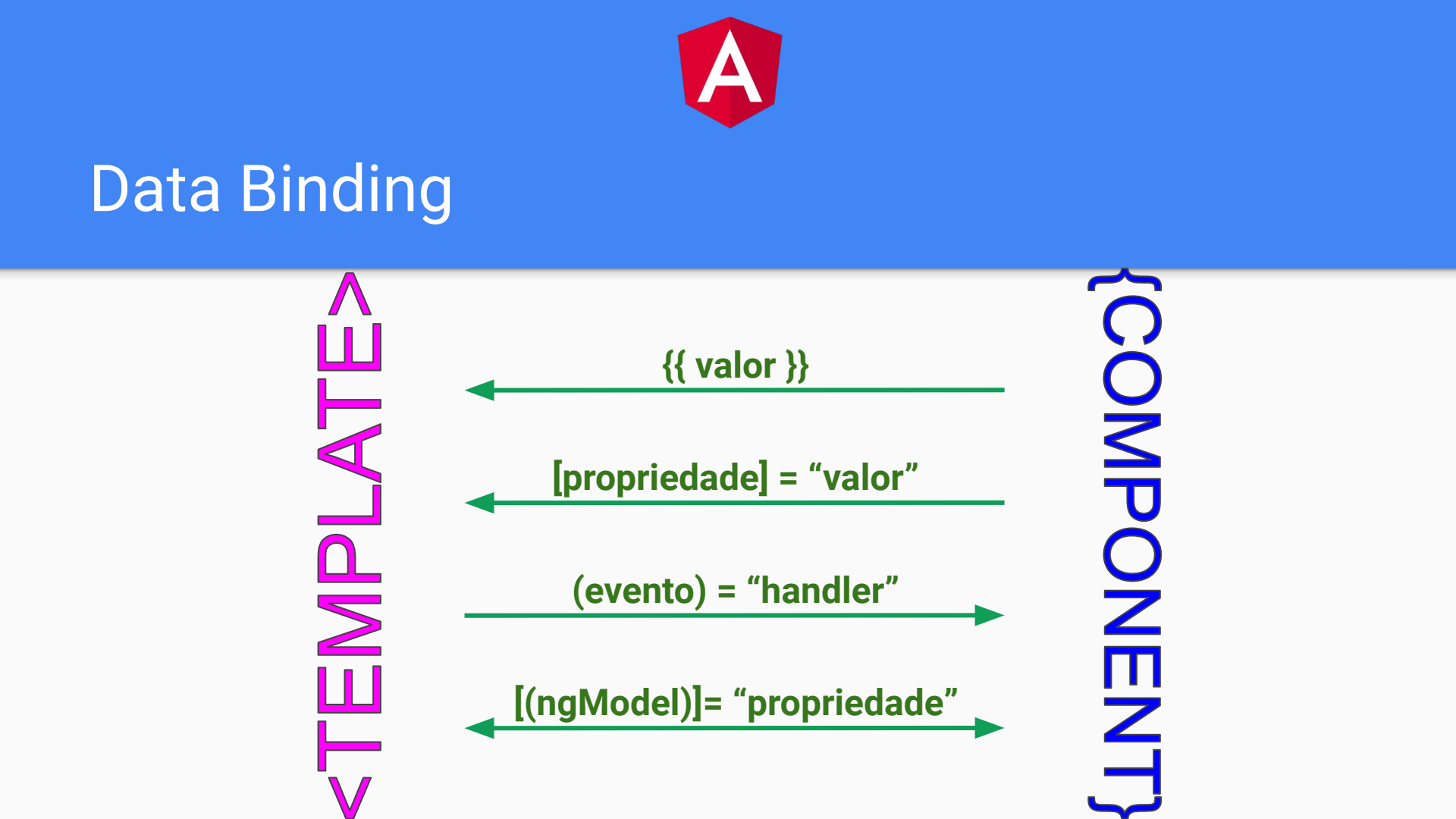
Data Binding


Data Binding
<TEMPLATE>
{{ valor }}
[propriedade] = “valor”
(evento) = “handler”
[(ngModel)]= “propriedade”
{COMPONENT}

Interpolation

- Valor do Component para o Template
- Usa-se chaves: `{{ valor }}`
- E o que se pode fazer?
  - Apresentar o valor da variável
  - Apresentar o retorno de uma função/método
  - Operações aritméticas
  - Operações booleanas

Property Binding

- Valor do Component para o Template
- Usa-se colchetes
  - `<img [src]="urlImg" />`
- Formato padrão é através de "bind-nomePropriedade"
  - `<img bind-src="urlImg" />`
- Quando não existe uma propriedade no elemento, usa-se [attr.colspan]

Class e Style Binding

- Tipos de Property Binding no CSS

```
<div [class.class-name]="class value">
    ...
</div>
```

```
<div [style.style-name]="style value">
    ...
</div>
```

Class Binding

- Class Binding:

```
<div class="alert" role="alert"
  [class.alert-success]="valor == true">
    Sucesso!
</div>
```

- Interpolation:

```
<div class="alert {{ class.value }}" role="alert">
    Alerta!
</div>
```

Style Binding

- Situação de uso do style binding:

```
<div class="alert alert-danger"
  role="alert"
style="display:none">
    Erro!
</div>
```

```
<div class="alert alert-danger"
  role="alert" style="display:block">
    Erro!
</div>
```

```
<div class="alert alert-danger" role="alert"
  [style.display]="error === true ? 'block'  : 'none'">
    Erro!
</div>
```

Event Binding

- Referência: https://developer.mozilla.org/pt-BR/docs/Web/Events
- Usa-se parênteses:
  - `(event-name)="event-action"`
- Formato padrão é através de "on-nomeEvento"
  - `<button (click)="button()" />`
  - `<button on-click="button()" />`

Two-Way Data Binding

- Valor do Template para o Componente e vice-versa
- Usa-se o binding de eventos + propriedades | Sintaxe “Banana na Caixa”
    - `<input [(ngModel)]=“nome” />`

```
<input type=“text”
  [value]=“nome”
  (input)=“nome =
$event.target.value”
/>
```

```
<input type=“text”
  [(ngModel)]=“nome”
/>
```

- Formato padrão: “bindon-ngModel”
- Diretiva que pertence ao FormsModule

Input Property

- Passar um valor do componente pai para o componente filho
- Usa-se a sintaxe do property binding no caso de uma variável

```
@Component({
  Selector: 'input-property',
  ...
})
Export class InputPropertyComponent
{
  @Input() name: string = '';
  ...
}
```

Componente pai

```
<input-property [name]="value">
</ input-property>
```

Componente filho

```
<p>{{ name }}<p>
```

Output Property

- Passar um valor/evento do componente filho para o componente pai
- Usa-se a sintaxe do property binding no caso de passar métodos

```
@Component({
  Selector: 'output-property',
  ...
})
Export class OutputPropertyComponent
{
  @Output() valueChanged = new
                EventEmitter();
  method() {
    this.valueChanged.emit({newValue:
        this.value}); }
```

Componente pai

```
<output-property [value]="someValue"
valueChanged="onChangedValue($event)
"
></ output-property>
```

Componente filho

```
<input [value]="someValue" />
```

Life Cycle Hooks

#	Evento (Hooks)	Quando?
1	ngOnChanges	Antes #2 e quando valor property-binding é atualizado
2	ngOnInit	Quando o Componente é inicializado
3	ngDoCheck	A cada ciclo de verificação de mudanças
4	ngAfterContentInit	Depois de inserir conteúdo externo na view
5	ngAfterContetChecked	A cada verificação de conteúdo inserido
6	ngAfterViewInit	Chamado uma vez depois do primeiro #5
7	ngAfterViewChecked	A cada verificação de conteúdo / conteúdo filho
8	ngOnDestroy	Antes da diretiva / componente ser destruído

Acesso ao DOM e ao Template

- Através do ViewChild decorator
  - `<input type="text" #valueField />`
  - `@ViewChild('valueField') valueInputField: ElementRef;`
- ViewChild:
  - Referência a um elemento do DOM
  - Referência para deretivas
  - Para injetar diretivas
  - ...

Observações:

- Questionário: https://forms.gle/7eZXa6Pa5uCdkLet6
- Código fonte: https://github.com/GabriellBP/Angular-course
- Dúvidas:
  - gabriel.pereira@edge.ufal.br
  - lucas.amorim@edge.ufal.br

Obrigado!