

Binding

Prof. Me. Gustavo Custodio

gustavo.custodio@anhembi.br

Em computação, lidamos com dados

Nos nossos *apps*, os dados vão viver **sempre** nos nossos arquivos TS.

Como fazemos a **comunicação** destes dados para a interface?

E ainda, como fazemos a interface **sinalizar** que um dado precisa ser alterado?

Binding

O que significa?

- **Víncular**
- Conectar
- Unir
- Amarrar
- Ligar
- Prender
- Colar
- Grudar



No nosso caso

A forma como vinculamos a interface com código

No Angular

Temos 4 tipos de binding

- Template Binding (Interpolação)
- Property Binding
- Event Binding
- Two-Way Data Binding

Exemplo

- No arquivo **home.page.ts**:

```
@Component(...)  
export class HomePage {  
  public currentValue = 0;  
  
  public increment(){  
    this.currentValue++;  
  }  
}
```

- Os dados são **propriedades** da classe
- As funcionalidades são **métodos** da classe
- Tudo que a interface consegue ver, deve ser `public`

Interpolação

Template Binding

Interpolação

Usamos quando queremos que algum valor apareça como texto no meio do HTML:

```
<h1>  
  Você tem {{ currentValue }} mensagens não lidas.  
</h1>
```

Mostrando dados complexos

- O valor sempre será convertido para **texto**
- Podemos escrever expressões mais complicadas se quisermos

```
<p>  
  0 seu saldo atual é {{ currentValue >= 0 ? 'positivo' : 'negativo' }}  
</p>
```

Property Binding

Property Binding

Usamos quando queremos determinar dinamicamente o valor de um **atributo/propriedade** do HTML.

```
<ion-progress-bar color="primary" value="0.5"></ion-progress-bar>  
<ion-progress-bar color="primary" [value]="currentValue"></ion-progress-bar>
```

Seria como o seguinte, se isso fosse válido:

```
<ion-progress-bar color="primary" value="{{ currentValue }}"></ion-progress-bar>
```

Também podemos fazer expressões mais complexas

```
<ion-progress-bar color="primary" [value]="currentValue / 100">  
</ion-progress-bar>
```

```
<h1>  
0 seu saldo atual é  
  <ion-text [color]="currentValue >= 0 ? 'success' : 'danger'">  
    {{ currentValue >= 0 ? 'positivo' : 'negativo' }}  
  </ion-text>  
</h1>
```

Começamos repetir código, depois veremos outras formas mais inteligentes de fazer isso 😊

Event Binding

Event Binding

Usamos quando queremos reagir a algum evento do componente, como um clique por exemplo.

```
<ion-button (click)="increment()">  
  Você clicou {{currentValue}} vezes  
</ion-button>
```

Precisa ser uma função?

Não! Mas é uma boa prática

```
<ion-button (click)="currentValue=currentValue+1">  
  Você clicou {{currentValue}} vezes  
</ion-button>  
  
<ion-button (click)="currentValue = 0">  
  Resetar  
</ion-button>
```


Criando um app

Faça um app, com a interface que preferir, que:

- Mostre o valor atual de um contador
- Mostre o valor máximo que o contador já atingiu
- Tenha um botão para aumentar este contador em 1
- Tenha um botão para reduzir este contador em 1
- Tenha um botão para resetar para 0
- Obs: o contador **nunca** pode ficar negativo!

Two-Way Data Binding

Antes de falar sobre Two-Way, vamos identificar o problema que ele resolve

```
<ion-input type="number" [value]="currentValue"></ion-input>  
  
<p>O valor atual é {{ currentValue }}</p>  
  
<ion-button (click)="currentValue=currentValue+1">  
  Incrementar  
</ion-button>
```

O código acima tem um problema sério

O vínculo valor ➡ interface só flui nessa direção

Muitas vezes, em formulários, precisamos dos dois sentidos

Two-Way Data Binding

Sinaliza que queremos a ligação de  mão-dupla

```
<ion-input type="number" [(ngModel)]="currentValue"></ion-input>
```

Mnemônico para lembrar: 🍌 banana dentro da 📦 caixa

Formulários

Campo texto

```
<ion-item>
  <ion-label position="floating">Floating Label</ion-label>
  <ion-input></ion-input>
</ion-item>
```

Podemos também definir o atributo `type` para ajudar o usuário, e forçar diferentes teclados no mobile

Campo textarea

Usado para textos mais longos (como comentários, descrições, etc)

```
<ion-item>
  <ion-label position="floating">Description</ion-label>
  <ion-textarea></ion-textarea>
</ion-item>
```


Checkbox

Usado para escolher múltiplas opções dentre um conjunto de opções.

```
<ion-item>
  <ion-label>São Paulo</ion-label>
  <ion-checkbox slot="start"></ion-checkbox>
</ion-item>
<ion-item>
  <ion-checkbox slot="start"></ion-checkbox>
  <ion-label>Rio de Janeiro</ion-label>
</ion-item>
<ion-item>
  <ion-checkbox slot="start"></ion-checkbox>
  <ion-label>Minas Gerais</ion-label>
</ion-item>
<ion-item>
  <ion-checkbox slot="start"></ion-checkbox>
  <ion-label>Espírito Santo</ion-label>
</ion-item>
```

Toggle

Usado para definir valores booleanos

```
<ion-list>
  <ion-item>
    <ion-label>Aceito os termos</ion-label>
    <ion-toggle [(ngModel)]="accept"></ion-toggle>
  </ion-item>
</ion-list>
```

Observação

Teoricamente, Toggle e Checkbox são iguais.

Mas, em termos de usabilidade, temos algumas diferenças:

<https://uxplanet.org/checkbox-vs-toggle-switch-7fc6e83f10b8>

Radio

```
<p>Método Seleccionado: {{ paymentMethod }}</p>
<ion-radio-group [(ngModel)]="paymentMethod">
  <ion-item>
    <ion-label>Dinheiro</ion-label>
    <ion-radio slot="start" value="cash"></ion-radio>
  </ion-item>

  <ion-item>
    <ion-label>Crédito</ion-label>
    <ion-radio slot="start" value="credit"></ion-radio>
  </ion-item>

  <ion-item>
    <ion-label>Débito</ion-label>
    <ion-radio slot="start" value="debit"></ion-radio>
  </ion-item>
</ion-radio-group>
```

Select

```
<ion-item>
  <ion-label>Método Seleccionado</ion-label>
  <ion-select placeholder="Select One">
    <ion-select-option value="cash"> Dinheiro </ion-select-option>
    <ion-select-option value="debit"> Débito </ion-select-option>
    <ion-select-option value="credit"> Crédito </ion-select-option>
  </ion-select>
</ion-item>
```

Observação

Novamente, temos dois componentes que serem o mesmo propósito.

- Usa-se o Radio quando temos poucas opções (até 3 é razoável, 4 já se começa a pensar)
- Select quanto temos mais opções


Exercício 1

Atualize o app criado anteriormente para simular uma conta bancária:

- Coloque uma caixa de texto para o usuário digitar o valor
- Troque os botões de incrementa/decrementa para saque/depósito, que não usem valores pré-definidos, e sim o valor atual da caixinha de texto
- Após o saque/depósito, volte o valor da caixa de texto para 0
- Troque o texto do botão de reset para "Sacar tudo"
- A conta também **não pode** entrar no negativo. No caso de um saque maior que o saldo atual, considere que o saldo vai para 0

Exercício 2

Faça um clone do seguinte app

- Ignore os botões no header e o teclado
- Use Select em vez de Radio
- O campo texto da direita deve estar desabilitado
- Use o evento `ionChange` para mudar o resultado quando um número é inserido.
- Para o botão de , consulte o componente `ion-fab`

