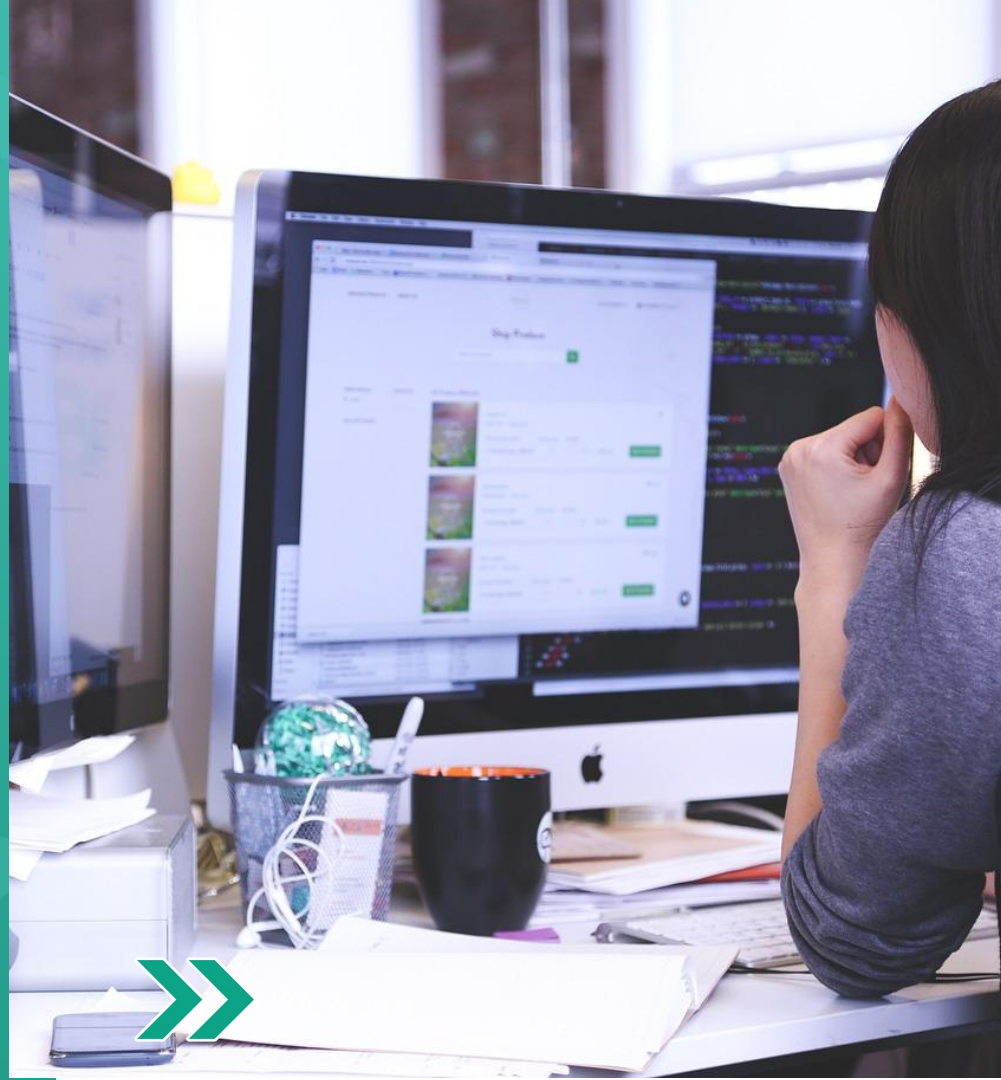
escola britânica de artes criativas & tecnologia

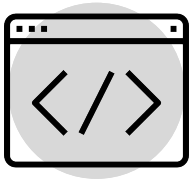
Profissão: Engenheiro Front-End

escola britânica de artes criativas & tecnologia

BOAS PRÁTICAS



VueJS - Visão geral



Confira boas práticas da comunidade de Front-End por assunto relacionado às aulas.

- **Conheça o VueJS**
- **Integre o VueJS**
- **Renderização condicional**
- **Utilize eventos para manipular dados**
- **Trabalhe com listas**
- **Crie componentes**

Conheça o VueJS

Vite.config.js

Acompanhe algumas das configurações comuns que você pode encontrar no vite.config.js

- **root**: O diretório raiz do projeto.
- **base**: A URL base do aplicativo.
- **publicDir**: O diretório onde os arquivos estáticos públicos são armazenados.
- **build**: Configurações relacionadas ao processo de construção (ex.: diretório de saída, opções de minificação etc.).
- **plugins**: Configurações para plugins adicionais usados pelo Vite.
- **server**: Configurações do servidor de desenvolvimento, como porta, proxy, etc.
- **resolve**: Configurações de resolução de módulos.

Conheça o VueJS

Volar

A volar foi criada pela comunidade Vue e fornece os seguintes recursos:

- **Detecção de erros aprimorada:** A extensão Volar é capaz de identificar erros de sintaxe, erros de tipo e problemas comuns no código Vue. Ela fornece indicações visuais no editor para ajudar a identificar e corrigir esses problemas rapidamente.

- **Refatoração avançada:** Com o Volar, você pode facilmente renomear componentes, props, métodos e outros elementos do seu código Vue de forma segura e consistente em todo o projeto. Ele também oferece recursos de extração de código, como extrair um componente em um arquivo separado.

- **Suporte a TypeScript:** O Volar possui suporte completo ao TypeScript para projetos Vue. Ele oferece recursos como inferência de tipo, navegação entre definições, sugestões inteligentes e muito mais para ajudar no desenvolvimento de aplicativos Vue com TypeScript.

Conheça o VueJS

Volar

A volar foi criada pela comunidade Vue e fornece os seguintes recursos:

- **Suporte a arquivos .vue:** O Volar entende completamente a estrutura de arquivos .vue e fornece recursos específicos para facilitar a edição desses arquivos. Isso inclui suporte a pre-processadores (por exemplo, SASS, Less), realces de sintaxe avançados, formatação automática e muito mais.
- **Performance otimizada:** O Volar foi projetado para ser rápido e eficiente, mesmo em projetos grandes. Ele é construído com base na biblioteca Language Server Protocol (LSP), que permite uma comunicação eficiente entre o servidor Volar e o editor VS Code.

Integre o VueJS

Scoped
A opção scoped cria estilos encapsulados apenas em nível de componente. Se você precisar de estilos compartilhados entre componentes ou estilos globais que afetam todo o aplicativo, considere o uso de outras abordagens, como classes globais, estilos em um arquivo separado ou o uso de frameworks de estilo externos. Acompanhe algumas dicas para usar a opção scoped:

- **Evite vazamento de estilos:** Ao usar a opção scoped, os seletores CSS definidos no componente não serão aplicados a elementos fora do componente. Isso evita o vazamento de estilos indesejados e ajuda a manter a consistência dos estilos dentro do componente.

- **Escolha seletivamente o escopo:** Nem todos os estilos em um componente precisam estar encapsulados. Use a opção scoped seletivamente apenas nos estilos que devem ser específicos do componente. Você pode ter estilos globais fora do bloco <style scoped> para aplicar estilos comuns a vários componentes.

- **Utilize modificadores de classe:** O uso de modificadores de classe é uma maneira eficaz de estilizar elementos dentro do componente. Os modificadores de classe permitem adicionar ou remover classes condicionalmente com base em propriedades ou estados do componente. Isso ajuda a criar estilos dinâmicos e reativos dentro do escopo do componente.

Integre o VueJS

Scoped
A opção scoped cria estilos encapsulados apenas em nível de componente. Se você precisar de estilos compartilhados entre componentes ou estilos globais que afetam todo o aplicativo, considere o uso de outras abordagens, como classes globais, estilos em um arquivo separado ou o uso de frameworks de estilo externos. Acompanhe algumas dicas para usar a opção scoped:

- **Estilize elementos filhos:** Ao usar a opção scoped, os seletores CSS definidos no componente se aplicam apenas aos elementos filhos diretos do componente. Isso permite que você estilize os elementos filhos com facilidade e mantenha o controle sobre os estilos dentro do componente.

- **Use estilização baseada em componentes:** A opção scoped facilita a criação de estilos baseados em componentes. Você pode definir estilos específicos de um componente em um bloco <style scoped> e reutilizá-los em vários lugares do componente. Isso promove a consistência visual e facilita a manutenção dos estilos.

- **Combine com pré-processadores CSS:** Se você estiver usando um pré-processador CSS, como SASS ou Less, a opção scoped ainda funciona. Você pode escrever estilos em sintaxe pré-processador dentro do bloco <style scoped>, e eles serão compilados corretamente para CSS encapsulado.

Integre o VueJS

Mustaches

Você pode usar os "mustaches" em várias situações dentro de um template Vue, por exemplo, exibindo valores de propriedades, operações e expressões, acesso a propriedades de objetos, entre outros.

- Use os mustaches para acessar e exibir o valor de propriedades de dados no template. Por exemplo, {{ mensagem }} irá renderizar o valor da propriedade mensagem definida no objeto data.

  Os mustaches são usados para exibir dados, mas não devem ser usados para alterar diretamente os dados. Evite fazer alterações de estado ou efeitos colaterais dentro de mustaches, pois eles são destinados apenas para exibição de dados.

  Os mustaches também podem ser usados para controle de fluxo simples no template, como exibir ou ocultar elementos com base em condições. Por exemplo, você pode usar {{ exibirMensagem ? 'Visível' : 'Oculto' }} para exibir o texto "Visível" ou "Oculto" com base no valor da propriedade exibirMensagem.

Integre o VueJS

Vbind

Acompanhe algumas dicas para usar a diretiva v-bind (ou :) no Vue.js:

- **Vinculação de atributos:** Use a diretiva v-bind (ou :) para vincular dinamicamente valores de propriedades ou expressões JavaScript a atributos HTML. Por exemplo, v-bind:href="url" vincula a propriedade url a um atributo href em um elemento HTML.

- **Expressões e cálculos:** Você pode usar expressões JavaScript dentro de v-bind para realizar cálculos ou manipulações antes de vincular um valor a um atributo. Por exemplo, v-bind:class="{ destaque: isActive }" adiciona a classe CSS destaque ao elemento somente quando a propriedade isActive é verdadeira.

- **Objetos e propriedades dinâmicas:** Use v-bind para vincular propriedades dinâmicas de objetos. Por exemplo, v-bind:[atributo]="valor" permite que o nome do atributo seja dinâmico, baseado na variável atributo, e o valor seja fornecido pela variável valor.

Integre o VueJS

Vbind

Acompanhe algumas dicas para usar a diretiva v-bind (ou :) no Vue.js:

- **Atalho de sintaxe:** Para maior concisão, você pode usar o atalho de sintaxe : em vez de v-bind. Por exemplo, :href="url" é equivalente a v-bind:href="url".

- **Ligação bidirecional:** v-bind é usado principalmente para realizar a vinculação de dados unidirecional, mas também pode ser usado em elementos de entrada de formulário para realizar a ligação de dados bidirecional. Por exemplo, v-bind:value="mensagem" em um campo de entrada <input> vincula o valor do campo à propriedade mensagem e atualiza automaticamente a propriedade quando o usuário digita no campo.

- **Atributos HTML personalizados:** v-bind também pode ser usado para vincular valores a atributos HTML personalizados. Certifique-se de que o atributo personalizado esteja em conformidade com as regras de nomenclatura, usando prefixos como data- ou seguindo convenções estabelecidas.

Renderização condicional

Reactive

- Utilize a função reactive para envolver objetos JavaScript e torná-los reativos. Isso permite que você rastreie automaticamente as mudanças nas propriedades do objeto.
- Acesse as propriedades reativas do objeto como faria com qualquer objeto JavaScript. Por exemplo, state.contador permite acessar e modificar a propriedade contador.
- O Vue.js detecta automaticamente as alterações nas propriedades reativas e atualiza a interface do usuário quando necessário. Isso inclui a atualização dos elementos do DOM que estão vinculados a essas propriedades.
- Ao trabalhar com arrays reativos, certifique-se de usar métodos específicos para arrays reativos, como push, pop, splice, etc. Isso garante que as alterações sejam detectadas corretamente.

Renderização condicional

vshow

Acompanhe algumas dicas para usar a diretiva v-show no Vue.js:

- **Comparação com v-if:** Ao contrário da diretiva v-if, que adiciona ou remove o elemento do DOM com base na expressão, v-show apenas altera a visibilidade do elemento. O elemento permanece no DOM, mas seu estilo de exibição é controlado.

- **Melhor para alternar visibilidade frequente:** Use v-show quando você espera alternar a visibilidade do elemento com frequência. Como o elemento permanece no DOM, a troca entre visível e invisível é mais rápida do que adicionar e remover o elemento com v-if.

- **Melhor para elementos pequenos:** v-show é mais adequado para elementos pequenos ou com conteúdo simples. Se o elemento a ser alternado com frequência for complexo ou possui um grande impacto na renderização, o uso de v-if pode ser mais eficiente.

Renderização condicional

vshow

Acompanhe algumas dicas para usar a diretiva v-show no Vue.js:

- **Use com cuidado em elementos de renderização intensiva:** Se o elemento controlado por v-show contiver muitos componentes filhos ou envolver cálculos intensivos, leve em consideração o impacto na performance e considere alternativas, como otimizar a renderização com o v-if.

- **Elementos invisíveis ocupam espaço:** Lembre-se de que, mesmo quando invisível, um elemento com v-show ainda ocupa espaço no layout. Isso pode afetar o posicionamento de outros elementos na página. Certifique-se de ajustar o layout adequadamente ou considere alternativas, como display: none via CSS em vez de v-show.

Renderização condicional

vshow

Acompanhe algumas dicas para usar a diretiva v-show no Vue.js:

- **Combine com transições:** Você pode combinar v-show com as transições do Vue.js para adicionar animações suaves ao alternar a visibilidade do elemento. Isso pode criar uma experiência mais agradável para o usuário.

- Lembre-se de considerar o contexto e as necessidades específicas do seu aplicativo ao decidir entre v-show e v-if. Ambas as diretivas têm seus casos de uso apropriados e cabe a você escolher a melhor opção para cada situação.

Utilize eventos para manipular dados

Two way data binging

- **Atualizações bidirecionais:** O two-way data binding atualiza automaticamente o valor do modelo e o valor exibido no elemento sempre que ocorrem alterações em ambos os lados. Isso permite que você mantenha o estado do modelo e a interface do usuário sincronizados sem a necessidade de código adicional.

- **Trate diferentes tipos de entrada:** O two-way data binding pode ser usado com diferentes tipos de entrada, como texto, números, checkboxes e seleções. Certifique-se de definir corretamente o tipo do elemento de entrada para garantir que a coerção de dados funcione adequadamente.

- **Valide e formate dados:** Você pode adicionar lógica adicional para validar e formatar os dados antes de serem armazenados ou exibidos. Isso pode ser feito usando computed properties ou métodos que manipulam os valores antes de serem atribuídos à propriedade vinculada.

Trabalhe com listas

Diretiva v-for

Acompanhe algumas dicas para usar a diretiva v-for:

- **Use uma chave única:** Sempre forneça uma chave única usando o atributo :key ao usar v-for. Isso ajuda o Vue a rastrear os elementos da lista e otimizar a renderização, melhorando o desempenho.
- **Acesso a índices:** Se você precisar acessar o índice do item atual na iteração, use a sintaxe (item, index).
- **Manipulação de objetos:** No caso de objetos, você pode usar a sintaxe (value, key) para iterar sobre as propriedades do objeto.

Trabalhe com listas

Diretiva v-for

Acompanhe algumas dicas para usar a diretiva v-for:

- **Usando uma lista filtrada:** Você pode usar a diretiva v-for com uma lista filtrada, aplicando uma condição usando uma computed property ou um método.
- **Iterar sobre um número fixo de vezes:** Se você precisar iterar sobre um número fixo de vezes, pode usar a função Array.from combinada com v-for.
- **Use v-for com componentes:** Além de iterar sobre elementos HTML, você também pode usar v-for para iterar sobre componentes personalizados. Isso pode ser útil quando você precisa renderizar uma lista de componentes repetitivos.

Crie componentes

Componentização

Acompanhe algumas dicas para aproveitar ao máximo a componentização no Vue.js:

- **Divida sua interface em componentes reutilizáveis:** A componentização permite dividir sua interface de usuário em componentes independentes, cada um responsável por uma parte específica da funcionalidade. Identifique elementos comuns e crie componentes para eles, promovendo a reutilização e a modularidade do código.

- **Pense em componentes autônomos:** Procure criar componentes que sejam autossuficientes e independentes. Isso significa que eles devem ter suas próprias dependências e lidar com sua própria lógica de estado. Isolar a lógica e o estado em componentes individuais torna o código mais limpo e fácil de entender.

Crie componentes

Componentização

Acompanhe algumas dicas para aproveitar ao máximo a componentização no Vue.js:

- **Defina propriedades para permitir personalização:** Ao projetar seus componentes, pense nas opções de personalização que você deseja fornecer. Defina propriedades (props) para permitir que os usuários do componente personalizem seu comportamento e aparência. Isso torna seus componentes mais flexíveis e adaptáveis a diferentes cenários de uso.

- **Use eventos personalizados para comunicação entre componentes:** Componentes muitas vezes precisam se comunicar uns com os outros. Use eventos personalizados para emitir e ouvir eventos entre componentes. Isso permite que os componentes interajam de maneira eficiente, mantendo a independência e a coesão.

Crie componentes

Componentização

Acompanhe algumas dicas para aproveitar ao máximo a componentização no Vue.js:

- **Organize seus componentes em uma estrutura de diretórios:** Conforme seu aplicativo cresce, pode se tornar difícil gerenciar todos os seus componentes em um único arquivo ou diretório. Organize seus componentes em uma estrutura de diretórios clara e lógica, compondo com os princípios de separação de preocupações e reutilização.

- **Reutilize componentes de terceiros:** Aproveite a comunidade Vue.js e a vasta coleção de componentes de terceiros disponíveis. Ao invés de reinventar a roda, procure bibliotecas e frameworks que ofereçam componentes que atendam às suas necessidades. Isso economizará tempo e esforço no desenvolvimento.

- **Teste seus componentes:** Ao criar componentes, não se esqueça de testá-los. Use ferramentas de teste, como o Vue Test Utils, para escrever testes automatizados para seus componentes. Isso ajudará a garantir que eles funcionem conforme o esperado e fornecerá confiança durante o desenvolvimento.

escola britânica de artes criativas & tecnologia

Bons estudos!