Introdução ao Vue.JS

Para este exercício utilizaremos a ferramenta online JSFiddle (https://jsfiddle.net/), que permite a execução de código JavaScript online.

Vamos desenvolver uma lista de tarefas simples, o exemplo mais comum para introdução ao Vue.js.

Para começar, abra um novo JSFiddle. Como estamos começando, não vamos usar nenhum código *boilerplate*.

Em seguida, selecione o Vue.js na combo-box que está no topo da seção de JavaScript.

Agora, vamos adicionar um HTML básico para a aplicação e rodar.

```
<div>
  <h2>Minhas Tarefas:</h2>
  <ul>
    <li><input type="checkbox"> Tarefa #1</li>
    <li><input type="checkbox"> Tarefa #2</li>
    <li><input type="checkbox"> Tarefa #3</li>
    <li><input type="checkbox"> Tarefa #4</li>
  </ul>

  <h2>Nova Tarefa:</h2>
  <input type="text"/>
  <input type="button" value=" + "/>
</div>
```

O próximo passo é melhorar a formatação, adicionando classes no HTML e no CSS.

```
<div>
  <h2 class='lista'>Minhas Tarefas:</h2>
  <ul>
    <li><input type="checkbox"> Tarefa #1</li>
    <li><input type="checkbox"> Tarefa #2</li>
    <li><input type="checkbox"> Tarefa #3</li>
    <li><input type="checkbox"> Tarefa #4</li>
  </ul>

  <h2 class='form'>Nova Tarefa:</h2>
  <input type="text"/>
  <input type="button" value=" + "/>
</div>
```

```
h2.lista {
  margin-bottom: 16px;
}

h2.form {
  margin-top: 48px;
  margin-bottom: 8px;
}

li {
  margin: 8px -40px;
  list-style: none;
}
```

Até agora, só temos HTML e CSS. Nada de Vue.js. Para começar a usar o framework, vamos criar a sua aplicação e ligar a um elemento HTML.

```
<div id='app'>
  <h2 class='lista'>Minhas Tarefas:</h2>
  <ul>
    <li><input type="checkbox"> Tarefa #1</li>
    <li><input type="checkbox"> Tarefa #2</li>
    <li><input type="checkbox"> Tarefa #3</li>
    <li><input type="checkbox"> Tarefa #4</li>
  </ul>

  <h2 class='form'>Nova Tarefa:</h2>
  <input type="text"/>
  <input type="button" value=" + "/>
</div>
```

```
h2.lista {
  margin-bottom: 16px;
}

h2.form {
  margin-top: 48px;
  margin-bottom: 8px;
}

li {
  margin: 8px -40px;
  list-style: none;
}
```

```
new Vue({
  el: "#app"
})
```

Com isso, podemos declarar dados no Vue e apresentar no HTML usando a notação “double-mustache'.

```
<div id='app'>
  {{ name }}

  <h2 class='lista'>Minhas Tarefas:</h2>
  <ul>
    <li><input type="checkbox"> Tarefa #1</li>
    <li><input type="checkbox"> Tarefa #2</li>
    <li><input type="checkbox"> Tarefa #3</li>
    <li><input type="checkbox"> Tarefa #4</li>
  </ul>

  <h2 class='form'>Nova Tarefa:</h2>
  <input type="text"/>
  <input type="button" value=" + "/>
</div>
```

```
h2.lista {
  margin-bottom: 16px;
}

h2.form {
  margin-top: 48px;
  margin-bottom: 8px;
}

li {
  margin: 8px -40px;
  list-style: none;
}
```

```
new Vue({
  el: "#app",
  data: {
    name: 'Marcio'
  }
})
```

Esta ligação pode ser transformada em bidirecional usando a diretiva “v-model'. Note a atualização do texto quando editamos na linha de texto. Chamamos este recurso de "Two-way data binding".

```
<div id='app'>
  {{ name }}

  <h2 class='lista'>Minhas Tarefas:</h2>
  <ul>
    <li><input type="checkbox"> Tarefa #1</li>
    <li><input type="checkbox"> Tarefa #2</li>
    <li><input type="checkbox"> Tarefa #3</li>
    <li><input type="checkbox"> Tarefa #4</li>
  </ul>

  <h2 class='form'>Nova Tarefa:</h2>
  <input type="text" v-model="name"/>
  <input type="button" value=" + "/>
</div>
```

```
h2.lista {
  margin-bottom: 16px;
}

h2.form {
  margin-top: 48px;
  margin-bottom: 8px;
}

li {
  margin: 8px -40px;
  list-style: none;
}
```

```
new Vue({
  el: "#app",
  data: {
    name: 'Marcio'
  }
})
```

Podemos declarar *arrays* e objetos usando a notação JavaScript dentro da lista de dados da aplicação Vue.js.

```
<div id='app'>
  <h2 class='lista'>Minhas Tarefas:</h2>
  <ul>
    <li><input type="checkbox"> Tarefa #1</li>
    <li><input type="checkbox"> Tarefa #2</li>
    <li><input type="checkbox"> Tarefa #3</li>
    <li><input type="checkbox"> Tarefa #4</li>
  </ul>

  <h2 class='form'>Nova Tarefa:</h2>
  <input type="text"/>
  <input type="button" value=" + "/>
</div>
```

```
h2.lista {
  margin-bottom: 16px;
}

h2.form {
  margin-top: 48px;
  margin-bottom: 8px;
}

li {
  margin: 8px -40px;
  list-style: none;
}
```

```
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 }
    ]
  }
})
```

Em seguida, podemos usar um loop para preencher a lista de tarefas (v-for).

```
<div id='app'>
  <h2 class='lista'>Minhas Tarefas:</h2>
  <ul>
    <li v-for="task in tasks">
      <input type="checkbox">
      <span>{{ task.title }}</span>
    </li>
  </ul>

  <h2 class='form'>Nova Tarefa:</h2>
  <input type="text"/>
  <input type="button" value=" + "/>
</div>
```

```
h2.lista {
  margin-bottom: 16px;
}

h2.form {
  margin-top: 48px;
  margin-bottom: 8px;
}

li {
  margin: 8px -40px;
  list-style: none;
}
```

```
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 }
    ]
  }
})
```

O loop também pode apresentar um índice.

```
<div id='app'>
  <h2 class='lista'>Minhas Tarefas:</h2>
  <ul>
    <li v-for="(task, index) in tasks">
      <input type="checkbox">
      <span>{{ index }}.{{ task.title }}</span>
    </li>
  </ul>

  <h2 class='form'>Nova Tarefa:</h2>
  <input type="text"/>
  <input type="button" value=" + "/>
</div>
```

```
h2.lista {
  margin-bottom: 16px;
}

h2.form {
  margin-top: 48px;
  margin-bottom: 8px;
}

li {
  margin: 8px -40px;
  list-style: none;
}
```

```
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 }
    ]
  }
})
```

Em seguida, ligamos ou desligamos o estado da check-box de acordo com o estado das tarefas usando a diretiva “v-bind' com um atributo do elemento.

```
<div id='app'>
  <h2 class='lista'>Minhas Tarefas:</h2>
  <ul>
    <li v-for="task in tasks">
      <input type="checkbox"
        v-bind:checked="task.done">
      <span>{{ task.title }}</span>
    </li>
  </ul>

  <h2 class='form'>Nova Tarefa:</h2>
  <input type="text"/>
  <input type="button" value=" + "/>
</div>
```

```
h2.lista {
  margin-bottom: 16px;
}

h2.form {
  margin-top: 48px;
  margin-bottom: 8px;
}

li {
  margin: 8px -40px;
  list-style: none;
}
```

```
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 }
    ]
  }
})
```

O mesmo pode ser feito com a classe do span que apresenta o título da tarefa.

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

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

  <h2 class='form'>Nova Tarefa:</h2>
  <input type="text"/>
  <input type="button" value=" + "/>
</div>
```

```
h2.lista {
  margin-bottom: 16px;
}

h2.form {
  margin-top: 48px;
  margin-bottom: 8px;
}

li {
  margin: 8px -40px;
  list-style: none;
}

span {
  cursor: pointer;
}

span.done {
  color: lightgray;
  text-decoration: line-through;
}
```

```
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 }
    ]
  }
})
```

Notem que clicar no check-box não altera o estado da tarefa. Precisamos de um método na aplicação Vue para alterar o estado. Este método deve ser ligado nas *checkbox* e nos *span* usando a notação @.

```
<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"/>
  <input type="button" value=" + "/>
</div>
```

```
h2.lista {
  margin-bottom: 16px;
}

h2.form {
  margin-top: 48px;
  margin-bottom: 8px;
}

li {
  margin: 8px -40px;
  list-style: none;
}

span {
  cursor: pointer;
}

span.done {
  color: lightgray;
  text-decoration: line-through;
}
```

```
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 }
    ]
  },
  methods: {
    toggle: function(task) {
      task.done = !task.done;
    }
  }
})
```

Também podemos esconder as tarefas concluídas usando a diretiva “v-show'. Este é apenas um exemplo de diretivas condicionais. Outros exemplos são “v-if', “v-else-if' e “v-else'.

```
<div id='app'>
  <h2 class='lista'>Minhas Tarefas:</h2>
  <ul>
    <li v-for="task in tasks" v-show="!task.done">
      <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"/>
  <input type="button" value=" + "/>
</div>
```

```
h2.lista {
  margin-bottom: 16px;
}

h2.form {
  margin-top: 48px;
  margin-bottom: 8px;
}

li {
  margin: 8px -40px;
  list-style: none;
}

span {
  cursor: pointer;
}

span.done {
  color: lightgray;
  text-decoration: line-through;
}
```

```
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 }
    ]
  },
  methods: {
    toggle: function(task) {
      task.done = !task.done;
    }
  }
})
```

Agora podemos adicionar o código para incluir novas tarefas na lista.

```
<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"/>
  <input type="button" value=" + " @click="add"/>
</div>
```

```
h2.lista {
  margin-bottom: 16px;
}

h2.form {
  margin-top: 48px;
  margin-bottom: 8px;
}

li {
  margin: 8px -40px;
  list-style: none;
}

span {
  cursor: pointer;
}

span.done {
  color: lightgray;
  text-decoration: line-through;
}
```

```
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 = '';
    }
  }
})
```

As tarefas também podem ter conteúdo HTML usando a diretiva “v-html'.

```
<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)"
        v-html="task.title">
      </span>
    </li>
  </ul>

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

```
h2.lista {
  margin-bottom: 16px;
}

h2.form {
  margin-top: 48px;
  margin-bottom: 8px;
}

li {
  margin: 8px -40px;
  list-style: none;
}

span {
  cursor: pointer;
}

span.done {
  color: lightgray;
  text-decoration: line-through;
}
```

```
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 = '';
    }
  }
})
```

Também podemos adicionar uma tarefa ao pressionar ENTER usando a notação @ com o evento keyup e a tecla ENTER.

```
<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)"
        v-html="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>
```

```
h2.lista {
  margin-bottom: 16px;
}

h2.form {
  margin-top: 48px;
  margin-bottom: 8px;
}

li {
  margin: 8px -40px;
  list-style: none;
}

span {
  cursor: pointer;
}

span.done {
  color: lightgray;
  text-decoration: line-through;
}
```

```
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 = '';
    }
  }
})
```

Finalmente, Vue.js permite definir filtros, que podem ser aplicados na notação “double-mustache'. A aplicação dos filtros em diretivas não é tão simples e está fora do nosso escopo (embora você possa ver rapidamente pesquisando no Google).

```
<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 | capitalize }}
      </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>
```

```
h2.lista {
  margin-bottom: 16px;
}

h2.form {
  margin-top: 48px;
  margin-bottom: 8px;
}

li {
  margin: 8px -40px;
  list-style: none;
}

span {
  cursor: pointer;
}

span.done {
  color: lightgray;
  text-decoration: line-through;
}
```

```
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 = '';
    }
  },
  filters: {
    capitalize: function (value) {
      return value.charAt(0).toUpperCase() + value.slice(1)
    }
  }
})
```

Agora é a sua vez de mostrar que aprendeu os recursos básicos do Vue.JS ...

1) Desenvolva uma aplicação para converter temperaturas expressas em graus Celsius para graus Fahrenheit.

2) Desenvolva uma aplicação para controlar despesas e receitas em uma conta corrente. A aplicação deve manter uma lista de transações, cada qual representada por uma descrição e um valor. Valores positivos representam créditos, enquanto valores negativos representam débitos.
   A aplicação deve apresentar a lista de débitos e créditos para o usuário, mantendo a ordem em que as transações foram registrados pelo usuário. Ao lado de cada transação deve ser apresentado o seu saldo, calculado como o somatório de todos os valores até a transação atual.
   Novas transações devem ser recebidas de um formulário contendo dois campos, sendo o primeiro para o histórico da nova transação e o segundo para entrar o valor da transação.

3) Funcionalidades complementares para o projeto de controle de despesas:

   a) Implementar a funcionalidade de remover uma transação.

   b) Implementar a funcionalidade de trocar a ordem das transações, subindo ou descendo uma transações na lista.

   c) Apresente os créditos em verde e os créditos em vermelho na lista de transações.

   d) Apresente saldos negativos com cor de fundo vermelha e cor de texto amarela.

   e) Faça com que o campo de valor do formulário fique com o fundo vermelho quando ele não contiver um valor numérico.