

02

Chamada de métodos

Transcrição

Agora, se quisermos executar a lógica de exclusão, precisamos criar um método no componente pai `Home` que será chamada assim que o botão for clicado:

```
<!-- alurapic/src/components/home/Home.vue -->
<template>
  <div>
    <h1 class="centralizado">Alurapic</h1>

    <input type="search" class="filtro" @input="filtro = $event.target.value" placeholder="Filtrar...">

    <ul class="lista-fotos">

      <li class="lista-fotos-item" v-for="foto in fotosComFiltro">

        <meu-painel :titulo="foto.titulo">

          <imagem-responsiva :url="foto.url" :titulo="foto.titulo"/>

          <meu-botao rotulo="remover" tipo="button" @click="remove()"/>

        </meu-painel>

      </li>

    </ul>
  </div>
</template>

<script>

import Painel from '../shared/painel/Painel.vue';
import ImagemResponsiva from '../shared/imagem-responsiva/ImagenResponsiva.vue'

// Fazendo o import do botão. Não esqueça de adicionar-lo em components

import Botao from '../shared/botao/Botao.vue';

export default {

  components: {

    'meu-painel': Painel,
    'imagem-responsiva': ImagemResponsiva,
    'meu-botao': Botao
  },

  methods: {

    remove() {
      alert('Precisa saber qual foto remover!');
    }
  }
}

</script>
```

```

        }
    }

    // código omitido
}
</script>
<style>

/* código omitido */
</style>

```

Veja que associamos a chamada do método `remove` declarada em `methods` através do evento `click` usando o atalho `@click` para binding deste tipo de evento. Mas quando clicamos no botão, nenhum evento é disparado!

Disparando eventos nativos

Isso acontece, porque o componente que criamos é uma caixa preta e só podemos lidar com o que ele oferece. No entanto, podemos usar o modificador `.native` no evento clique para que o evento `click`, nativo de toda tag do mundo HTML seja disparado. Isso parece que vai resolver nosso problema:

```

<!-- alurapic/src/components/home/Home.vue -->
<!-- código anterior omitido -->

<meu-botao rotulo="remover" tipo="button" @click.native="remove()"/>

<!-- código posterior omitido -->

```

Agora, quando clicamos no botão, nosso alert é exibido. Excelente, mas precisamos saber qual foto clicamos para futuramente realizarmos um pedido à nossa API para que a delete.

Como o nosso botão está sendo construído dentro de um elemento com a diretiva `v-for`, podemos passar o elemento que está sendo iterado na lista diretamente para o método. Lembre-se que usamos o nome `foto` para referenciar cada elemento da lista. É este nome que passaremos para o método `remove`:

```

<!-- alurapic/src/components/home/Home.vue -->

<template>
    <div>
        <h1 class="centralizado">Alurapic</h1>

        <input type="search" class="filtro" @input="filtro = $event.target.value" placeholder="Fazer uma busca...">

        <ul class="lista-fotos">

            <li class="lista-fotos-item" v-for="foto in fotosComFiltro">

                <meu-painel :titulo="foto.titulo">

                    <imagem-responsiva :url="foto.url" :titulo="foto.titulo"/>

                </meu-painel>

            </li>
        </ul>
    </div>
</template>
<script>
    ...
</script>
<style>
    ...
</style>

```

```

</meu-painel>

</li>

</ul>
</div>
</template>

<script>

import Painel from '../shared/painel/Painel.vue';
import ImagemResponsiva from '../shared/imagem-responsiva/ImagenResponsiva.vue'

// Fazendo o import do botão. Não esqueça de adicioná-lo em components

import Botao from '../shared/botao/Botao.vue';

export default {

components: {

'meu-painel': Painel,
'imagem-responsiva': ImagemResponsiva,
'meu-botao': Botao
},

methods: {

remove(foto) {
    // exibindo o título da foto selecionado
    alert(foto.titulo);
}
}

// código omitido
}

</script>
<style>

/* código omitido */
</style>

```

Excelente, quando clicamos no botão remove de cada foto vemos o título da foto correspondente sendo exibida. Mas vocês devem lembrar que precisamos confirmar a exclusão. Nesse sentido, podemos usar a função `confirm` para conseguir rapidamente essa funcionalidades:

```

<!-- alurapic/src/components/home/Home.vue -->

<!-- código anterior omitido -->
methods: {

remove(foto) {
    if(confirm('Confirma?')) {
        alert(foto.titulo);
    }
}
}

```

```
        }
    }
}

<!-- código posterior omitido -->
```

Perfeito, só exibimos o alerta se confirmarmos. Mas pensem comigo. Todo lugar que precisarmos de uma confirmação teremos que ter esse código que chama o `confirm`. E se no lugar do `confirm` fosse exibido um model bonitão? Teríamos que repetir esse código em vários lugares. A ideia é colocar o código de confirmação no próprio botão, além disso, podemos ativar ou não a confirmação configurando nosso componente para tornar o botão ainda mais reutilizável. Como faremos isso? É o que veremos no próximo vídeo!