**ĐỊNH NGHĨA**

Directives là một đối tượng giúp chúng ta dễ dàng thay đổi một đối tượng khác và cách áp dụng rất đơn giản và linh hoạt. Có 3 loại Directives trong angular: 1. Component-Directives với template 2. Structural directives-thay đổi cấu trúc DOM bằng việc thêm bớt các phần tử trong DOM 3. Attribute directives-thay đổi giao diện và tương tác của các đối tượng hoặc thay đổi directive khác

**CÁC LOẠI DIRECTIVE**

### DIRECTIVE CHO COMPONENT  (Directive thành phần)

Là directive được angular cung cấp sẵn. Component Directive cho phép ta khai báo selector, cách hoạt động, xử lý của component.

Ng g c [name]

### Structural directive (Directive cấu trúc)

Đúng như tên gọi, structural directive hay directive cấu trúc sẽ quyết định DOM element nào được thực thi. Các structural directive thường có dấu '**\*'** ở trước của directive. Ví dụ điển hình của structural directive chính là **\*ngIf** , **\*ngFor và [ngSwitch]**

## **NgIf**

Sử dụng khi muốn thêm hoặc xóa bỏ một phần tử khi render. Ví dụ: hiển thị thông báo lỗi khi người dùng nhập form chưa đúng.

Cú pháp:

<h2 \*ngIf="printable">{{ message }}</h2>

Chú thích: printable là 1 property dạng Boolean, nếu printable = true thì element h2 sẽ hiện ra, ngược lại sẽ bị ẩn đi.

## **NgFor**

Sử dụng khi muốn render một list các phần tử. Ví dụ: render list các liên hệ trong 1 danh bạ chẳng hạn.

Cú pháp:

<div \*ngFor="let contact of contacts">

<h3>{{ contact.name }}</h3>

<div>

<img \*ngIf="contact.avatar?.url" [src]="contact.avatar?.url" alt="Avatar of {{ contact.name }}">

</div>

</div>

Lưu ý: đừng quên dấu \* phía trước ngFor directive và sử dụng cấu trúc let [item] of [array]

### NgSwitchCase

Sử dụng thay thế việc if nhiều lần, tương tự như switch-case trong Javascript.

<div [ngSwitch]="tabIndex">

<div \*ngSwitchCase="1">

<div>

Tab content 1

</div>

<p>

Đây là nội dung tab 1 </p>

</div>

<div \*ngSwitchCase="2">

<div>

Tab content 2

</div>

<p>

Đây là nội dung tab 2

</p>

</div>

<div \*ngSwitchCase="3">

<div>

Tab content 3

</div>

<p>

Nội dung tab 3 </p>

</div>

</div>

Lưu ý: không có dấu \*ở phía trước ngSwitch directive. Thay vào đó, sử dụng property binding.

Đặt dấu \* ở phía trước ngSwitchCasevà ngSwitchDefault.

### Attribute directive (Directive thuộc tính)

Directive thuộc tính là các directive được sử dụng như một thuộc tính của thẻ HTML .

### Tạo attribute directive

Hôm nay mình muốn nói sâu hơn về Attribute directives. Nó được dùng như một thuộc tính của đối tượng, cho nên khi build thì directive và các thuộc tính thông thường khác được build cùng một lúc dẫn đến dự thay đổi của directive là đồng thời khi render đối tượng đó.

**1. Tạo một attributes directive đơn giản.**

Một attribute directive cần requires decorate class đối tượng Directive của angular bằng các dùng [@Directive](https://viblo.asia/u/Directive) trước class. Ví dụ đây là HighlightDirective mục đích sẽ làm thay đổi màu background của đối tượng khi người dùng hover qua nó.

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

@Directive({

selector: '[appHighlight]'

})

export class HighlightDirective {

constructor() { }

}

Khi làm vậy đối tượng của chúng ta sẽ có 1 tên selector , ở đây là appHighlight Trang mà muốn dử dụng HighlightDirective cho một đối tượng để set background color khi người dùng hover. Chúng ra chỉ cần áp dụng như sau:

<p appHighlight>Highlight me!</p>

Implement cho directive. Chúng ta có thể sử dụng CLI command để tạo ra đối tượng directive.

ng generate directive highlight

CLI sẽ tạo ra file src/app/highlight.directive.ts và khai báo nó trong AppModule Cấu trúc của file src/app/highlight.directive.ts

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

@Directive({

selector: '[appHighlight]'

})

export class HighlightDirective {

constructor() { }

}

Import định danh Directive để sử dụng nó decorate cho đối tượng trong angular. Gọi [@Directive](https://viblo.asia/u/Directive)trước class HighlightDirective là để sư dụng decorate, khi sử dùng chúng ta cần khai báo tên selector để sử dụng như một thuộc tính, [appHighlight] dấu ([]) là cách mà angular hiểu nó là một thuộc tính, Khi biên dịch mà thấy phần tử nào có thuộc tính có tên là appHighlight sẽ được apsp dụng thay đổi bởi directive.

Sau khi sử dụng [@Directive](https://viblo.asia/u/Directive), đừng quên export class HighlightDirective để có thể import và sử dụng. Bây giờ chúng ta hãy implement cho cho HighlightDirective để làm thay đổi màu background:

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

@Directive({

selector: '[appHighlight]'

})

export class HighlightDirective {

constructor(el: ElementRef) {

el.nativeElement.style.backgroundColor = 'yellow';

}

}

ElementRef là Class trong thư viện của angular. Chúng ta dùng ElementRef trong hàm construct để inject nó tham chiếu đến các phần tử DOM trong component hiện tại. Sau đó chi cần gọi thuộc tính ElementRef để lấy về đối tượng DOM để thay đổi style background sang mày vàng.

**2. Áp dụng attribute directive.**

Để dùng HighlightDirective, thêm thẻ

như sau:

<p appHighlight>Highlight me!</p>

Chạy server để thấy kết quả.

ng serve

**3. Tương tác directive với người dung( lắng nghe hành động ).**

Hiện tại appHighlight chỉ c màu cố dịnh cho background, chưa hề có sự linh hoạt và tương tác nào. Chúng ta sẽ implement nó để thay đổi màu cho các sự kiện chuột và người dùng hành động. Trước tiên cần import HostListener.

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

Tiếp theo là thêm hàm xử lý khi sự kiện xảy ra bằng cách sử dụng Decorator HostListener.

@HostListener('mouseenter') onMouseEnter() {

this.highlight('yellow');

}

@HostListener('mouseleave') onMouseLeave() {

this.highlight(null);

}

private highlight(color: string) {

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

}

[@HostListener](https://viblo.asia/u/HostListener) decorator sẽ theo dõi và bắt các sự kiện của phần tử trong DOM mà có dử dụng directive appHighlight Hàm highlight sẽ thay đổi background color theo màu được truyền vào tham số, nên trong các hàm xử lý chỉ cần gọi tới highlight với tham số là màu cần hiển thị. Chạy serve và thấy kêt quả nhé.

**4. Pass dữ liệu vào directive thông qua** [@Input](https://viblo.asia/u/Input" \t "_blank)

Hiện tại các màu cho các sự kiện vẫn là cố định, sử dụng ở đâu thì các màu vẫn vậy . Để tạo nên tính linh hoạt cho directive chúng ta sẽ truyền các màu vào cho nó. Đầu tiên cần import Input:

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

Sử dụng decorator cho biến highlightColor.

@Input() highlightColor: string;

Sử dụng kết hợp appHighlight và input binding cho đối tượng.

<p appHighlight highlightColor="yellow">Highlighted in yellow</p>

<p appHighlight [highlightColor]="'orange'">Highlighted in orange</p>

Đó là cách sử dụng ban đầu, nhưng directive đã được cải thiện để rút ngắn code và thuaanjt iện hơn bằng cách sử dụng director như một thuộc tính(đây là lý do vì sao selector của nó có dấu [])

<p [appHighlight]="color">Highlight me!</p>

Thuộc tính [appHighlight] là sự kết hợp của highlighting directive và set àu cho biến appHighlight Chúng ta cũng có thể đổi tên cho biến nếu không muốn đặt tên biến là appHighlight theo selector

@Input('appHighlight') highlightColor: string;

Chúng ta cũng hoàn toàn có thể kết hợp cách trên với input binding thông thường.

<p [appHighlight]="color" defaultColor="violet">

Highlight me too!

</p>

@Input() defaultColor: string;

Angular sẽ tự hiểu bạn binding defaultColor cho HighlightDirective vì bạn đã khai báo decorator Input cho nó. binding thêm một defaultColor thông qua input để làm màu mặc định. trong hàm xử lý cho sự kiện chuột thay đổi như sau:

@HostListener('mouseenter') onMouseEnter() {

this.highlight(this.highlightColor || this.defaultColor || 'red');

}

**5. Bây giờ hãy cùng làm một ví dụ đơn giản nhé.** Update file app.component.html:

<h1>My First Attribute Directive</h1>

<h4>Pick a highlight color</h4>

<div>

<input type="radio" name="colors" (click)="color='lightgreen'">Green

<input type="radio" name="colors" (click)="color='yellow'">Yellow

<input type="radio" name="colors" (click)="color='cyan'">Cyan

</div>

<p [appHighlight]="color">Highlight me!</p>

File src/app/app.component.ts:

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

@Directive({

selector: '[appHighlight]'

})

export class HighlightDirective {

color: string;

constructor(private el: ElementRef) { }

@Input('appHighlight') highlightColor: string;

@HostListener('mouseenter') onMouseEnter() {

this.highlight(this.highlightColor || 'red');

}

@HostListener('mouseleave') onMouseLeave() {

this.highlight(null);

}

private highlight(color: string) {

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

}

}