1. Generate a directive called btn
   1. This will be an attribute directive

ng g d btn

1. Import ElementRef, Renderer2 and HostListener from @angular/core
2. Create a setFontColor method that uses renderer and setStyle() to set the font color (‘color’) of the native element to the variable passed in (color)

setFontColor(color: string) {

this.renderer.setStyle(

this.elementRef.nativeElement,

'color',

color

)

}

1. Inside the constructor, use the setFontColor method to set an initial font color. Set variables for ElementRef and Renderer2.

constructor(

private elementRef: ElementRef,

private renderer: Renderer2

) {

this.setFontColor('red')

}

1. Use @HostListener to change the font color on an event. The event inside @HostListener is the native event. If you use mouseenter, you can create another one for mouseleave.

@HostListener('mouseenter') OnMouseEnter()() {

this.setFontColor('blue')

}

1. Apply the attribute directive in the template.

<button appBtn>My Button</button>

1. Generate another directive, a structural directive that will do the opposite of \*ngIf

ng g d not

1. Import Directive, Input, TemplateRef, ViewContainerRef from '@angular/core'
2. The Input decorator is used to communicate between the two components. It sends data from one component to the other using property binding.
3. TemplateRef represents the embedded template which is used to instantiate the embedded views. These embedded views are linked to the template which is to be rendered.
4. ViewContainerRef is a container where one or more views can be attached. We can use createEmbeddedView() function to attach the embedded templates in the container.

constructor(

private templateRef: TemplateRef<any>,

private viewContainer: ViewContainerRef) { }

@Input() set appNot(condition: boolean) {

if (!condition) {

this.viewContainer.createEmbeddedView(this.templateRef);

} else {

this.viewContainer.clear(); }

}

1. Apply \*appNot in the template

<h1 \*appNot="true">True</h1><h1 \*appNot="false">False</h1>