📌 Diretiva com Seletor de Elemento: Modificando Imagens no Angular

No Angular, **diretivas** são usadas para modificar o comportamento de elementos HTML. Neste guia, vamos explorar **diretivas com seletor de elemento**, aplicadas especificamente para modificar imagens.

Com isso, conseguimos **automatizar estilos, adicionar efeitos e tornar o código mais organizado e reutilizável**.

1️⃣ Criando uma Diretiva Simples para Modificar Imagens

Vamos criar uma diretiva que **modifica imagens**, adicionando um **borda arredondada** e um **efeito de sombra**.

📌 Código da Diretiva

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

@Directive({
  selector: 'app-Rounded-Image' // Define que a diretiva será usada como um elemento HTML
})
export class RoundedImageDirective {
  constructor(private el: ElementRef, private renderer: Renderer2) {
    // Aplica um estilo automático na imagem quando a diretiva é usada
    this.renderer.setStyle(this.el.nativeElement, 'border-radius', '15px');
    this.renderer.setStyle(this.el.nativeElement, 'box-shadow', '5px 5px 10px rgba(0,0,0,0.3)');
  }
}
```

📌 Como Usar no HTML

Agora podemos simplesmente usar a **diretiva como um elemento**:

```
<app-Rounded-Image>
  <img src="assets/imagens/exemplo.jpg" width="200">
</app-Rounded-Image>
```

🔹 A imagem dentro da diretiva receberá **automaticamente uma borda arredondada e um efeito de sombra**.
🔹 Sem precisar adicionar manualmente classes CSS!

2️⃣ Personalizando a Aparência com Inputs

Podemos tornar a diretiva **mais flexível** permitindo que o usuário escolha o **raio da borda** e a **cor da sombra**.

📌 Código Atualizado com Inputs

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

@Directive({
  selector: 'app-Custom-Image'
})
export class CustomImageDirective {
  @Input() borderRadius: string = '10px'; // Valor padrão de borda arredondada
  @Input() shadowColor: string = 'rgba(0,0,0,0.3)'; // Valor padrão da sombra

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

  ngOnInit() {
    // Aplica os estilos na imagem com base nos valores recebidos como Input
    this.renderer.setStyle(this.el.nativeElement, 'border-radius',
this.borderRadius);
    this.renderer.setStyle(this.el.nativeElement, 'box-shadow', `5px 5px 10px
${this.shadowColor}`);
  }
}
```

📌 Como Usar no HTML

Agora podemos passar valores personalizados para a diretiva:

```
<app-Custom-Image borderRadius="20px" shadowColor="rgba(255,0,0,0.5)">
  <img src="assets/imagens/exemplo.jpg" width="200">
</app-Custom-Image>
```

✅ Aqui a imagem terá **bordas arredondadas de 20px** e **sombra vermelha**.

✅ Se nenhum valor for informado, serão usados os valores padrão (`10px` de borda e sombra preta `rgba(0,0,0,0.3)`).

3️⃣ Adicionando um Efeito ao Passar o Mouse

Agora, vamos modificar a diretiva para que **a imagem aumente levemente quando o usuário passar o mouse**.

📌 Código Atualizado

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

@Directive({
  selector: 'app-Hover-Image'
})
export class HoverImageDirective {
  constructor(private el: ElementRef, private renderer: Renderer2) {
    this.renderer.setStyle(this.el.nativeElement, 'transition', 'transform 0.3s
ease-in-out');
  }

  // Evento para aumentar a imagem ao passar o mouse
  @HostListener('mouseenter') onMouseEnter() {
    this.renderer.setStyle(this.el.nativeElement, 'transform', 'scale(1.1)');
  }

  // Evento para voltar ao tamanho normal quando o mouse sai
  @HostListener('mouseleave') onMouseLeave() {
    this.renderer.setStyle(this.el.nativeElement, 'transform', 'scale(1)');
  }
```

```
}
```

📌 Como Usar no HTML

```
<app-Hover-Image>
  <img src="assets/imagens/exemplo.jpg" width="200">
</app-Hover-Image>
```

✅ Agora, ao **passar o mouse sobre a imagem, ela aumentará levemente de tamanho**.

✅ Ao **tirar o mouse**, a imagem volta ao tamanho original.

4️⃣ Fazendo uma Imagem Preto e Branco e Colorida ao Passar o Mouse

Outra modificação legal é aplicar um **efeito de preto e branco**, deixando a imagem colorida apenas quando o mouse passar por cima.

📌 Código da Diretiva

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

@Directive({
  selector: 'app-Grayscale-Image'
})
export class GrayscaleImageDirective {
  constructor(private el: ElementRef, private renderer: Renderer2) {
    this.renderer.setStyle(this.el.nativeElement, 'filter', 'grayscale(100%)');
    this.renderer.setStyle(this.el.nativeElement, 'transition', 'filter 0.3s
ease-in-out');
  }

  @HostListener('mouseenter') onMouseEnter() {
    this.renderer.setStyle(this.el.nativeElement, 'filter', 'grayscale(0%)');
  }
```

```
  @HostListener('mouseleave') onMouseLeave() {
    this.renderer.setStyle(this.el.nativeElement, 'filter', 'grayscale(100%)');
  }
}
```

📌 Como Usar no HTML

```
<app-Grayscale-Image>
  <img src="assets/imagens/exemplo.jpg" width="200">
</app-Grayscale-Image>
```

✅ A imagem ficará **preto e branco por padrão**.
✅ Quando o usuário passar o mouse sobre a imagem, ela **ficará colorida**.

5️⃣ Tornando Todas as Imagens Redondas Automaticamente

Agora vamos criar uma diretiva que transforma **todas as imagens dentro dela em imagens redondas**.

📌 Código da Diretiva

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

@Directive({
  selector: 'app-Round-Images'
})
export class RoundImagesDirective {
  constructor(private el: ElementRef, private renderer: Renderer2) {
    const images = this.el.nativeElement.querySelectorAll('img'); // Seleciona
todas as imagens internas
    images.forEach((img: HTMLElement) => {
      this.renderer.setStyle(img, 'border-radius', '50%');
    });
  }
}
```

📌 Como Usar no HTML

```
<app-Round-Images>
  <img src="assets/imagens/usuario1.jpg" width="150">
  <img src="assets/imagens/usuario2.jpg" width="150">
</app-Round-Images>
```

✅ **Todas as imagens dentro do `<app-Round-Images>` se tornarão redondas automaticamente!** 🎯

🎯 Conclusão

Agora vimos **diferentes exemplos de diretivas para modificar imagens** no Angular. Com essas técnicas, conseguimos:

✅ **Aplicar estilos automaticamente**, como bordas e sombras.
✅ **Permitir personalização**, como cor da sombra e raio da borda.
✅ **Criar efeitos visuais interativos**, como aumentar a imagem ou remover preto e branco ao passar o mouse.
✅ **Facilitar a manutenção**, pois **uma única diretiva pode modificar várias imagens sem precisar alterar todo o HTML**.

🚀 Próximos Passos

🔹 Criar uma diretiva para **deixar todas as imagens em um formato polaroid**.
🔹 Criar uma **diretiva dinâmica** que permite **aplicar diferentes filtros de imagem** (exemplo: desfoque, brilho, contraste).

Caso precise de mais exemplos ou queira alguma funcionalidade específica, me avise! 🎨✨