**Angular Attribute and Structural Directives**

Directives are classes that add additional behavior to elements in your Angular applications. There are two main types of directives: Attribute directives and Structural directives. In this tutorial, we'll explore both types with examples in Angular.

**Attribute Directives**

Attribute directives change the appearance or behavior of an element, component, or another directive.

**Built-in Attribute Directive: ngStyle**

Let's start with a built-in attribute directive, ngStyle:

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

import { NgStyle } from '@angular/common';

@Component({

selector: 'app-root',

template: `

<h2 [ngStyle]="{'color': textColor, 'font-size': fontSize}">

Styled Heading

</h2>

<button (click)="toggleColor()">Toggle Color</button>

`,

standalone: true,

imports: [NgStyle]

})

export class AppComponent {

textColor = 'blue';

fontSize = '24px';

toggleColor() {

this.textColor = this.textColor === 'blue' ? 'red' : 'blue';

}

}

Expected output:

Styled Heading (in blue, 24px font)

[Toggle Color] (button)

Clicking the button will toggle the heading color between blue and red.

**Custom Attribute Directive**

Now, let's create a custom attribute directive:

// highlight.directive.ts

import { Directive, ElementRef, HostListener, Input } from '@angular/core';

@Directive({

selector: '[appHighlight]',

standalone: true

})

export class HighlightDirective {

@Input() appHighlight = '';

constructor(private el: ElementRef) {}

@HostListener('mouseenter') onMouseEnter() {

this.highlight(this.appHighlight || 'yellow');

}

@HostListener('mouseleave') onMouseLeave() {

this.highlight('');

}

private highlight(color: string) {

this.el.nativeElement.style.backgroundColor = color;

}

}

// app.component.ts

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

import { HighlightDirective } from './highlight.directive';

@Component({

selector: 'app-root',

template: `

<p appHighlight="lightblue">Hover over me!</p>

<p appHighlight>Hover over me too!</p>

`,

standalone: true,

imports: [HighlightDirective]

})

export class AppComponent {}

Expected output:

Hover over me!

Hover over me too!

The first paragraph will highlight in light blue when hovered, while the second will use the default yellow color.

**Structural Directives**

Structural directives change the structure of the DOM by adding or removing elements.

**Built-in Structural Directive: \*ngIf**

The \*ngIf directive is used to conditionally render elements:

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

import { NgIf } from '@angular/common';

@Component({

selector: 'app-root',

template: `

<button (click)="toggleShow()">Toggle Message</button>

<p \*ngIf="showMessage">Hello, Angular!</p>

`,

standalone: true,

imports: [NgIf]

})

export class AppComponent {

showMessage = false;

toggleShow() {

this.showMessage = !this.showMessage;

}

}

Expected output (initially):

[Toggle Message] (button)

After clicking the button:

[Toggle Message] (button)

Hello, Angular!

**Built-in Structural Directive: \*ngFor**

The \*ngFor directive is used for rendering lists:

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

import { NgFor } from '@angular/common';

@Component({

selector: 'app-root',

template: `

<ul>

<li \*ngFor="let item of items; let i = index">

{{ i + 1 }}. {{ item }}

</li>

</ul>

`,

standalone: true,

imports: [NgFor]

})

export class AppComponent {

items = ['Apple', 'Banana', 'Cherry', 'Date'];

}

Expected output:

- 1. Apple

- 2. Banana

- 3. Cherry

- 4. Date

**Custom Structural Directive**

Let's create a custom structural directive that repeats an element a specified number of times:

// repeat.directive.ts

import { Directive, Input, TemplateRef, ViewContainerRef } from '@angular/core';

@Directive({

selector: '[appRepeat]',

standalone: true

})

export class RepeatDirective {

constructor(

private templateRef: TemplateRef<any>,

private viewContainer: ViewContainerRef

) {}

@Input() set appRepeat(count: number) {

this.viewContainer.clear();

for (let i = 0; i < count; i++) {

this.viewContainer.createEmbeddedView(this.templateRef, { index: i });

}

}

}

// app.component.ts

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

import { RepeatDirective } from './repeat.directive';

@Component({

selector: 'app-root',

template: `

<div \*appRepeat="3">

<p>This is repeated content!</p>

</div>

`,

standalone: true,

imports: [RepeatDirective]

})

export class AppComponent {}

Expected output:

This is repeated content!

This is repeated content!

This is repeated content!

**Combining Attribute and Structural Directives**

You can use both types of directives together:

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

import { NgFor, NgStyle } from '@angular/common';

import { HighlightDirective } from './highlight.directive';

@Component({

selector: 'app-root',

template: `

<ul>

<li \*ngFor="let item of items; let i = index"

[ngStyle]="{'color': i % 2 === 0 ? 'blue' : 'green'}"

[appHighlight]="i % 2 === 0 ? 'lightblue' : 'lightgreen'">

{{ item }}

</li>

</ul>

`,

standalone: true,

imports: [NgFor, NgStyle, HighlightDirective]

})

export class AppComponent {

items = ['Item 1', 'Item 2', 'Item 3', 'Item 4'];

}

Expected output:

- Item 1 (in blue text, highlights lightblue on hover)

- Item 2 (in green text, highlights lightgreen on hover)

- Item 3 (in blue text, highlights lightblue on hover)

- Item 4 (in green text, highlights lightgreen on hover)

**Conclusion**

Directives in Angular provide powerful ways to manipulate the DOM and add dynamic behavior to your applications. Attribute directives change the appearance or behavior of an element, while structural directives change the structure of the DOM itself. By combining these directives, you can create rich, interactive user interfaces with clean, declarative template syntax.