📌 Como Passar Parâmetros para uma Diretiva de Atributo no Angular

No Angular, podemos passar parâmetros para uma **Diretiva de Atributo** usando o `@Input()`. Isso permite modificar o comportamento da diretiva de forma dinâmica.

✅ Exemplo 1: Mudando a Cor de Fundo de um Input

Aqui está um exemplo de diretiva que permite alterar a cor de fundo de um campo de entrada (`<input>`) ao receber foco.

📌 Código da Diretiva

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

@Directive({
  selector: '[appHighlight]'
})
export class HighlightDirective {

  // 🔹 Definindo uma propriedade que recebe um valor externo
  @Input() appHighlight: string = 'yellow';

  // 🔹 Definindo uma segunda propriedade para receber uma cor opcional do texto
  @Input() textColor: string = 'black';

  // 🔹 Vinculamos a propriedade ao estilo do elemento
  @HostBinding('style.backgroundColor') bgColor: string = '';
  @HostBinding('style.color') fontColor: string = '';

  // 🔹 Quando o usuário coloca o mouse sobre o elemento
  @HostListener('mouseover') onMouseOver() {
    this.bgColor = this.appHighlight;
    this.fontColor = this.textColor;
  }
```

```
  // 🔹 Quando o mouse sai do elemento
  @HostListener('mouseout') onMouseOut() {
    this.bgColor = 'transparent';
    this.fontColor = 'black';
  }
}
```

📌 Como Usar no HTML

<input appHighlight="blue" textColor="white" type="text" placeholder="Passe o mouse aqui!">

🔹 **O que acontece?**

- Quando o usuário passa o mouse sobre o `<input>`, o fundo fica azul e o texto branco.
- Quando o mouse sai do campo, volta para as cores padrões.

✅ Exemplo 2: Alterando a Largura do Elemento

Podemos criar uma diretiva que altera a largura de qualquer elemento HTML.

📌 Código da Diretiva

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

@Directive({
  selector: '[appWidth]'
})
export class WidthDirective {

  // 🔹 Definindo uma propriedade que recebe o valor da largura
  @Input() appWidth: string = '200px';

  // 🔹 Aplicando essa largura ao elemento
  @HostBinding('style.width') elementWidth: string = '';
```

```
  // 🔹 Quando o Angular inicializa o componente, aplicamos a largura recebida
  ngOnInit() {
    this.elementWidth = this.appWidth;
  }
}
```

📌 Como Usar no HTML

```
<div appWidth="300px" style="background-color: lightgray;">Esse div tem 300px de largura!</div>
```

🔹 **O que acontece?**

- O `div` receberá uma largura de **300px**, mas esse valor pode ser alterado no HTML.

✅ Exemplo 3: Criando uma Classe CSS Dinâmica

Podemos criar uma diretiva que adiciona/remova classes CSS dinamicamente.

📌 Código da Diretiva

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

@Directive({
  selector: '[appDynamicClass]'
})
export class DynamicClassDirective {

  // 🔹 Recebe um nome de classe como parâmetro
  @Input() appDynamicClass: string = '';

  // 🔹 Aplica essa classe ao elemento
  @HostBinding('class') elementClass: string = '';
```

```
  // 🔹 Quando o Angular inicializa o componente, aplicamos a classe recebida
  ngOnInit() {
    this.elementClass = this.appDynamicClass;
  }
}
```

📌 Como Usar no HTML

```
<p appDynamicClass="meu-texto-vermelho">Esse texto terá uma classe CSS dinâmica!</p>

<!-- No arquivo CSS -->
<style>
  .meu-texto-vermelho {
    color: red;
    font-weight: bold;
  }
</style>
```

🔹 **O que acontece?**

- O `<p>` receberá a classe `meu-texto-vermelho`, e o texto ficará **vermelho e em negrito**.

📌 Conclusão

- **Usamos `@Input()` para passar valores da diretiva pelo HTML**.
- **Com `@HostBinding()` podemos modificar estilos ou classes dinamicamente**.
- **O Angular nos permite criar diretivas reutilizáveis para diferentes elementos**.

Isso permite que criemos interfaces dinâmicas e reutilizáveis de maneira eficiente! 🚀