Vue.JS: Componentes

Para este exercício continuaremos utilizando a ferramenta online JSFiddle (https://jsfiddle.net/), que permite a execução de código JavaScript online. Vamos transformar a implementação do aplicativo de lista de tarefas em um conjunto de componentes para estudar o modelo de componentes do Vue.JS.

Terminamos a última aula com o código abaixo. Retiramos o filtro do JavaScript e do HTML para simplificar o código, assim como o CSS que não foi alterado (permanece o mesmo da última aula).

```
<div id='app'>
  <h2 class='lista'>Minhas Tarefas:</h2>
  <ul>
    <li v-for="task in tasks">
      <input type="checkbox"
        @change="toggle(task)"
        v-bind:checked="task.done">

      <span v-bind:class="[task.done ? 'done':'']"
        @click="toggle(task)">
         {{ task.title }}
      </span>
    </li>
  </ul>

  <h2 class='form'>Nova Tarefa:</h2>
  <input type="text"
    v-model="newTask"
    @keyup.enter="add"/>
  <input type="button" value=" + " @click="add"/>
</div>
```

CSS usado na última aula

```
new Vue({
  el: "#app",
  data: {
    tasks: [
      { title: "Aprender Java", done: true },
      { title: "Aprender back-end Web com Java", done: true },
      { title: "Aprender Vue.js básico", done: false },
      { title: "Aprender Vue.js avançado", done: false },
      { title: "Aprender roteamento com Vue.js", done: false }
    ],
    newTask: ''
  },
  methods: {
    toggle: function(task) {
      task.done = !task.done;
    },
    add: function() {
      var task = { title: this.newTask, done: false };
      this.tasks.push(task);
      this.newTask = '';
    }
  }
})
```

Componentes são utilizados para dividir a funcionalidade de uma aplicação em partes. Um componente Vue.JS possui um template HTML, propriedades, dados locais e um conjunto de métodos. Abaixo temos um primeiro componente. É importante observar que o *template* do componente deve ter apenas um elemento HTML raiz!

```
<div id='app'>
  <h2 class='lista'>Minhas Tarefas:</h2>
  <ul>
    <li v-for="task in tasks">
      <input type="checkbox"
        @change="toggle(task)"
        v-bind:checked="task.done">

      <span v-bind:class="[task.done ? 'done':'']"
        @click="toggle(task)">
         {{ task.title }}
      </span>
    </li>
  </ul>

  <task-list></task-list>

  <h2 class='form'>Nova Tarefa:</h2>
  <input type="text"
    v-model="newTask"
    @keyup.enter="add"/>
  <input type="button" value=" + " @click="add"/>
</div>
```

CSS usado na última aula

```
Vue.component('task-list', {
  template: `
    <ul>
      <li>Teste</li>
    </ul>`
})

new Vue({
  el: "#app",
  component: ['task-list'],
  data: {
    tasks: [
      { title: "Aprender Java", done: true },
      { title: "Aprender back-end Web com Java", done: true },
      { title: "Aprender Vue.js básico", done: false },
      { title: "Aprender Vue.js avançado", done: false },
      { title: "Aprender roteamento com Vue.js", done: false }
    ],
    newTask: ''
  },
  methods: {
    toggle: function(task) {
      task.done = !task.done;
    },
    add: function() {
      var task = { title: this.newTask, done: false };
      this.tasks.push(task);
      this.newTask = '';
    }
  }
})
```

Podemos adicionar dados no componente, utilizando suas propriedades e os atributos do elemento HTML. Note como a lista apresenta os mesmos elementos do componente principal.

```
<div id='app'>
  <h2 class='lista'>Minhas Tarefas:</h2>
  <ul>
    <li v-for="task in tasks">
      <input type="checkbox"
        @change="toggle(task)"
        v-bind:checked="task.done">

      <span v-bind:class="[task.done ? 'done':'']"
        @click="toggle(task)">
         {{ task.title }}
      </span>
    </li>
  </ul>

  <task-list v-bind:tasks='tasks'></task-list>

  <h2 class='form'>Nova Tarefa:</h2>
  <input type="text"
    v-model="newTask"
    @keyup.enter="add"/>
  <input type="button" value=" + " @click="add"/>
</div>
```

CSS usado na última aula

```
Vue.component('task-list', {
  props: ['tasks'],
  template: `
    <ul>
      <li>Teste</li>
    </ul>`
})

new Vue({
  el: "#app",
  component: ['task-list'],
  data: {
    tasks: [
      { title: "Aprender Java", done: true },
      { title: "Aprender back-end Web com Java", done: true },
      { title: "Aprender Vue.js básico", done: false },
      { title: "Aprender Vue.js avançado", done: false },
      { title: "Aprender roteamento com Vue.js", done: false }
    ],
    newTask: ''
  },
  methods: {
    toggle: function(task) {
      task.done = !task.done;
    },
    add: function() {
      var task = { title: this.newTask, done: false };
      this.tasks.push(task);
      this.newTask = '';
    }
  }
})
```

Agora podemos mover a funcionalidade associada à apresentação da lista. Note que o código de parte da aplicação Vue.JS foi suprimido para que possa caber na página. Ao rodar a aplicação, você perceberá duas listas com a mesma funcionalidade e compartilhando as informações. Note também que a diretiva “v-bind” foi removida do componente, ficando apenas o marcador “:”.

```
<div id='app'>
  <h2 class='lista'>Minhas Tarefas:</h2>
  <ul>
    <li v-for="task in tasks">
      <input type="checkbox"
        @change="toggle(task)"
        v-bind:checked="task.done">

      <span v-bind:class="[task.done ? 'done':'']"
        @click="toggle(task)">
         {{ task.title }}
      </span>
    </li>
  </ul>

  <task-list v-bind:tasks='tasks'></task-list>

  <h2 class='form'>Nova Tarefa:</h2>
  <input type="text"
    v-model="newTask"
    @keyup.enter="add"/>
  <input type="button" value=" + " @click="add"/>
</div>
```

CSS usado na última aula

```
Vue.component('task-list', {
  props: ['tasks'],
  template: `
    <ul>
      <li v-for="task in tasks">
        <input type="checkbox"
          @change="toggle(task)"
          :checked="task.done">

        <span :class="[task.done ? 'done' : '']"
          @click="toggle(task)">
            {{ task.title }}
        </span>
      </li>
    </ul>`,
  methods: {
    toggle: function(task) {
      task.done = !task.done;
    }
  }
})

new Vue({
  el: "#app",
  component: ['task-list'],
  ...
})
```

Agora podemos simplificar a aplicação removendo o método que foi para o componente e usar apenas o componente no HTML.

```
<div id='app'>
  <h2 class='lista'>Minhas Tarefas:</h2>
  <task-list v-bind:tasks='tasks'></task-list>

  <h2 class='form'>Nova Tarefa:</h2>
  <input type="text"
    v-model="newTask"
    @keyup.enter="add"/>
  <input type="button" value=" + " @click="add"/>
</div>
```

CSS usado na última aula

```
Vue.component('task-list', {
  props: ['tasks'],
  template: `
    <ul>
      <li v-for="task in tasks">
        <input type="checkbox" @change="toggle(task)" :checked="task.done">
        <span :class="[task.done ? 'done':'']" @click="toggle(task)">
          {{ task.title }}
        </span>
      </li>
    </ul>`,
  methods: {
    toggle: function(task) {
      task.done = !task.done;
    }
  }
})

new Vue({
  el: "#app",
  component: ['task-list'],
  data: {
    tasks: [
      { title: "Aprender Java", done: true },
      { title: "Aprender back-end Web com Java", done: true },
      { title: "Aprender Vue.js básico", done: false },
      { title: "Aprender Vue.js avançado", done: false },
      { title: "Aprender roteamento com Vue.js", done: false }
    ],
    newTask: ''
  },
  methods: {
    add: function() {
      var task = { title: this.newTask, done: false };
      this.tasks.push(task);
      this.newTask = '';
    }
  }
})
```

Podemos simplificar ainda mais o código criando um segundo componente para o formulário. Note que este componente tem dados locais e que estes dados são retornados como uma função (é uma demanda do Vue.JS, devido ao seu modelo de compartilhamento de dados entre componentes).

```
<div id='app'>
  <h2 class='lista'>Minhas Tarefas:</h2>
  <task-list :tasks='tasks'></task-list>

  <h2 class='form'>Nova Tarefa:</h2>
  <task-form :tasks='tasks'></task-form>
</div>
```

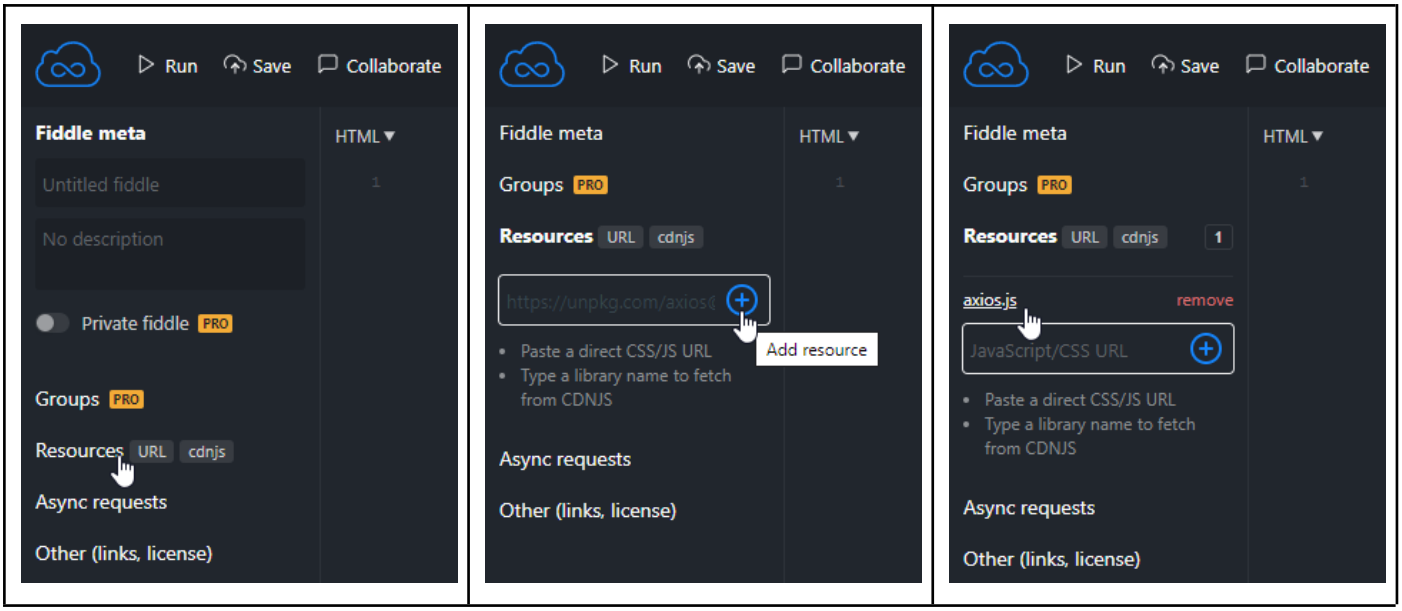
CSS usado na última aula

```
Vue.component('task-list', {
  props: ['tasks'],
  template: `
    <ul>
      <li v-for="task in tasks">
        <input type="checkbox" @change="toggle(task)" :checked="task.done">
        <span :class="[task.done ? 'done':'']" @click="toggle(task)">{{ task.title }}</span>
      </li>
    </ul>`,
  methods: {
    toggle: function(task) {
      task.done = !task.done;
    }
  }
})

Vue.component('task-form', {
  props: ['tasks'],
  data: function () {
    return { newTask: '' }
  },
  template: `
    <div>
      <input type="text" v-model="newTask" @keyup.enter="add"/>
      <input type="button" value=" + " @click="add"/>
    </div>`,
  methods: {
    add: function() {
      var task = { title: this.newTask, done: false };
      this.tasks.push(task);
      this.newTask = '';
    }
  }
})

new Vue({
  el: "#app",
  component: ['task-list', 'task-form'],
  data: {
    tasks: [
      { title: "Aprender Java", done: true },
      { title: "Aprender back-end Web com Java", done: true },
      { title: "Aprender Vue.js básico", done: false },
      { title: "Aprender Vue.js avançado", done: false },
      { title: "Aprender roteamento com Vue.js", done: false }
    ]
  }
})
```

O JSFiddle permite a importação de recursos externos para o script. Por exemplo, a biblioteca Axios (acesso a recursos HTTP) pode ser importada do endereço https://unpkg.com/axios@0.16.2/dist/axios.js.



O programa abaixo usa esta biblioteca para pegar preços de Bitcoin. Note o uso do método mounted, que pertence ao ciclo de vida da aplicação (ou componente) e indica que ela foi conectada ao HTML.

```
<div id="app">
  <h2>Bitcoin price at {{ time }}</h2>
  <ul>
    <li v-html="dollars"></li>
    <li v-html="euros"></li>
    <li v-html="pounds"></li>
  </ul>
  <input type="button" value="Update" @click="update()"/>
</div>
```

```
li {
  margin: 8px 0;
}
```

```
new Vue({
  el: '#app',
  data: {
    time: null,
    dollars: null, euros: null, pounds: null
  },
  methods: {
    update: function() {
      axios
        .get('https://api.coindesk.com/v1/bpi/currentprice.json')
        .then(response => {
          var info = response.data;
          this.time = info.time.updateduk;
          this.dollars = info.bpi.USD.symbol + " " + info.bpi.USD.rate;
          this.euros = info.bpi.EUR.symbol + " " + info.bpi.EUR.rate;
          this.pounds = info.bpi.GBP.symbol + " " + info.bpi.GBP.rate;
        })
    }
  },
  mounted() {
    this.update();
  }
})
```

Agora é a sua vez de mostrar que aprendeu como usar componentes no Vue.JS ...

1) Separe a aplicação de controle de despesas e receitas em componente. Crie um componente para a lista, com as funcionalidades de apresentar as transações, mudar a sua ordem e remover transações. Crie um segundo componente para o formulário de registro de novas transações.

2) Desenvolva uma aplicação que apresente quem está na estação espacial internacional neste momento, usando a API do “Who is in Space Right Now”.

   API endpoint: https://www.howmanypeopleareinspacerightnow.com/peopleinspace.json

3) Desenvolva uma aplicação que apresente os repositórios mais populares do dia, semana e mês utilizando a API do GitHub.

4) Desenvolva uma aplicação que apresenta um Meme aleatório baseado no Chuck Norris.

   API endpoint: https://api.chucknorris.io/jokes/random

5) Desenvolva uma aplicação com um formulário adequado aos parâmetros de uma mensagem mal educada, utilizando o serviço (peculiar) abaixo.

   API endpoint: https://www.foaas.com/