CSS Style Binding é um recurso poderoso do Angular que permite aplicar estilos diretamente a elementos HTML de forma dinâmica. Ele funciona vinculando propriedades CSS específicas a expressões no TypeScript. Isso permite que os estilos de um elemento mudem automaticamente com base no estado da aplicação.

Formas de usar CSS Style Binding

1. Vinculando propriedades CSS individuais

Você pode usar a sintaxe `[style.<property>]` para vincular uma propriedade CSS a uma expressão dinâmica no TypeScript.

Exemplo:

```
<button [style.color]="'red'">Texto Vermelho</button>
<button [style.backgroundColor]="'blue'">Fundo Azul</button>
<button [style.fontSize.px]="20">Fonte Tamanho 20px</button>
```

---

2. Vinculando várias propriedades CSS com um objeto

Atributo `[style]` permite passar um objeto com pares de chave-valor representando as propriedades e valores CSS.

Exemplo:

```
<div [style]="{ color: 'white', backgroundColor: 'black', padding: '10px' }">
  Texto com múltiplos estilos
</div>
```

---

3. Vinculando propriedades com unidades específicas

Para valores que requerem unidades (como `px`, `%`, `em`), você pode usar a sintaxe `[style.<property>.<unit>]`.

Exemplo:

```
<div [style.width.px]="150" [style.height.px]="100">
  Dimensão fixa (150px x 100px)
</div>
```

---

4. Vinculando estilos com variáveis ou métodos do TypeScript

Você pode usar variáveis ou métodos para definir estilos dinamicamente.

TypeScript:

```
export class AppComponent {
  dynamicColor = 'purple';
  dynamicSize = 24;

  getDynamicStyles() {
    return {
      color: this.dynamicColor,
      fontSize: `${this.dynamicSize}px`,
    };
  }
}
```

HTML:

```
<button [style.color]="dynamicColor">Texto Roxo</button>
<button [style.fontSize.px]="dynamicSize">Fonte Tamanho Dinâmico</button>
<button [style]="getDynamicStyles()">Estilos Dinâmicos</button>
```

---

5. Alterando estilos com eventos

Você pode alterar os estilos dinamicamente em resposta a eventos do usuário, como `click`, `hover`, etc.

TypeScript:

```
export class AppComponent {
  isHovered = false;

  getHoverStyles() {
```

```
    return this.isHovered
      ? { backgroundColor: 'lightgreen', fontWeight: 'bold' }
      : { backgroundColor: 'lightgray', fontWeight: 'normal' };
  }
}
```

HTML:

```
<div
  [style]="getHoverStyles()"
  (mouseover)="isHovered = true"
  (mouseout)="isHovered = false">
  Passe o mouse aqui!
</div>
```

---

6. Combinando CSS Style Binding com Classes

Embora Style Binding seja usado para propriedades específicas, você pode combiná-lo com **Class Binding** para controlar estilos complexos.

Exemplo:

```
<div [style.backgroundColor]="isActive ? 'green' : 'red'" [class]="isActive ? 'active-class' : 'inactive-class'">
  Estilo dinâmico com classe
</div>
```

Exemplo completo

TypeScript:

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.scss'],
})
export class AppComponent {
```

```
  width = 100;
  height = 50;
  bgColor = 'yellow';

  updateStyles() {
    this.width += 20;
    this.height += 10;
    this.bgColor = this.bgColor === 'yellow' ? 'orange' : 'yellow';
  }
}
```

HTML:

```
<div
  [style.width.px]="width"
  [style.height.px]="height"
  [style.backgroundColor]="bgColor">
  Caixa dinâmica
</div>
<button (click)="updateStyles()">Atualizar Estilos</button>
```

O que pode ser feito com Style Binding

1. **Alterar estilos com base em estados dinâmicos**:
    - Exibir diferentes cores ou tamanhos baseados em valores condicionais.
2. **Aplicar estilos dependentes de eventos**:
    - Alterar aparência de elementos com eventos como `click`, `mouseover`, `keydown`.
3. **Criar componentes altamente personalizáveis**:
    - Vincular propriedades de estilo a valores do TypeScript para tornar os componentes adaptáveis.
4. **Controlar dimensões responsivas**:
    - Ajustar propriedades como `width` e `height` dinamicamente.

5. **Combinar com animações ou transições CSS**:
    - Modificar estilos suavemente usando transições, controladas dinamicamente.

Melhores práticas

1. Use **Style Binding** para estilos dinâmicos individuais ou de pequena escala.
2. Prefira **Class Binding** para conjuntos de estilos mais complexos.
3. Mantenha os estilos reutilizáveis e centralizados no CSS/SCSS, sempre que possível.
4. Substitua objetos de estilo inteiros em vez de modificar diretamente suas propriedades, para melhor reatividade no Angular.