Diretivas

O que são as Diretivas?

- Uma forma de passar instruções para o template

```
<ul>
  <li *ngFor="let event of events">
    {{ course }}
  </li>
</ul>
```

"Itere por todos os itens do array **eventos**, e a cada iteração, atribua o valor do elemento atual à uma variável chamada **evento**. A cada iteração, replique também o elemento **<li>** com o valor da variável **evento**."

O que são as Diretivas?

- Componentes também são diretivas (diretivas com template)

```
...
  template: `
     <events></events>
  `,
...
```

“Instancie um componente do Tipo (classe) e renderize a view (template) desse componente nesse lugar especificado.”

Tipos de Diretivas

Estruturais	Utilizadas para modificar a estrutura do DOM e/ou do código HTML. Essas diretivas interagem diretamente com a view.	*ngFor *ngIf
De Atributo	Não modificam a estrutura do DOM, mas interagem diretamente com o elemento em que foram aplicadas	ng-class ng-style
Componentes	Diretivas com um template	<events> </events>

Condicional IF

- Executa alguma lógica de programação a partir de uma condicional

```
var events = [...];

if (events.length > 0) {
  // lógica de programação
} else {
  // outra lógica de programação
}
```

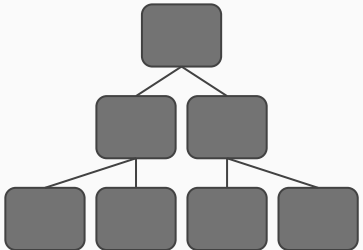
```
// Componente
let events = [...];

// Template
<div *ngIf="events.length > 0">
  Lista de eventos
  ...
</div>
```

- Não existe *ngElse

*ngIf x Hidden

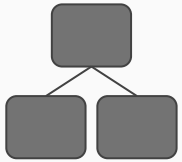
***ngIf**

- Mais custoso
- Mais seguro
- Recomendado para árvore de elementos grandes



[hidden]

- Menos custoso
- Menos seguro
- Recomendado para árvore de elementos pequenos



Condicional Switch-Case

- Executa alguma lógica de programação a partir de uma condicional

```
var pages = [‘home’, ‘profile’];

switch (pages) {
  case ‘home’: // lógica home
    break;
  default: // lógica padrão
}
```

```
// Componente
var pages = [‘home’, ‘profile’];

// Template
<div [ngSwitch]=“pages”>
  <p
*ngSwitchCase=“‘home’”>HOME</p>
  <p *ngSwitchDefault>PROFILE</p>
</div>
```

Loop For

- Os loops podem executar um bloco de código várias vezes.

```
var events= [‘event-1’, ‘event-2’, ...];

for (let i = 0; i < events.length; i++) {
  let event = events[i];
  console.log(event);
}
```

```
// Componente
let events= [‘event-1’, ‘event-2’,...];

// Template
<ul>
  <li *ngFor=“let event of events”>
    {{ event }}
  </li>
</ul>
```

ngClass

- Diretiva de atributo que visa facilitar o uso de diferentes classes

```
// Componente
myFavorite: boolean = false;

// Template
<i class="glyphicon"
[class.glyphicon-star-empty]="!myFavorite"
[class.glyphicon-star]="myFavorite"
(click)="onClick()"
></i>
```

```
// Componente
myFavorite: boolean = false;

//Template
<i class="glyphicon"
[ngClass]="{
  'glyphicon-star-empty': !myFavorite
  'glyphicon-star': myFavorite,
}"
(click)="onClick()"
></i>
```

ngStyle

- Diretiva de atributo que visa facilitar o uso de diferentes estilos

```
// Componente
active: boolean = false;

// Template
<button
[style.color]="active ? 'blue' : 'gray'"
(click)="changeActiveButton()"
>
My button
</button>
```

```
// Componente
active: boolean = false;

// Template
<button
[ngStyle]="{
  'color': active ? 'white' : 'black'
}"
(click)="changeActiveButton()"
>My button</button>
```

Operador Elvis

- Maneira segura de realizar a navegação entre os objetos
- Representado através do ponto de exclamação: ?
- Executa uma lógica parecida com o operador ternário

```
// Componente

event: any = {
    desc: 'Event description',
    responsable: null
    // responsable : {
        // name: 'João'
    // }
};
```

```
// Template

<p> Descrição: {{ event.desc }}
<p>
    Responsável:
    {{ event.responsable?.name }}
</p>
```

ng-content

- Maneira de passar ‘conteúdo’ entre os componentes
- Mais robusto se comparado ao Input Property
- Permite o uso de múltiplos seletores e objetos mais complexos

```
// Componente pai

<app-component>
    <p>
        Conteúdo da minha div.
    </p>
</app-component>
```

```
// Componente Filho

<h1>Título do meu componente</h1>

<div>
    <ng-content></ng-content>
</div>
```

Criando uma diretiva de atributo

- Através do uso do ElementRef e do Renderer
  - ElementRef: Faz referência ao elemento do DOM
  - Renderer: Responsável por realizar manipulações no DOM
- Utilizando HostListener e HostBinding
  - HostListener
    - Tipo de metadado
    - “Escuta” os eventos que acontecem no elemento hospedeiro da nossa diretiva
  - HostBinding: Permite a associação de um atributo/classe do html para uma variável

Criando uma diretiva de estrutura

- Cada diretiva de estrutura é um input property
- Uso de duas classes do Angular:
  - TemplateRef: Faz referência ao template
  - ViewContainerRef: Faz referência ao conteúdo do container/template
- Dica:
  - Referência: https://angular.io/api/common/NgIf

Observações:

- Questionário: https://forms.gle/31931ukERT9BTodz7
- Código fonte: https://github.com/GabriellBP/Angular-course
- Dúvidas:
  - gabriel.pereira@edge.ufal.br
  - lucas.amorim@edge.ufal.br

Obrigado!