Multiple Choice Questions (MCQs)

1. **Angular Components:**

What is the purpose of components in Angular?

a) To handle routing

b) To create reusable UI pieces

c) To manage data models

d) To handle form submission

2. **Data Binding:**

Which type of data binding is used to send data from the parent component to the child component?

a) One-way binding

b) Two-way binding

c) Property binding

d) Event binding

3. **Dependency Injection:**

What is Dependency Injection (DI) used for in Angular?

a) To increase coupling between classes

b) To create new instances of services

c) To inject services into components or other services

d) To manage routing and navigation

4. **Directives**

What is the purpose of the `ngIf` directive in Angular?

a) To loop through an array

b) To conditionally display an element

c) To handle user input

d) To transform data

5. **Services**

Which decorator is used to define a service in Angular?

a) `@Injectable()`

b) `@Component()`

c) `@Directive()`

d) `@NgModule()`

6. **Lifecycle hooks**

Which lifecycle hook is called after Angular has fully initialized a component's view?

a) `ngOnInit()`

b) `ngOnChanges()`

c) `ngAfterViewInit()`

d) `ngOnDestroy()`

7. **Routing**

How do you specify a parameter in an Angular route path?

a) `/route/:param`

b) `/route/{param}`

c) `/route?(param)`

d) `/route/<param>`

8. **Reactive forms**

Which class is used in Angular to track the value and validation status of an individual form control?

a) `FormGroup`

b) `FormControl`

c) `FormArray`

d) `FormBuilder`

9. Dependency injection

What is a primary benefit of Dependency Injection in Angular?

a) Increases coupling between classes

b) Automatically generates components

c) Helps in managing state

d) Promotes loose coupling between classes

10. Pipes

Which built-in pipe would you use to format a date in Angular?

a) `datePipe`

b) `currencyPipe`

c) `numberPipe`

d) `jsonPipe`

11. Angular modules

What is the purpose of `NgModule` in Angular?

a) To encapsulate and group components, services, and directives

b) To create standalone applications

c) To manage routing exclusively

d) To handle HTTP requests

12. Observables and RxJS

What is an Observable in Angular?

a) A function that executes when an event occurs

b) A continuous stream of data that you can subscribe to

c) A type of array

d) A service for HTTP requests

13. Change detection

How does Angular detect changes within the application?

a) By comparing virtual DOM with the actual DOM

b) By periodically refreshing the application

c) By using a zone to detect asynchronous operations

d) By manually triggering change detection

Direct Questions

1. **Directives:**
   - Explain the difference between structural and attribute directives in Angular.
2. **Services:**
   - Describe how services are used in Angular and why they are beneficial.
3. **Routing:**
   - How do you configure a route in Angular to navigate to a component?

Coding Questions

4. **Create a Component:**
   - Write a simple Angular component to display a user's name and age. Include the necessary decorator and metadata.
5. **Event Handling:**
   - Provide an example of how to handle a button click event in Angular to update a variable's value.
6. **Service and Dependency Injection:**
   - Create a service that retrieves user data from a mock API and inject it into a component to display the user's details.

Advanced Questions

7. **RxJS and Observables:**
   - Explain how you would use Observables and RxJS to handle asynchronous data streams in Angular.
8. **Change Detection:**
   - Discuss Angular's change detection strategy and how you can optimize it for better performance.
9. **ngModel and Forms:**
   - Provide an example of how to use ngModel within a form to capture and validate user input.

Basic Coding Questions

1. **Component Creation:**
   - Write an Angular component that displays a simple "Hello, World!" message.

2. **Event Binding:**
    - Create a component with a button. When clicked, the button should display an alert with a custom message.
3. **Property Binding:**
    - Bind a component property to the paragraph tag in the template and update it dynamically.
4. **ngFor Directive:**
    - Use the `ngFor` directive to render a list of items from an array in the component.

Intermediate Coding Questions

5. **Custom Pipe:**
    - Create a custom pipe that transforms a string to uppercase and appends an exclamation mark.
6. **Service and HTTP:**
    - Write a service that makes an HTTP GET request to fetch user data from a public API and display the data in a component.
7. **Form Handling:**
    - Create a reactive form with fields for a user's first name, last name, and email. Include basic validation and display the form values on submit.
8. **Routing and Navigation:**
    - Set up a basic routing configuration to navigate between two components.

Advanced Coding Questions

9. **State Management:**
    - Implement a simple state management solution using services and RxJS to share data between unrelated components.
10. **Dynamic Component Loader:**
    - Write a directive that dynamically loads and displays a component upon a specific user action (like clicking a button).
11. **Directives with HostListeners:**
    - Create a directive that changes the color of the text in a paragraph when hovered over.
12. **Optimizing with Pure Pipes:**
    - Create a pure pipe that performs a complex calculation and demonstrate how it avoids recalculations when input data doesn't change.

Expert-Level Coding Questions

13. **Advanced Routing:**
    - Configure lazy-loaded routes in your Angular application to improve load times.
14. **Content Projection:**
    - Create a component that accepts projected content and displays it in a specific layout.
15. **Custom Structural Directive:**
    - Write a structural directive that conditionally renders an element based on a custom condition.

16. **Higher-Order Observables:**
    - Demonstrate using a higher-order Observable in a service to manage multiple API requests efficiently.

Code fixing :

Question 1: Component Property Binding

typescript Copy code

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

@Component({
  selector: 'app-user-profile',
  template: `<h1>Welcome, {{username}}</h1>`
})
export class UserProfileComponent {
  userName: string;
  constructor() {
    this.userName = 'John Doe';
  }
}
```

Task: Identify and fix the issue in the code above to ensure the username is displayed correctly.

Question 2: Event Binding

typescript Copy code

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

@Component({
  selector: 'app-counter',
  template: `
    <button (click)="incrementCounter">Increment</button>
    <p>Counter: {{counter}}</p>
  `
})
export class CounterComponent {
  counter = 0;

  incrementCounter() {
    this.counter += 1;
  }
}
```

Task: The button is supposed to increment the counter, but it's not working as expected. Find and fix the issue.

Question 3: ngFor Directive Usage

typescript Copy code

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

@Component({
  selector: 'app-item-list',
  template: `
    <ul>
      <li *ngFor="let item of items">{{item}}</li>
    </ul>
  `
})
export class ItemListComponent {
  items = ['Item 1', 'Item 2', 'Item 3'];
}
```

Task: The list of items isn't displaying as intended. Identify and resolve any issues with the code.

Question 4: HTTP Service Error Handling

typescript Copy code

```typescript
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';

@Injectable({
  providedIn: 'root',
})
export class DataService {
  constructor(private http: HttpClient) {}

  fetchData() {
    this.http.get('https://api.example.com/data').subscribe(
      (data) => console.log('Data:', data),
      (error) => console.error('There was an error!', error)
    );
  }
}
```

Task: There's an error occurring when fetching data, but it's not being handled correctly. Improve the error handling in this service.

Question 5: Reactive Forms Validation

```
import { Component } from '@angular/core';
import { FormGroup, FormControl } from '@angular/forms';

@Component({
  selector: 'app-profile-editor',
  template: `
    <form [formGroup]="profileForm">
      <label for="first-name">First Name:</label>
      <input id="first-name" formControlName="firstName">

      <label for="last-name">Last Name:</label>
      <input id="last-name" formControlName="lastName">
    </form>
  `
})
export class ProfileEditorComponent {
  profileForm = new FormGroup({
    firstName: new FormControl(''),
    lastName: new FormControl(''),
  });
}
```

Task: The form is supposed to include validation for both first and last names to ensure they are not empty. Implement the necessary validators