PROGRAMAÇÃO WEB I

TECNOLOGIAS E SISTEMAS DE INFORMAÇÃO PARA A WEB

POLITÉCNICO
DO PORTO
ESCOLA
SUPERIOR
DE MEDIA ARTES E
DESIGN

POLITÉCNICO DO PORTO
ESCOLA SUPERIOR DE MEDIA ARTES E DESIGN

P.PORTO

PROGRAMAÇÃO WEB I

TECNOLOGIAS E SISTEMAS DE INFORMAÇÃO PARA A WEB

2019-20

M02T01 – COMPONENTES

ÍNDICE

ÍNDICE

M02T01 – INTRODUÇÃO AOS COMPONENTES

1. INTRODUÇÃO

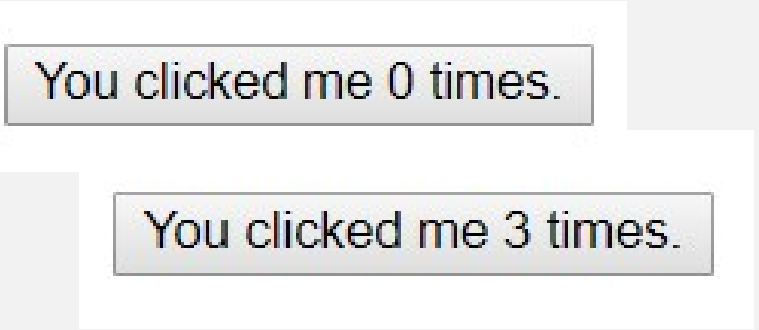
- OS **COMPONENTES** SÃO INSTÂNCIAS VUE REUTILIZÁVEIS COM UM NOME
- EXEMPLO DE UM COMPONENTE

```
// Define um novo componente chamado button-counter
Vue.component('button-counter', {
  data: function() {
    return {
     count: 0
    }
  },
  template: '<button v-on:click="count++">You clicked me {{ count }} times.</button>'
})
```

- NESTE CASO, PODEMOS USAR ESTE COMPONENTE COMO UM ELEMENTO PERSONALIZADO DENTRO DE UMA INSTÂNCIA RAIZ DO VUE CRIADA COM O **NEW VUE**:

```
const vm = new Vue({el: "#app"})
```

```
<div id="app">
  <button-counter></button-counter>
</div>
```



M02T01 – INTRODUÇÃO AOS COMPONENTES

1. INTRODUÇÃO

- OS **COMPONENTES** SÃO INSTÂNCIAS VUE REUTILIZÁVEIS COM UM NOME
- EXEMPLO DE UM COMPONENTE

```
// Define um novo componente chamado button-counter
Vue.component('button-counter', {
  data: function() {
    return {
      count: 0
    }
  },
  template: '<button v-on:click="count++">You clicked me {{ count }} times.</button>'
})
```

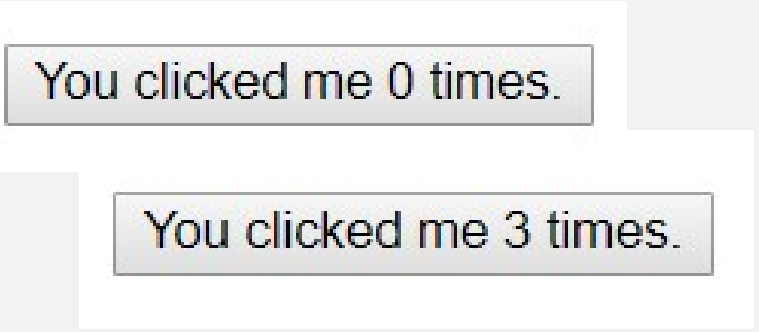
A propriedade data **deve ser uma função, para que cada instância possa manter uma cópia independente do objeto de dados retornado**

A propriedade template **define a renderização a ser feita baseada nos dados do componente**

- NESTE CASO, PODEMOS USAR ESTE COMPONENTE COMO UM ELEMENTO PERSONALIZADO DENTRO DE UMA INSTÂNCIA RAIZ DO VUE CRIADA COM O **NEW VUE**:

```
const vm = new Vue({el: "#app"})
```

```
<div id="app">
  <button-counter></button-counter>
</div>
```



M02T01 – INTRODUÇÃO AOS COMPONENTES

1. INTRODUÇÃO

- COMPONENTES PODEM SER REUTILIZADOS QUANTAS VEZES QUISER

```
<div id="intro">
  <button-counter></button-counter>
  <button-counter></button-counter>
  <button-counter></button-counter>
</div>
```

You clicked me 0 times. | You clicked me 2 times. | You clicked me 6 times.

- OBSERVE QUE, AO CLICAR NOS BOTÕES, CADA UM MANTÉM SUA PRÓPRIA CONTAGEM SEPARADA
- ISSO PORQUE, CADA VEZ QUE USA UM COMPONENTE, UMA NOVA INSTÂNCIA É CRIADA!

M02T01 – INTRODUÇÃO AOS COMPONENTES

1. INTRODUÇÃO

- CADA COMPONENTE PODE TER OS SEUS PRÓPRIOS DADOS, MÉTODOS, PROPRIEDADES CALCULADAS

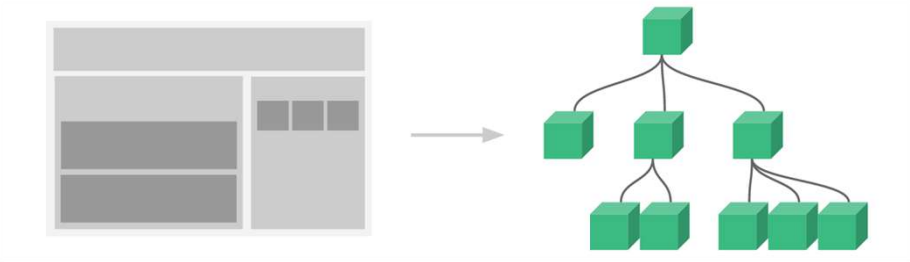
```
Vue.component('positive-numbers', {
  template: '<p>{{ positiveNumbers.length }} positive numbers</p>',
  data() {
    return {
      numbers: [-5, 0, 2, -1, 1, 0.5]
    };
  },
  computed: {
    positiveNumbers() {
      return this.numbers.filter( number => number >= 0)
    }
  }
});
```

ÍNDICE

M02T01 – INTRODUÇÃO AOS COMPONENTES

2. REGISTO

- É COMUM QUE UMA APLICAÇÃO SEJA ORGANIZADA NUMA ÁRVORE DE COMPONENTES ANINHADOS



- POR EXEMPLO, PODE TER COMPONENTES PARA UM CABEÇALHO, BARRA LATERAL E ÁREA DE CONTEÚDO, CADA UM CONTENDO OUTROS COMPONENTES PARA LINKS DE NAVEGAÇÃO, POSTS DE BLOG, ETC.

- PARA USAR ESSES COMPONENTES EM TEMPLATES, ELES DEVEM SER **REGISTADOS** PARA QUE O VUE OS CONHEÇA

- EXISTEM DOIS TIPOS DE REGISTO DE COMPONENTES:
  - **GLOBAL**
  - **LOCAL**

M02T01 – INTRODUÇÃO AOS COMPONENTES

2. REGISTO

REGISTO GLOBAL

- COMPONENTES REGISTADOS DE FORMA GLOBAL SIGNIFICA QUE ELES PODEM SER USADOS NO TEMPLATE DE QUALQUER INSTÂNCIA RAIZ DO VUE (**NEW VUE**) CRIADA APÓS O REGISTO

```
Vue.component('my-component-name', {
  // Opções
})
```

Nome do componente

Para o registo global use Vue.component

- POR EXEMPLO

```
Vue.component('component-a', { /* ... */ })
Vue.component('component-b', { /* ... */ })
Vue.component('component-c', { /* ... */ })

const vm = new Vue({el: "#app"})
```

JS

```
<div id="app">
  <component-a></component-a>
  <component-b></component-b>
  <component-c></component-c>
</div>
```

HTML

- COMPONENTES REGISTADOS GLOBALMENTE PODEM ATÉ SER USADOS DENTRO DE TODOS OS SUBCOMPONENTES DA ÁRVORE DE COMPONENTES DESSA INSTÂNCIA DO VUE

M02T01 – INTRODUÇÃO AOS COMPONENTES

2. REGISTO

REGISTO LOCAL

- O REGISTRO GLOBAL GERALMENTE NEM SEMPRE É O IDEAL
- POR EXEMPLO, SE ESTIVER USANDO UM SISTEMA DE AGREGAÇÃO/CONSTRUÇÃO COMO O **WEBPACK**, REGISTAR TODOS OS COMPONENTES GLOBALMENTE SIGNIFICA QUE, MESMO QUE PARE DE USAR UM COMPONENTE, ELE AINDA PODERÁ SER INCLUÍDO NO BUNDLER FINAL
- AUMENTO DESNECESSÁRIO DA QUANTIDADE DE JAVASCRIPT QUE OS UTILIZADORES PRECISAM DESCARREGAR!

M02T01 – INTRODUÇÃO AOS COMPONENTES
2. REGISTO

REGISTO LOCAL

- NESSES CASOS, PODE DEFINIR OS SEUS COMPONENTES COMO OBJETOS JAVASCRIPT SIMPLES:

```js
const ComponentA = { /* ... */ }
const ComponentB = { /* ... */ }
const ComponentC = { /* ... */ }
```

JS

- EM SEGUIDA, DEFINA OS COMPONENTES QUE GOSTARIA DE USAR NA PROPRIEDADE **COMPONENTS**:

```js
new Vue({
  el: “#app”,
  components: {
    ‘component-a’: ComponentA,
    ‘component-b’: ComponentB,
  }
})
```

JS

- PARA CADA PROPRIEDADE NO OBJETO **COMPONENTS**, A CHAVE SERÁ O NOME DO ELEMENTO PERSONALIZADO, ENQUANTO O VALOR CONTERÁ O OBJETO DE OPÇÕES PARA O COMPONENTE

M02T01 – INTRODUÇÃO AOS COMPONENTES

2. REGISTO

REGISTO LOCAL

- OS COMPONENTES REGISTADOS LOCALMENTE TAMBÉM NÃO ESTÃO DISPONÍVEIS NOS SUBCOMPONENTES
- POR EXEMPLO, SE QUISESSE QUE O **COMPONENTA** ESTIVESSE DISPONÍVEL NO **COMPONENTB**, TERIA QUE USAR:

```js
const ComponentA = { /* ... */ }

const ComponentB = {
  components: {
    'component-a': ComponentA
  },
  // ...
})
```

JS

M02T01 – INTRODUÇÃO AOS COMPONENTES
2. REGISTO

CONVENÇÃO DE NOMES

- TEM DUAS OPÇÕES AO DEFINIR NOMES DE COMPONENTES:

```
// Kebab-case
Vue.component('my-component-name', { /* ... */ })

// PascalCase
Vue.component('MyComponentName', { /* ... */ })
```

  - AO DEFINIR UM COMPONENTE EM

    - **KEBAB**, DEVE USAR A MESMA CONVENÇÃO AO REFERENCIAR O SEU ELEMENTO PERSONALIZADO, COMO EM **<MY-COMPONENT-NAME>**

    - **PASCAL**, PODE USAR QUALQUER CONVENÇÃO AO REFERENCIAR O SEU ELEMENTO PERSONALIZADO

ÍNDICE

M02T01 – INTRODUÇÃO AOS COMPONENTES

3. PROPS

- OS COMPONENTES SÃO ÚTEIS, MAS ELES REALMENTE MOSTRAM O SEU PODER QUANDO PASSA DADOS PARA ELES. ISSO É FEITO ATRAVÉS DOS **PROPS**
- **PROPS** SÃO ATRIBUTOS PERSONALIZADOS QUE PODE REGISTAR NUM COMPONENTE
- QUANDO UM VALOR É PASSADO PARA UM ATRIBUTO PROP, ELE TORNA-SE UMA PROPRIEDADE NESSA INSTÂNCIA
- POR EXEMPLO, PARA PASSAR UM TÍTULO A UM COMPONENTE DE POSTS DE UM BLOG, PODEMOS INCLUIR O TÍTULO DO POST NA LISTA DE PROPS QUE ESSE COMPONENTE ACEITA:

Uso da propriedade props

```
Vue.component('blog-post', {
  props: ['title'],
  template: '<h3>{{title}}</h3>'
})
```

JS

- UM COMPONENTE PODE TER QUANTOS PROPS QUISER E QUALQUER VALOR PODE SER PASSADO A UM PROP
- NO TEMPLATE EM CIMA, VÊ-SE QUE SE PODE ACEDER AO VALOR DO PROP NA INSTÂNCIA DO COMPONENTE

M02T01 – INTRODUÇÃO AOS COMPONENTES

3. PROPS

Uso da propriedade props

```
Vue.component('blog-post', {
  props: ['title'],
  template: '<h3>{{title}}</h3>'
})
```

JS

- UMA VEZ QUE UM PROP É REGISTADO, PODE PASSAR DADOS PARA ELE COMO UM ATRIBUTO CUSTOMIZADO:

```
<blog-post title="O meu primeiro post"></blog-post>
<blog-post title="Aprendendo Vue"></blog-post>
<blog-post title="Componentes em Vue"></blog-post>
```

HTML

O meu primeiro post
Aprendendo Vue
Componentes em Vue

M02T01 – INTRODUÇÃO AOS COMPONENTES

3. PROPS

- EM VEZ DE PASSAR UM ARRAY SIMPLES CONTENDO OS NOMES DOS OBJETOS QUE O COMPONENTE PODE RECEBER, TAMBÉM É POSSÍVEL PASSAR UM **OBJETO COM INFORMAÇÕES SOBRE OS PROPS**, POR EXEMPLO:
    - TIPOS DOS PROPS
    - OBRIGATORIEDADE
    - VALORES PADRÃO
    - FUNÇÕES DE VALIDAÇÃO PERSONALIZADAS

M02T01 – INTRODUÇÃO AOS COMPONENTES

3. PROPS

TIPOS DOS PROPS

- PODE LISTAR OS PROPS COMO UM **OBJETO**, ONDE OS NOMES E VALORES DAS PROPRIEDADES CONTÊM OS NOMES E TIPOS DO PROP, RESPETIVAMENTE:

```
props: {
  title: String,
  likes: Number,
  isPublished: Boolean,
  commentIds: Array,
  author: Object
}
```

Define o tipo de um prop, com um construtor nativo, como `Number`, `String` **ou** `Object`, **ou uma função construtora personalizada**

- ISSO NÃO APENAS DOCUMENTA O COMPONENTE, MAS TAMBÉM AVISA OS UTILIZADORES NA CONSOLA DO NAVEGADOR, CASO ESTES PASSEM O TIPO ERRADO

- UM **PROP PODE SER DE VÁRIOS TIPOS**, NESSE CASO PODE PASSAR TODOS OS TIPOS VÁLIDOS NUM ARRAY, COMO POR EXEMPLO: [NUMBER, STRING]

M02T01 – INTRODUÇÃO AOS COMPONENTES

3. PROPS

OBRIGATORIEDADE E VALORES PADRÃO

- PODE ESPECIFICAR SE UM PROP É OBRIGATÓRIO OU ATRIBUIR UM VALOR PADRÃO SE UM NÃO FOR DEFINIDO
- PARA TAL, DEFINA UM OBJETO EM VEZ DO CONSTRUTOR E PASSE O TIPO COM A PROPRIEDADE **TYPE** DO OBJETO

Prop obrigatório

```
Vue.component('price-display', {
  props: {
    price: {
      type: Number,
      required: true
    },
    unit: {
      type: String,
      dafault: '€'
    }
  }
})
```

Prop com valor padrão

- NESTE EXEMPLO:
  - O **PRICE** É OBRIGATÓRIO E UM AVISO SERÁ ACIONADO SE NÃO FOR ESPECIFICADO
  - O **UNIT** NÃO É OBRIGATÓRIO, MAS TEM UM VALOR PADRÃO DE **€**, POR ISSO, SE NÃO ATRIBUIR NENHUM VALOR, **UNIT** SERÁ IGUAL A **€** DENTRO DO COMPONENTE

M02T01 – INTRODUÇÃO AOS COMPONENTES

3. PROPS

FUNÇÕES DE VALIDAÇÃO

- PODE PASSAR UMA **FUNÇÃO VALIDADORA** QUE RECEBERÁ O VALOR DO PROP E DEVERÁ:
  - RETORNAR **TRUE** SE O PROP FOR VÁLIDO
  - RETORNAR **FALSE** SE O PROP NÃO FOR VÁLIDO

- O EXEMPLO A SEGUIR VALIDA SE O NÚMERO ESTÁ ACIMA DE ZERO, PARA QUE NÃO POSSA DAR OS PREÇOS NEGATIVOS ACIDENTALMENTE:

Função de validação

```
price: {
  type: Number,
  required: true
  validator(value) {
    return value >= 0
  }
}
```

M02T01 – INTRODUÇÃO AOS COMPONENTES

3. PROPS

REATIVIDADE

- QUANDO O VALOR DO OBJETO **DATA**, MÉTODO OU PROPRIEDADE CALCULADA É ALTERADO, O TEMPLATE TAMBÉM É ATUALIZADO, E ISSO TAMBÉM FUNCIONA COM OS PROPS
- A DIRETIVA **V-BIND** PODE SER USADA AO DEFINIR O PROP NO PAI PARA VINCULÁ-LO A UM VALOR E, EM SEGUIDA, SEMPRE QUE ESSE VALOR FOR ALTERADO, QUALQUER LUGAR EM QUE ELE FOR USADO NO COMPONENTE TAMBÉM SERÁ ATUALIZADO

```
Vue.component('display-number', {
 template: '<p>Number: {{number}}</p>',
 props: {
  number: {
   type: Number,
   required: true
  }
 }
})
```

Valor passado ao componente é incrementado a cada segundo

```
<div id="app">
  <display-number v-bind:number="number">
  </display-number>
</div>
<script>
  new Vue({
    el: '#app',
    data: {
      number: 0
    },
    created() {
      setInterval(() => {
        this.number++;
      }, 1000);
    }
  });
</script>
```

Vinculação da propriedade number ao componente através do props

M02T01 – INTRODUÇÃO AOS COMPONENTES

3. PROPS

PASSAGEM DE ARRAYS DE OBJETOS

```
<div id="app">
  <my-table v-bind:castles="castles">
  </my-table>
</div>
```

```
Vue.component("my-table", {
  props: ["castles"],
  template: `
    <table>
      <tr><th>id</th><th>name</th><th>link</th></tr>
      <tr v-for="castle in castles" v-bind:key="castle.id">
        <td>{{castle.id}}</td>
        <td>{{castle.name}}</td>
        <td>{{castle.link}}</td>
      </tr>
    </table>`
})

const vm = new Vue({
    el: "#app",
    data: { castles: []},
    created() {
     this.castles = [
       {id: 1, name: "Castelo ", link: "htt...", year:1642},
     ],...
    }
})
```

ÍNDICE

M02T01 – INTRODUÇÃO AOS COMPONENTES

4. EVENTOS

INTRODUÇÃO

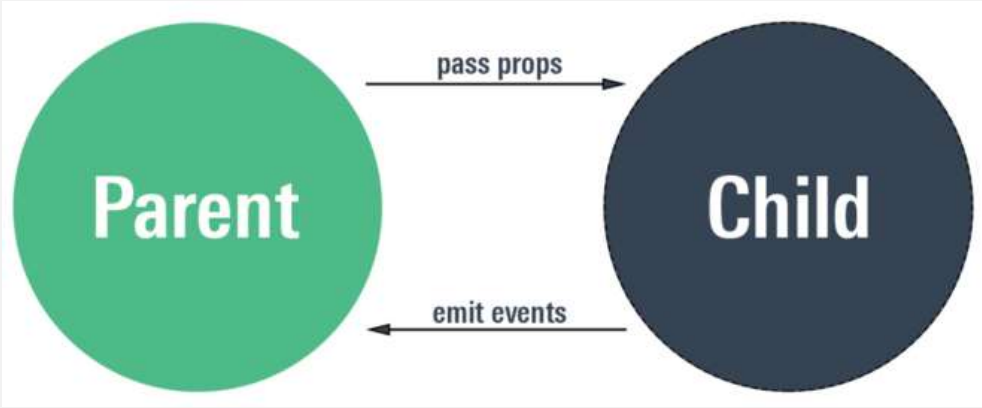
- REGRAS PARA DEFINIR UM COMPONENTE:
  - DEVEM SER **REUTILIZÁVEIS**
  - DEVEM TER APENAS A **LÓGICA NECESSÁRIA** PARA O SEU FUNCIONAMENTO
  - DEVEM SER **“DUMMY”** - O COMPONENTE DEVE APENAS PARA O SEU TRABALHO E ISSO É, DEVE ESTAR PREPARADO PARA ACEITAR DADOS DE UM PAI.
    - EXEMPLO: UMA TABELA DEVE CLASSIFICAR OS DADOS EM COLUNAS E LINHAS, DEPENDENDO DOS DADOS TRANSMITIDOS PELO PAI.
  - **NÃO DEVEM INTERAGIR DIRETAMENTE COM O "EXTERIOR"**. A ÚNICA FORMA DE FAZER ISSO É COM EVENTOS
  - DEVEM SER **INDEPENDENTES**. ISSO SIGNIFICA QUE ELES DEVEM E DEVEM TRABALHAR POR CONTA PRÓPRIA SEM ESTAR VINCULADOS A UMA ARQUITETURA ESPECÍFICA, ETC..
  - DEVEM SER CAPAZES DE SER ESTILIZADOS PELO PAI (CASO NECESSÁRIO)

M02T01 – INTRODUÇÃO AOS COMPONENTES

4. EVENTOS

INTRODUÇÃO

- REGRA DE OURO:




```html
<blog-post
  title="O meu primeiro post">
</blog-post>
```

HTML

```js
Vue.component('blog-post', {
  props: ['title'],
  template: '<h3>{{title}}</h3>'
})
```

JS

M02T01 – INTRODUÇÃO AOS COMPONENTES

4. EVENTOS

INTRODUÇÃO

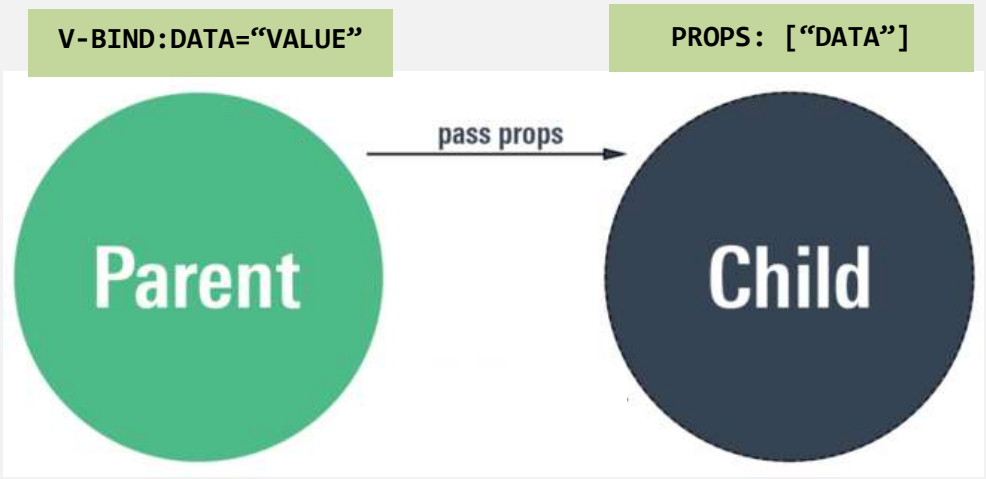
- OS DADOS SÃO PASSADOS PARA UM FILHO DE UM PAI POR MEIO DE UM PROP
- QUANDO OS DADOS SÃO ATUALIZADOS NO PAI, O PROP PASSADO AO FILHO É ATUALIZADO
- NO ENTANTO, NÃO PODE MODIFICAR O PROP DO COMPONENTE FILHO
- ISSO É CONHECIDO COMO UMA LIGAÇÃO UNIDIRECIONAL (**ONE-WAY-DOWN BINDING**) E IMPEDE QUE OS COMPONENTES ALTEREM ACIDENTALMENTE O ESTADO DE UM PAI
- PODE CONTORNAR ESTE FLUXO COM O MODIFICADOR **.SYNC** E O MÉTODO **$EMIT** PARA EMITIR ALTERAÇÕES DESDE O FILHO PARA O PAI

M02T01 – INTRODUÇÃO AOS COMPONENTES

4. EVENTOS

ENVIANDO MENSAGENS AOS PAIS COM EVENTOS

- À MEDIDA QUE DESENVOLVEMOS UM COMPONENTE ALGUNS RECURSOS PODEM EXIGIR A COMUNICAÇÃO DE VOLTA PARA O PAI
- ATÉ AGORA SÓ PASSAMOS DADOS PARA O COMPONENTE FILHO:

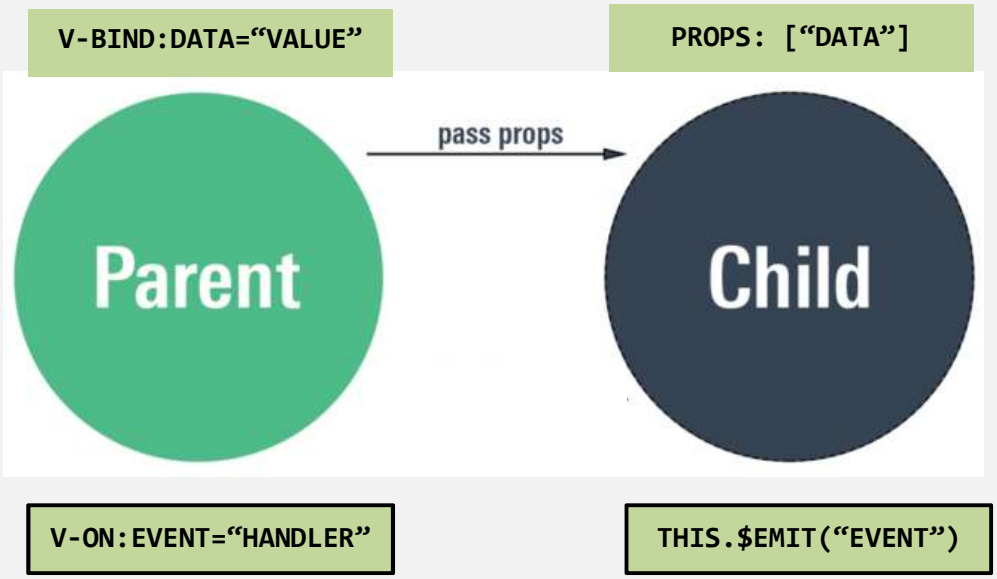


M02T01 – INTRODUÇÃO AOS COMPONENTES

4. EVENTOS

ENVIANDO MENSAGENS AOS PAIS COM EVENTOS

- PARA PASSAR DADOS DO FILHO PARA O PAI TEMOS DE EMITIR UM EVENTO ATRAVÉS DE **THIS.$EVENT**
- O VENTO TERÁ DE SER CAPTURADO PELO PAI COM A DIRETIVA **V-ON**:



M02T01 – INTRODUÇÃO AOS COMPONENTES

4. EVENTOS

ENVIANDO MENSAGENS AOS PAIS COM EVENTOS

- EXEMPLO: CONTROLAR O TAMANHO DOS POSTS DO BLOG
  - 1. CRIAR UM COMPONENTE **BLOG-POST**

```
Vue.component('blog-post', {
  props: [post],
  template: `
    <div class="blog-post">
      <h3>{{ post.title }}</h3>
      <button> Enlarge text </button>
      <div v-html="post.content"></div>
    </div>
  `
})
```

M02T01 – INTRODUÇÃO AOS COMPONENTES

4. EVENTOS

ENVIANDO MENSAGENS AOS PAIS COM EVENTOS

- EXEMPLO: CONTROLAR O TAMANHO DOS POSTS DO BLOG
  - 2. DEFINIR A INSTÂNCIA VUE

```
new Vue({
    el: '#blog-app',
    data: {
      posts: [/*…*/],
      postFontSize: 1
    }
});
```

  - 3. USO DO COMPONENTE NA PÁGINA HTML

```
<div id="blog-app">
  <div v-bind:style="fontSize: postFontSize + 'em'">
    <blog-post
      v-for="post in posts"
      v-bind:key="post.id"
      v-bind:post="post">
    </blog-post>
  </div>
</div>
```

M02T01 – INTRODUÇÃO AOS COMPONENTES

4. EVENTOS

ENVIANDO MENSAGENS AOS PAIS COM EVENTOS

- EXEMPLO: CONTROLAR O TAMANHO DOS POSTS DO BLOG
  - QUANDO CLICAMOS NO BOTÃO, PRECISAMOS COMUNICAR AO PAI DE FORMA A QUE ESTE AMPLIE O TEXTO DE TODOS OS POSTS
  - FELIZMENTE, AS INSTÂNCIAS DO VUE FORNECEM UM SISTEMA DE EVENTOS PERSONALIZADO PARA RESOLVER ESSE PROBLEMA. PARA EMITIR UM EVENTO PARA O PAI, PODEMOS CHAMAR O MÉTODO BUILT-IN **$EMIT**, PASSANDO O NOME DO EVENTO

```
<button>
  Enlarge text
</button>
```

```
<button v-on:click="$emit('enlarge-text')">
  Enlarge text
</button>
```

  - NO POST DO BLOG (PÁGINA HTML), PODEMOS OUVIR ESSE EVENTO COM **V-ON**, ASSIM COMO FARÍAMOS COM UM EVENTO DOM NATIVO:

```
<div v-bind:style="fontSize: postFontSize + 'em'">
    <blog-post
      ...
      v-on:enlarge-text="postFontSize +=0.1"
    >
    </blog-post>
  </div>
```

M02T01 – INTRODUÇÃO AOS COMPONENTES

4. EVENTOS

EMITINDO UM VALOR COM UM EVENTO

- ÀS VEZES, É ÚTIL EMITIR UM VALOR ESPECÍFICO COM UM EVENTO. POR EXEMPLO, PODEMOS QUERER QUE O COMPONENTE **<BLOG-POST>** SEJA RESPONSÁVEL POR AUMENTAR O TEXTO

- NESSES CASOS, PODEMOS USAR O SEGUNDO PARÂMETRO DO **$EMIT** PARA FORNECER ESTE VALOR:

```
<button v-on:click="$emit('enlarge-text, 0.1')">
  Enlarge text
</button>
```

Novo valor a passar com o evento

- QUANDO OUVIRMOS O EVENTO NO PAI, PODEREMOS ACEDER AO VALOR DO EVENTO EMITIDO COM **$EVENT**:

```
v-on:enlarge-text="postFontSize += $event"
```

- OU SE O MANIPULADOR DE EVENTOS (EVENT HANDLER) FOR UM MÉTODO:

```
v-on:enlarge-text="onEnlargeText"
```

```
methods: {
  onEnlargeText: function (amount) {
    this.postFontSize += amount
  }
}
```

M02T01 – INTRODUÇÃO AOS COMPONENTES

4. EVENTOS

USANDO V-MODEL EM COMPONENTES

- EVENTOS TAMBÉM PODEM SER USADOS PARA CRIAR INPUTS PERSONALIZADOS QUE FUNCIONAM COM O **V-MODEL**

```
<custom-input v-model="searchText"></custom-input>
```

```
Vue.component('custom-input', {
  props: ['value'],
  template: `
    <input
      v-bind:value="value"
      v-on:input="$emit('input', $event.target.value)"
    >
  `
})
```