Módulo 1: Introdução ao Vue.js

- diretiva diretiva v-on, manipulando eventos

A diretiva `v-on` no Vue.js é utilizada para lidar com eventos DOM, permitindo que você execute ações quando certos eventos acontecem em elementos HTML, como cliques, teclas pressionadas, movimentação do mouse, entre outros. Ela facilita a associação de eventos JavaScript a componentes Vue, tornando seu código mais organizado e reativo.

Sintaxe básica

A sintaxe da diretiva `v-on` segue este padrão:

```
<!-- Sintaxe completa -->
<button v-on:click="metodo">Clique aqui</button>

<!-- Sintaxe abreviada -->
<button @click="metodo">Clique aqui</button>
```

Aqui estão os detalhes:

- `v-on:evento`: O `evento` é o nome do evento DOM que você deseja manipular, como `click`, `mouseover`, `keyup`, etc.
- `metodo`: O método no Vue.js que será executado quando o evento for disparado.

A versão abreviada, utilizando `@`, é equivalente à sintaxe completa de `v-on` e é comumente usada por ser mais concisa.

Manipulação de eventos comuns

Aqui estão alguns dos eventos mais utilizados com `v-on`:

1. **Clique (click)** Manipula o evento de clique em um elemento:

```
<button @click="mostrarAlerta">Clique aqui</button>
```

No Vue.js:

```
new Vue({
  el: '#app',
  methods: {
    mostrarAlerta() {
      alert('Botão clicado!');
    }
  }
});
```

Passar o mouse (mouseover) Manipula o evento de passar o mouse sobre um elemento:

```
<div @mouseover="mudarCor">Passe o mouse aqui</div>
```

No Vue.js:

```
new Vue({
  el: '#app',
  methods: {
    mudarCor() {
      console.log('O mouse passou sobre o elemento');
    }
  }
});
```

Tecla pressionada (keyup) Manipula o evento de uma tecla pressionada:

```
<input @keyup="mostrarTecla">
```

No Vue.js:

```
new Vue({
  el: '#app',
  methods: {
    mostrarTecla(event) {
      console.log('Tecla pressionada:', event.key);
    }
  }
});
```

1. Aqui, o objeto `event` contém informações sobre a tecla pressionada, e o `event.key` retorna o valor da tecla.

Passando parâmetros para o método

Você pode passar parâmetros para o método associado ao evento diretamente da diretiva `v-on`. Por exemplo:

```
<button @click="cumprimentar('Olá')">Cumprimentar</button>
```

No método Vue.js:

```
new Vue({
  el: '#app',
  methods: {
    cumprimentar(mensagem) {
      alert(mensagem);
    }
  }
});
```

Aqui, quando o botão for clicado, o método `cumprimentar` será chamado com o argumento `'Olá'`.

Usando o objeto `event`

O objeto `event` contém informações sobre o evento que foi disparado. Ele é automaticamente passado para o método quando você utiliza `v-on`. Se você precisar acessá-lo explicitamente, pode incluí-lo como um parâmetro:

```
<button @click="mostrarCoordenadas($event)">Clique aqui</button>
```

No Vue.js:

```
new Vue({
  el: '#app',
  methods: {
    mostrarCoordenadas(event) {
      console.log('Posição do clique:', event.clientX, event.clientY);
    }
  }
});
```

Neste exemplo, `event.clientX` e `event.clientY` retornam as coordenadas do clique no eixo X e Y.

Modificadores de evento

Os modificadores são sufixos especiais adicionados a eventos para alterar o comportamento padrão. Alguns dos modificadores mais comuns incluem:

1. **`.stop`**: Impede que a propagação do evento continue para outros elementos pais.

```
<button @click.stop="acao">Clique com stop</button>
```

`.prevent`: Chama o `event.preventDefault()`, que impede o comportamento padrão do navegador (por exemplo, impedir que um link seja seguido).

```
<form @submit.prevent="enviarFormulario">Enviar</form>
```

`.self`: Garante que o evento seja acionado apenas se o próprio elemento foi clicado (não seus filhos).

```
<div @click.self="acao">Clique no div</div>
```

.once: O evento será disparado apenas uma vez.

```
<button @click.once="acao">Clique uma vez</button>
```

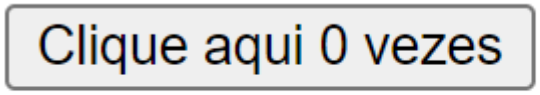
Vamos para um exemplo prático:

Exemplo 1: Botão de contagem

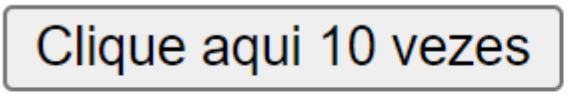
Estrutura:

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Exemplo Contagem</title>
</head>
<body>
    <div id="app">
        <button @click="incrementarContador">Clique aqui {{ contador }} vezes</button>
      </div>


      <script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>
      <script>
        new Vue({
          el: '#app',
          data: {
            contador: 0
          },
          methods: {
            incrementarContador() {
              this.contador++;
            }
          }
        });
      </script>

</body>
</html>
```

Neste exemplo, o botão incrementa o valor de `contador` a cada clique, e esse valor é exibido no texto do botão.



Ao clicarmos é incrementado valor ao botão:



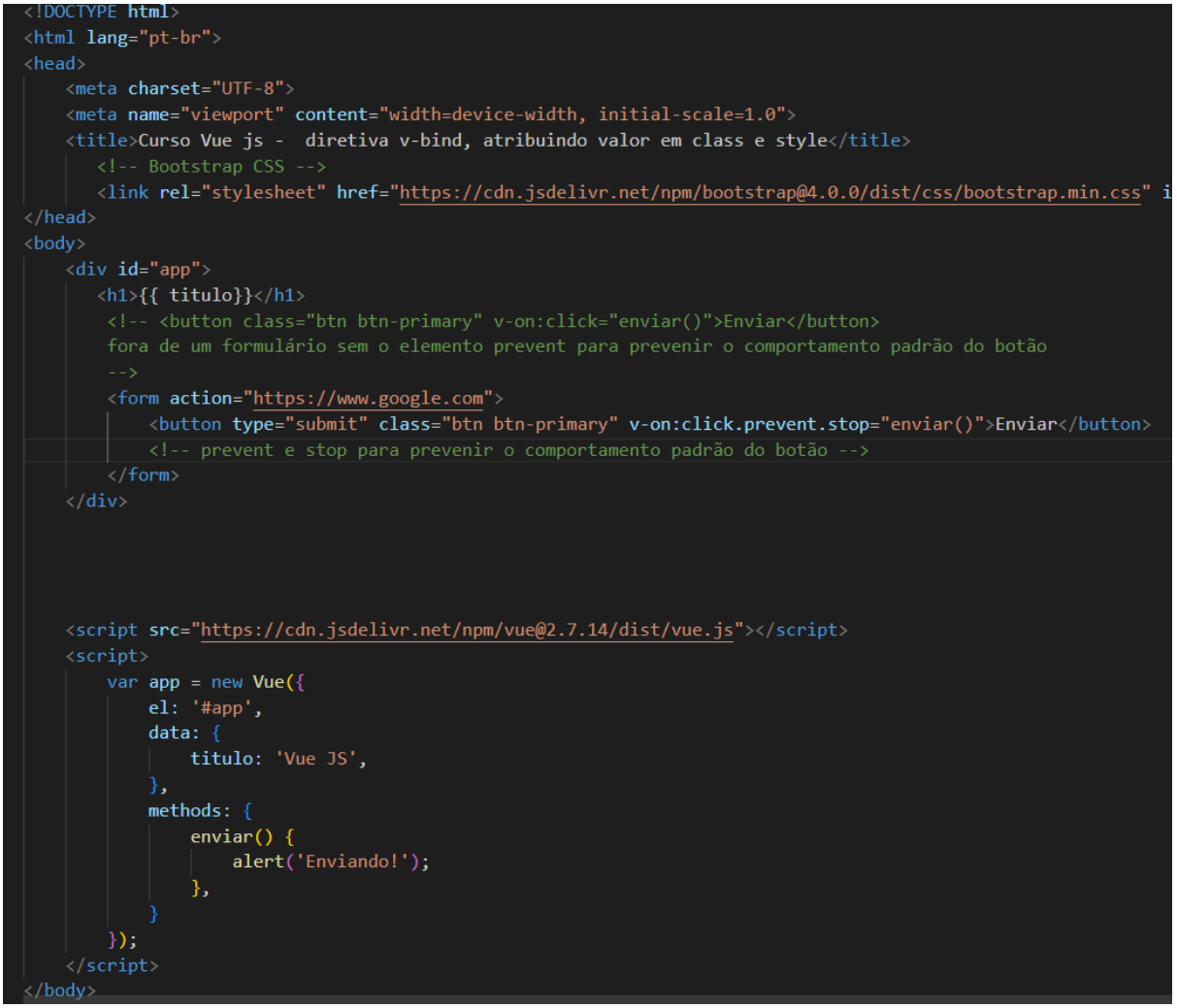
Exemplo 2: Formulário com prevenção do comportamento padrão

Aqui, o formulário é enviado via JavaScript, sem recarregar a página, graças ao modificador `.prevent` no evento `submit`.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Exemplo form 02 com prevenção de comportamento</title>
</head>
<body>
    <div id="app">
        <form @submit.prevent="enviarFormulario">
            <input type="text" v-model="nome">
            <button type="submit">Enviar</button>
        </form>
    </div>


    <script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>
    <script>
        new Vue({
            el: '#app',
            data: {
                nome: ''
            },
            methods: {
                enviarFormulario() {
                    alert('Formulário enviado com o nome: ' + this.nome);
                }
            }
        });
    </script>

</body>
</html>
```

E por último e não menos importante um evento utilizando bootstrap chamando uma função através de um botão:

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" i
</head>
<body>
    <div id="app">
      <h1>{{ titulo}}</h1>
       <!-- <button class="btn btn-primary" v-on:click="enviar()">Enviar</button>
       fora de um formulário sem o elemento prevent para prevenir o comportamento padrão do botão
       -->
       <form action="https://www.google.com">
          <button type="submit" class="btn btn-primary" v-on:click.prevent.stop="enviar()">Enviar</button>
          <!-- prevent e stop para prevenir o comportamento padrão do botão -->
       </form>
    </div>



    <script src="https://cdn.jsdelivr.net/npm/vue@2.7.14/dist/vue.js"></script>
    <script>
        var app = new Vue({
            el: '#app',
            data: {
                titulo: 'Vue JS',
            },
            methods: {
                enviar() {
                    alert('Enviando!');
                },
            }
        });
    </script>
</body>
```

No seu código, a combinação de `.prevent` e `.stop` na diretiva `v-on` dentro do botão serve para controlar como o evento `click` no botão de "Enviar" se comporta no contexto de um formulário. Vamos explicar o que esses modificadores fazem e como eles são usados na prática.

1. `.prevent`

O modificador `.prevent` é usado para chamar o método `event.preventDefault()` em eventos. Isso impede que o comportamento padrão do navegador ocorra quando o evento é disparado.

No contexto de um formulário HTML, o comportamento padrão de um botão com `type="submit"` é enviar o formulário e recarregar a página ou redirecionar o usuário para o valor do atributo `action`, que no seu exemplo é `https://www.google.com`.

Quando você usa `.prevent` no `v-on:click`, ele impede que o formulário seja enviado ao clicar no botão e, consequentemente, evita o redirecionamento.

Aqui, o modificador `.prevent` impede que o formulário seja enviado e a página seja redirecionada para o Google, permitindo que você controle o que acontece ao clicar no botão, como a exibição de um alerta com a mensagem `'Enviando!'`.

2. `.stop`

O modificador `.stop` é usado para parar a propagação do evento na árvore do DOM. Ele chama o método `event.stopPropagation()`, o que significa que o evento não continuará "subindo" (ou "borbulhando") para elementos pais.

Em um contexto como o de um formulário, o evento de clique no botão "Enviar" normalmente poderia propagar-se para elementos pais, como o próprio `<form>` ou até o `<body>`. Isso poderia acionar outros manipuladores de evento definidos nesses elementos, caso existissem. O modificador `.stop` impede essa propagação, garantindo que o evento seja tratado exclusivamente no elemento em que ocorreu (neste caso, o botão).

No seu exemplo, você usou `.stop` para garantir que o clique no botão não vá além dele, e nenhum outro manipulador de evento fora do botão (como no `<form>`) seja acionado.

Combinação de `.prevent` e `.stop`

Quando você combina os dois modificadores, como no seu caso:

`.prevent`: Impede o envio do formulário (que é o comportamento padrão de um botão de envio dentro de um formulário).

`.stop`: Evita que o evento `click` continue subindo pela árvore do DOM, garantindo que outros elementos fora do botão não lidem com esse evento.

Resumo:

- `.prevent`: É usado para impedir o comportamento padrão de um evento, como o envio do formulário ou o redirecionamento.
- `.stop`: Interrompe a propagação do evento, evitando que outros elementos ancestrais na árvore do DOM também tratem o mesmo evento.

No seu exemplo, o comportamento esperado é que, ao clicar no botão, o formulário não seja enviado (por causa do `.prevent`), a página não redirecione, e o evento de clique não se propague para elementos pais (graças ao `.stop`). Apenas o método `enviar()` será executado, disparando o alerta `'Enviando!'`.

Se você remover `.prevent`, o formulário será enviado e a página será redirecionada para o Google. Se remover `.stop`, o evento poderá se propagar, potencialmente acionando outros eventos em elementos pais, se houver.