Projeto de Interfaces WEB

Introdução ao Angular
Aula 02

- Agenda
  - Diretivas
    - *ngFor
    - *ngIf
  - @Input
  - @Output e EventEmitter
  - Diretivas personalizadas
  - NgClass e NgStyle
  - Pipes

Diretivas

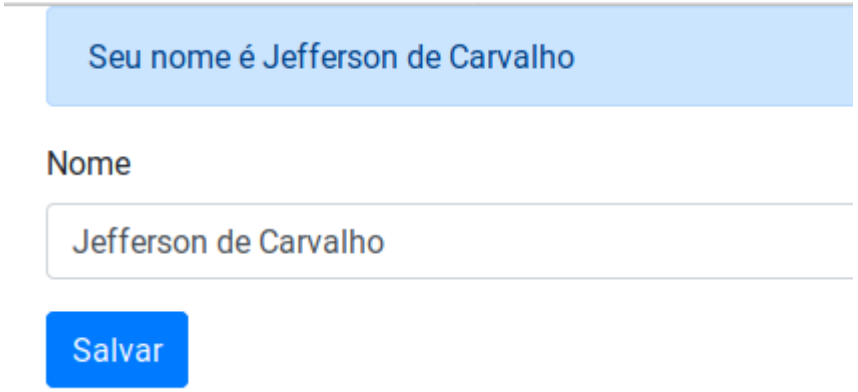
- Instruções passada no template.
  - **Componentes**
    - Código html atrelado ao componente (.ts)
    - <app-hello></app-hello>, por exemplo (instrução)
  - **Diretivas Estruturais**
    - Altera o DOM da página
    - <h1 ***ngIf**=”autenticado”> Olá {{usuario}}!</h1>
  - **Diretivas de Atributos**
    - Modifica o comportamento ou a aparência do elemento **sem** alterar o DOM
    - <h2 [style.color]=”red”> Olá {{usuario}}</h2>

Diretivas - DOM

- https://tableless.com.br/entendendo-o-dom-document-object-model/
- “O DOM (Document Object Model) é uma interface que representa como os documentos HTML e XML são lidos pelo seu browser. Após o browser ler seu documento HTML, ele cria um objeto que faz uma representação estruturada do seu documento e define meios de como essa estrutura pode ser acessada. Nós podemos acessar e manipular o DOM com JavaScript, é a forma mais fácil e usada.”

Diretivas ngIf e Hidden

- No exemplo anterior, vamos mostrar o alerta apenas quando o usuário clicar no botão salvar.



Diretivas ngIf e Hidden

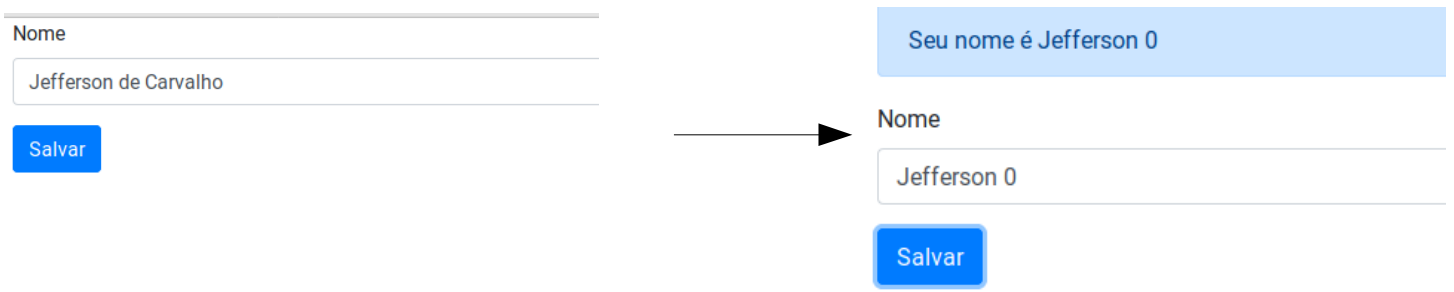
- No app.component.ts
  - Adicionar uma variável booleana “clicou”

```
export class AppComponent {
  nome = 'Jefferson de Carvalho';
  contador = 0;
  clicou = false;

  salvar(nome:string){
    console.log(`Salvando ${this.nome}`);
    this.nome = "Jefferson " + this.contador;
    this.contador++;
    this.clicou = true;
  }

  mudar(event:any){
    this.nome = event.target.value;
  }
}
```

Diretivas ngIf e Hidden

- No app.component.html (diretiva estrutural)
  - Inserir a diretiva *ngIf na div do alerta:

```
<div class="alert alert-primary" role="alert" *ngIf="clicou">
    Seu nome é {{nome}}
</div>
```



Diretivas ngIf e Hidden

- https://angular.io/api/common/NgIf
- “A structural directive that conditionally **includes** a **template** based on the **value** of an expression coerced to Boolean. When the expression evaluates to true, Angular renders the template provided in a **then** clause, and when false or null, Angular renders the template provided in an optional **else** clause. The default template for the else clause is blank”.

Diretivas ngIf e Hidden

- Exercício
  - Use um eventBiding (focus) para que quando o input fique em foco, o alerta desapareça!
  - Resposta no próximo slide. Tente não ver...

Diretivas ngIf e Hidden

- Resposta:

```
<input type="text" class="form-control"
        [(ngModel)]="nome"
        (focus)="clicou=false">
```

Diretivas ngIf e Hidden

- Há também a possibilidade de usar a diretiva de atributo, não alterando assim o DOM.

```
<div class="alert alert-primary" role="alert" [hidden]="!clicou">
    Seu nome é {{nome}}
</div>
```

Diretiva ngFor

- Vamos adicionar uma lista de pessoas no nosso HTML.
- Inicialmente, vamos até a página de componentes do bootstrap.
- Procure pelo componente “**cards**” e copie seu exemplo.

```
<div class="card" style="width: 18rem;">
  <img src="..." class="card-img-top" alt="...">
  <div class="card-body">
    <h5 class="card-title">Card title</h5>
    <p class="card-text">Some quick example text to build on the card title and make up the bulk of the card's
content.</p>
    <a href="#" class="btn btn-primary">Go somewhere</a>
  </div>
</div>
```

Diretiva ngFor

- No app.component.html

```
…
<button type="button" class="btn btn-primary" (click)="salvar()"
[disabled]="nome.length==0">Salvar</button>

<div class="card" style="width: 18rem;">
    <img src="https://randomuser.me/api/portraits/women/1.jpg" class="card-img-top" alt="...">
    <div class="card-body">
      <h5 class="card-title">Card title</h5>
      <p class="card-text">Some quick example text to build on the card title and make up the bulk of the
        card's content.</p>
      <a href="#" class="btn btn-primary">Go somewhere</a>
    </div>
  </div>
```

Diretiva ngFor

- Melhorando: (apagando informações desnecessárias)
- Usando também a class=”row”, do bootstrap.

```
...
<button type="button" class="btn btn-primary" (click)="salvar()"
[disabled]="nome.length==0">Salvar</button>

  <div class="row" style="margin-top: 20px">
    <div class="col-2">

      <div class="card" >
        <img src="https://randomuser.me/api/portraits/women/1.jpg" class="card-img-top" alt="...">
        <div class="card-body">
          Maria
        </div>
      </div>

    </div>
  </div>
```

Diretiva ngFor

- Em app.component.ts, criar o array de pessoas.

```
export class AppComponent {
  nome = 'Jefferson de Carvalho';
  contador = 0;
  clicou = false;
  pessoas = [];

  salvar(nome:string){
    console.log(`Salvando ${this.nome}`);
    this.nome = "Jefferson " + this.contador;
    this.contador++;
    this.clicou = true;
    this.pessoas.push(this.nome);
  }
```

Diretiva ngFor

- Em app.component.html, usar o ngFor.

```
<div class="row" style="margin-top: 20px">
   <div class="col-2" *ngFor="let pessoa of pessoas">
    <div class="card">
     <img src="https://randomuser.me/api/portraits/women/1.jpg"
          class="card-img-top" alt="...">
     <div class="card-body">
      {{pessoa}}
     </div>
    </div>
   </div>
</div>
```

Diretiva ngFor

- Finalizando o código (app.component.ts)

```
export class AppComponent {
  nome = 'Jefferson de Carvalho';
  contador = 0;
  clicou = false;
  pessoas = [];

  salvar(nome:string){
    console.log(`Salvando ${this.nome}`)
    this.contador++;
    this.clicou = true;
    this.pessoas.push({nome:this.nome,
                       id:this.contador});
  }
...
```

Diretiva ngFor

- Finalizando o código (app.component.html)

```
<div class="row" style="margin-top: 20px">
   <div class="col-2" *ngFor="let pessoa of pessoas">
     <div class="card">
       <img src="https://randomuser.me/api/portraits/women/{{pessoa.id}}.jpg"
            class="card-img-top" alt="...">
       <div class="card-body">
         {{pessoa.nome}}
       </div>
     </div>
   </div>
</div>
```

Binding @Input

- Como passar dados para propriedade customizadas dos nossos componentes.
- Vamos criar um novo componente:
  - Abra o terminal do VS
  - ng g c pessoa-card

Binding @Input

- Em pessoa-card html:
  - Cole o código do card

```
<div class="card">
  <img src="https://randomuser.me/api/portraits/women/{{pessoa.id}}.jpg"
       class="card-img-top" alt="...">
  <div class="card-body">
    {{pessoa.nome}}
  </div>
</div>
```

Binding @Input

- Agora chame o seletor de pessoas dentro de app.component.html

```
<div class="row" style="margin-top: 20px">
   <div class="col-2" *ngFor="let pessoa of pessoas">
     <app-pessoa-card></app-pessoa-card>
   </div>
 </div>
```

Binding @Input

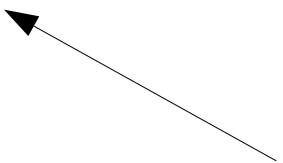
- Deve-se agora fazer o “binding” entre a variável pessoa de app.component e alguma variável dentro do novo componente.
- No novo componente, criaremos:

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

@Component({
  selector: 'app-pessoa-card',
  templateUrl: './pessoa-card.component.html',
  styleUrls: ['./pessoa-card.component.css']
})
export class PessoaCardComponent {

  @Input() pessoa:any; //NOTE QUE O INPUT SERVE PARA TORNAR A VARIÁVEL VISÍVEL PARA OUTROS COMPONENTES..

}
```

Binding @Input

- Voltando ao app.component.html:

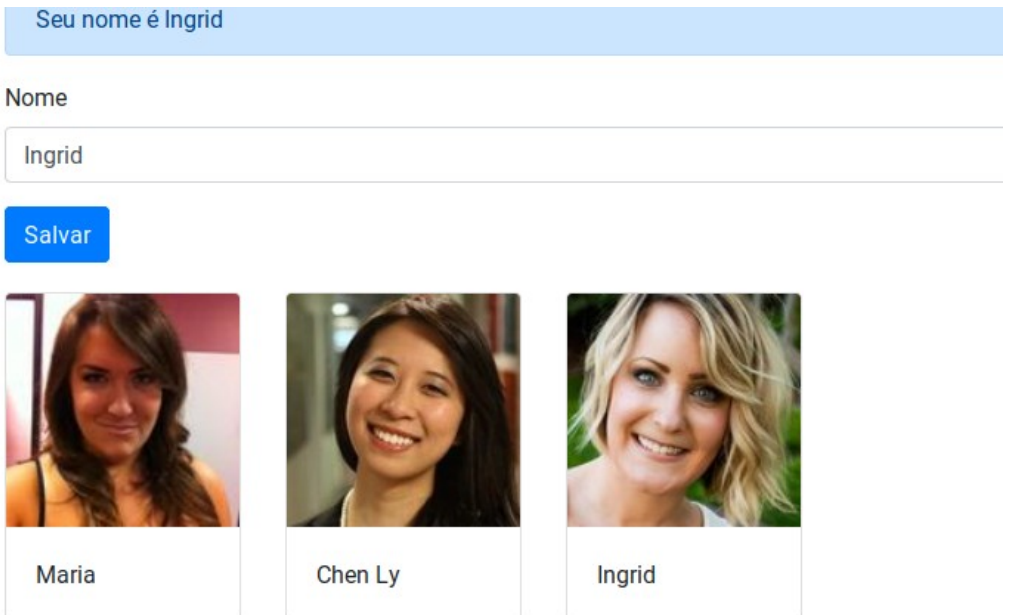
```
<div class="row" style="margin-top: 20px">
   <div class="col-2" *ngFor="let p of pessoas">
     <app-pessoa-card [pessoa]="p"></app-pessoa-card>
   </div>
</div>
```



Objeto sendo passado por parâmetro.

Binding @Input

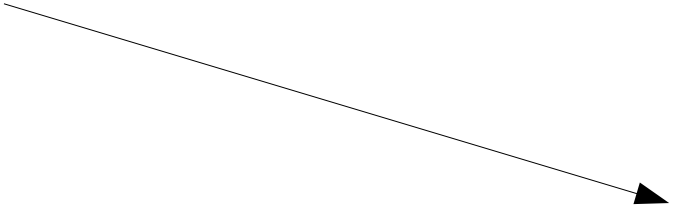
- Teste novamente a página:



Binding @Input

- É possível também criar um alias no input para o nome da propriedade. Só não esqueça de mudar também no seletor.

```
@Input('obj-pessoa') pessoa:any;
```



```
<app-pessoa-card [obj-pessoa]="p"></app-pessoa-card>
```

Binding @Output e EventEmitter

- Binding de eventos.
- Comunicação entre componentes, e duas vias.
- Objetivo: criar um novo componente para o formulário que irá passar o valor de “nome” para app.component. Ou seja, alimentar o seu vetor de pessoas.
- Vamos criar um novo componente:
  - ng g c pessoa-form --spec=false

Binding @Output e EventEmitter

- Em pessoa-form html, copie e cole o código do label, input e button do app.component.html. Não esqueça de chamar o seletor de pessoa-form em app.componente.html

```
<div class="alert alert-primary" role="alert" [hidden]="!clicou">
    Seu nome é {{nome}}
  </div>

  <div class="form-group">
    <label>Nome</label>
    <input type="text" class="form-control" [(ngModel)]="nome" (focus)="clicou=false">
  </div>
  <button type="button" class="btn btn-primary" (click)="salvar()"
[disabled]="nome.length==0">Salvar</button>
```

Binding @Output e EventEmitter

- pessoa-form.ts

```
import { Component, OnInit, Output, EventEmitter } from '@angular/core';
..

export class PessoaFormComponent   {

  nome = 'Jefferson de Carvalho';
  contador = 0;
  clicou = false;

  @Output('criado') pessoaSalva = new EventEmitter();

  salvar(){

    this.contador++;
    this.clicou = true;
    const pessoa = {nome:this.nome,
                    id:this.contador};

    this.pessoaSalva.emit(pessoa);
  }
}
```

Será visto como um evento.

Disparando o evento e passando um objeto
Para quem chamar.

Binding @Output e EventEmitter

- App.component.html

```
<div class="container">
  <app-pessoa-form (criado) = 'aoSalvar($event)'></app-pessoa-form>
  <div class="row" style="margin-top: 20px">
    <div class="col-2" *ngFor="let p of pessoas">
      <app-pessoa-card [obj-pessoa]="p"></app-pessoa-card>
    </div>
  </div>
</div>
```

Método que deve ser implementado em App.component.ts

Binding @Output e EventEmitter

- App.component.html

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  pessoas = [];

  aoSalvar(pessoa){
    this.pessoas.push(pessoa);
  }
}
```

Recebendo o objeto pessoa.

Binding @Output e EventEmitter

- Como funciona?
  - Em pessoa-form.html, (click)="salvar() chama o método salvar de seu componente.
  - pessoa-form.ts faz:
    - **@Output('criado') pessoaSalva = new EventEmitter();** Onde é criado um objeto “pessoaSalva” do tipo EVENTO!
    - No método “salvar”, um objeto “pessoa” é criado normalmente com id e nome. Logo depois, a chamada **this.pessoaSalva.emit(pessoa);** faz com o objeto “pessoaSalva” se ligue com app.component.html, pssando como parâmetro uma pessoa.
  - Em app.component.html, **<app-pessoa-form (criado) = 'aoSalvar($event)'></app-pessoa-form>** fica esperando ser chamado pelo emit, explicado anteriormente. Note que o objeto “pessoa” passado pelo emite é capturado pelo $event e repassado ao método “aoSalvar”.
  - Finalmente, o objeto “pessoa” recebido no event é passado para app.component.ts via método “aoSalvar”, onde é adicionado ao array de **pessoas[],** sendo assim visto por outros componentes filhos, como o pessoa-card, por exemplo.

CSS

- Como adicionar estilos CSS nos nossos componentes.
- Meta dados styleUrls
- Basta adicionar algo ao template indicado no metadados.
- Ou, você pode usar template literals dentro do próprio componente.

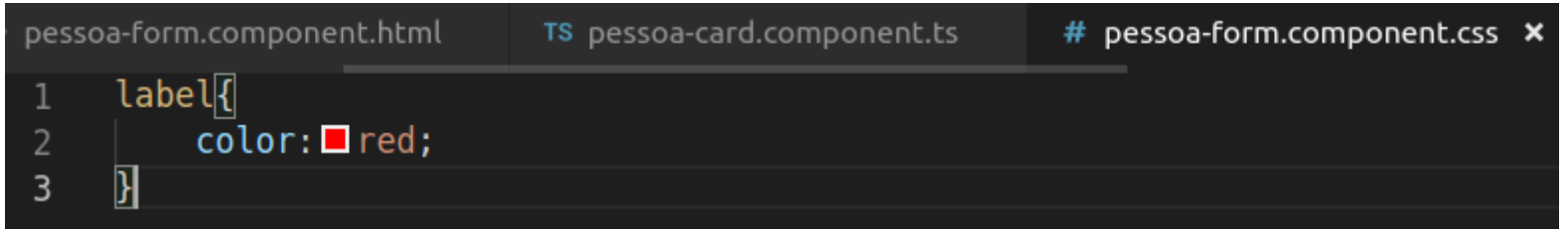
CSS

- Mexendo dentro do metadados:

```
@Component({
  selector: 'app-pessoa-card',
  templateUrl: './pessoa-card.component.html',
  //styleUrls: ['./pessoa-card.component.css']
  styles: [`
    .card-body{
      text-transform: uppercase;
      color:blue;
    }
  `]
})
```

CSS

- No arquivo css:



CSS com ngStyle

- Como adicionar estilos de forma dinâmica, através do ngStyle.
- https://angular.io/api/common/NgStyle
- Vamos trabalhar no pessoa-card.

CSS com ngStyle

- pessoa-card.html

```
<div class="card" [ngStyle]="getEstilosCard()">
  <img src="https://randomuser.me/api/portraits/women/{{pessoa.id}}.jpg"
       class="card-img-top" alt="...">
  <div class="card-body">
    {{pessoa.nome}}
  </div>
</div>
```

Chamada da diretiva.

CSS com ngStyle

- pessoa-card.ts

```
export class PessoaCardComponent {

  @Input('obj-pessoa') pessoa:any;

  getEstilosCard(){
    return {
      'border-width.px':this.pessoa.id,
      backgroundColor:this.pessoa.id%2==0?'lightblue':'lightgreen'
    };
  }

}
```

CSS com ngClass

- Classes dinâmicas.
- The CSS classes are updated as follows, depending on the type of the expression evaluation:
  - string - the CSS classes listed in the string (space delimited) are added,
  - Array - the CSS classes declared as Array elements are added,
  - Object - keys are CSS classes that get added when the expression given in the value evaluates to a truthy value, otherwise they are removed.
- Vamos usar o componente “badge”, do bootstrap. Caso o usuário seja adm, mostraremos um tipo de badge diferente.

CSS com ngClass

```
<some-element [ngClass]="'first second'">...</some-element>

<some-element [ngClass]="['first', 'second']">...</some-element>

<some-element [ngClass]="{'first': true, 'second': true, 'third': false}">...</some-
element>

<some-element [ngClass]="stringExp|arrayExp|objExp">...</some-element>

<some-element [ngClass]="{'class1 class2 class3' : true}">...</some-element>
```

CSS com ngClass

- pessoa.card.html

```
<div class="card" [ngStyle]="getEstilosCard()">
  <img src="https://randomuser.me/api/portraits/women/{{pessoa.id}}.jpg"
       class="card-img-top" alt="...">
  <div class="card-body">
    <span class="badge" [ngClass]=
                                "{'badge-danger':isAdmin(),
                                  'badge-primary':!isAdmin()}">
      {{pessoa.nome}}
    </span>
  </div>
</div>
```

CSS com ngClass

- pessoa-card.ts

```
export class PessoaCardComponent {

  @Input('obj-pessoa') pessoa:any;

  isAdmin(){
    return this.pessoa.nome.startsWith('J');
  }

  getEstilosCard(){
    return {
      'border-width.px':this.pessoa.id,
      backgroundColor:this.pessoa.id%2==0?'lightblue':'lightgreen'
    };
  }

}
```

Diretivas Personalizadas

- Vamos criar diretivas personalizadas a partir de classes da nossa própria autoria.
- Objetivo: inicialmente, uma diretiva personalizada para mudar a cor do input.
- Execute o comando:
  - **ng g d campo-colorido --spec==false;**

Diretivas Personalizadas

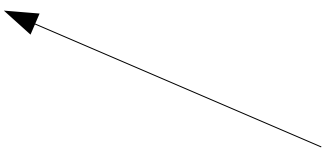
- “Injetando” objeto no construtor da diretiva:

```
import { Directive, ElementRef, Renderer2 } from '@angular/core';

@Directive({
  selector: '[appCampoColorido]'
})
export class CampoColoridoDirective {

  constructor(
    private elementRef:ElementRef, //elemento hospedeiro
    private renderer:Renderer2 //abstração de renderização
  ) {
    this.renderer.setStyle(this.elementRef.nativeElement,
      'background-color','yellow');
  }

}
```

Diretivas Personalizadas

- Aplique a diretiva no input

```
...
<div class="form-group">
        <label>Nome</label>
        <input type="text"
                class="form-control"
                #nomeInput
                (focus)="clicou=false"
                appCampoColorido>
</div>
...
```



Diretivas Personalizadas

- @HostListener
  - Vamos mudar a cor do input apenas quando o ele ganhar o foco. Saindo de foco, ele volta pra cor “normal”. Adicione o seguinte método ao que já existe.

```
@HostListener('focus') aoGanharFoco(){
    this.renderer.setStyle(this.elementRef.nativeElement,
      'background-color','yellow');
}
```

Diretivas Personalizadas

- @HostListener
  - E agora, iremos voltar pro normal quando perde o foco. Adicione o seguinte método:

```
@HostListener('blur') aoPerderFoco(){
    this.renderer.setStyle(this.elementRef.nativeElement,
      'background-color','transparent');
  }
```

Diretivas Personalizadas

- @HostBiding
  - Forma mais simples. Comente o código anterior e substitua por esse:

```
export class CampoColoridoDirective {

  @HostBinding('style.backgroundColor') corDeFundo:string;

  @HostListener('focus') aoGanharFoco(){
    this.corDeFundo = 'yellow';
  }

  @HostListener('blur') aoPerderFoco(){
    this.corDeFundo = 'transparent';
  }

}
```

Já funciona sem modificar o HTML

Diretivas Personalizadas

- Exercício:
  - Como, através do @Input, podemos entrar com uma cor para o foco via seletor?
  - Crie uma variável corDeEntrada do tipo @Input que receba uma cor de entrada via HTML.

Diretivas Personalizadas

- **Resposta.**
- Property biding (no TS). Ou seja, vamos entrar com a cor via input, no seletor. Apenas uma modificação do exemplo anterior.

```
  ...
  @Input() cor:string = 'gray';

  @HostBinding('style.backgroundColor') corDeFundo:string;

  @HostListener('focus') aoGanharFoco(){
    this.corDeFundo = this.cor;
  }
....
```

Diretivas Personalizadas

- Property biding (no HTML)

```
<input type="text"
        class="form-control"
        #nomeInput
        (focus)="clicou=false"
        appCampoColorido [cor]="'red'">
```

Diretivas Personalizadas

- Exportando a diretiva...

TS

```
@Directive({
  selector: '[appCampoColorido]',
  exportAs: 'campoColorido'
})
```

1 - Exporte a diretiva como uma variável chamada “campoColorido”.

HTML

```
<div class="form-group">
      <label>Nome</label>
      <input type="text"
            class="form-control"
            #nomeInput
            (focus)="clicou=false"
            appCampoColorido [cor]="'red'"
            #campo="campoColorido">
</div>

   <button type="button" class="btn btn-primary"
(click)="salvar(nomeInput.value)">Salvar</button>

   <button type="button" class="btn btn-primary"
(click)="campo.aoGanharFoco()">Colorir</button>
   <button type="button" class="btn
btn-primary">Descolorir</button>
```

2 - Inicialize o objeto e o atribua a #campo

3 - Chame os métodos da diretiva via variável campo.

Pipes |

- Criando saídas personalizadas para as interpolações.
- Crie as seguintes variáveis em app.component.ts:

```
cliente = "Fulano de Tal";
dataAniversario = new Date(1982,3,2);
preco = 255.89
```

Pipes |

- No app.component.html

```
<h1>{{cliente | uppercase}}</h1>
<h1>{{dataAniversario | date:'dd/MM/yyyy'}}</h1>
<h1>{{preco | currency:'BRL':'':'0.2':'pt'}}</h1>
<h1>{{preco | currency:'BRL':'symbol':'0.2':'pt'}}</h1>
<h1>{{preco | currency:'BRL':'code':'0.2':'pt'}}</h1>
<h1>{{preco | number:'':'pt'}}</h1>
```

Pipes |

- Só funciona se importar no app.module.ts, a localização :

```
import localePt from '@angular/common/locales/pt';
import { registerLocaleData } from '@angular/common';
registerLocaleData(localePt);

@NgModule({
……
```