02

Capturando dados

Transcrição

Agora que já temos nosso formulário, precisamos capturar a entrada do usuário para que possamos enviá-la para a nossa API responsável pelo cadastro. O primeiro passo é adicionar no objeto retornado pela função `data` do nosso componente a propriedade `foto`. Seu valor será um objeto com as propriedades `titulo`, `url` e `descricao`:

```
<!-- alurapic/src/components/cadastro/Cadastro.vue -->

<template>

    <!-- código omitido -->

</template>

<script>

import ImagemResponsiva from '../shared/imagem-responsiva/ImagemResponsiva.vue'
import Botao from '../shared/botao/Botao.vue';

export default {

  components: {

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

  data() {
    return {

      foto: {
        titulo: '',
        url: '',
        descricao: ''
      }
    }
  }
}

</script>
<style scoped>

  /* código omitido */

</style>
```

Agora, precisamos de alguma maneira fazer com que cada entrada do usuário em nosso formulário seja gravada na sua respectiva propriedade do objeto `foto` que acabamos de criar. Já aprendemos a realizar um data binding da view para a fonte de dados. Vamos usar o evento `@input` em cada campo do nosos formulário:

```
<!-- alurapic/src/components/cadastro/Cadastro.vue -->

<template>

  <div>
    <h1 class="centralizado">Cadastro</h1>
    <h2 class="centralizado"></h2>

    <form @submit.prevent="grava()">
      <div class="controle">
        <label for="titulo">TÍTULO</label>
        <input @input="foto.titulo = $event.target.value" id="titulo" autocomplete="off">
      </div>

      <div class="controle">
        <label for="url">URL</label>
        <input @input="foto.url = $event.target.value" id="url" autocomplete="off">
        <imagem-responsiva/>
      </div>

      <div class="controle">
        <label for="descricao">DESCRIÇÃO</label>
        <textarea @input="foto.descricao= $event.target.value" id="descricao" autocomplete="off'
      </div>

      <div class="centralizado">
        <meu-botao rotulo="GRAVAR" tipo="submit"/>
        <router-link to="/"><meu-botao rotulo="VOLTAR" tipo="button"/></router-link>
      </div>

    </form>
  </div>
</template>

<!-- código posterior omitido -->
```

Agora que realizamos uma associação de cada campo do formulário com cada propriedade do nosso objeto `foto` podemos fazer o seguinte teste. Quando clicarmos no botão "GRAVAR" vamos exibir no console nosso objeto `foto`. Como ele já terá recebido os dados do formulário poderemos visualizar esses dados. Dessa forma, teremos a garantia que tudo esta funcionando para depois enviarmos esses dados para nossa API.

No mundo JavaScript, o evento `submit` é disparado toda vez que um formulário é submetido. Sendo assim, vamos criar o método `grava` em nosso componente `Cadastro` que será chamado para este evento. É nele que exibiremos os dados do nosso objeto no console:

```
<!-- alurapic/src/components/cadastro/Cadastro.vue -->

<template>

  <div>
    <h1 class="centralizado">Cadastro</h1>
    <h2 class="centralizado"></h2>

    <!-- associando o evento com método do componente -->
```

```
    <form @submit="grava()">

      <div class="controle">
        <label for="titulo">TÍTULO</label>
        <input @input="foto.titulo = $event.target.value" id="titulo" autocomplete="off">
      </div>

      <div class="controle">
        <label for="url">URL</label>
        <input @input="foto.url = $event.target.value" id="url" autocomplete="off">
        <imagem-responsiva/>
      </div>

      <div class="controle">
        <label for="descricao">DESCRIÇÃO</label>
        <textarea id="descricao" autocomplete="off" @input="foto.titulo = $event.target.value">
      </div>

      <div class="centralizado">
        <meu-botao rotulo="GRAVAR" tipo="submit"/>
        <router-link to="/"><meu-botao rotulo="VOLTAR" tipo="button"/></router-link>
      </div>

    </form>
  </div>
</template>

<script>

import ImagemResponsiva from '../shared/imagem-responsiva/ImagemResponsiva.vue'
import Botao from '../shared/botao/Botao.vue';

export default {

  components: {

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

  data() {
    return {

      foto: {
        titulo: '',
        url: '',
        descricao: ''
      }
    }
  },

  methods: {

    grava() {

      console.log(this.foto);
      this.foto = {};
```

```
    }
  }
}

</script>
<style scoped>

  /* código omitido */

</style>
```

Parece que ainda falta algo. Quando clicamos no botão "GRAVAR", o formulário é submetido e nossa aplicação recarregada. Isso não faz sentido em uma SPA, além disso, pelo fato da aplicação recarregar não conseguimos ver o resultado no console. A boa notícia é que podemos usar um `event modifier`. O Vue nos disponibiiza o modificador `prevent` que equivale ao `event.preventDefault()` padrão do JavaScript que cancela o comportamento padrão do evento, em nosso caso, o comportamento padrão do evento `submit` é submeter o formulário:

```
<!-- alurapic/src/components/cadastro/Cadastro.vue -->

<template>

  <div>
    <h1 class="centralizado">Cadastro</h1>
    <h2 class="centralizado"></h2>

    <form @submit.prevent="grava()">

    <!-- código posterior omitido -->
```

Agora sim! Quando clicamos no botão e o formulário é submetido nossa aplicação não recarrega e o método `grava` é chamado exibindo no console os dados do objeto `foto` que foram alimentados através do formulário.

No entanto, se vocês estão bem atentos, devem ter percebido que o formulário continua preenchido. Isso pode confundir o usuário fazendo-o pensar que a operação não funcionou.

Podemos resolver isso facilmente no método `grava` atribuindo para a propriedade `foto` um novo objeto com as propriedade `titulo`, `url` e `descricao` em branco:

```
<!-- alurapic/src/components/cadastro/Cadastro.vue -->

<template>

  <div>
    <h1 class="centralizado">Cadastro</h1>
    <h2 class="centralizado"></h2>

    <form @submit.prevent="grava()">
      <div class="controle">
        <label for="titulo">TÍTULO</label>
        <input @input="foto.titulo = $event.target.value" id="titulo" autocomplete="off">
      </div>
```

```
        <div class="controle">
          <label for="url">URL</label>
          <input @input="foto.url = $event.target.value" id="url" autocomplete="off">
          <imagem-responsiva/>
        </div>

        <div class="controle">
          <label for="descricao">DESCRIÇÃO</label>
          <textarea id="descricao" autocomplete="off" @input="foto.titulo = $event.target.value">
        </div>

        <div class="centralizado">
          <meu-botao rotulo="GRAVAR" tipo="submit"/>
          <router-link to="/"><meu-botao rotulo="VOLTAR" tipo="button"/></router-link>
        </div>

      </form>
    </div>
  </template>

  <script>

  import ImagemResponsiva from '../shared/imagem-responsiva/ImagemResponsiva.vue'
  import Botao from '../shared/botao/Botao.vue';

  export default {

    components: {

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

    data() {
      return {

        foto: {
          titulo: '',
          url: '',
          descricao: ''
        }
      }
    },

    methods: {

      grava() {

        console.log(this.foto);

        this.foto = {
            titulo: '',
            url: '',
            descricao: ''
        };

      }
    }
```

```
}

</script>
<style scoped>

/* código omitido */

</style>
```

Nenhum erro aconteceu, mas também o formulário não foi limpo. O que houve? A resposta esta no tipo de data binding que foi realizado. Usamos um data binding de eventos que vai da view para a fonte de dados, mas não fizemos o data binding inverso que vai da fonte de dados para view. Por isso nossa view não foi atualizada quando alteramos a propriedade `foto`. Aprendemos a realizar este último tipo de data binding através de `v-bind` ou seu atalho `:`. Vamos fazer uma asssociação que vai da fonte de dados para a a view através de `:value`:

```
<!-- alurapic/src/components/cadastro/Cadastro.vue -->

<template>

  <div>
    <h1 class="centralizado">Cadastro</h1>
    <h2 class="centralizado"></h2>

    <form @submit.prevent="grava()">
      <div class="controle">
        <label for="titulo">TÍTULO</label>
        <input :value="foto.titulo" @input="foto.titulo = $event.target.value" id="titulo" autoc
      </div>

      <div class="controle">
        <label for="url">URL</label>
        <input :value="foto.url" @input="foto.url = $event.target.value" id="url" autocomplete='
        <imagem-responsiva/>
      </div>

      <div class="controle">
        <label for="descricao">DESCRIÇÃO</label>
        <textarea :value="foto.descricao" id="descricao" autocomplete="off" @input="foto.descri
      </div>

      <div class="centralizado">
        <meu-botao rotulo="GRAVAR" tipo="submit"/>
        <router-link to="/"><meu-botao rotulo="VOLTAR" tipo="button"/></router-link>
      </div>

    </form>
  </div>
</template>

<!-- código posterior omitido -->
```

Agora sim! Nosso formulário é limpo! Temos aqui dois tipos de data binding ativos ao mesmo tempo, cada um com uma direção diferente o que caracteriza um **two way-data binding**. Podemos até usar como subtítulo do nosso componente o

título da foto que estamos digitando. A media que formos digitando no campo, a interpolação exibirá o valor mais atual:

```
<!-- alurapic/src/components/cadastro/Cadastro.vue -->

<template>

  <div>
    <h1 class="centralizado">Cadastro</h1>
    <h2 class="centralizado">{{ foto.titulo }}</h2>

 <!-- código posterior omitido -->
```

Tudo funciona, mas o Vue possui um atalho que pode nos poupar muito código na hora de usarmos esses dois tipos de associação de dados ao mesmo tempo. É o que veremos no próximo vídeo.