Angular Lab Assessment: Testing Understanding in Data Binding

Objective:

To assess students' understanding of Angular data binding concepts, including property binding, event binding, two-way binding, and interpolation. This assessment will test their ability to implement these bindings in a real-world scenario.

Prerequisites:

- Familiarity with Angular components.
- Basic understanding of TypeScript.
- Knowledge of Angular data binding mechanisms.

Scenario Overview:

Students are required to build a simple user profile management form. The form will allow users to input their name, email, and age, and display the entered information dynamically. The form should also include a button to reset the form fields to their default values.

Assessment Instructions

1. **Create a New Angular Project:**
    - Students should create a new Angular project using the Angular CLI.
    - Name the project user-profile-app.

ng new user-profile-app

cd user-profile-app

2. **Generate a User Profile Component:**
    - Create a new component named UserProfile.

ng generate component user-profile

3. **Task 1: Implement Property Binding**
    - In the UserProfileComponent, define a property userName in the component class.
    - Use property binding to display the userName in the template.
    - Initially set userName to "John Doe".

Steps:

- Define userName: string = 'John Doe'; in user-profile.component.ts.
- Display the userName using property binding in the template.

<h2>User Name: {{ userName }}</h2>

4. **Task 2: Implement Event Binding**
    - Add a button that, when clicked, changes the userName to "Jane Doe".
    - Use event binding to handle the button click event.

Steps:

- Add a method changeUserName() in user-profile.component.ts that sets userName to "Jane Doe".
- Bind the click event of the button to this method.

<button (click)="changeUserName()">Change User Name</button>

5. **Task 3: Implement Two-Way Binding**
    - Add input fields for the user to update their name, email, and age.
    - Use two-way binding to bind the input fields to the respective properties in the component class.

Steps:

- Define properties userEmail: string and userAge: number in the component class.
- Use [(ngModel)] to bind these properties to the input fields.

<input type="text" [(ngModel)]="userName" placeholder="Enter your name">

<input type="email" [(ngModel)]="userEmail" placeholder="Enter your email">

<input type="number" [(ngModel)]="userAge" placeholder="Enter your age">

6. **Task 4: Display User Information Using Interpolation**
    - Below the form, display the user's information using interpolation.

Steps:

- Use interpolation to display userName, userEmail, and userAge below the input fields.

Name: {{ userName }}

Email: {{ userEmail }}

Age: {{ userAge }}

7. **Task 5: Implement Reset Functionality**
    - Add a "Reset" button that clears the form fields and resets the user information to default values.

Steps:

- Add a method resetForm() in user-profile.component.ts that resets the form fields.

- Bind the click event of the "Reset" button to this method.

<button (click)="resetForm()">Reset</button>

8. **Task 6: Validation (Optional for Extra Credit)**
    - Add basic form validation to ensure that all fields are filled out before allowing submission.

Steps:

- Use Angular's form validation techniques to add required validation on the input fields.
- Display error messages if the fields are not filled out.

<div *ngIf="!userName">Name is required.

<div *ngIf="!userEmail">Email is required.

<div *ngIf="!userAge">Age is required.

Assessment Submission

- Students should submit their project via a Git repository or as a zipped project folder.
- Ensure the application is fully functional and meets all the requirements outlined in the tasks.

Assessment Criteria

Criteria	**Points**
Property binding implementation	10
Event binding implementation	10
Two-way binding implementation	20
Displaying user information using interpolation	10
Reset functionality implementation	20
Code quality and best practices	10
Form validation (extra credit)	10
Total	**80-90**

Hints and Tips:

- Make sure you use proper naming conventions for your variables and methods.
- Keep your component class clean and organize your code properly.
- Test your application thoroughly to ensure that all bindings are working as expected.

This assessment is designed to test your ability to apply data binding concepts in a real-world scenario. Good luck!