

03

Filtrando efetivamente a lista

Transcrição

Agora que já temos o valor digitado pelo usuário podemos utilizá-lo para filtrar nossa lista de fotos.

Pensem comigo. Alguma lógica terá que ser aplicada no dado `fotos` para que apenas as fotos que tenham parte do título sejam consideradas. Certo? Mas não podemos simplesmente sair removendo os itens da lista, porque se o usuário desistir de um filtro e utilizar outro temos que exibir o item. Outro ponto, se o campo estiver em branco vamos exibir todas as fotos para ele. E agora, como lidaremos com essa lógica?

O Vue oferece uma solução elegante chamada de *computed property*, em português, propriedade computada. Hoje temos um dado que é `fotos`, mas essa lista precisará ser computada podendo retornar uma lista diferente da original. Sempre que tivemos que realizar algum cálculo ou aplicar alguma lógica dinamicamente podemos usar `computed property`.

Vamos adicionar em nosso componente `App` a propriedade `computed` e nela passar um objeto. Cada propriedade do objeto é obrigatoriamente uma função. Não poderia ser diferente, já que executaremos uma lógica:

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

<template>
  <div class="corpo">

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

    <input type="search" class="filtro" v-on:input="filtro = $event.target.value" placeholder="Filtre as fotos...">

    <ul class="lista-fotos">
      <li class="lista-fotos-item" v-for="foto of fotos">
        <meu-painel :titulo="foto.titulo">
          
        </meu-painel>
      </li>
    </ul>

  </div>
</template>

<script>

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

export default {

  components: {

    'meu-painel': Painel
  },

  data () {
    return {

```

```

        titulo: 'Alurapic',
        fotos: [],
        filtro: ''
    },
    computed: {
        fotosComFiltro() {
            if (this.filtro) {
                // filtra a lista, por enquanto vamos retornar uma lista em branco
                return [];
            } else {
                // se o campo estiver vazio, não filtramos, retornamos a lista
                return this.fotos;
            }
        }
    },
    created() {
        this.$http
            .get('http://localhost:3000/v1/fotos')
            .then(res => res.json())
            .then(fotos => this.fotos = fotos, err => console.log(err));
    }
}
</script>
<style>
/* código omitido */
</style>

```

Uma *computed property* pode ser acessada como uma propriedade em nossa view. Sendo assim, na diretiva `v-for` vamos usar `fotosComFiltro` no lugar de `fotos`:

```

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

<template>
<div class="corpo">

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

    <input type="search" class="filtro" v-on:input="filtro = $event.target.value" placeholder="-"

    <ul class="lista-fotos">
        <li class="lista-fotos-item" v-for="foto of fotosComFiltro">
            <meu-painel :titulo="foto.titulo">
                
            </meu-painel>
        </li>
    </ul>

</div>
</template>

```

```

<script>

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

export default {

components: {

'meu-painel': Painel
},

data () {
return {
    titulo: 'Alurapic',


    fotos: [],

    filtro: ''
}
},

computed: {
fotosComFiltro() {
    if (this.filtro) {
        // filtra a lista, por enquanto vamos retornar uma lista em branco
        return [];
    } else {
        // se o campo estiver vazio, não filtramos, retornamos a lista
        return this.fotos;
    }
}
},
created() {

this.$http
    .get('http://localhost:3000/v1/fotos')
    .then(res => res.json())
    .then(fotos => this.fotos = fotos, err => console.log(err));
}

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

```

Faça um teste. Se o campo estiver em branco, tudo será exibido. Se qualquer coisa for digitada no campo do filtro, nada será exibido. Chegou a hora de aplicarmos a lógica que retorna a lista filtrada pelo que digitamos. Para isso precisamos saber um pouquinho de expressão regular e o uso da poderosa função `filter` que todo array possui:

```

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

<template>
<div class="corpo">
```

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

<input type="search" class="filtro" v-on:input="filtro = $event.target.value" placeholder="-"

<ul class="lista-fotos">
  <li class="lista-fotos-item" v-for="foto of fotosComFiltro">
    <meu-painel :titulo="foto.titulo">
      
    </meu-painel>
  </li>
</ul>

</div>
</template>

<script>

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

export default {

  components: {

    'meu-painel': Painel
  },

  data () {
    return {
      titulo: 'Alurapic',

      fotos: [],

      filtro: ''
    }
  },
}

computed: {

  fotosComFiltro() {

    if (this.filtro) {
      // criando uma expressão com o valor do filtro, insensitivo
      let exp = new RegExp(this.filtro.trim(), 'i');
      // retorna apenas as fotos que condizem com a expressão
      return this.fotos.filter(foto => exp.test(foto.titulo));
    } else {
      return this.fotos;
    }

  }
},
}

created() {

  this.$http
    .get('http://localhost:3000/v1/fotos')
    .then(res => res.json())
}
```

```
.then(fotos => this.fotos = fotos, err => console.log(err));
}
}
</script>
<style>
/* código omitido */
</style>
```

Veja que dentro da nossa *computed property* podemos acessar dado filtro através de `this`. Isso é possível porque Vue internamente aplica sua mágica para que o `this` tenha acesso a todos as propriedade definidas na função `data`, o que é excelente para o desenvolvedor.

Nosso componente final fica assim:

```
<template>
<div class="corpo">

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

  <input type="search" class="filtro" v-on:input="filtro = $event.target.value" placeholder="-"

  <ul class="lista-fotos">
    <li class="lista-fotos-item" v-for="foto of fotosComFiltro">
      <meu-painel :titulo="foto.titulo">
        
      </meu-painel>
    </li>
  </ul>

</div>
</template>

<script>
import Painel from './components/shared/painel/Painel.vue'

export default {

  components: {

    'meu-painel': Painel
  },

  data () {
    return {
      titulo: 'Alurapic',

      fotos: [],

      filtro: ''
    }
  },

  computed: {
```

```
fotosComFiltro() {  
  
    if (this.filtro) {  
        let exp = new RegExp(this.filtro.trim(), 'i');  
        return this.fotos.filter(foto => exp.test(foto.titulo));  
    } else {  
        return this.fotos;  
    }  
  
}  
},  
  
created() {  
  
    this.$http  
        .get('http://localhost:3000/v1/fotos')  
        .then(res => res.json())  
        .then(fotos => this.fotos = fotos, err => console.log(err));  
}  
}  
}  
</script>  
<style>  
  
.centralizado {  
    text-align: center;  
}  
  
.corpo {  
    font-family: Helvetica, sans-serif;  
    margin: 0 auto;  
    width: 96%;  
}  
  
.lista-fotos {  
    list-style: none;  
}  
  
.lista-fotos .lista-fotos-item {  
    display: inline-block;  
}  
  
.imagem-responsiva {  
    width: 100%;  
}  
  
.filtro {  
    display: block;  
    width: 100%;  
}  
</style>
```