Template Variables no Angular

As **template variables** (variáveis de template) em Angular permitem que você acesse e interaja diretamente com elementos HTML ou diretivas em um template. Essas variáveis são declaradas usando `#` no template, e podem ser usadas para acessar propriedades, métodos e valores associados ao elemento HTML.

Exemplo básico de Template Variable

```
<!-- Criação de uma template variable chamada 'meuInput' para acessar o campo de texto -->
<input type="text" #meuInput placeholder="Digite algo" />

<!-- Exibição do valor atual do campo de texto usando a template variable -->
<p>Você digitou: {{meuInput.value}}</p>

<!-- Botão para exibir o valor no console -->
<button (click)="mostrarValor(meuInput.value)">Mostrar no console</button>
```

Comportamento:

1. A variável de template `#meuInput` permite acessar o elemento `<input>` diretamente.
2. O valor atual do campo é exibido dinamicamente no parágrafo.
3. O botão chama o método `mostrarValor` do componente e passa o valor do campo como argumento.

Exemplo com Interação de Elementos

```
<!-- Template variable para acessar o parágrafo e o botão -->
<input type="text" #campoTexto (input)="atualizarTexto(campoTexto.value)" placeholder="Digite algo" />
<p #paragrafoTexto>Texto atualizado aparecerá aqui.</p>
<button #meuBotao (click)="mostrarParagrafo(paragrafoTexto)">Clique para ver no console</button>
```

Componente Angular:

```
export class AppComponent {
  // Atualiza o texto do parágrafo diretamente
  atualizarTexto(valor: string) {
    const paragrafo = document.querySelector('#paragrafoTexto');
    if (paragrafo) paragrafo.textContent = valor;
  }

  // Mostra o conteúdo do parágrafo no console
  mostrarParagrafo(paragrafo: HTMLElement) {
    console.log('Texto no parágrafo:', paragrafo.textContent);
  }
}
```

Exemplo com Diretivas (e.g., ngIf e ngFor)

As template variables também podem ser usadas para acessar elementos dentro de estruturas de diretivas como `ngIf` e `ngFor`.

Usando `ngIf`

```
<div *ngIf="exibeTexto; else outroTemplate" #minhaDiv>
  Este texto será exibido se "exibeTexto" for verdadeiro.
</div>
<ng-template #outroTemplate>
  Texto alternativo se "exibeTexto" for falso.
</ng-template>

<button (click)="toggleTexto()">Alternar Texto</button>
```

Componente Angular:

```
export class AppComponent {
  exibeTexto = true;

  toggleTexto() {
    this.exibeTexto = !this.exibeTexto;
  }
```

```
}
```

Usando `ngFor`

```
<ul>
  <li *ngFor="let item of itens; let i = index" #minhaLista>
    {{ i + 1 }}. {{ item }}
  </li>
</ul>
<button (click)="logarPrimeiroItem(minhaLista)">Ver primeiro item</button>
```

Componente Angular:

```
export class AppComponent {
  itens = ['Item 1', 'Item 2', 'Item 3'];

  logarPrimeiroItem(lista: HTMLElement) {
    console.log('Primeiro item:', lista.textContent);
  }
}
```

Usando Template Variables com Change Detection

As template variables são úteis para interagir diretamente com o DOM, mas o Angular trabalha com o **ciclo de vida de detecção de mudanças (Change Detection)** para atualizar a interface.

Forçando uma Detecção de Mudanças

Às vezes, precisamos forçar o Angular a atualizar o DOM manualmente usando `ChangeDetectorRef`.

Exemplo:

```
<input type="text" #inputText placeholder="Digite algo" />
<p>Valor no campo: {{valor}}</p>
<button (click)="atualizar(inputText)">Atualizar manualmente</button>
```

Componente Angular:

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html'
})
export class AppComponent {
  valor: string = '';

  constructor(private cdRef: ChangeDetectorRef) {}

  atualizar(input: HTMLInputElement) {
    this.valor = input.value; // Atualiza o valor da variável
    this.cdRef.detectChanges(); // Força o Angular a processar a mudança
  }
}
```

O que mais pode ser feito com Template Variables

Acessar Métodos ou Propriedades de Elementos HTML

```
 <input type="file" #fileInput />
<button (click)="carregarArquivo(fileInput.files[0])">Upload</button>
```

1.

Trabalhar com Formulários

```
 <form #meuFormulario="ngForm"
(ngSubmit)="enviarFormulario(meuFormulario)">
  <input name="nome" ngModel required />
  <button type="submit">Enviar</button>
</form>
```

2.

Usar com Componentes Filhos

```
 <app-child #meuComponente></app-child>
```

```
<button
(click)="chamarMetodoDoComponente(meuComponente)">Executar</button>
 chamarMetodoDoComponente(componente: any) {
  componente.metodoQualquer();
}
```

3.

Alterar Estilos Dinamicamente

```
 <div #meuDiv [style.backgroundColor]="cor">Caixa colorida</div>
<button (click)="alterarCor(meuDiv)">Alterar cor</button>
 alterarCor(elemento: HTMLElement) {
  elemento.style.backgroundColor = 'blue';
}
```

4.

Dicas para Template Variables

- **Escopo restrito**: As template variables estão disponíveis apenas dentro do próprio template, e não podem ser acessadas no componente TypeScript.
- **Boas práticas**: Use template variables para acesso rápido e simples, mas prefira usar a vinculação de dados (`ngModel`) para sincronização entre a interface e o componente.

Template Variables são incrivelmente úteis para criar interações dinâmicas no Angular e facilitam o acesso direto aos elementos DOM sem depender exclusivamente de serviços ou manipulação direta.