

Tipos de Diretivas

ngStyle

A diretiva `ngStyle` serve para alterar em tempo real valores de propriedades de CSS dentro de um elemento HTML ou componente. Essa diretiva receberá um objeto com as propriedades que serão mudadas dinamicamente. Para usar essa diretiva, adicione-o como atributo do seu elemento ou componente com a sintaxe `[ngStyle]="{}"` o objeto receberá as propriedades que serão mudadas que serão interpretadas pelo Angular e executadas visualmente para o usuário.

```
<p [ngStyle]="{ color: 'red', 'font-family': 'sans-serif', fontSize:'20px' }">  
  Este é um parágrafo com o ngStyle!  
</p>
```

ngClass

A diretiva `ngClass` serve para adicionar ou remover uma classe de algum elemento HTML ou componente que está sendo usado. O `ngClass`, como o `ngStyle`, pode receber um objeto como valor, onde os atributos desse objeto será o nome das classes e os valores alguma expressão booleana que se verdadeira, adicionará a classe, se falsa, não adicionará. Mas também pode receber diretamente um atributo do componente onde o valor desse atributo é uma string com o nome da classe. A sintaxe é a mesma do `ngStyle`. Basta colocar a diretiva como atributo `[ngClass]="{}"`.

```
<p [ngClass]="{ 'minha-classe': 5 > 2 }">  
  Este é um parágrafo com o ngClass!  
</p>
```

ngIf

O `ngIf` é uma diretiva estrutural que, a partir de uma condição booleana, pode ou não adicionar elementos a DOM, dinamizando ainda mais o site a interatividade do usuário.

Diferente das outras diretivas apresentadas, essa não ficará em volta de colchetes, mas sim acompanhada de um asterisco (*), sendo assim sua sintaxe ***ngIf="expressaoBooleana"**. Adicionando o ngIf a algum elemento, ele pode ou não ser criado dentro da DOM do seu website.

```
<p *ngIf="5 > 2">
  Este parágrafo apareceu na sua tela por causa da ngIf!
</p>

<p *ngIf="5 < 2">
  Este parágrafo não apareceu na sua tela por causa da ngIf!
</p>
```


ngSwitchCase

O ngSwitch também é uma diretiva estrutural e similar a estrutura **switch** das linguagens de programação. O diferencial dessa diretiva é que ela se divide em 3 partes: **ngSwitch**, **ngSwitchCase** e **ngSwitchDefault**, onde o ngSwitch recebe o atributo que será testado, ngSwitchCase receberá o valor que tornará o case verdadeiro e o ngSwitchDefault será um elemento padrão que será renderizado caso nenhum dos cases for verdadeiro. A sintaxe do ngSwitch é **[ngSwitch]="attribute"**. Enquanto do ngSwitchCase e do ngSwitchDefault recebem o asterisco na frente deles.

```
<div [ngSwitch]="atributo">
  <p *ngSwitchCase="valor1">
    Será que eu apareci?
  </p>
  <p *ngSwitchCase="valor2">
    Será que eu apareci?
  </p>
  <p *ngSwitchCase="valor3">
    Será que eu apareci?
  </p>
  <p *ngSwitchDefault>
    Se nenhum dos três ali aparecer, eu apareço!
  </p>
</div>
```

ngFor

Essa diretiva, sem dúvida, é uma das mais importantes. Com ela, você constrói laços de repetição para arrays que contêm dados que serão apresentados na tela do usuário. A sintaxe dessa diretiva é parecida com a do `ngIf`. No entanto, o jeito de usá-lo é diferente. Para usá-lo, a sintaxe dele é **`*ngFor="let value of values"`**. A variável *value* possui um único valor de alguma posição do array *values*, podendo acessar assim o seu valor. Se caso for necessário, você pode também acessar o índice atual do elemento do array. Para isso, basta adicionar um ponto-e vírgula (;) depois da primeira expressão e depois criar uma nova variável que receberá o índice, ficando da seguinte forma: **`*ngFor="let value of values; let idx = index"`**. Fazendo assim, o índice atual do elemento estará acessível.



```
<p *ngFor="let valor of array">
  {{ valor }}
</p>

<p *ngFor="let valor of array; let i = index">
  {{ id }} -> {{ valor }}
</p>
```