Angular ?

a) Configurer les options de compilation TypeScript pour le projet.
b) Définir les configurations de test unitaire.
c) Gérer les dépendances des modules Angular.
d) Spécifier les configurations de styles globaux.

Question 2 : Création et l'Intégration de Composants Angular

1. Quelle commande Angular CLI permet de générer un nouveau composant nommé profil ?

a) ng new component profil
b) ng generate profil

c) ng generate component profil

d) ng create component profil

2. Après avoir généré un composant, lequel des fichiers suivants n'est pas créé automatiquement ?

a) profil.component.html

b) profil.component.ts

c) profil.component.css

d) profil.service.ts

3. Quel décorateur est utilisé pour définir une classe en tant que composant Angular ?

a) @NgModule

b) @Component

c) @Injectable

d) @Directive

4. Pour intégrer un nouveau composant dans l'application, que devez-vous ajouter dans le template du composant parent ?

a) La balise HTML personnalisée correspondant au sélecteur du composant.

b) Une importation du composant dans le fichier TypeScript du parent.

c) Le nom du composant dans le tableau providers du module.

d) Une référence au composant dans le fichier angular.json.

5. Quelle est la commande pour créer un dossier de composants et générer un composant nommé utilisateur à l'intérieur en une seule commande ?

a) ng generate component composants/utilisateur

b) ng create composants/utilisateur

c) ng new component utilisateur in composants

d) ng generate composants/utilisateur component

Question 3 : Liaison de Données Bidirectionnelle avec ngModel

1. Quelle est la syntaxe correcte pour utiliser ngModel dans un élément de formulaire pour la liaison de données bidirectionnelle ?

a) [ngModel]="nom"

b) (ngModel)="nom"

c) [(ngModel)]="nom"

d) {{ ngModel:nm }}

2. Pour que ngModel fonctionne dans un composant, quel module devez-vous importer dans app.module.ts ?

a) ReactiveFormsModule

b) FormsModule

c) HttpClientModule

d) BrowserModule

3. Quelle est la principale différence entre la liaison de données unidirectionnelle et bidirectionnelle ?

a) La liaison unidirectionnelle met à jour le modèle lorsque la vue change, tandis que la bidirectionnelle met à jour la vue lorsque le modèle change.

b) La liaison unidirectionnelle met à jour la vue lorsque le modèle change, tandis que la bidirectionnelle synchronise automatiquement le modèle et la vue dans les deux sens.

c) La liaison unidirectionnelle n'est pas supportée par Angular, seule la bidirectionnelle l'est.

d) La liaison bidirectionnelle est utilisée uniquement avec les services, tandis que la unidirectionnelle est utilisée avec les composants.

4. Dans quel package Angular se trouve la directive ngModel ?

a) @angular/core

b) @angular/common

c) @angular/forms

d) @angular/platform-browser

5. Pourquoi est-il important d'utiliser la liaison de données bidirectionnelle dans les formulaires ?

a) Pour simplifier la manipulation du DOM directement.

b) Pour synchroniser automatiquement les données entre le modèle (composant) et la vue (template).

c) Pour permettre l'utilisation des services au sein des composants.

d) Pour améliorer les performances de l'application en évitant le changement de détection.

Question 4 : l'AppModule

1. Quel est le rôle principal de l'AppModule dans une application Angular ?

a) Il définit le point d'entrée HTML de l'application.

b) Il configure les routes de l'application.

c) Il déclare les composants, directives, et pipes, et importe les modules nécessaires.

d) Il gère les styles globaux de l'application.

2. Quel décorateur est utilisé pour définir une classe en tant que module Angular ?

a) @NgModule

b) @Component

c) @Injectable

d) @Directive

3. Dans quel tableau de l'@NgModule devez-vous ajouter le FormsModule pour utiliser ngModel ?

a) declarations

b) imports

c) providers

d) bootstrap

4. Que se passe-t-il si vous oubliez d'importer un module nécessaire dans l'AppModule ?

a) L'application fonctionne normalement sans problème.

b) Le compilateur génère une erreur indiquant que le module est introuvable.

c) Les fonctionnalités liées au module manquant ne fonctionneront pas, et des erreurs pourraient survenir à l'exécution.

d) Angular importe automatiquement les modules manquants.

5. Quel est le fichier qui contient la définition de l'AppModule par défaut dans un projet Angular ?

a) main.ts

b) app.module.ts

c) app.component.ts

d) angular.json

Question 5 : QCM sur la Déclaration des Composants dans l'AppModule

1. Pourquoi est-il nécessaire de déclarer les composants dans l'AppModule ?

a) Pour que les composants puissent être utilisés dans les templates d'autres composants du même module.

b) Pour que les composants soient compilés en production uniquement.

c) Pour que les services soient injectés correctement dans les composants.

d) Pour que les composants soient automatiquement enregistrés dans le système de routage.

2. Dans quel tableau de l'@NgModule déclarez-vous les composants, directives et pipes de votre module ?

a) imports

b) declarations

c) providers

d) exports

3. Que se passe-t-il si vous utilisez un composant dans un template sans l'avoir déclaré dans un module ?

a) Le composant sera ignoré et ne sera pas rendu.

b) Angular génère une erreur indiquant que le composant n'est pas une entité connue.

c) Le composant sera automatiquement déclaré par Angular.

d) Le compilateur supprime le composant lors de la compilation.

4. Est-il possible de déclarer un composant dans plusieurs modules simultanément ?

a) Oui, c'est recommandé pour le partage de composants.

b) Non, un composant ne peut être déclaré que dans un seul module.

c) Oui, mais uniquement si les modules sont chargés paresseusement.

d) Non, sauf si vous utilisez le module racine.

5. Quelle est la bonne pratique pour organiser les déclarations des composants dans les modules Angular ?

a) Déclarer tous les composants dans l'AppModule principal.

b) Créer des modules fonctionnels ou de fonctionnalités pour regrouper les composants liés et déclarer les composants dans ces modules.

c) Ne pas déclarer les composants et laisser Angular les gérer automatiquement.

d) Déclarer les composants uniquement dans les modules de routage.

Questions ouvertes pour évaluer les acquis

1. Expliquez le rôle du fichier app.component.ts dans une application Angular et comment il s'intègre avec app.module.ts et main.ts.
2. Décrivez le processus complet de création d'un nouveau composant Angular et son intégration dans une application, en mentionnant les étapes clés et leur importance.
3. Pourquoi est-il nécessaire d'importer le FormsModule pour utiliser ngModel ? Que contient ce module et comment contribue-t-il au fonctionnement des formulaires dans Angular ?

4. Expliquez ce qu'est la liaison de données bidirectionnelle et donnez un exemple concret de son utilisation dans un formulaire Angular.

5. Discutez des conséquences potentielles de ne pas déclarer un composant dans l'AppModule ou le module approprié. Comment cela affecte-t-il le comportement de l'application ?