Israel Ferreira de Moraes	2018003534
Ivan Leoni Vilas Boas	2018009073
Marcelo Cavalca Filho	33161
Sandro Ricardo dos Reis	2017005104
Ygor Salles Aniceto Carvalho	2017014382

Algumas características do Vue.js

- A biblioteca principal é focada exclusivamente na camada visual (view layer);
- Requer uma configuração mínima na criação de um projeto;
- Fácil de integrar com uma aplicação ou bibliotecas já existentes através de uma simples tag script;
- Versátil, modular, com estrutura limpa;
- Fácil aprendizado e integração de novos membros a equipe;
- É reativo: Isso quer dizer que qualquer mudança feita nos elementos do framework irá alterar automaticamente todos os locais em que esse item aparecerá para o usuário.

O que é Vue.js

- Vue JS é um framework Javascript progressivo e open source, utilizado para a construção de interfaces de usuário.

- Foi lançado em Fevereiro de 2014 por Evan You, desenvolvedor que atuava em um dos projetos do Google no AngularJS.

- Descobriu a necessidade de criar uma ferramenta mais completa e ágil para lidar com varias e grandes interfaces de usuário

Onde Utilizar ?

- Utilizado para criar aplicações *single page* (página única);

- E também para desenvolver vários tipos de interfaces, que possuem necessidades de maior interação e experiência mais valorosa para o usuário.

- Com mais de 150k de estrelas no Github, Vue.JS já está entre os frameworks Javascript para criação de interfaces mais populares do mundo.

Utilização do Vue.js

Várias empresas brasileiras já fazer uso da tecnologia que o Vue fornece;

A lista completa pode ser encontrada neste link : https://empresas-usando-vuejs.netlify.app/

Arquitetura do Vue.js

- **Arquitetura enxuta:**

➔ Aplicações são constituídas de **componentes** criados com a sintaxe **HTML, CSS e Javascript** em um **único** arquivo .vue. Tornando assim fácil o isolamento e a manutenção de suas funcionalidades.

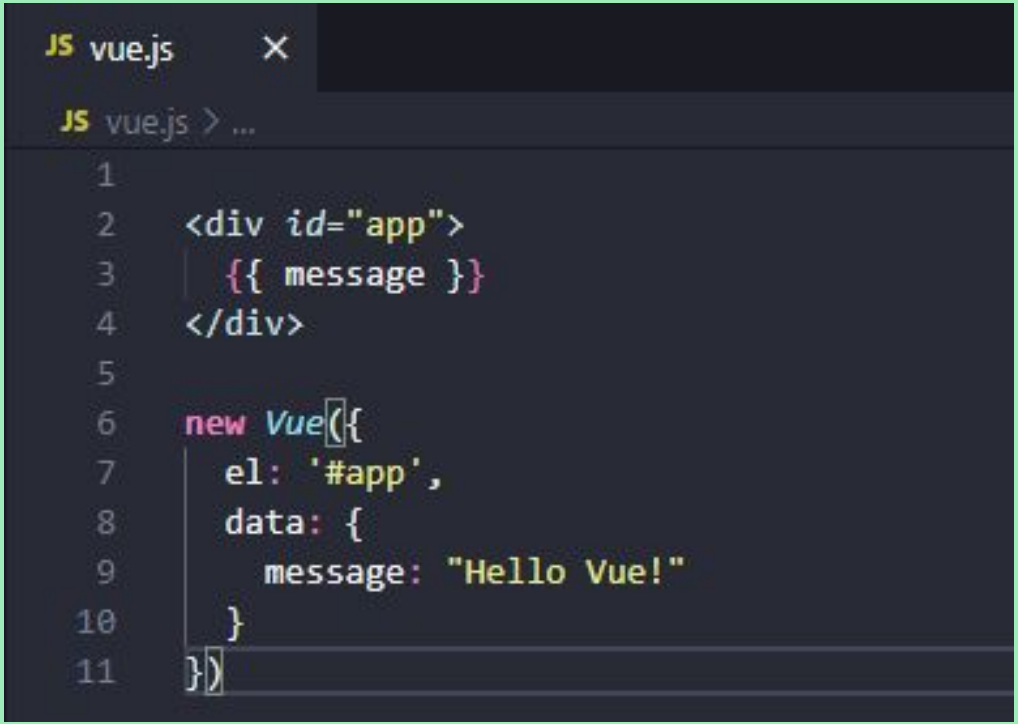
➔ Cada componente constitui um escopo isolado dos demais, tanto em lógica quantos nos estilos.

Arquitetura do Vue.js

- ➔ A renderização dos dados é feita baseada em uma virtual DOM que é atualizada apenas quando os dados de um componentes são alterados, aumentando muito assim o desempenho e descartando atualizações desnecessárias.

- ➔ Cada componente é criado usando a sintaxe HTML para estruturação com os dados atrelados via Javascript, o que supre as limitações do HTML como a capacidade de iterar sobre uma coleção de dados ou decidir se uma tag deve ser renderizada ou não.

Exemplo simples de um componente Vue.js



- O Vue renderiza declarativamente os dados no DOM (Document Object Model) usando uma sintaxe de *template* simples : {{atributo}}
- O identificador do *el* = “app”, (corresponde ao ID da div) definirá que tudo dentro da div #app será contemplado pelo Vue
- Data: estabelece onde ficará as variáveis

Instalação do Vue.js

- Vue não suporta IE8 e versões anteriores, pois usa funcionalidades ECMAScript 5 incompatíveis nestes. Entretanto, suporta todos os navegadores compatíveis com ECMAScript 5.

- A instalação consiste simplesmente em adicionar o pacote do Vue à tag script. Ou seja, é só colar o caminho do Vue.js (fonte) e o framework já estará instalado.

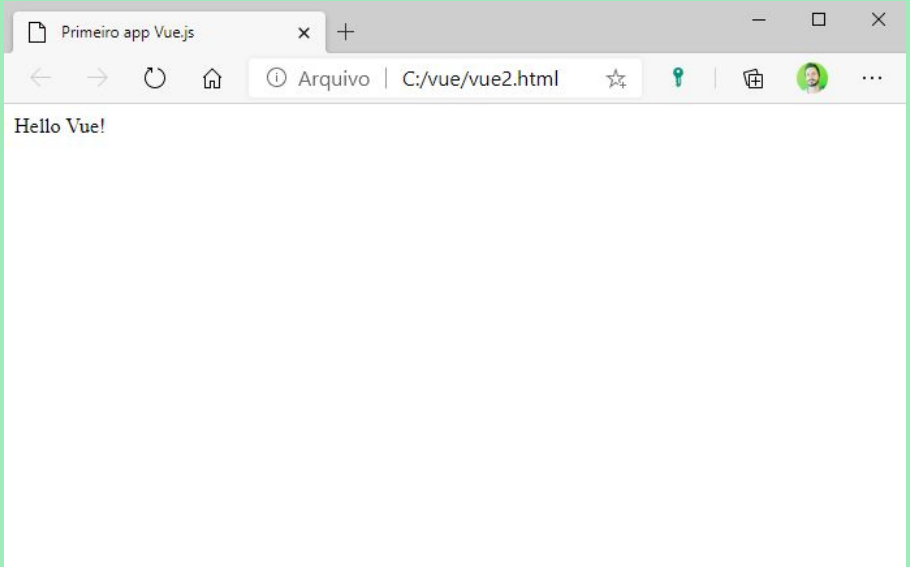
```
<script src="https://unpkg.com/vue"></script>
```

Instalação do Vue.js

vue.html ● vue2.html ×

vue2.html > ...

```
1  <html>
2      <head>
3        <title>Primeiro app Vue.js</title>
4        <script src="https://unpkg.com/vue"></script>
5      </head>
6      <body>
7            <div id="app">
8              {{ message }}
9            </div>
10           <script>
11                   new Vue({
12                       el: '#app',
13                       data: {
14                           message: "Hello Vue!"
15                       }
16                   })
17           </script>
18     </body>
19   </html>
```



Instalação do Vue.js

Com CDN

- Para propósitos de aprendizado, usaremos a versão mais recente com:

  <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>

Instalação do Vue.js

Com Inclusão Direta com <script>

- Realize o *download* da versão desenvolvedor em :

  https://br.vuejs.org/v2/guide/installation.html

- Depois inclua a *tag* **<script>** com o caminho correto para que o Vue seja registrado como uma variável global.

NPM: $ npm install vue

Instalação do Vue.js

CLI

- Possui um conjunto de configurações de *build* prontas para um processo de trabalho de *front-end* moderno.

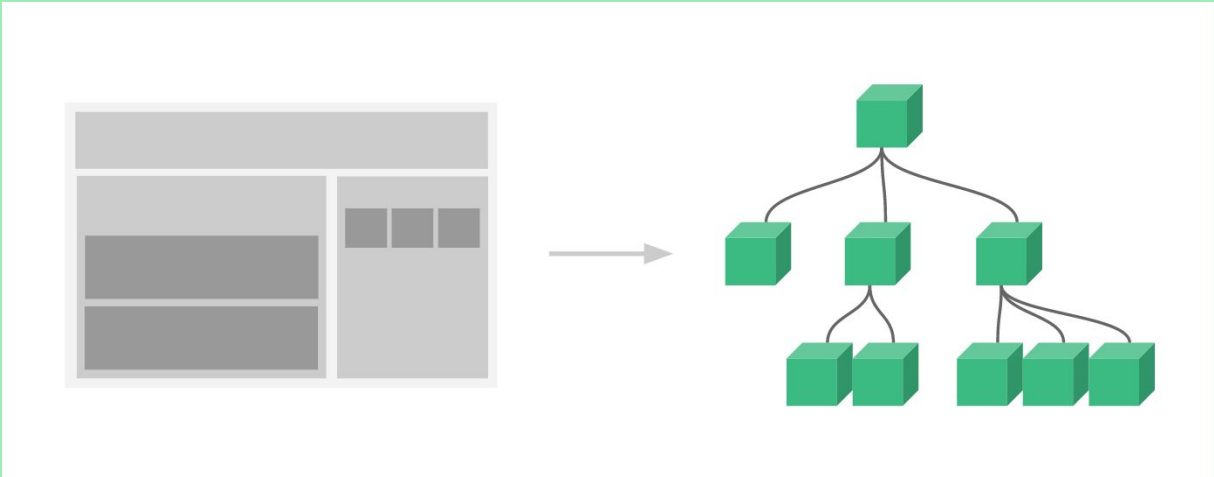
  $ npm install -g @vue/cli (ou $ yarn global add @vue/cli)

- Para a criação do projeto : $ vue create my-project (ou $ vue ui)

- Veja mais sobre a instalação do CLI em https://cli.vuejs.org/

Composição com componentes

➔ A construção de aplicações de larga escala são compostas por pequenos componentes, auto-contidos e frequentemente reutilizáveis. Assim quase qualquer tipo de interface de uma aplicação pode ser abstraída em uma árvore de componentes:



Composição com componentes

```
<html>
    <head>
      <title> Desvendando Vue.js</title>
      <script src="https://unpkg.com/vue"></script>
    </head>
    <body>
          <div id="app">
            <h1>{{titulo}}</h1>
            <menu-bar v-for="item in linguagens" v-bind:linguagem="item"></menu-bar>
          </div>
          <script>
            Vue.component('menu-bar',{
              props: ['linguagem'],
              template:"<ul><li>{{linguagem}}</li></ul>"
            });
            new Vue({
                        el: '#app',
                        data: {
                            titulo: "Aprendendo  componetes Reutilizáveis",
                            linguagens:["java","c++","PHP"]
                        },
            });
          </script>
    </body>
  </html>
```

No Vue, um componente é essencialmente uma instância Vue com opções predefinidas. Registrar um componente no Vue é simples

Composição com componentes

vue3.html > ...

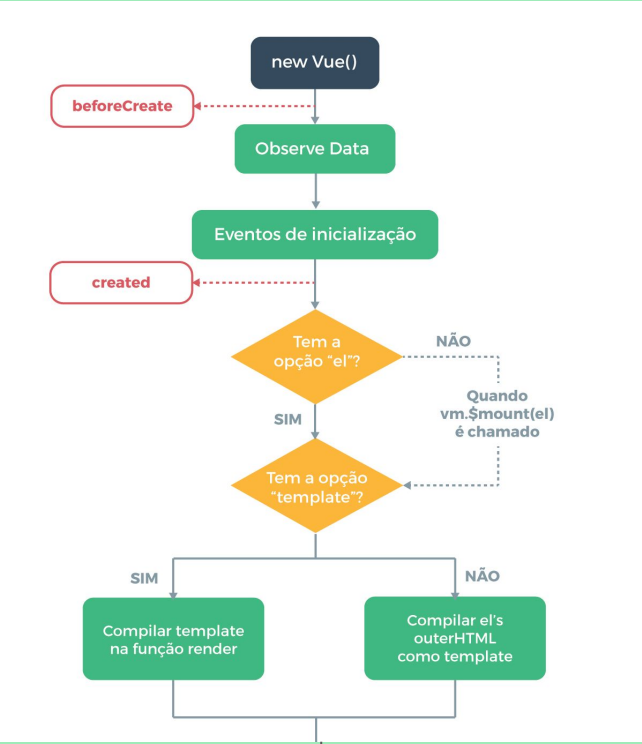
```
<html>
    <head>
      <title> Desvendando Vue.js</title>
      <script src="https://unpkg.com/vue"></script>
    </head>
    <body>
          <div id="app">
            <h1>{{titulo}}</h1>
            <menu-bar v-for="item in linguagens" v-bind:linguagem="item"></menu-bar>
          </div>
          <template id="linguas">
              <ul>
                  <li>{{linguagem}}</li>
              </ul>
          </template>
          <script>
            Vue.component('menu-bar',{
              props: ['linguagem'],
              template:"#linguas"
            });
            new Vue({
                        el: '#app',
                        data: {
                            titulo: "Aprendendo  componetes Reutilizáveis",
                            linguagens:["java","c++","PHP"]
                        },
            });
          </script>
    </body>
  </html>
```

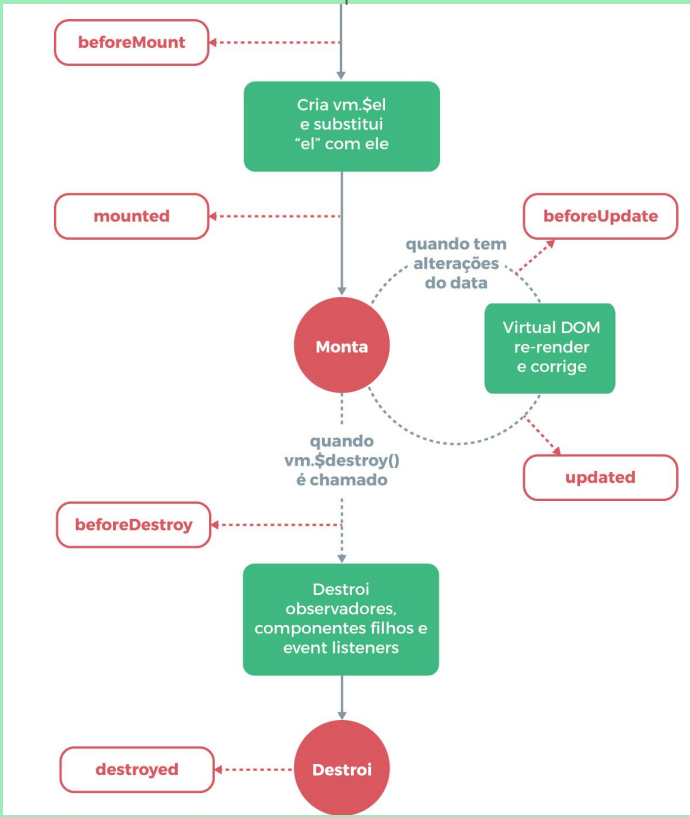
Código melhor estruturado

Ciclo de vida da instância

- Cada instância Vue passa por uma série de etapas em sua inicialização - por exemplo, é necessário configurar a observação de dados, compilar o *template*, montar a instância no DOM, atualizar o DOM quando os dados forem alterados. Ao longo do caminho, ocorrerá a invocação de alguns gatilhos de ciclo de vida, oferecendo a oportunidade de executar lógicas personalizadas em etapas específica.
- Por exemplo, o gatilho *created* pode ser utilizado para executar código logo após a instância ser criada. Existem outros gatilhos que são chamados em diferentes etapas do ciclo de vida da instância, como *mounted, updated e destroyed*. Qualquer gatilho de ciclo de vida é executado com seu contexto **this** apontando para a instância Vue que o invocou.

Ciclo de vida da instância





Sintase de Templates

- É baseada em HTML, permitindo que você vincule declarativamente o DOM renderizado aos dados da instância Vue. Todos os *templates* do Vue.js são compostos por HTML válido que pode ser compilado por navegadores compatíveis com as especificações e também por compiladores HTML.
- Internamente, Vue compila os *templates* dentro de funções de renderização de Virtual DOM. Combinado com o sistema de reatividade, Vue é capaz de identificar de forma inteligente a menor quantidade possível de componentes a serem “re-renderizados” e aplica o mínimo possível de manipulações DOM quando o estado da aplicação muda.
- Também é possível escrever diretamente funções de renderização em vez de utilizar *templates*.

Sintase de Templates

#Texto

O mais básico *data binding*, interpolando texto com a sintaxe *Mustache* (chaves duplas):

```
<span>Mensagem: {{ msg }}</span>
```

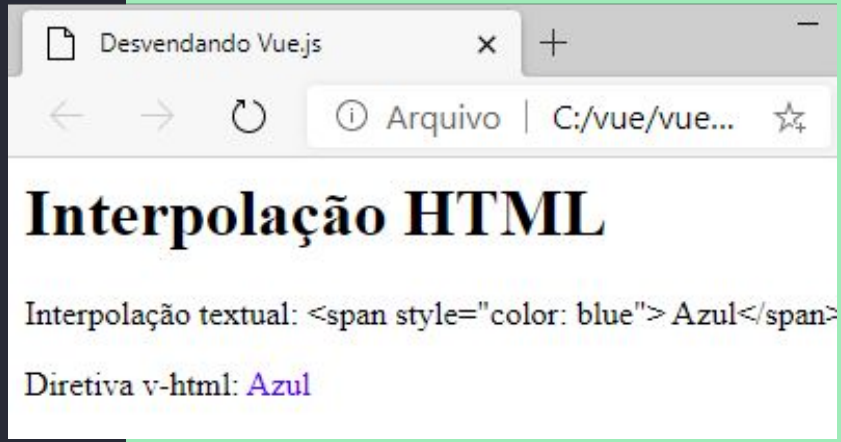
#HTML

Para que você exiba HTML, utilize a diretiva **v-html**

Sintase de Templates: HTML

vue4.html > ...

```
1   <html>
2       <head>
3         <title> Desvendando Vue.js</title>
4         <script src="https://unpkg.com/vue"></script>
5       </head>
6       <body>
7             <div id="app">
8               <h1>{{titulo}}</h1>
9               <p>Interpolação textual: {{ rawHtml }}</p>
10              <p>Diretiva v-html: <span v-html="rawHtml"></span></p>
11            </div>
12            <script>
13              new Vue({
14                        el: '#app',
15                        data: {
16                            titulo: "Interpolação HTML",
17                            rawHtml: '<span style="color: blue"> Azul</span>'
18                        },
19              });
20            </script>
21      </body>
22    </html>
```



Sintase de Templates

#Atributo

Chaves duplas não podem ser usadas em atributos HTML. Para isso, utilize a diretiva v-bind (Como visto em diretivas)

obs.: No caso de atributos booleanos, onde sua mera existência implica em true, v-bind funciona um pouco diferente.

```
<button v-bind:disabled="isDisabled">Botão</button>
```

Se **isDisabled** possui um valor null, undefined ou false, o atributo disabled nem mesmo será incluído no elemento <button> renderizado.

Sintase de Templates

 # Expressões JavaScript

➔ O Vue.js suporta todo o poder das expressões JavaScript dentro de qualquer tipo de *data binding*

```
{{ number + 1 }}

{{ ok ? 'SIM' : 'NÃO' }}

{{ message.split('').reverse().join('') }}

<div v-bind:id=" 'list - ' + id"></div>
```

Diretivas do Vue.js

- Na manipulação de dados são utilizadas diretivas diretamente integradas ao HTML via Javascript de **forma dinâmica** por meio de interpolação, essa integração existe para dar maior flexibilidade à linguagem de marcação.

- Existem **diretivas específicas** para cada utilização, como para renderização condicional: *v-for para interação, v-if*, ou para demonstração de textos: v-text e para conexão com eventos: *v-on:event*, onde o *event* seria o evento que chamaria a função desejada (click, submit, scroll).

Diretiva v-if

vue.html | vue2.html

vue.html > html > body

```
1
2   <html>
3     <head>
4       <title>Primeiro app Vue.js</title>
5       <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
6     </head>
7     <body>
8           <div id="app">
9             <h1 v-if='status'>{{ message }}</h1>
10          </div>
11
12          <script>
13                  new Vue({
14                      el: '#app',
15                      data: {
16                          message: "Hello Vue!",
17                          status: true
18                      }
19                  })
20          </script>
21    </body>
22  </html>
```

Com o v-if no código ao lado e se status: false a mensagem não aparecia para na tela do usuário e nem no código.

Diretiva v-show

vue.html | vue2.html

vue.html > ...

```
1
2   <html>
3     <head>
4       <title>Primeiro app Vue.js</title>
5       <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
6     </head>
7     <body>
8           <div id="app">
9             <h1 v-show='status'>{{ message }}</h1>
10          </div>
11
12          <script>
13                  new Vue({
14                      el: '#app',
15                      data: {
16                          message: "Hello Vue!",
17                          status: false
18                      }
19                  })
20          </script>
21    </body>
22  </html>
```

v-if e v-show tem mesmo resultado para o usuário final, a diferença está em que v-show torna o display: none, ou seja, no código ao lado com o if-show e com status: false a mensagem não aparecia na tela do usuário, mas apareceria no código.

Diretiva v-if / v-else

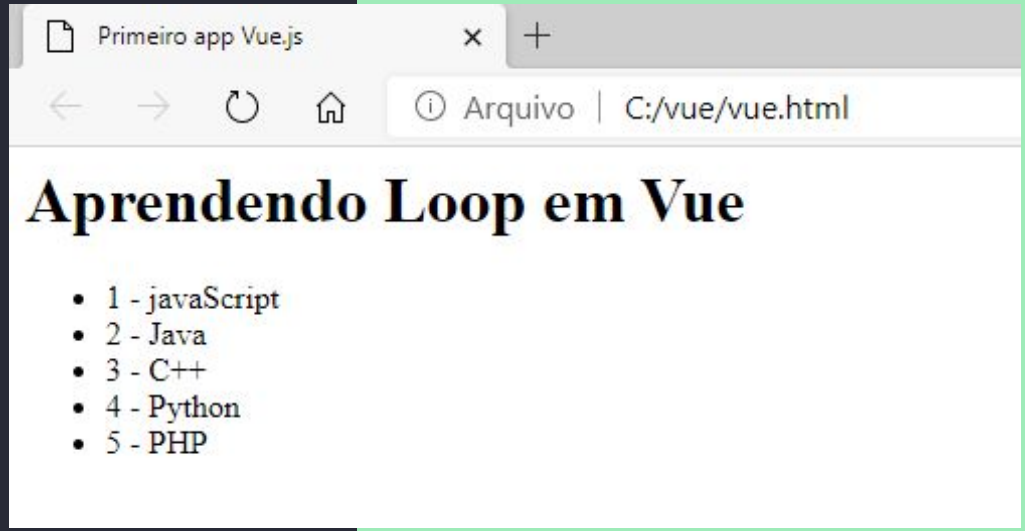
```
7   <body>
8         <div id="app">
9           <h1>{{ titulo }}</h1>
10          <p v-if = "usuario.id == 1">
11            ID: {{usuario.id}}, <br>
12            Nome: {{usuario.nome}}, <br>
13            Profissão: {{usuario.profissao}}, <br>
14            Idade: {{usuario.idade}}, <br>
15            Email: {{usuario.email}}, <br>
16          </p>
17          <p v-else>
18            Usuário dom ID == 1 não encontrado!
19          </p>
20        </div>S
21        <script>
22                new Vue({
23                    el: '#app',
24                    data: {
25                      titulo: "Cadastro de Login",
26                      usuario:{
27                        id:1,
28                        nome: "Ivan Leoni",
29                        profissao: "estudante",
30                        idade: 30,
31                        email: "ivanleoni30@unifei.edu.br"
32                      }
33                    }
34                })
35        </script>
36  </body>
```

Diretiva v-if / v-else

```
7   <body>
8       <div id="app">
9         <h1>{{ titulo }}</h1>
10        <p v-if = "usuario.id == 1">
11          ID: {{usuario.id}}, <br>
12          Nome: {{usuario.nome}}, <br>
13          Profissão: {{usuario.profissao}}, <br>
14          Idade: {{usuario.idade}}, <br>
15          Email: {{usuario.email}}, <br>
16        </p>
17        <p v-else>
18          Usuário com ID == 1 não encontrado!
19        </p>
20      </div>
21      <script>
22              new Vue({
23                  el: '#app',
24                  data: {
25                    titulo: "Cadastro de Login",
26                    usuario:{
27                      id:2,
28                      nome: "Ivan Leoni",
29                      profissao: "estudante",
30                      idade: 30,
31                      email: "ivanleoni30@unifei.edu.br"
32                    }
33                  }
34              })
35      </script>
36  </body>
```

Diretiva v-for

```
<body>
        <div id="app">
          <h1>{{ titulo }}</h1>
          <ul>
            <li v-for="(item, index) in linguagens">{{index+1}} - {{item.aprender}}</li>
          </ul>
        </div>
        <script>
                new Vue({
                    el: '#app',
                    data: {
                      titulo: "Aprendendo Loop em Vue",
                      linguagens:[
                        {aprender:"javaScript"},
                        {aprender:"Java"},
                        {aprender:"C++"},
                        {aprender:"Python"},
                        {aprender:"PHP"},
                      ]
                    }
                })
        </script>
</body>
```



Diretiva v-model

- Com o Vue 3, a API para vinculação de dados bidirecional foi padronizada para reduzir a confusão e permitir aos desenvolvedores mais flexibilidade com a diretiva v-model.

- O v-model para criar interligações de mão dupla (*two-way binding*) entre os dados e elementos *input*, *textarea* e *select* de formulários.

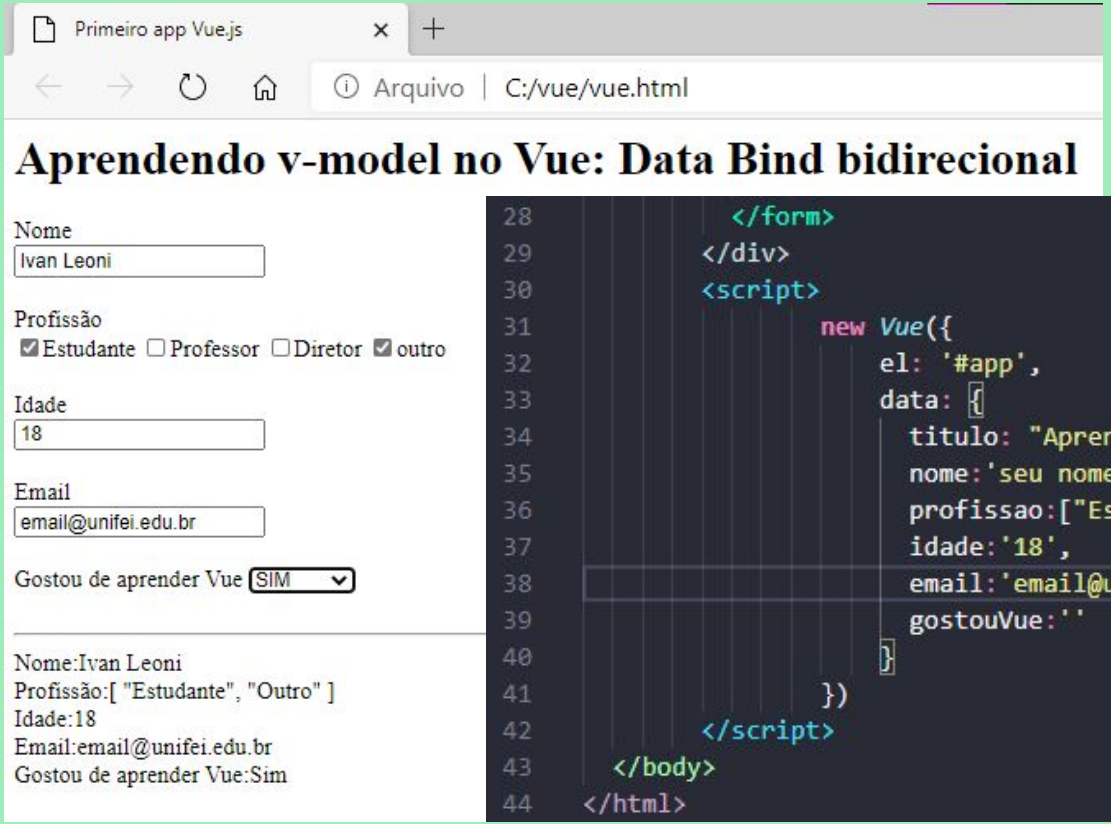
- A diretiva automaticamente busca a maneira correta de atualizar o elemento com base no tipo de entrada.

Diretiva v-model

vue.html > html > body > script > data

```
<html>
  <head>
    <title>Primeiro app Vue.js</title>
    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
  </head>
  <body>
        <div id="app">
          <h1>{{ titulo }}</h1>
          <form>
                <label for='nome'>Nome</label><br><input type="text" v-model="nome"><br><br>
                <label for='profissao'>Profissão</label><br>
                <input type ="checkbox" value = "Estudante" v-model="profissao">Estudante</imput>
                <input type ="checkbox" value = "Professor" v-model="profissao">Professor</imput>
                <input type ="checkbox" value = "Diretor" v-model="profissao">Diretor</imput>
                <input type ="checkbox" value = "Outro" v-model="profissao">outro</imput>
                <br><br><label for=''>Idade</label><br><input type="text" v-model="idade"><br><br>
                <label for=''>Email</label><br><input type='text' v-model="email">  <br><br>
                <label for='gostou'>Gostou de aprender Vue</label>
                <select v-model="gostouVue">
                  <option value="">Escolha</option>
                  <option value="Sim">SIM</option>
                  <option value="Nao">Não</option>
                </select>
                <br><br><hr>
                Nome:{{nome}} <br>Profissão:{{profissao}} <br> Idade:{{idade}}<br>
                Email:{{email}} <br> Gostou de aprender Vue:{{gostouVue}} <br>
          </form>
```

Diretiva v-model




```
28            </form>
29          </div>
30          <script>
31                  new Vue({
32                      el: '#app',
33                      data: {
34                        titulo: "Aprendendo v-model no Vue: Data Bind bidirecional",
35                        nome:'seu nome',
36                        profissao:["Estudante"],
37                        idade:'18',
38                        email:'email@unifei.edu.br',
39                        gostouVue:''
40                      }
41                  })
42          </script>
43    </body>
44  </html>
```

Diretiva v-bind (:)

- Uma necessidade comum de interligação de dados é manipular as classes dos elementos e seus estilos *inline*. Uma vez que ambos são atributos, podemos usar v-bind para lidar com eles.

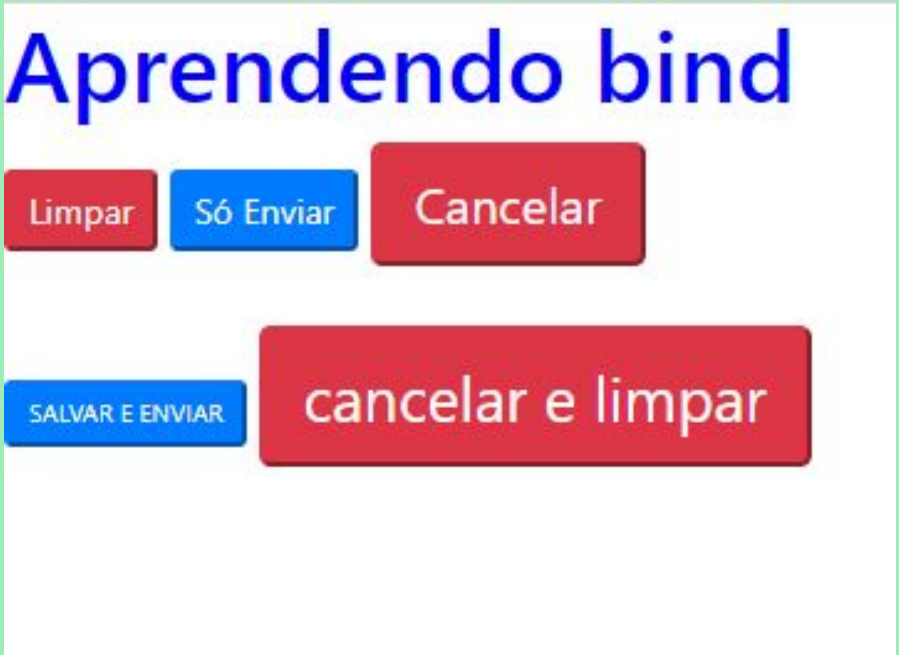
- Vue fornece aprimoramentos especiais quando v-bind é usado com class e style. Além de Strings, as expressões também podem avaliar Objetos ou Arrays.

Diretiva v-bind (:)

```
<html>
    <head>
      <title>Primeiro app Vue.js</title>
      <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css"
      integrity="sha384-TX8t27EcRE3e/ihU7zmQxVncDAy5uIKz4rEkgIXeMed4M0jlfIDPvg6uqKI2xXr2" crossorigin="anonymous">
      <script src="https://unpkg.com/vue"></script>
    </head>
    <style>
      .red { color: red}
      .blue {color: blue}
    </style>
    <body>
          <div id="app">
            <h1 v-bind:class="cor">{{ message }}</h1>
            <button v-bind:class= "clear">Limpar</button>
            <button :class="ClassEnviar"> Só Enviar</button>
            <button :class="ClassCancelar">Cancelar</button><br><br>
            <button :class="ClassEnviar" v-bind:style="StyleEnviar">Salvar e Enviar</button>
            <button :class="ClassCancelar" v-bind:style="StyleCancelar">Cancelar e Limpar</button>
          </div>
          <script>
                  //Incluir o codigo do proximo slide aqui
          </script>
    </body>
  </html>
```

Diretiva v-bind (:)

```
<script>
        new Vue({
            el: '#app',
            data: {
                message: "Aprendendo bind",
                cor:"blue",
                //com objeto
                clear: {
                "btn-danger":true,
                "btn-sm": true
                },
                //com array
                ClassEnviar:[
                "btn-primary","btn-sm"
                ],
                //com objeto dentro do array
                ClassCancelar:[
                "btn-danger",
                {"btn-lg": true}
                ],
                StyleEnviar: {
                "font-size":"10px",
                "text-transform":"uppercase"
                },
                StyleCancelar:[
                {"font-size":"25px"},
                {"text-transform":"lowercase"}
            ]
            },
        });
</script>
```



Métodos em Vue (methods)

➔ Um **método Vue** é uma função associada à instância **Vue**.

➔ Os **métodos** são especialmente úteis quando você precisa executar uma ação e anexar uma diretiva v-on em um elemento para manipular eventos

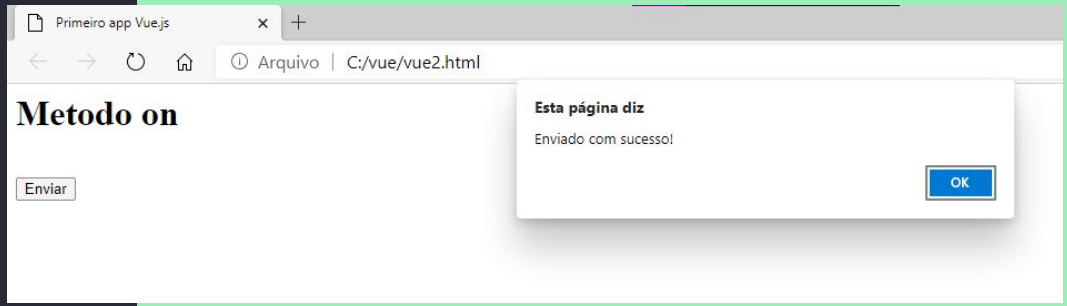
```
<template> <a v-on:click="clickAlert('Aprendendo Metodo')">Clique aqui!</a></template>

new Vue({
        methods: {
            clickAlert: function(text) { alert(text) }
        }
})
```

Diretiva v-on (@)

```html
<html>
    <head>
      <title>Primeiro app Vue.js</title>
      <script src="https://unpkg.com/vue"></script>
    </head>
    <body>
          <div id="app">
            <h1>{{ message }}</h1> <br>
            <button v-on:click="enviar()">Enviar</button>
          </div>
          <script>
                  new Vue({
                      el: '#app',
                      data: {
                          message: "Metodo on"
                      },

                      methods:{
                        enviar(){
                          alert("Enviado com sucesso!");
                        },
                      }
                  })
          </script>
    </body>
  </html>
```



Abreviações Úteis:

- ➔ O prefixo **v-** serve como dica visual para identificar atributos específicos do Vue nos *templates*. Isso é útil quando se está utilizando o Vue para aplicar comportamento dinâmico em HTML existente.

- ➔ Porém o uso do prefixo v- se torna menos importante quando se está construindo uma SPA, onde o Vue gerencia cada *template*.

- ➔ Vue oferece duas abreviações especiais para as diretivas mais utilizadas:
  - v-bind pode ser abreviado por dois pontos :
  - v-on pode ser abreviado por @

Dados Computados

- ➔ Expressões dentro de *templates* são muito convenientes, mas são destinadas a operações simples. Colocar muita lógica neles pode fazer com que fiquem inchados e que a sua manutenção fique mais complicada.

- ➔ Dados computados são cacheados de acordo com suas dependências reativas. Um dado computado somente será reavaliado quando alguma de suas dependências for alterada

Dados Computados. Exemplo 1

vue5.html > html > body > div#app > br

```
<html>
    <head>
      <title> Desvendando Vue.js</title>
      <script src="https://unpkg.com/vue"></script>
    </head>
    <body>
          <div id="app">
            <h1>{{titulo}} -- {{toUpperCase}}</h1>
            <label for='nome'>Nome</label><br>
            <input type="text" v-model="nome"><br><br>
            <label for='sobrenome'>Sobre Nome</label><br>
            <input type="text" v-model="sobreNome"><br><br><hr>
            <p>Nome Completo: {{nomeCompleto}}</p>
          </div>
          <script>
            new Vue({
                      el: '#app',
                      data: {
                          titulo: "Computed",
                          nome: "",
                          sobreNome: ""                    },
                      //sempre que a dependencia é alterada a computer é executada
                      computed:{
                          toUpperCase(){
                              return this.titulo.toUpperCase();
                            },
                          nomeCompleto(){
                               return  this.nome + ' ' + this.sobreNome;
                          },
                      },
            });
          </script>
    </body>
  </html>
```

Computed -- COMPUTED

Nome

Ivan Leoni

Sobre Nome

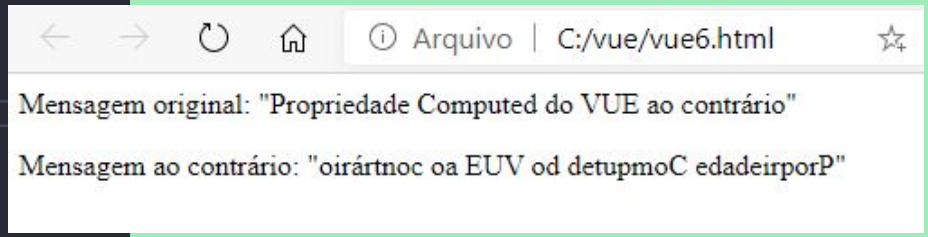
Martins

Nome Completo: Ivan Leoni Martins

Dados Computados. Exemplo 2

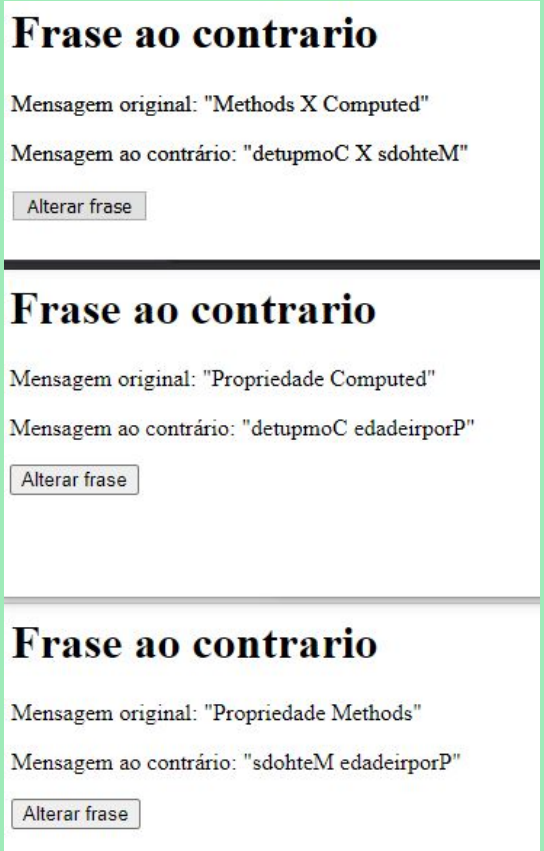
vue6.html > html > body > div#app > p

```
1  <html>
2      <head>
3        <title> Desvendando Vue.js</title>
4        <script src="https://unpkg.com/vue"></script>
5      </head>
6      <body>
7            <div id="app">
8                <p>Mensagem original: "{{ message }}"</p>
9                <p>Mensagem ao contrário: "{{ reversedMessage }}"</p>
10             </div>
11           </div>
12           <script>
13             new Vue({
14                       el: '#app',
15                       data: {
16                           message: "Propriedade Computed do VUE ao contrário",
17                       },
18                       //sempre que a dependencia é alterada a computer é executada
19                       computed: {
20                             // uma função "getter" computada (computed getter)
21                             reversedMessage: function () {
22                             // `this` aponta para a instância Vue da variável `vm`
23                                 return this.message.split('').reverse().join('')
24                             }
25                       }
26             });
27           </script>
28     </body>
29   </html>
```



Dados Computados x Métodos

```
6     <body>
7           <div id="app">
8               <h1>{{titulo}}</h1>
9                 <p>Mensagem original: "{{ message }}"</p>
10                <p>Mensagem ao contrário: "{{ reversedMessage }}"</p>
11                <button v-on:click="mudar()">Alterar frase</button>
12              </div>
13          </div>
14          <script>
15            new Vue({
16                      el: '#app',
17                      data: {
18                          titulo:"Frase ao contrario",
19                          message:"Methods X Computed",
20                          aux:0
21                      },
22                      //sempre que a variavel menssage é alterada a computer é executada
23                      computed: {
24                            // uma função "getter" computada (computed getter)
25                            reversedMessage: function () {
26                                return this.message.split('').reverse().join('')
27                            }
28                      },
29                      //Somente quando o click e dado a função é acionada
30                      methods:{
31                          mudar(){
32                              if(this.aux==0){
33                                this.message = "Propriedade Methods",//executa computed
34                                this.aux=1
35                              }
36                              else{
37                                this.message= "Propriedade Computed",//executa computed
38                                this.aux=0
39                              }
40                          }
41                      }
42            });
43          </script>
44    </body>
```



Filtros

- ➔ Vue permite utilizar filtros para aplicação de formatações de texto corriqueiras.

- ➔ Filtros são permitidos em interpolações mustache e expressões v-bind (sendo a última suportada em 2.1.0+).

- ➔ Filtros podem ser acrescidos ao final de uma expressão JavaScript, sendo denotados pelo símbolo “pipe”.

Filtros

```
<html>
    <head>
      <title> Desvendando Vue.js</title>
      <script src="https://unpkg.com/vue"></script>
    </head>
    <body>
          <div id="app">
            <h1>{{titulo | minusculo | concatena}}</h1>
            <menu-bar v-for="item in linguagens" v-bind:linguagem="item | maiuscula()"></menu-bar>
          </div>
          <script>
            Vue.component('menu-bar',{
              props: ['linguagem'],
              template:"<ul><li>{{linguagem}}</li></ul>"
            });
            new Vue({
                        el: '#app',
                        data: {
                            titulo: "FILTROS",
                            linguagens:["java","c++","php","python"]
                        },
                        filters:{
                            maiuscula(str){
                                return str.toUpperCase();
                            },
                            minusculo(str){
                              return str.toLowerCase();
                            },
                            concatena(str){
                              return str.concat(" ", " podem ser usados em CADEIA");
                            },
                        },
            });
          </script>
    </body>
  </html>
```

filtros podem ser usados em CADEIA

- JAVA
- C++
- PHP
- PYTHON

Modificador de Eventos

- ➔ Vue fornece modificadores de evento para v-on. Modificadores são sufixos da diretiva, indicados após um ponto:
- ➔ .stop
- ➔ .prevent
- ➔ .capture
- ➔ .self
- ➔ .once
- ➔ .passive

Modificador de Eventos - Exemplos

.stop para que propagação do evento click será interrompida:

```
<a v-on:click.stop="doThis"></a>
```

.prevent para que o evento submit deixe de recarregar a página:

```
<form v-on:submit.prevent="onSubmit"></form>
```

Os Modificadores podem ser usados encadeados :

```
<a v-on:click.stop.prevent="doThat"></a>
```

Modificador de Eventos - Exemplos

.capture usa modo de captura ao adicionar o evento ou seja, um evento em um elemento interno é tratado aqui após ser tratado por aquele elemento:

```
<div v-on:click.capture="doThis">...</div>
```

.self que só aciona o manipulador se event.target é o próprio elemento, isto é, não aciona a partir de um elemento filho:

```
<div v-on:click.self="doThat">...</div>
```

Modificador de Teclado

- Quando escutamos eventos do teclado, precisamos muitas vezes verificar a ocorrência de teclas específicas. O Vue também permite a adição de modificadores v-on ao escutar eventos de teclado:
  - enter
  - .tab
  - .delete (captura tanto “Delete” quanto “Backspace”)
  - .esc
  - .space
  - .up
  - .down
  - .left
  - .right

Modificador de Teclado - Exemplos

- Só fará chamada a submit() quando a `key` é `Enter`:

```
<input v-on:keyup.enter="submit">
```

- O manipulador só será chamado se $event.key for igual a 'PageDown'.

```
<input v-on:keyup.page-down="onPageDown">
```

Modificador de mouse de Sistema

- O Vue permite utilizar os modificadores que restringem o manipulador de eventos ser disparados por um botão específico do *mouse*:
  - .left
  - .right
  - .middle

- O Vue aciona algum eventos de *mouse* ou teclado apenas quando o modificador correspondente estiver acionado:
  - .ctrl
  - .alt
  - .shift
  - .meta

Modificador de Sistema - Exemplo

```
<!-- Alt + C -- >

                    <input v-on:keyup.alt.67="clear">


<!-- Ctrl + Click -- >

                    <div v-on:click.ctrl="doSomething">Faça alguma coisa</div>
```

Porque Escutas no HTML?

Como todas as funções de manipuladores e expressões Vue são estritamente ligadas ao *ViewModel* que está manipulando o modo de exibição atual, as escutas não causaram qualquer dificuldade de manutenção.

Os benefícios em usar v-on no *template*:

1. É mais fácil localizar as implementações de função de manipulador dentro de seu código JS deslizando sobre o *template* HTML.
2. Não tem que anexar manualmente escutas a eventos em JS, seu código de *ViewModel* pode conter apenas a lógica pura e está livre de manipulação DOM. Isto torna mais fácil de testar.
3. Quando um *ViewModel* é destruído, todas escutas a eventos são removidas automaticamente. Você não precisa se preocupar em removê-las explicitamente.

Interligação em Formulários

➔ O v-model usa internamente diferentes propriedades e emite diferentes eventos para diferentes elementos *input*:

- os elementos *text* e *textarea* usam a propriedade value e o evento input;
- *checkboxes* e *radiobuttons* usam a propriedade checked e o evento change;
- campos de seleção usam value como prop e change como um evento.

Interligação em Formulários

- Input :

```
<input v-model="message" placeholder="Me edite">
<p>A mensagem é: {{ message }}</p>
```

- Textarea:

```
<span>Mensagem com múltiplas linhas:</span>
<p style="white-space: pre-line;">{{ message }}</p>
<br>
<textarea v-model="message" placeholder="Escreva bastante"></textarea>
```

Interligação em Formulários

- Checkboxes:

```
<div id="inteligacaoFormCheckboxes">
  <input type="checkbox" id="jack" value="Jack" v-model="checkedNames">
  <label for="jack">Jack</label>
  <input type="checkbox" id="john" value="John" v-model="checkedNames">
  <label for="john">John</label>
  <input type="checkbox" id="mike" value="Mike" v-model="checkedNames">
  <label for="mike">Mike</label>  <br>
  <span>Nomes assinalados: {{ checkedNames }}</span>
</div>
```

```
new Vue({
  el: '#inteligacaoFormCheckboxes',
  data: {
    checkedNames: []
  }
})
```

Interligação em Formulários

- Radio:

```
<input type="radio" id="one" value="Um" v-model="picked">
<label for="one">Um</label>
<br>
<input type="radio" id="two" value="Dois" v-model="picked">
<label for="two">Dois</label>
<br>
<span>Escolhido: {{ picked }}</span>
```

Interligação em Formulários

- Select:

```
<select v-model="selected">
  <option disabled value="">Escolha um item</option>
  <option>A</option>
  <option>B</option>
  <option>C</option>
</select>
<br>
<span>Selecionado: {{ selected }}</span>
```

```
new Vue({
  el: '...',
  data: {
    selected: ''
  }
})
```

Interligação em Formulários

Para *radio*, *checkbox* e *options* de *select*, os valores de vinculação do v-model são normalmente Strings estáticas (ou booleano no caso do *checkbox*).

```
<!-- `picked` é uma String "a" quando assinalado -->
<input type="radio" v-model="picked" value="a">

<!-- `toggle` é verdadeiro ou falso -->
<input type="checkbox" v-model="toggle">

<!-- `selected` é uma String "abc" se a primeira opção está selecionada -->
<select v-model="selected">
  <option value="abc">ABC</option>
</select>
```

Interligação em Formulários

- *checkbox* :

```
<input
  type="checkbox"
  v-model="toggle"
  true-value="sim"
  false-value="não"
>
// quando está assinalado:
vm.toggle === 'sim'
// quando não está
assinalado:
vm.toggle === 'não'
```

- *Select*:

```
<select v-model="selected">
  <option v-bind:value="{ number: 123}">123 </option>
</select>
// quando está assinalado:
typeof vm.selected // => 'object'
vm.selected.number // => 123
```

- *Radio*:

```
<input type="radio" v-model="pick" v-bind: value="a" >
// quando está assinalado:
vm.pick === vm.a
```

Modificadores

.lazy

Por padrão, v-model sincroniza o elemento com os dados após cada evento do tipo input. Mas adicionando o modificador lazy, a sincronização ocorrerá *após* o evento change:

```
<!-- sincronizado depois do "change" ao invés de "input" -->
<input v-model.lazy="msg">
```

Modificadores

.number

Se quiser que a entrada do usuário seja automaticamente convertida para Number, pode adicionar o modificador number ao v-model do elemento:

```
<input v-model.number="age" type="number">
```

Isso é bastante útil, porque mesmo no caso de type="number", o valor retornado pelo HTML é sempre uma String. Se o valor não puder ser convertido através de parseFloat(), o valor original é retornado.

Modificadores

.trim

Se você quiser que a entrada do usuário seja automaticamente isenta de espaços no início e no fim do texto, você pode adicionar o modificador trim ao v-model do elemento:

```
<input v-model.trim="msg">
```

Execução real um vue **com cli**

Demonstração =+ 10 minutos de explicação (cavalca)

Conclusões:

- ➔ O Vue js é um framework Javascript, de código aberto, indicado para criação de componentes reativos para interfaces web. É flexível, simples e garante bom desempenho em aplicativos móveis.

- ➔ O Vue não é melhor nem pior que o angular ou React, cada um tem suas especificidades e o seu uso no projeto pode ser definido conforme melhor afinidade de cada um ou da própria organização.

FIM