Por que utilizar diretivas no Angular?

As diretivas no Angular permitem **manipular elementos do DOM de forma reutilizável**. Elas ajudam a criar **comportamentos personalizados** que podem ser aplicados a qualquer componente sem a necessidade de modificar diretamente o código HTML.

Vantagens de usar diretivas:

✅ **Reutilização** – Você pode criar um comportamento e aplicá-lo a vários elementos.
✅ **Organização do código** – Separação da lógica de manipulação do DOM do componente principal.
✅ **Facilidade de manutenção** – Se precisar alterar o comportamento, basta modificar a diretiva e não todos os componentes que a utilizam.
✅ **Boas práticas** – Segue o princípio **DRY (Don't Repeat Yourself)**, evitando código duplicado.

Como Criar uma Diretiva de Atributo no Angular

Passo 1: Criar a diretiva

Podemos criar uma diretiva usando o comando CLI:

ng generate directive nome-da-diretiva

Exemplo:

ng generate directive highlight

Isso criará um arquivo `highlight.directive.ts`.

Passo 2: Implementação de uma Diretiva Simples

Criamos uma diretiva que altera a cor do texto de um elemento ao passar o mouse sobre ele.

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

@Directive({
  selector: '[appHighlight]' // Nome da diretiva para ser usada no HTML
})
export class HighlightDirective {

  constructor(private el: ElementRef) {} // ElementRef permite manipular o elemento diretamente

  @HostListener('mouseenter') onMouseEnter() {
    this.highlight('yellow'); // Quando o mouse entra, muda a cor para amarelo
  }

  @HostListener('mouseleave') onMouseLeave() {
    this.highlight(null); // Quando o mouse sai, remove o destaque
  }

  private highlight(color: string | null) {
    this.el.nativeElement.style.backgroundColor = color; // Aplica a cor ao fundo do elemento
  }
}
```

Passo 3: Usando a Diretiva no HTML

Agora podemos aplicar essa diretiva a qualquer elemento HTML usando `[appHighlight]`:

```
<p appHighlight>Passe o mouse aqui para destacar!</p>
```

Quando o usuário passar o mouse sobre esse `<p>`, ele ficará com fundo amarelo.

Exemplos do que mais pode ser feito com diretivas

1️⃣ Alterar a cor do texto

```
@Directive({
  selector: '[appTextColor]'
})
export class TextColorDirective {
  constructor(private el: ElementRef) {
    this.el.nativeElement.style.color = 'blue';
  }
}
```

Uso no HTML:

```
<p appTextColor>Esse texto será azul!</p>
```

2️⃣ Adicionar uma borda ao passar o mouse

```
@Directive({
  selector: '[appBorder]'
})
export class BorderDirective {
  @HostListener('mouseenter') onMouseEnter() {
    this.el.nativeElement.style.border = '2px solid red';
  }

  @HostListener('mouseleave') onMouseLeave() {
    this.el.nativeElement.style.border = 'none';
  }

  constructor(private el: ElementRef) {}
}
```

Uso no HTML:

```
<div appBorder>Passe o mouse aqui para ver a borda vermelha!</div>
```

3️⃣ Desabilitar um botão baseado em uma condição

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

@Directive({
  selector: '[appDisable]'
})
export class DisableDirective implements OnInit {
  @Input() appDisable = false;

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

  ngOnInit() {
    if (this.appDisable) {
      this.renderer.setAttribute(this.el.nativeElement, 'disabled', 'true');
    }
  }
}
```

Uso no HTML:

```
<button [appDisable]="true">Esse botão estará desabilitado</button>
```

Conclusão

As diretivas são ferramentas poderosas no Angular para **modificar o comportamento e aparência de elementos HTML** de forma reutilizável e organizada. Elas ajudam a reduzir a complexidade dos componentes e a manter um código mais limpo. 🚀