01

Que tal filtrarmos nossa lista? Precisamos de dados!

Transcrição

Começando deste ponto? Você pode fazer o DOWNLOAD (https://s3.amazonaws.com/caelum-online-public/vue/stages/05-alurapic.zip) completo do projeto do capítulo anterior e continuar seus estudos a partir deste capítulo. Será necessário abrir seu terminal, entrar dentro da pasta alurapic e executar o comando npm install para baixar novamente todas as dependências da aplicação.

Ainda não somos capazes de incluir novas fotos em nossa aplicação, mas com certeza o número de fotos pode ser bem grande e ajudar o usuário a encontrar uma foto por parte do título é de grande valia.

A primeira coisa que precisamos fazer é adicionar um `input` que capturará o título para que possamos utilizá-lo como critério de filtro da nossa lista. Inclusive, já vamos adicionar um estilo para ele na tag `style` do nosso componente `App.vue`:

```
<!-- alurapic/src/App.vue -->

<template>
  <div class="corpo">

    <h1 class="centralizado">{{ titulo }}</h1>
```

```
    <input type="search" class="filtro" placeholder="filtre
pelo título da foto">

    <ul class="lista-fotos">
      <li class="lista-fotos-item" v-for="foto in fotos">
        <meu-painel :titulo="foto.titulo">
          <img class="imagem-responsiva" :src="foto.url"
:alt="foto.titulo">
        </meu-painel>
      </li>
    </ul>

  </div>
</template>

<script>

// código omitido

</script>
<style>

  /* código anterior omitido */

  .filtro {
    display: block;
    width: 100%;
  }
</style>
```

COPIAR CÓDIGO

Excelente, agora precisamos arrumar uma forma de conseguirmos termos acesso ao valor digitado pelo usuário a cada dígito. Temos uma situação diferente. Antes, aprendemos a criar uma associação de dados, no inglês *data binding* que fluía d dados para a view, ou seja, para o template do nosso componente. Um exemplo e

a propriedade `fotos` retornada pela nossa função data. Quando ela é modificada pelo retorno de `$http` os dados fluem para a view fazendo com que ela seja atualizada. Queremos algo agora um pouco diferente, queremos que o dado flua da view para dentro do nosso componente.

Primeiro, vamos adicionar no objeto retornado pela função `data` do nosso componente a propriedade `filtro` :

```
<!-- alurapic/src/App.vue -->

<template>
 <!-- código do template omitido -->
</template>


<input class="filtro" placeholder="filtre pelo título da
foto">

<script>

import Painel from './components/shared/painel/Painel.vue'

export default {

  components: {

    'meu-painel': Painel
  },

  data () {

    return {

      titulo: 'Alurapic',
```

```
      fotos: [],

      filtro: ''

    }
  },

  created() {

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

/* código omitido */

</style>
```

COPIAR CÓDIGO

A diretiva v-on e mais um tipo de data binding

Agora, precisamos fazer uma associação unidirecional que flua da view para a fonte de dados, no caso, para a propriedade `filtro`. Queremos que a propriedade `filtro` seja atualizada a cada dígito no campo. Sabemos que no mundo JavaScript há o evento `input` disparado toda vez que algum valor é inserido no campo.

Vamos alterar a tag `input` do template e adicionar a diretiva **v-on:input**:

```
<!-- alurapic/src/App.vue -->
```

```
<!-- código anterior omitido -->

<input type="search" class="filtro" v-on:input
placeholder="filtre pelo título da foto">

<!-- código posterior omitido -->
```

COPIAR CÓDIGO

É através da diretiva `v-on:` que podemos elaborar uma resposta para eventos do JavaScript. No caso, adicionamos o nome do evento logo após os dois pontos. No entanto, precisamos associar algum código para este evento. Completando o código:

```
<!-- alurapic/src/App.vue -->

<!-- código anterior omitido -->

<input class="filtro" v-on:input="filtro =
$event.target.value" placeholder="filtre pelo título da foto">

<!-- código posterior omitido -->
```

COPIAR CÓDIGO

Quando digitamos no campo, a expressão adicionada entre aspas será executada. Veja que ela atribui à `filtro` o valor de `$event.target.value`. Quando um evento em JavaScript é disparado, há um objeto especial chamado `event` que detém um monte de informação sobre o evento disparado. No caso usamos `$event` pois esse é um objeto especial do `Vue`. Dizemos que é um `event` original encapsulado pelo `Vue`. É através dele, assim como o `event` padrão que podemos ter acesso ao alvo do evento, no caso, o seu `target`. No caso o `target` é o próprio `input`. É por isso que do target podemos fazer `.value` para obter o valor do input.

Se vocês ainda desconfiam que o `filtro` esteja recebendo o valor do digitado por nós, coloque a seguinte interpolação logo abaixo do `input` :

```
<!-- alurapic/src/App.vue -->

<!-- código anterior omitido -->

<input type="search" class="filtro" v-on:input="filtro =
$event.target.value" placeholder="filtre pelo título da foto">
{{ filtro }}

<!-- código posterior omitido -->
```

COPIAR CÓDIGO

Veja que a cada dígito no local da interpolação será exibido o valor que digitamos no input. Isso é fantástico, pois com pouquíssimo esforço estamos capturando o valor digitado pelo usuário e exibindo-o automaticamente na tela.

Por fim, vale ressaltar que `v-on` realiza um data binding unidirecional que flui da view para os dados e a interpolação ou `v-bind` realiza uma associação unidirecional que flui dos dados para view.

Agora que já aprendemos a capturar o filtro do usuário, que fazemos uso dele para filtrarmos a nossa lista?