Módulo 1: Introdução ao Vue.js

- diretiva v-bind, atribuindo valor em class e style

A diretiva `v-bind` no Vue.js é uma poderosa ferramenta que pode ser usada para vincular dinamicamente valores a atributos HTML, incluindo `class` e `style`. Essas duas propriedades são amplamente usadas para estilizar elementos na página. Ao utilizar `v-bind`, você pode controlar de forma dinâmica as classes CSS aplicadas a um elemento e até mesmo os estilos inline.

1. `v-bind` com `class`

A diretiva `v-bind` permite que você vincule classes CSS a elementos HTML de forma reativa. Isso é útil quando você quer aplicar diferentes estilos a um elemento com base em variáveis ou condições do seu aplicativo.

Sintaxe básica:

Você pode usar `v-bind` para aplicar uma ou várias classes dinamicamente a um elemento:

```
<p :class="classe">Este é um parágrafo com classe dinâmica.</p>
```

Aqui, o valor de `classe` é definido no objeto `data` no Vue.js. Dependendo do valor dessa variável, a classe será aplicada ao elemento.

Aplicando múltiplas classes:

Você pode vincular várias classes dinamicamente usando objetos ou arrays.

Com um objeto:

No caso de um objeto, você pode atribuir classes com base em condições booleanas:

```
<p :class="{ red: isRed, green: isGreen }">Texto dinâmico</p>
```

Aqui, as classes `red` e `green` serão aplicadas de acordo com os valores das variáveis `isRed` e `isGreen`. Se `isRed` for `true`, a classe `red` será aplicada ao elemento, e o mesmo vale para `isGreen`.

Com um array:

Você também pode usar um array para aplicar várias classes:

```
<p :class="[baseClass, conditionalClass]">Texto com várias classes</p>
```

Neste exemplo, `baseClass` e `conditionalClass` são variáveis que contêm os nomes das classes a serem aplicadas.

2. `v-bind` com `style`

Você pode usar `v-bind` para vincular dinamicamente estilos inline a elementos HTML. Isso permite que você controle as propriedades CSS de um elemento diretamente no JavaScript, tornando-as reativas.

Sintaxe básica:

Para vincular um estilo dinamicamente, você usa `v-bind:style`:

```
<p :style="{ color: textColor, fontSize: fontSize + 'px' }">Texto estilizado dinamicamente</p>
```

No exemplo acima:

- A cor do texto (`color`) é vinculada à variável `textColor` do Vue.
- O tamanho da fonte (`fontSize`) é calculado dinamicamente, onde o valor de `fontSize` é concatenado com a unidade `'px'`.

Estilizando múltiplas propriedades:

Você pode adicionar vários estilos de maneira simples, passando um objeto onde cada chave é o nome de uma propriedade CSS, e cada valor é a variável associada no Vue:

```
<p :style="{ color: 'red', backgroundColor: bgColor, padding: padding + 'px' }">Texto com estilos dinâmicos</p>
```

Aqui, `bgColor` e `padding` são variáveis controladas pelo Vue, permitindo que o fundo e o espaçamento interno (padding) do parágrafo mudem de forma reativa.

Sintaxe curta de `v-bind`

Você pode simplificar a sintaxe do `v-bind` para `:class` e `:style`. No lugar de `v-bind:class` e `v-bind:style`, você pode usar apenas `:class` e `:style`, como visto nos exemplos anteriores. Essa forma mais curta é amplamente utilizada por ser mais concisa.

Exemplo prático:

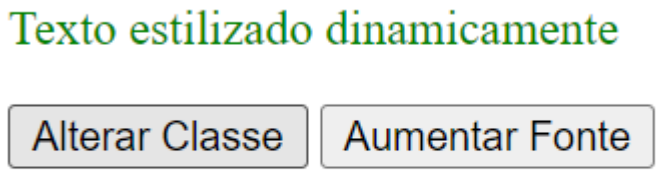
Aqui está um exemplo que ilustra como usar `v-bind` com `class` e `style`:

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>v-bind com style e class</title>
</head>
<style>
    .red {
      color: red;
    }
    .green {
      color: green;
    }
  </style>
```

```html
<body>
    <div id="app">
        <p :class="{ red: isRed, green: isGreen }" :style="{ fontSize: fontSize + 'px' }">
          Texto estilizado dinamicamente
        </p>
        <button @click="toggleClass">Alterar Classe</button>
        <button @click="increaseFontSize">Aumentar Fonte</button>
      </div>


      <script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>
      <script>
        new Vue({
          el: '#app',
          data: {
            isRed: true,
            isGreen: false,
            fontSize: 16
          },
          methods: {
            toggleClass() {
              this.isRed = !this.isRed;
              this.isGreen = !this.isGreen;
            },
            increaseFontSize() {
              this.fontSize += 2;
            }
          }
        })
      </script>

</body>
</html>
```

Ao rodarmos no navegador o nosso exemplo:

Quando clicamos no botão esquerdo com o nome “Alterar Classe” ocorre este efeito:



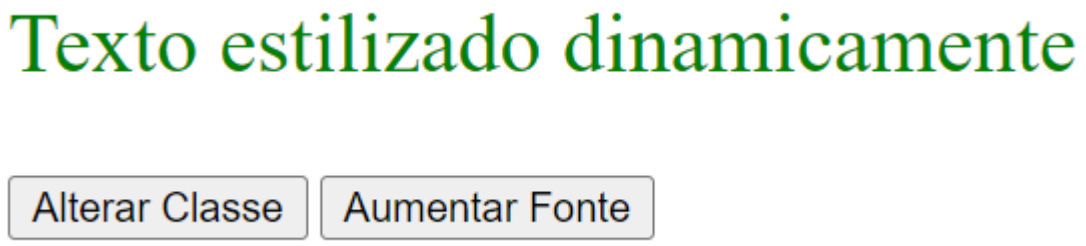
Fica alternando a cor pela configuração que colocamos, com um método de efeito toggle ou seja se não for uma cor será a outra, e com isso eles ficam invertendo toda vez que o botão é clicado.

```
<div id="app">
    <p :class="{ red: isRed, green: isGreen }" :style="{ fontSize: fontSize + 'px' }">
      Texto estilizado dinamicamente
    </p>
    <button @click="toggleClass">Alterar Classe</button>
    <button @click="increaseFontSize">Aumentar Fonte</button>
  </div>


  <script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>
  <script>
    new Vue({
      el: '#app',
      data: {
        isRed: true,
        isGreen: false,
        fontSize: 16
      },
      methods: {
        toggleClass() {
          this.isRed = !this.isRed;
          this.isGreen = !this.isGreen;
        },
        increaseFontSize() {
          this.fontSize += 2;
        }
      }
    })
  </script>
```

E quando clicamos no botão direito “Aumentar a fonte” ele irá aumentar gradativamente o fontsize que colocamos no style, pois chama o método que criamos.

```
body>
  <div id="app">
    <p :class="{ red: isRed, green: isGreen }" :style="{ fontSize: fontSize + 'px' }">
      Texto estilizado dinamicamente
    </p>
    <button @click="toggleClass">Alterar Classe</button>
    <button @click="increaseFontSize">Aumentar Fonte</button>
  </div>


  <script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>
  <script>
    new Vue({
      el: '#app',
      data: {
        isRed: true,
        isGreen: false,
        fontSize: 16
      },
      methods: {
        toggleClass() {
          this.isRed = !this.isRed;
          this.isGreen = !this.isGreen;
        },
        increaseFontSize() {
          this.fontSize += 2;
        }
      }
    })
  </script>
```

Ocasionando assim o aumento da fonte do elemento ao ser clicado.



Vamos criar outro exemplo para clarear um pouco mais sobre esse tema.

Agora vamos criar uma estrutura com botões utilizando bootstrap, para manipularmos os atributos do bootstrap por class e v-bind.

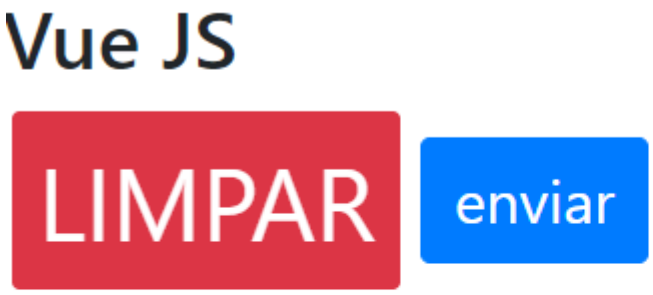
Estrutura do exemplo:

```html
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Curso Vue js -  diretiva v-bind, atribuindo valor em class e style</title>
      <!-- Bootstrap CSS -->
      <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css" integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4
</head>
<body>
    <div id="app">
      <h1>{{ titulo}}</h1>
    <button class="btn" v-bind:class="btnClassLimpar" v-bind:style="btnStyleLimpar">LIMPAR</button>
    <button class="btn" :class="btnClassEnviar" :style="btnStyleEnviar">ENVIAR</button>
    </div>
```

```html
<div id="app">
    <h1>{{ titulo}}</h1>
  <button class="btn" v-bind:class="btnClassLimpar" v-bind:style="btnStyleLimpar">LIMPAR</button>
  <button class="btn" :class="btnClassEnviar" :style="btnStyleEnviar">ENVIAR</button>
</div>
```

```html
    <script src="https://cdn.jsdelivr.net/npm/vue@2.7.16/dist/vue.js"></script>
    <script>
        var app = new Vue({
            el: '#app',
            data: {
                titulo: 'Vue JS',
                btnClassLimpar: {
                    'btn-danger': true,
                    'btn-sm': false
                },
                // btnClassEnviar: 'btn-primary btn-lg'
                // btnClassEnviar: [
                //     'btn-primary',
                //     'btn-lg'
                // ]
                btnClassEnviar: [
                    'btn-primary',
                    { 'btn-lg': true }
                ],
                btnStyleLimpar: 'margin: 5px; font-size: 50px',
                // btnStyleEnviar: 'font-size: 10px',
                btnStyleEnviar: [
                    {'font-size': '30px'},
                    {'text-transform': 'lowercase'}
                ],
            }
        })
    </script>
</body>
</html>
```

Resultado final :



São 02 botões estilizados com bootstrap onde suas configurações são passadas por v-bind, como podem verificar no exemplo eu posso tanto criar configurações com booleanos ou passar inclusive um array de objetos com os valores combinados.