Lesson 9 - Directives Deep Dive

What We'll Cover:

1. Types of Directives (Component, Attribute, Structural)
2. Built-in Attribute Directives (ngClass, ngStyle)
3. Creating a Custom Attribute Directive (ng generate directive)
4. Using @if and @for Effectively (and *ngIf, *ngFor Legacy)
5. Practical Use Cases (Highlighting, Toggling Visibility)

Types of Directives

- **Concept**: Directives extend HTML behavior
- **Three Types**:
    - **Component Directives**: Components are directives with templates (e.g., @Component)
    - **Attribute Directives**: Modify element behavior or appearance (e.g., ngClass)
    - **Structural Directives**: Alter DOM structure (e.g., @for, @if)

- **Syntax**:
    - Attribute: [directiveName]="value"
    - Structural: Modern @directive or legacy *directive

- **Key Point**: Directives are Angular’s secret sauce

Built-in Attribute Directives

- **Concept**: Pre-built tools for styling and behavior
- **Examples**:
    - **ngClass**:

```
template: `<p [ngClass]="{'active': isActive}">Text</p>`

isActive = true; // Adds 'active' class
```

ngStyle:

```
template: `<p [ngStyle]="{'color': textColor}">Text</p>`

textColor = 'blue'; // Text turns blue
```

Key Point: Dynamic styling without CSS files

Creating a Custom Attribute Directive

- **How**: Use Angular CLI
- **Command**:

```
ng generate directive highlight
```

Example:

```
import { Directive, ElementRef, HostListener } from '@angular/core';
@Directive({
  standalone: true,
  selector: '[appHighlight]'
})
export class HighlightDirective {
  constructor(private el: ElementRef) {
        this.el.nativeElement.style.backgroundColor = 'yellow';
  }
  @HostListener('mouseenter') onMouseEnter() {
        this.el.nativeElement.style.backgroundColor = 'lightgreen';
  }
  @HostListener('mouseleave') onMouseLeave() {
        this.el.nativeElement.style.backgroundColor = 'yellow';
  }
}
```

Usage:

```
template: `<p appHighlight>Hover me!</p>`
```

Output: Yellow background, green on hover

Using @if and @for Effectively

- **Concept**: Modern structural directives (Angular 17+)
- **Legacy Note**: *ngIf and *ngFor still work but are legacy
- **Examples**:
    - **@if**:

```
template: `
  @if (isVisible) {
      <p>Visible!</p>
  }
`

isVisible = true;
```

@for:

```
template: `
  <ul>@for (item of items; track item) { <li>{{ item }}</li> }</ul>
`

items = ['A', 'B', 'C'];
```

Tips:

- Use track with @for for performance
- @if can include @else

Legacy *ngIf and *ngFor

- **Why Mention?**: Still common in older code
- **Examples**:
    - ***ngIf**:

```
<p *ngIf="isVisible">Visible!</p>
```

***ngFor**:

```
<ul><li *ngFor="let item of items">{{ item }}</li></ul>
```

Key Point: @if/@for are cleaner, more JavaScript-like

Transition Tip: Update old code to modern syntax

Practical Use Cases

- **Highlighting**:

```
template: `<p appHighlight [ngClass]="{'bold': isBold}">Text</p>`
isBold = true;
```

Toggling Visibility:

```
template: `
  <button (click)="toggle()">Toggle</button>
  @if (show) {<p>Peek-a-boo!</p>
  } @else {  <p>Hidden</p> }
`
show = false;
toggle() { this.show = !this.show; }
```

Demo: Highlight on hover, toggle content

Putting It Together

- **Mini-Project**: Task list with directives
- **Code**:

```
@Component({
  standalone: true,
  imports: [CommonModule, HighlightDirective],
  template: `
    <button (click)="toggle()">Show/Hide</button>
    @if (visible) {  <ul>  @for (task of tasks; track task) {  <li appHighlight [ngStyle]="{'color': taskColor}">{{ task }}</li>  }
</ul>  }
  `})
export class TaskListComponent {
  tasks = ['Learn Directives', 'Build App'];
  visible = false;
  taskColor = 'blue';
  toggle() { this.visible = !this.visible; }
}  //
```

Demo: Toggle list, highlight on hover

Key Takeaways

- Directives: Components, attributes, structural
- Built-in: ngClass, ngStyle for dynamic styling
- Custom: Build your own with ng generate directive
- Modern @if/@for replace *ngIf/*ngFor
- Use cases: Enhance UI with toggles, highlights