04

Adicionando comportamento ao nosso componente

Transcrição

Tornamos a experiência do usuário ainda melhor com a possibilidade de filtrar nossa lista de fotos. No entanto, que tal permitirmos que nosso componente `Painel` seja colapsável? Se implementarmos esse comportamento no componente, todos os lugares que ele for utilizado também terá o recurso.

A diretiva v-show

Queremos esconder o conteúdo do painel, não removê-lo. Para isso há a diretiva `v-show` . Quando adicionada em um elemento, quando seu valor for true, o elemento será exibido, se for false, será ocultado. Por debaixo dos panos a diretiva realiza um `display: none` .

Vamos adicioná-la no `slot` de `App` . Lembrando que este é o local onde serão inseridos os conteúdos do painel.

```
<!-- alurapic/src/components/shared/painel/Painel.vue -->

<template>

  <div class="painel">

    <h2 class="painel-titulo">{{ titulo }}</h2>
    <slot class="painel-conteudo" v-show="false">
```

```
    </slot>
  </div>

</template>

<script>

export default {

   props: ['titulo']

}
</script>

<style>
/* código omitido */
</style>
```

COPIAR CÓDIGO

Como atribuímos o valor `false`, nossos painéis devem exibir apenas o título do painel, mas isso não acontece. O painel e seu conteúdo continuam sendo exibidos.

Pegadinha do v-show e slots

O problema é que não podemos usar a diretiva `v-show` diretamente na tag `slot`. Se quisermos fazer com que o slot e todo seu conteúdo desaparece, precisamos envolvê-lo em uma tag `div` e nela usar a diretiva. Faremos isso, inclusive vamos mudar a class `painel-conteudo` de `slot` para a nova `div`:

```
<!-- alurapic/src/components/shared/painel/Painel.vue -->

<template>
```

```
  <div class="painel">

    <h2 class="painel-titulo">{{ titulo }}</h2>
    <div class="painel-conteudo" v-show="false">
      <slot></slot>
    </div>

  </div>

</template>

<script>

export default {

   props: ['titulo']

}

</script>

<style>

 .painel {
    padding: 0 auto;
    border: solid 2px grey;
    display: inline-block;
    margin: 5px;
    box-shadow: 5px 5px 10px grey;
    width: 200px;
    height: 100%;
    vertical-align: top;
    text-align: center;
  }
```

```
  .painel .painel-titulo {
    text-align: center;
    border: solid 2px;
    background: lightblue;
    margin: 0 0 15px 0;
    padding: 10px;
    text-transform: uppercase;
  }
</style>
```

COPIAR CÓDIGO

Perfeito! Todos os painéis são exibidos, mas apenas seu título. Se quisermos voltar a exibir seu conteúdo basta atribuirmos `true` ao valor de `v-show`. No entanto, queremos que essa troca do valor de `v-show` seja feita pelo usuário toda vez que ele clicar no título do painel.

O primeiro passo é adicionar no objeto retornando pela função `data` a propriedade `visivel` que começa como `true` e associá-la à diretiva `v-show` ao invés de deixarmos um valor fixo como `true` ou `false`:

```
<!-- alurapic/src/components/shared/painel/Painel.vue -->

<template>

  <div class="painel">

    <h2 class="painel-titulo">{{ titulo }}</h2>
    <div class="painel-conteudo" v-show="visivel">
      <slot></slot>
    </div>

  </div>
```

```
</template>

<script>

export default {

    props: ['titulo'],

    data() {

        return {

            visivel: true
        }

    }

}

</script>

<style>
/* código omitido */
</style>
```

COPIAR CÓDIGO

Bom, ainda continuamos na mesma, a diferença é que fizemos usa associação de `v-show` com `visivel` da nossa função `data` . Agora é que vem o truque.

Bind de eventos, mais uma vez

Aprendemos que através da diretiva v-on podemos executar um código a partir de um evento do JavaScript.Nesse caso, vamos adicionar um `v-on:dblclick` para responder ao evento `click` na tag `<h2>` que representa o título do nosso painel:

```
<!-- alurapic/src/components/shared/painel/Painel.vue -->

<template>

  <div class="painel">

    <h2 class="painel-titulo" v-on:dblclick="visivel =
!visivel">{{ titulo }}</h2>
    <div class="painel-conteudo" v-show="visivel">
      <slot></slot>
    </div>

  </div>

</template>

<script>

export default {

   props: ['titulo'],

   data() {

       return {

           visivel: true
       }
   }

}

</script>

<style>
```

```
/* código omitido */
</style>
```

COPIAR CÓDIGO

Veja que no evento `dblclick` estamos mudando o valor da propriedade `visivel` para o seu valor oposto. Se ela vale `true`, mudaremos no clique para `false`. Se clicarmos novamente, mudaremos para `true`.

Um novo teste demonstra que nossa estratégia funcionou. Veja que nosso painel agora possui seu primeiro comportamento e, não importa onde nosso componente `Painel` seja utilizado, esse comportamento o acompanhará.

Atalho para v-on

Da mesma forma que aprendemos um atalho para `v-bind`, há um atalho para a diretiva `v-on`. No caso, basta adicionarmos o nome do evento com o prefixo `@`:

```
<!-- alurapic/src/components/shared/painel/Painel.vue -->

<template>

  <div class="painel">

    <h2 class="painel-titulo" @dblclick="visivel = !visivel">
{{ titulo }}</h2>
    <div class="painel-conteudo" v-show="visivel">
      <slot></slot>
    </div>

  </div>

</template>
```

```
<script>

export default {

   props: ['titulo'],

   data() {

       return {

           visivel: true
       }
   }

}

</script>

<style>
/* código omitido */
</style>
```

COPIAR CÓDIGO

Inclusive podemos alterar em `App` de `v-on:input` para `@input`.

Tudo muito bacana, mas que tal realizar esse processo de exibe e oculta com efeito, por exemplo, um fade? É isso que veremos no próximo vídeo.