A diretiva estrutural `*ngIf` no Angular é utilizada para adicionar ou remover elementos do DOM com base em uma condição booleana. Aqui está uma explicação detalhada e exemplos de tudo o que pode ser feito com `*ngIf`.

Funcionamento Básico de `*ngIf`

`*ngIf` verifica uma condição. Se for avaliada como `true`, o elemento é adicionado ao DOM; caso contrário, é removido.

Sintaxe básica:

```
<div *ngIf="condicao">
  Este conteúdo será exibido se `condicao` for verdadeira.
</div>
```

Exemplo 1: Condição Simples

```
<div *ngIf="isLoggedIn">
  Bem-vindo, usuário!
</div>
<div *ngIf="!isLoggedIn">
  Por favor, faça login.
</div>

export class AppComponent {
  isLoggedIn: boolean = false;
}
```

- Quando `isLoggedIn` for `true`, a mensagem de boas-vindas será exibida.
- Quando `isLoggedIn` for `false`, será exibido o pedido de login.

Exemplo 2: Usando `else`

`*ngIf` pode ser combinado com o bloco `else` para exibir um conteúdo alternativo.

```
<div *ngIf="isLoggedIn; else notLoggedIn">
  Bem-vindo, usuário!
</div>
<ng-template #notLoggedIn>
  <div>Por favor, faça login.</div>
</ng-template>
```

- Se `isLoggedIn` for `true`, o conteúdo "Bem-vindo, usuário!" será exibido.
- Se `isLoggedIn` for `false`, o bloco definido no `ng-template` será exibido.

Exemplo 3: Usando `then` e `else`

`*ngIf` também suporta o uso explícito de `then` e `else` para maior clareza.

```
<ng-container *ngIf="isLoggedIn; then loggedInTemplate; else
loggedOutTemplate"></ng-container>

<ng-template #loggedInTemplate>
  <div>Bem-vindo, usuário!</div>
</ng-template>

<ng-template #loggedOutTemplate>
  <div>Por favor, faça login.</div>
</ng-template>
```

- Essa abordagem separa completamente a lógica de exibição.

Exemplo 4: Usando `*ngIf` com Variáveis Locais

Você pode declarar uma variável local com o valor de `*ngIf`.

```
<div *ngIf="user as loggedInUser; else noUser">
  Bem-vindo, {{ loggedInUser.name }}!
</div>
<ng-template #noUser>
  <div>Nenhum usuário está logado.</div>
</ng-template>

export class AppComponent {
  user = { name: 'Jessica', loggedIn: true }; // ou null
}
```

- Se `user` for um objeto, ele será exibido como `loggedInUser`.
- Se `user` for `null`, o bloco `noUser` será exibido.

Exemplo 5: Uso Avançado com Múltiplas Condições

Você pode aninhar vários `*ngIf` para controlar exibições complexas.

```
<div *ngIf="isLoggedIn; else notLoggedIn">
  <div *ngIf="isAdmin">
    Bem-vindo, Administrador!
  </div>
  <div *ngIf="!isAdmin">
    Bem-vindo, Usuário!
  </div>
</div>

<ng-template #notLoggedIn>
  <div>Por favor, faça login.</div>
</ng-template>

export class AppComponent {
  isLoggedIn = true;
  isAdmin = false;
}
```

- Exibe mensagens diferentes com base no status de login e no tipo de usuário.

Exemplo 6: Alternância de Estilos e Classes

Embora `*ngIf` remova elementos do DOM, ele pode ser combinado com classes ou estilos dinâmicos.

```
<div *ngIf="isLoggedIn" class="logged-in">
  Você está logado.
</div>
<div *ngIf="!isLoggedIn" class="not-logged-in">
  Você não está logado.
</div>

.logged-in {
  color: green;
}
.not-logged-in {
  color: red;
}
```

Exemplo 7: Uso com Componentes

Você pode usar `*ngIf` diretamente em componentes para renderizá-los condicionalmente.

```
<app-dashboard *ngIf="isLoggedIn"></app-dashboard>
<app-login-form *ngIf="!isLoggedIn"></app-login-form>
```

- O componente `app-dashboard` será renderizado somente se `isLoggedIn` for `true`.
- O componente `app-login-form` será renderizado somente se `isLoggedIn` for `false`.

Resumo do que pode ser feito com `*ngIf`

1. **Condições Simples:** Exibe ou oculta elementos com base em um valor booleano.
2. **Bloco `else`:** Define conteúdo alternativo.
3. **Blocos `then` e `else`:** Permite maior clareza e flexibilidade.
4. **Variáveis Locais:** Declara variáveis locais para uso no template.
5. **Múltiplas Condições:** Controla exibições mais complexas.
6. **Com Classes e Estilos:** Altera estilos dinamicamente com base na condição.
7. **Componentes Dinâmicos:** Renderiza componentes condicionalmente.

Vantagens do `*ngIf`

- Remoção real de elementos do DOM, otimizando o desempenho.
- Controle de exibição baseado em condições simples ou complexas.
- Combinação com outros recursos do Angular, como `ng-template` e variáveis locais.

Esses exemplos e explicações cobrem praticamente tudo o que pode ser feito com `*ngIf`.