Creating a lab exercise focused on Angular's structural directive `*ngIf` offers an excellent opportunity to practice conditional rendering within templates. This exercise will guide you through the process of building a simple Angular application that demonstrates the use of `*ngIf` to conditionally display content based on user input and component state.

Lab Exercise: Exploring *ngIf in Angular

Objective

Learn how to use the `*ngIf` directive to conditionally display elements in the DOM based on specific conditions.

Requirements

- Angular CLI installed on your machine.
- Basic understanding of Angular application structure.
- Basic knowledge of TypeScript and HTML.

Setup

1. Create a new Angular project by running: `ng new ngif-practice`.
2. Navigate into your project directory: `cd ngif-practice`.
3. Serve your application to ensure it's working: `ng serve`.
4. Open the project in your preferred code editor.

Exercises

Exercise 1: Simple Conditional Rendering

1. In the app component (`app.component.ts`), add a boolean property `isVisible` with a default value of `false`.
2. In the app component template (`app.component.html`), add a button that toggles the `isVisible` property between `true` and `false` when clicked.
3. Below the button, add a paragraph (`<p>`) element that only displays when `isVisible` is `true`. Use the `*ngIf` directive. The paragraph should contain any text, e.g., "This text is conditionally displayed."

Exercise 2: Using *ngIf with an Else Clause

1. Add another paragraph element to the template that will display when `isVisible` is `false`. This requires using `*ngIf` with an `else` clause.
2. Define a template reference for the else condition using the `<ng-template>` tag. For example, `<ng-template #elseBlock>Text when isVisible is false</ng-template>`.
3. Update the `*ngIf` directive on the first paragraph to use the `else` clause, referencing the `elseBlock`.

Exercise 3: *ngIf with Component Property

1. Add a new property `user` to the app component, which is an object containing `name` and `isLoggedIn` (boolean).
2. Based on the `isLoggedIn` property, use `*ngIf` to conditionally display a welcome message, e.g., "Welcome, [user's name]!" when the user is logged in.
3. Implement an else condition that displays a login button (it doesn't have to function for this exercise) when the user is not logged in.

Exercise 4: Combining *ngIf with *ngFor

1. In the app component, add an array property of items (e.g., `items = ['Item 1', 'Item 2', 'Item 3'];`).
2. Use `*ngFor` to display each item in a list (`<ul><li>`).
3. Above the list, add an input field for filtering items based on user input.
4. Implement a method in the app component that filters the displayed items based on the input field's value.
5. Use `*ngIf` to display a message when no items match the filter criteria, e.g., "No items found."

Conclusion

Upon completing these exercises, you will have practiced using Angular's `*ngIf` directive to show or hide elements based on conditions. This exercise covers basic usage, combining `*ngIf` with an `else` clause, using component properties with `*ngIf`, and combining `*ngIf` with `*ngFor` for more complex conditional rendering scenarios.