No Angular, você pode definir estilos diretamente no componente utilizando a propriedade `styles` no decorador `@Component`. Essa abordagem é útil para aplicar estilos específicos a um componente sem a necessidade de arquivos CSS ou SCSS separados.

Propriedade `styles`:

A propriedade `styles` aceita um array de strings, onde cada string contém as regras de estilo CSS que serão aplicadas ao componente. Esses estilos são encapsulados no componente, ou seja, não afetam outros componentes.

Exemplo básico:

Definindo estilos no componente:

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

@Component({
  selector: 'app-card',
  template: `
    <div class="card">
      <h1>{{ title }}</h1>
      <p>Este é um cartão estilizado com a propriedade "styles".</p>
    </div>
  `,
  styles: [
    `
      .card {
        border: 2px solid #4caf50;
        border-radius: 10px;
        padding: 16px;
        background-color: #f9f9f9;
        box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
      }

      .card h1 {
        color: #4caf50;
```

```
      font-size: 24px;
    }

    .card p {
      color: #333;
      font-size: 16px;
    }
  `,
 ],
})
export class CardComponent {
 title = 'Cartão Verde';
}
```

Encapsulamento de estilos:

Os estilos definidos na propriedade `styles` afetam apenas o componente atual. Isso ocorre porque o Angular adiciona atributos específicos para garantir que os estilos não vazem para outros componentes.

Exemplo com encapsulamento:

Componente `app-header` e `app-footer` têm estilos diferentes, mas ambos podem usar a classe `.title` sem conflitos:

```
@Component({
 selector: 'app-header',
 template: `<h1 class="title">Cabeçalho</h1>`,
 styles: [
  `
    .title {
      color: blue;
    }
  `,
 ],
})
export class HeaderComponent {}

@Component({
```

```
  selector: 'app-footer',
  template: `<h1 class="title">Rodapé</h1>`,
  styles: [
    `
      .title {
        color: green;
      }
    `,
  ],
})
export class FooterComponent {}
```

Utilizando variáveis CSS:

É possível aproveitar variáveis CSS nos estilos inline para criar temas dinâmicos:

```
@Component({
  selector: 'app-card',
  template: `
    <div class="card">
      <h1>{{ title }}</h1>
      <p>Este é um cartão com variáveis CSS.</p>
    </div>
  `,
  styles: [
    `
      :host {
        --primary-color: #ff5722;
      }

      .card {
        border: 2px solid var(--primary-color);
        background-color: #fff3e0;
        color: var(--primary-color);
        padding: 16px;
        border-radius: 8px;
      }
```

```
      .card h1 {
        font-size: 24px;
      }
    `,
  ],
})
export class CardComponent {
  title = 'Cartão Temático';
}
```

Animações CSS no componente:

Você também pode definir estilos para animações:

```
@Component({
  selector: 'app-loader',
  template: `
    <div class="spinner"></div>
  `,
  styles: [
    `
      .spinner {
        width: 40px;
        height: 40px;
        border: 4px solid #ccc;
        border-top-color: #ff5722;
        border-radius: 50%;
        animation: spin 1s linear infinite;
      }

      @keyframes spin {
        from {
          transform: rotate(0deg);
        }
        to {
          transform: rotate(360deg);
        }
      }
    `,
```

```
  ],
})
export class LoaderComponent {}
```

Múltiplos estilos:

Você pode definir múltiplas strings dentro do array `styles`:

```
@Component({
  selector: 'app-banner',
  template: `
    <div class="banner">
      <h1>{{ title }}</h1>
    </div>
  `,
  styles: [
    `
      .banner {
        background-color: #4caf50;
        color: white;
        text-align: center;
        padding: 20px;
      }
    `,
    `
      .banner h1 {
        font-size: 36px;
        font-weight: bold;
      }
    `,
  ],
})
export class BannerComponent {
  title = 'Bem-vindo ao Angular!';
}
```

Interações com variáveis do componente:

Embora não seja possível interpolar diretamente dentro da propriedade `styles`, você pode passar valores dinâmicos para o template utilizando `ngStyle` ou `ngClass`.

Exemplo:

```
@Component({
  selector: 'app-dynamic-card',
  template: `
    <div [ngStyle]="{ backgroundColor: bgColor, color: textColor }"
class="card">
      <h1>{{ title }}</h1>
      <p>Estilo dinâmico com ngStyle.</p>
    </div>
  `,
  styles: [
    `
      .card {
        padding: 16px;
        border-radius: 10px;
        border: 2px solid #333;
      }
    `,
  ],
})
export class DynamicCardComponent {
  bgColor = '#ffeb3b';
  textColor = '#000';
  title = 'Cartão Dinâmico';
}
```

Vantagens de usar `styles`:

1. Encapsulamento automático dos estilos.
2. Ideal para componentes pequenos ou estilos específicos.
3. Evita dependência de arquivos externos para estilos simples.

Quando evitar:

1. Quando os estilos são complexos ou muito extensos (prefira `styleUrls`).
2. Se você precisa reutilizar estilos entre vários componentes (use CSS global ou SCSS).

Com isso, você pode estilizar seus componentes de maneira encapsulada, prática e eficiente!