

## vue 3 exemplo bootstrap popover com vite

🕒 Thought for 51 seconds ▾

Vamos criar um exemplo de uso do popover do Bootstrap em um projeto Vue 3 com Vite. Como o Bootstrap 5 requer o Popper.js para popovers, podemos incluir o Bootstrap via CDN ou instalar via npm.

Neste exemplo, vou mostrar como fazer via npm, que é mais comum em projetos Vite.

Passos:

1. Criar um projeto Vue 3 com Vite (se já não tiver).
2. Instalar o Bootstrap e o @popperjs/core.
3. Importar o CSS do Bootstrap e o JS do Bootstrap (que inclui o popover).
4. Inicializar os popovers no Vue.

No entanto, note que o Bootstrap não é otimizado para Vue, então podemos usar uma biblioteca que integra Bootstrap com Vue, como "bootstrap-vue-3" (mas para Bootstrap 5, a integração pode ser direta com o Bootstrap original, pois não há muitos componentes Vue específicos para Bootstrap 5). Para popovers, podemos usar a diretiva do Bootstrap JS diretamente.

Outra opção é usar o pacote "bootstrap" e inicializar os popovers manualmente no Vue.

Vou mostrar uma maneira simples, inicializando os popovers no evento mounted do componente.

Vamos criar um componente de exemplo.

Se preferir, também pode usar a biblioteca "bootstrap-vue-3" que já tem componentes Vue para Bootstrap 5, mas aqui vou fazer de forma mais direta.

No entanto, note que o Bootstrap 5 não depende do jQuery, então podemos usar o JavaScript puro.

Vamos lá:

Primeiro, crie um projeto Vue 3 com Vite (se não tiver):

```
npm create vite@latest meu-projeto -- --template vue  
cd meu-projeto
```

Instale o Bootstrap e o Popper (que é uma dependência do Bootstrap para componentes como popovers):

```
npm install bootstrap @popperjs/core
```

Agora, no seu componente (por exemplo, App.vue), importe o CSS do Bootstrap e o JS do Bootstrap.

Como o Vite lida com CSS, podemos importar o CSS no main.js ou diretamente no componente.

Vou mostrar no main.js e no componente.

No main.js (ou main.ts):

```
import { createApp } from 'vue'  
import App from './App.vue'  
  
// Importe o CSS do Bootstrap  
import 'bootstrap/dist/css/bootstrap.min.css'  
// Importe o JS do Bootstrap (opcional: você pode importar apenas os componentes que  
precisa, mas aqui vamos importar tudo)  
import 'bootstrap/dist/js/bootstrap.bundle.min.js'  
  
createApp(App).mount('#app')
```

Agora, no componente, podemos usar o popover do Bootstrap.

No template, adicionaremos um botão com o atributo data-bs-toggle="popover" e inicializaremos no mounted.

No entanto, o Bootstrap não inicializará automaticamente os popovers em elementos dinâmicos (se o componente for montado após o carregamento da página). Portanto, precisamos inicializá-los manualmente.

Podemos usar a API do Bootstrap para inicializar o popover no Vue.

Exemplo de componente:

App.vue:

```
<template>  
<div class="container mt-5">  
  <h1>Vue 3 + Bootstrap Popover com Vite</h1>  
  <button ref="popoverButton" type="button" class="btn btn-lg btn-danger" data-  
  bs-toggle="popover" title="Título do Popover" data-bs-content="Conteúdo do  
  popover. E está funcionando!">  
    Clique para abrir o popover  
  </button>  
</div>  
</template>  
  
<script setup> import { onMounted, ref } from 'vue' // Referência para o botão  
const popoverButton = ref(null) onMounted(() => { // Inicializa o popover no  
botão // Note: estamos usando a API do Bootstrap para inicializar o popover //  
0 Bootstrap 5 disponibiliza o objeto bootstrap no window, mas como importamos  
o bundle, podemos acessar via window.bootstrap // No entanto, para garantir,  
podemos importar o popover do bootstrap diretamente. // Vou fazer de duas  
formas, a primeira é usando o window. // Método 1: Usando window.bootstrap (se
```

```
o bundle foi importado globalmente) // const popover = new
window.bootstrap.Popover(popoverButton.value) // Método 2: Importando
dinamicamente o Popover do módulo bootstrap (recomendado para melhor controle
de bundle) // Vou usar o método 2 para evitar dependência global e para tree-
shaking. import('bootstrap/js/dist/popover').then((module) => { //
module.default é o construtor do Popover new
module.default(popoverButton.value) }) }) </script> <style> /* Estilos
adicionais se necessário */ </style>
```

No entanto, o método 2 (importação dinâmica) pode ser um pouco pesado se você tiver muitos componentes, pois carregará o módulo separadamente. Mas é uma boa prática para carregar apenas o necessário.

Outra forma é importar o Popover do bootstrap no topo do arquivo, mas isso incluirá o código no bundle principal.

Se você não se importar com o bundle, pode importar diretamente no topo:

```
import Popover from 'bootstrap/js/dist/popover'

onMounted(() => {
  new Popover(popoverButton.value)
})
```

Vou ajustar para usar a importação no topo, que é mais simples.

No entanto, note que o Bootstrap 5 é modular, então podemos importar apenas o popover.

Vou ajustar o código:

```
<script setup> import { onMounted, ref } from 'vue' import Popover from
'bootstrap/js/dist/popover' const popoverButton = ref(null) onMounted(() => {
  new Popover(popoverButton.value) }) </script>
```

Isso deve funcionar.

Vamos testar.

Se você quiser que o popover seja mostrado em diferentes direções, pode usar o atributo data-bs-placement.

Exemplo: data-bs-placement="top", "bottom", "left", "right".

Vamos adicionar um exemplo com placement.

Também, lembre-se de que o popover requer que o elemento tenha um title (ou data-bs-title) e data-bs-content (ou content via opções).

Você também pode configurar o popover via opções no JavaScript, em vez de usar data attributes.

Exemplo com opções:

```
new Popover(popoverButton.value, {
  title: 'Título via JS',
  content: 'Conteúdo via JS',
  placement: 'right',
  trigger: 'click'
})
```

Mas note que se você passar opções, irá sobrescrever os data attributes.

Vou deixar com data attributes no template.

Agora, se você quiser múltiplos popovers, pode usar uma referência para cada um ou inicializar todos os elementos com o atributo data-bs-toggle="popover" em um wrapper.

Por exemplo, se você tiver vários botões com a classe "popover-button", pode fazer:

```
document.querySelectorAll('.popover-button').forEach(el => new Popover(el))
```

Mas no Vue, é mais comum usar refs em um array ou usar querySelector no elemento pai.

Vou mostrar um exemplo com múltiplos popovers usando uma ref no container e inicializando todos os botões dentro.

No entanto, para um exemplo simples, vou fazer com um único botão.

Vamos ao código final do componente:

App.vue:

Aqui está um exemplo de como usar Popovers do Bootstrap 5 com Vue 3 e Vite:

## 1. Instalar as dependências

bash

```
npm install bootstrap @popperjs/core
```

## 2. Configurar o Bootstrap no projeto

main.js / main.ts:

javascript

```
import { createApp } from 'vue'
import App from './App.vue'

// Importar CSS do Bootstrap
import 'bootstrap/dist/css/bootstrap.min.css'
```

```
createApp(App).mount('#app')
```

### 3. Componente de exemplo

PopoverExample.vue:

vue

```
<template>
<div class="container mt-5">
  <h2>Bootstrap Popovers com Vue 3</h2>

  <!-- Exemplo 1: Popover básico -->
  <button
    ref="popoverButton1"
    type="button"
    class="btn btn-primary m-2"
    data-bs-toggle="popover"
    data-bs-title="Título do Popover"
    data-bs-content="Conteúdo do popover aqui."
  >
    Popover básico
  </button>

  <!-- Exemplo 2: Popover com HTML -->
  <button
    ref="popoverButton2"
    type="button"
    class="btn btn-success m-2"
    data-bs-toggle="popover"
    data-bs-title="Popover com HTML"
    :data-bs-content="htmlContent"
    data-bs-html="true"
  >
    Popover com HTML
  </button>

  <!-- Exemplo 3: Popover com posicionamento -->
  <button
    ref="popoverButton3"
    type="button"
    class="btn btn-warning m-2"
    data-bs-toggle="popover"
    data-bs-title="Popover à direita"
    data-bs-content="Aparece à direita do elemento"
  >
```

```
        data-bs-placement="right"
    >
        Popover à direita
    </button>

    <!-- Exemplo 4: Popover programático -->
    <button
        ref="popoverButton4"
        type="button"
        class="btn btn-info m-2"
        @click="toggleCustomPopover"
    >
        Popover programático
    </button>

    <!-- Exemplo 5: Popover em elemento customizado -->
    <div class="card mt-4" style="width: 18rem;">
        <div class="card-body">
            <h5 class="card-title" ref="titlePopover">Card Title</h5>
            <p class="card-text">Passe o mouse sobre o título para ver o popover.</p>
        </div>
    </div>
</div>
</template>

<script setup>
import { ref, onMounted, onBeforeUnmount } from 'vue'
import { Popover } from 'bootstrap'

// Refs para os elementos
const popoverButton1 = ref(null)
const popoverButton2 = ref(null)
const popoverButton3 = ref(null)
const popoverButton4 = ref(null)
const titlePopover = ref(null)

// Instâncias dos popovers
let popover1 = null
let popover2 = null
let popover3 = null
let popover4 = null
let titlePopoverInstance = null

// Conteúdo HTML para o popover
const htmlContent = `
<div>
    <strong>Conteúdo HTML</strong>
```

```
<ul>
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
</ul>
</div>
`
```

  

```
// Inicializar popovers quando o componente for montado
onMounted(() => {
  // Popovers com data attributes
  if (popoverButton1.value) {
    popover1 = new Popover(popoverButton1.value)
  }

  if (popoverButton2.value) {
    popover2 = new Popover(popoverButton2.value)
  }

  if (popoverButton3.value) {
    popover3 = new Popover(popoverButton3.value)
  }

  // Popover programático
  if (popoverButton4.value) {
    popover4 = new Popover(popoverButton4.value, {
      title: 'Popover Programático',
      content: 'Este popover foi criado via JavaScript.',
      placement: 'bottom',
      trigger: 'manual'
    })
  }
}

// Popover em elemento customizado (hover)
if (titlePopover.value) {
  titlePopoverInstance = new Popover(titlePopover.value, {
    title: 'Informação',
    content: 'Este é um card de exemplo.',
    placement: 'top',
    trigger: 'hover'
  })
}
})
```

  

```
// Método para alternar popover programático
const toggleCustomPopover = () => {
  if (popover4) {
```

```

    if (popoverButton4.value.classList.contains('show')) {
      popover4.hide()
    } else {
      popover4.show()
    }
  }

// Limpar instâncias ao desmontar
onBeforeUnmount(() => {
  if (popover1) popover1.dispose()
  if (popover2) popover2.dispose()
  if (popover3) popover3.dispose()
  if (popover4) popover4.dispose()
  if (titlePopoverInstance) titlePopoverInstance.dispose()
})
</script>

<style scoped>
.container {
  padding: 20px;
}
.btn {
  margin: 5px;
}
</style>

```

## 4. Componente wrapper reutilizável (opcional)

**BootstrapPopover.vue:**

```

vue

<template>
  <component
    :is="tag"
    ref="popoverElement"
    :data-bs-toggle="dataBsToggle ? 'popover' : null"
    :data-bs-title="title"
    :data-bs-content="content"
    :data-bs-placement="placement"
    :data-bs-html="html"
    :data-bs-trigger="trigger"
  >
    <slot></slot>
  </component>

```

```
</template>

<script setup>
import { ref, onMounted, onBeforeUnmount } from 'vue'
import { Popover } from 'bootstrap'

const props = defineProps({
  tag: {
    type: String,
    default: 'button'
  },
  title: String,
  content: String,
  placement: {
    type: String,
    default: 'top'
  },
  html: {
    type: Boolean,
    default: false
  },
  trigger: {
    type: String,
    default: 'click'
  },
  dataBsToggle: {
    type: Boolean,
    default: true
  },
  options: {
    type: Object,
    default: () => ({})
  }
})

const popoverElement = ref(null)
let popoverInstance = null

onMounted(() => {
  if (popoverElement.value) {
    popoverInstance = new Popover(popoverElement.value, {
      title: props.title,
      content: props.content,
      placement: props.placement,
      html: props.html,
      trigger: props.trigger,
      ...props.options
    })
  }
})
```

```

        })
    }
})

onBeforeUnmount(() => {
  if (popoverInstance) {
    popoverInstance.dispose()
  }
})

// Métodos expostos
defineExpose({
  show: () => popoverInstance?.show(),
  hide: () => popoverInstance?.hide(),
  toggle: () => popoverInstance?.toggle(),
  dispose: () => popoverInstance?.dispose()
})
</script>

```

## Uso do componente wrapper:

vue

```

<template>
  <BootstrapPopover
    title="Popover Reutilizável"
    content="Este é um popover usando componente wrapper"
    placement="right"
  >
    <button class="btn btn-primary">
      Popover Reutilizável
    </button>
  </BootstrapPopover>
</template>

```

## 5. Importar JavaScript do Bootstrap (alternativa)

Se preferir importar todo o JavaScript do Bootstrap, adicione no **main.js**:

javascript

```

import { createApp } from 'vue'
import App from './App.vue'

import 'bootstrap/dist/css/bootstrap.min.css'

```

```
import 'bootstrap/dist/js/bootstrap.bundle.min.js'

createApp(App).mount('#app')
```

## Pontos importantes:

1. Bootstrap 5 não requer jQuery, apenas Popper.js (já incluído no bundle)
2. Use data-bs- prefix para atributos do Bootstrap 5
3. Lembre-se de destruir as instâncias com dispose() no lifecycle onBeforeUnmount
4. Para triggers diferentes, use: click , hover , focus ou combinações

Este exemplo mostra diferentes formas de usar popovers, desde o uso simples com data attributes até controle programático completo.