No Angular, você pode usar a propriedade `styleUrls` no decorador `@Component` para referenciar um ou mais arquivos externos de estilos CSS ou SCSS. Essa abordagem é ideal para organizar e reutilizar estilos em componentes, separando o código HTML, a lógica TypeScript e os estilos.

Sintaxe básica do `styleUrls`:

```
@Component({
  selector: 'app-card',
  templateUrl: './card.component.html',
  styleUrls: ['./card.component.css'], // Referência a um arquivo de estilos
})
export class CardComponent {}
```

No exemplo acima, os estilos contidos em `card.component.css` serão aplicados exclusivamente ao componente `CardComponent`.

Exemplo prático com um arquivo CSS:

Arquivo `card.component.css`:

```
.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;
}
```

Arquivo `card.component.ts`:

```
@Component({
  selector: 'app-card',
  templateUrl: './card.component.html',
  styleUrls: ['./card.component.css'],
})
export class CardComponent {
  title = 'Cartão Verde';
}
```

Arquivo `card.component.html`:

```
<div class="card">
  <h1>{{ title }}</h1>
  <p>Este é um cartão estilizado com CSS externo.</p>
</div>
```

Múltiplos arquivos de estilos:

A propriedade `styleUrls` aceita um array, permitindo referenciar vários arquivos de estilos.

```
@Component({
  selector: 'app-multi-style',
  templateUrl: './multi-style.component.html',
  styleUrls: ['./style1.css', './style2.css'], // Referenciando múltiplos arquivos
})
export class MultiStyleComponent {}
```

Utilizando SCSS:

Se você preferir usar SCSS, basta referenciar o arquivo `.scss` no `styleUrls`.

Arquivo `card.component.scss`:

```
$primary-color: #4caf50;

.card {
  border: 2px solid $primary-color;
  border-radius: 10px;
  padding: 16px;
```

```
  background-color: lighten($primary-color, 40%);
  box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);

  h1 {
    color: $primary-color;
    font-size: 24px;
  }

  p {
    color: darken($primary-color, 20%);
    font-size: 16px;
  }
}
```

Arquivo `card.component.ts`:

```
@Component({
  selector: 'app-card',
  templateUrl: './card.component.html',
  styleUrls: ['./card.component.scss'],
})
export class CardComponent {
  title = 'Cartão Verde com SCSS';
}
```

Interpolação dinâmica com `ngStyle` e `ngClass`:

Embora os estilos em `styleUrls` sejam estáticos, você pode complementar com estilos dinâmicos usando `ngStyle` ou `ngClass`.

Exemplo:

```
<div
  class="card"
  [ngStyle]="{ backgroundColor: bgColor, color: textColor }"
>
  <h1>{{ title }}</h1>
  <p>Estilo dinâmico com ngStyle.</p>
</div>
```

Arquivo CSS/SCSS:

```
.card {
```

```
  border: 2px solid #333;
  padding: 16px;
  border-radius: 10px;
}
```

Benefícios de `styleUrls`:

1. **Reutilização:** Permite usar os mesmos arquivos de estilo em diferentes componentes.
2. **Organização:** Mantém o código mais limpo e separado por responsabilidade.
3. **Escalabilidade:** Ideal para projetos maiores com estilos complexos.

Considerações importantes:

1. **Encapsulamento de estilos:** Os estilos de `styleUrls` são encapsulados automaticamente para o componente, não afetando outros componentes.
2. **Global styles:** Use o arquivo global `styles.css` ou `styles.scss` para estilos compartilhados por toda a aplicação (definido no `angular.json`).
3. **Performance:** Arquivos externos são otimizados e minificados durante o build.

Arquivo Global de Estilos:

Definição no `angular.json`:

```
"styles": [
  "src/styles.css"
]
```

Estilos globais definidos aqui afetarão toda a aplicação.

Com a propriedade `styleUrls`, você pode estruturar seus componentes Angular com clareza e eficiência, aproveitando toda a flexibilidade e organização de arquivos externos de estilos.