Criando seu Primeiro Projeto React com Vite

Por Escola Dnc

Introdução

O React é uma das bibliotecas JavaScript mais populares para desenvolvimento de interfaces de usuário. Neste ebook, vamos explorar como criar seu primeiro projeto React utilizando o Vite, uma ferramenta moderna e eficiente para construção de aplicações web.

Ao longo deste material, você aprenderá:

- O que é o Vite e por que usá-lo para projetos React
- Como configurar seu ambiente de desenvolvimento
- Os passos para criar um novo projeto React com Vite
- A estrutura básica de um projeto React
- Como executar e visualizar sua aplicação
- Conceitos fundamentais do React

Vamos mergulhar nesse mundo fascinante do desenvolvimento front-end com React!

Capítulo 1: Introdução ao Vite

O que é o Vite?

O Vite (pronuncia-se "vit") é uma ferramenta de construção de projetos front-end que tem ganhado muita popularidade recentemente. Desenvolvido por Evan You, o criador do Vue.js, o Vite tem como objetivo proporcionar uma experiência de desenvolvimento mais rápida e leve para projetos web modernos.

Por que usar o Vite?

1. Velocidade: O Vite utiliza ES modules nativos do navegador durante o desenvolvimento, o que resulta em um tempo de inicialização extremamente rápido.
2. Flexibilidade: Embora tenha sido criado inicialmente para o Vue.js, o Vite suporta diversos frameworks e bibliotecas, incluindo React, Preact, Svelte e Vanilla JavaScript.
3. Configuração mínima: O Vite vem com configurações padrão sensatas, permitindo que você comece a desenvolver rapidamente sem a necessidade de configurações complexas.
4. Hot Module Replacement (HMR): O Vite oferece HMR ultrarrápido, permitindo que você veja as mudanças no seu código quase instantaneamente no navegador.
5. Otimização para produção: Quando você está pronto para fazer o build do seu projeto, o Vite utiliza Rollup para criar bundles altamente otimizados.

Vite vs Create React App

Anteriormente, a forma mais comum de iniciar um projeto React era usando o Create React App (CRA). No entanto, a própria equipe do React agora recomenda o uso de ferramentas mais modernas como o Vite. Vamos comparar as duas abordagens:

1. Velocidade de inicialização:
    - Vite: Extremamente rápido, geralmente leva poucos segundos.
    - CRA: Pode levar vários minutos, especialmente em máquinas menos potentes.
2. Tempo de compilação:
    - Vite: Utiliza ES modules nativos, resultando em compilações quase instantâneas.
    - CRA: Usa webpack, que pode ser mais lento, especialmente em projetos maiores.
3. Configuração:
    - Vite: Configuração mínima necessária, fácil de personalizar.
    - CRA: Configuração mais robusta, mas pode ser difícil de modificar sem ejetar.
4. Suporte a outros frameworks:
    - Vite: Suporta múltiplos frameworks além do React.
    - CRA: Focado exclusivamente em React.
5. Manutenção:
    - Vite: Ativamente mantido e em constante evolução.
    - CRA: Ainda mantido, mas com menos atualizações frequentes.

Embora o Create React App ainda funcione e seja uma opção válida, o Vite oferece uma experiência de desenvolvimento mais moderna e eficiente, alinhada com as práticas atuais de desenvolvimento web.

Capítulo 2: Preparando o Ambiente de Desenvolvimento

Antes de começarmos a criar nosso projeto React com Vite, precisamos garantir que nosso ambiente de desenvolvimento esteja corretamente configurado. Vamos passar pelos passos necessários:

1. Instalando o Node.js

O Node.js é essencial para o desenvolvimento com React e Vite. Ele nos fornece o npm (Node Package Manager), que usaremos para instalar dependências e executar scripts.

Para instalar o Node.js:

1. Visite o site oficial do Node.js: https://nodejs.org/
2. Baixe a versão LTS (Long Term Support) recomendada para a maioria dos usuários.
3. Execute o instalador e siga as instruções na tela.

Para verificar se a instalação foi bem-sucedida, abra o terminal e digite:

```
node --versionnpm --version
```

Ambos os comandos devem retornar números de versão.

2. Escolhendo um Editor de Código

Embora você possa usar qualquer editor de texto para escrever código React, um bom IDE (Integrated Development Environment) pode melhorar significativamente sua produtividade. Algumas opções populares incluem:

- Visual Studio Code (recomendado)
- WebStorm
- Atom
- Sublime Text

Neste ebook, usaremos o Visual Studio Code (VS Code) como exemplo, devido à sua popularidade e excelente suporte para desenvolvimento React.

Para instalar o VS Code:

1. Visite https://code.visualstudio.com/
2. Baixe e instale a versão apropriada para seu sistema operacional.

3. Extensões Úteis para o VS Code

Após instalar o VS Code, considere adicionar algumas extensões que tornarão o desenvolvimento React mais fácil:

1. ESLint: Para linting de JavaScript e JSX.
2. Prettier: Para formatação automática de código.
3. ES7+ React/Redux/React-Native snippets: Para snippets úteis de React.
4. vscode-icons: Para ícones de arquivo mais intuitivos.

Para instalar extensões no VS Code:

1. Abra o VS Code

2. Clique no ícone de extensões na barra lateral esquerda (ou use o atalho Ctrl+Shift+X)
3. Pesquise pela extensão desejada e clique em "Install"

4. Configurando o Git (opcional, mas recomendado)

O Git é um sistema de controle de versão que ajuda a gerenciar e rastrear mudanças no seu código. Embora não seja estritamente necessário para desenvolver com React e Vite, é uma ferramenta valiosa para qualquer desenvolvedor.

Para instalar o Git:

1. Visite https://git-scm.com/
2. Baixe e instale a versão apropriada para seu sistema operacional.

Após a instalação, configure seu nome de usuário e email no Git:

```
git config --global user.name "Seu Nome"git config --global
user.email "seuemail@exemplo.com"
```

5. Familiarizando-se com o Terminal

Muitas operações no desenvolvimento web moderno são realizadas através do terminal. No Windows, você pode usar o PowerShell ou o novo Windows Terminal. No macOS ou Linux, o terminal padrão é suficiente.

Alguns comandos básicos que você deve conhecer:

- `cd` : Mudar de diretório
- `ls` (ou `dir` no Windows): Listar conteúdo do diretório
- `mkdir` : Criar um novo diretório
- `rm` (ou `del` no Windows): Remover arquivos ou diretórios

6. Testando a Instalação do npm

O npm vem instalado com o Node.js. Para verificar se está funcionando corretamente, você pode tentar instalar um pacote globalmente:

```
npm install -g serve
```

Este comando instala o `serve`, uma ferramenta útil para servir aplicações web estáticas localmente.

Com essas configurações, seu ambiente de desenvolvimento está pronto para começar a criar projetos React com Vite!

Capítulo 3: Criando seu Primeiro Projeto React com Vite

Agora que temos nosso ambiente configurado, vamos criar nosso primeiro projeto React utilizando o Vite. Siga os passos abaixo:

1. Escolhendo um Local para o Projeto

Primeiro, decida onde você quer criar seu projeto. Você pode criar uma pasta específica para seus projetos de código, por exemplo:

```
mkdir ~/Projetoscd ~/Projetos
```

2. Criando o Projeto

Para criar um novo projeto React com Vite, use o seguinte comando:

```
npm create vite@latest
```

Este comando irá iniciar o processo de criação do projeto. Você será guiado por algumas perguntas:

1. **Nome do projeto**: Digite o nome desejado para seu projeto, por exemplo "meu-primeiro-react".
2. **Framework**: Selecione "React" na lista de opções.
3. **Variante**: Escolha "JavaScript" (ou "TypeScript" se você preferir usar TypeScript).

Após responder essas perguntas, o Vite criará a estrutura básica do projeto para você.

3. Navegando para o Diretório do Projeto

Após a criação, navegue para o diretório do projeto:

```
cd meu-primeiro-react
```

4. Instalando as Dependências

Agora, precisamos instalar as dependências do projeto. Execute:

```
npm install
```

Este comando lê o arquivo `package.json` e instala todas as dependências listadas.

5. Estrutura do Projeto

Após a instalação, você terá uma estrutura de projeto semelhante a esta:

```
meu-primeiro-react/├── node_modules/├── public/│   └── vite.svg├──
src/│   ├── assets/│   │   └── react.svg│   ├── App.css│   ├──
App.jsx│   ├── index.css│   └── main.jsx├── .eslintrc.cjs├──
.gitignore├── index.html├── package.json├── package-lock.json└──
vite.config.js
```

Vamos entender cada parte:

- `node_modules/` : Contém todas as dependências instaladas.
- `public/` : Arquivos estáticos que serão servidos diretamente.
- `src/` : Contém o código-fonte da sua aplicação.
- `App.jsx` : O componente principal da sua aplicação.
- `main.jsx` : O ponto de entrada da sua aplicação.
- `index.html` : O arquivo HTML principal.
- `package.json` : Lista as dependências e scripts do projeto.
- `vite.config.js` : Configurações do Vite.

6. Executando o Projeto

Para iniciar o servidor de desenvolvimento, execute:

```
npm run dev
```

Isso iniciará o servidor de desenvolvimento do Vite. Você verá uma mensagem no terminal com um URL local, geralmente `http://localhost:5173` . Abra este URL

no seu navegador para ver sua aplicação React rodando!

7. Explorando o Projeto

Abra o projeto no VS Code:

```
code .
```

Explore os arquivos, especialmente `src/App.jsx`. Este é o componente principal da sua aplicação. Experimente fazer algumas alterações neste arquivo e veja-as refletidas instantaneamente no navegador graças ao Hot Module Replacement (HMR) do Vite.

8. Entendendo o Código Inicial

Vamos dar uma olhada mais detalhada no `App.jsx`:

```
import { useState } from 'react'import reactLogo from
'./assets/react.svg'import viteLogo from '/vite.svg'import
'./App.css'function App() {  const [count, setCount] = useState(0)
return (    <>      <div>        <a href="https://vitejs.dev"
target="_blank">          <img src={viteLogo} classNAme="logo"
alt="Vite logo" />        </a>        <a href="https://react.dev"
target="_blank">          <img src={reactLogo} classNAme="logo
react" alt="React logo" />        </a>      </div>      <h1>Vite +
React</h1>      <div classNAme="card">        <button onClick={()
=> setCount((count) => count + 1)}>          count is {count}
</button>        <p>          Edit <code>src/App.jsx</code> and
save to test HMR        </p>      </div>      <p classNAme="read-
the-docs">        Click on the Vite and React logos to learn more
</p>    </>  )}export default App
```

Este código demonstra alguns conceitos básicos do React:

- Importação de módulos e assets

- Uso do hook `useState` para gerenciar estado
- JSX para descrever a UI
- Manipulação de eventos (no botão)

Parabéns! Você criou com sucesso seu primeiro projeto React usando Vite. Nas próximas seções, exploraremos mais a fundo os conceitos do React e como desenvolver componentes mais complexos.

Capítulo 4: Fundamentos do React

Agora que temos nosso projeto React configurado e rodando, vamos explorar alguns dos conceitos fundamentais do React que você encontrará ao desenvolver aplicações.

1. Componentes

Componentes são os blocos de construção fundamentais de qualquer aplicação React. Um componente é uma parte isolada e reutilizável da interface do usuário. Em React, existem dois tipos principais de componentes:

Componentes Funcionais

Estes são funções JavaScript que retornam JSX. São a forma moderna e recomendada de escrever componentes React.

Exemplo:

```
function Saudacao(props) {  return <h1>Olá, {props.nome}!</h1>;}
```

Componentes de Classe

Estes são classes ES6 que estendem `React.Component`. Embora ainda suportados, são menos comuns em código React moderno.

Exemplo:

```
class Saudacao extends React.Component {  render() {    return
<h1>Olá, {this.props.nome}!</h1>;  }}
```

2. JSX

JSX é uma extensão de sintaxe para JavaScript que parece com HTML. Ele permite que você escreva estruturas de UI diretamente no seu código JavaScript.

Exemplo:

```
const elemento = <h1>Isto é JSX</h1>;
```

JSX é transformado em chamadas de função React.createElement() durante a compilação.

3. Props

Props (abreviação de "properties") são a forma de passar dados de um componente pai para um componente filho. São somente leitura.

Exemplo:

```
function BemVindo(props) {  return <h1>Bem-vindo, {props.nome}!
</h1>;}// Uso:<BemVindo nome="Maria" />
```

4. Estado (State)

O estado é um objeto que contém dados que podem mudar ao longo do tempo. Em componentes funcionais, usamos o hook useState para gerenciar o estado.

Exemplo:

```
import React, { useState } from 'react';function Contador() {
const [count, setCount] = useState(0);  return (    <div>
<p>Você clicou {count} vezes</p>      <button onClick={() =>
setCount(count + 1)}>        Clique aqui      </button>    </div>
);}
```

5. Eventos

React tem um sistema de eventos sintéticos que funciona de forma semelhante

Exemplo:

```
function Botao() {  function handleClick() {    alert('O botão foi
clicado!');  }  return <button onClick={handleClick}>Clique em
mim</button>;}
```

6. Renderização Condicional

Em React, você pode criar UIs distintas baseadas em condições.