Criando Diretivas no Angular

Além das diretivas padrões que o Angular traz, como o *ngStyle* ou o *ngIf*, o Angular nos permite criar nossas próprias diretivas. As diretivas customizadas nos permitem manipular a estilização de elementos do HTML, ouvir eventos da DOM, adicionar ou remover classes, dentre várias outras coisas. Essas diretivas customizadas podem ser comparadas a componentes, pois eles também nos permitem estilizar elementos do HTML e ouvir eventos da DOM, por exemplo. Mas, diferentemente dos componentes, as diretivas não possuem um template HTML e manipulam somente um elemento do HTML por vez.

Para criar uma diretiva, nós podemos utilizar o *angular-cli* para gerar automaticamente a diretiva. Nós podemos utilizar o comando:

```
$ ng generate directive minha-diretiva
```

ou:

```
$ ng g d minha-diretiva
```

Com esse código, nós teremos um arquivo chamado *minha-diretiva.directive.ts*, com essa estrutura

Agora, para podermos manipular o elemento em que essa diretiva está, devemos injetar uma propriedade do tipo *ElementRef*. O ElementRef permite com que o arquivo Typescript possa ter uma referência de algum elemento do HTML, como uma div, uma lista ou qualquer outro elemento HTML. A partir dele nós podemos acessar atributos, adicionar estilos ao elemento e várias outras coisas. Vamos, de início, fazer com que um elemento tenha a cor do seu texto mudada para verde.

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

@Directive({
  selector: '[appMinhaDiretiva]'
})
export class MinhaDiretivaDirective {

  constructor(
    private el: ElementRef
  ) {
    this.el.nativeElement.style.color = 'green'
  }
}
```

Agora, vamos adicionar esta diretiva a um elemento do HTML, como um parágrafo. Após adicionar, teste no navegador e veja o resultado.

Diretivas com valores dinâmicos

Nós podemos receber valores dinâmicos para algumas propriedades da nossa diretiva. Por exemplo, na diretiva que criamos anteriormente, podemos, ao invés de colocar uma cor fixa, podemos receber uma cor que é informada fora da diretiva. Para isso, nós iremos usar o *Input()*, visto anteriormente.

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

@Directive({
  selector: '[appMinhaDiretiva]'
})
export class MinhaDiretivaDirective {

  @Input()
  color: string = ''

  constructor(
    private el: ElementRef
  ) {
    this.el.nativeElement.style.color = this.color
  }
}
```

Agora, para passar um valor para essa propriedade, ela vai funcionar como um atributo no HTML. Basta apenas informar o valor desejado.

```
<p appMinhaDiretiva color="black">
  Parágrafo com a cor preta
</p>

<p appMinhaDiretiva [color]="'green'">
  Parágrafo com a cor verde
</p>

<p appMinhaDiretiva color="red">
  Parágrafo com a cor vermelha
</p>

<p appMinhaDiretiva color="#77ffe4">
  Parágrafo com a cor em hexadecimal
</p>
```

Nós podemos adicionar múltiplos inputs nas nossas diretivas. Cada input da diretiva funcionará como um atributo no HTML, como visto no exemplo acima.

Outras maneiras de estilizar com as diretivas

Como vimos anteriormente, diretamente com a referência do elemento que pegamos do HTML, nós podemos estilizar o elemento adicionando as propriedades CSS que desejarmos, adicionando classes entre várias outras coisas. Mas nós podemos utilizar um elemento do Angular chamado **Renderer2**. O Renderer2 nos permite adicionar estilos, classes e atributos a um elemento com uma sintaxe diferente. Assim como o ElementRef, nós injetaremos o Renderer2 na nossa diretiva.

```
import { Directive, Input, Renderer2 } from '@angular/core';

@Directive({
  selector: '[appMinhaDiretiva]'
})
export class MinhaDiretivaDirective {

  @Input()
  color: string = ''

  constructor(
    private el: ElementRef,
    private renderer: Renderer2
  ) {
    this.renderer.setStyle(this.el.nativeElement, 'background-color', 'purple')
    this.renderer.setStyle(this.el.nativeElement, 'color', '#ffffff')
    /*
      Colocamos primeiro o elemento em que ele tem que adicionar o estilo,
      a propriedade e depois o valor
    */

    this.renderer.addClass(this.el.nativeElement, 'classe1')
    this.renderer.addClass(this.el.nativeElement, 'classe2')
    /*
      Colocamos primeiro o elemento em que ele tem que adicionar classe e
      depois o nome da classe
    */

    this.renderer.setAttribute(this.el.nativeElement, 'id', 'el')
    /*
      Colocamos primeiro o elemento em que ele tem que adicionar o id e
      depois o nome do id
    */
  }
}
```

Ouvindo Eventos nas Diretivas

Nós podemos ouvir eventos dentro de uma diretiva utilizando os *HostListeners*. Os HostListener é um Decorator que adicionando em métodos da classe que nos permite ouvir eventos diretamente pela diretiva e adicionar um comportamento para ser executado quando o evento acontecer. Vamos fazer um exemplo para um evento de clique. Quando o elemento for clicado, aparecerá um *alert* informando que o elemento foi clicado.

```
import { Directive, Input, Renderer2, HostListener } from '@angular/core';

@Directive({
  selector: '[appMinhaDiretiva]'
})
export class MinhaDiretivaDirective {

  @Input()
  color: string = ''

  constructor(
    private el: ElementRef,
    private renderer: Renderer2
  ) {}

  @HostListener('click')
  showAlert() {
    alert('Você clicou neste elemento!')
  }
}
```

No HostListener nós informamos qual evento nós queremos ouvir, nesse caso, o elemento de click. Quando o evento acontecer, o método *showAlert()* será executado e mostrará um alert na tela.