

Desenvolvimento com React: Introdução e Primeiro Projeto

Site: [Geração Tech](#)
Curso: Formação em Desenvolvedor Web – Online
Livro: Desenvolvimento com React: Introdução e Primeiro Projeto

Impresso por: JOÃO VITOR DE MELO FREITAS
Data: quinta-feira, 18 jul. 2024, 22:38

Índice

1. React

1.1. Criando o Projeto com Vite

1.2. Criando Componentes

1. React

Preparação do Ambiente

Olá pessoal, tudo bem com vocês? Sou eu, Gleidson Teixeira, e estamos começando nosso desenvolvimento com [React](#). Vamos configurar nosso ambiente e criar nosso primeiro projeto com [React](#).

Acessando a Documentação Oficial

Primeiramente, acesse a página oficial do [React react.dev](#). Na seção "Learn [React](#)", você encontrará tutoriais e documentação que ajudarão a entender a biblioteca. Embora a documentação sugira usar o comando `create-react-app`, vamos utilizar uma abordagem diferente e mais leve com o [Vite](#).

Por que Não Usar `create-react-app`?

O comando `create-react-app` é robusto, mas pode ser pesado e baixar mais pacotes do que realmente precisamos. Em vez disso, vamos utilizar o [Vite](#), uma ferramenta de build rápida que simplifica a configuração inicial do projeto.

Instalando Node.js

Antes de começar, certifique-se de ter o Node.js instalado no seu computador. Você pode baixá-lo no site oficial [nodejs.org](#).

dia 16 video 1



1.1. Criando o Projeto com Vite

Vamos criar nosso primeiro projeto [React](#) usando o [Vite](#). Siga os passos abaixo:

1. Abrir o Terminal:

- Abra o terminal integrado do VSCode ou o terminal do seu sistema operacional.

2. Navegar até a Pasta de Documentos:

- Use o comando `cd` para navegar até a pasta onde você deseja criar o projeto. Exemplo:

```
bash
```

```
Copiar código
```

```
cd /caminho/para/sua/pasta
```

3. Executar o Comando de Criação do Projeto:

- Execute o comando para criar o projeto com Vite:

```
bash
```

```
Copiar código
```

```
npm create vite@latest
```

- O Vite pedirá o nome do projeto. Digite `react-aula1` e escolha [React](#) como o template e `JavaScript` como a linguagem.

4. Instalar Dependências:

- Navegue até a pasta do projeto e instale as dependências:

```
bash
```

```
Copiar código
```

```
cd react-aula1 npm install
```

5. Abrir o Projeto no VSCode:

- Abra o projeto no VSCode:

```
bash
```

```
Copiar código
```

```
code .
```

Estrutura do Projeto

O Vite cria uma estrutura básica de pastas e arquivos para o projeto [React](#). Vamos explorar os principais arquivos:

- `index.html`: O ponto de entrada da aplicação.
- `src/main.jsx`: O arquivo principal que inicializa o [React](#).
- `src/App.jsx`: O componente principal da aplicação.

Executando a Aplicação

Para ver sua aplicação [React](#) em ação, execute o comando:

```
bash
```

```
Copiar código
```

```
npm run dev
```

Isso iniciará um servidor de desenvolvimento. Abra o navegador e acesse `http://localhost:3000` para ver a aplicação rodando.

Editando o Componente Principal

Vamos editar o componente principal para entender como o [React](#) funciona. Abra o arquivo `src/App.jsx` e modifique o conteúdo:

jsx

 Copiar código

```
import { useState } from 'react'; import './App.css'; function App() { const [count, setCount] = useState(0); return ( <div  
  className="App"> <h1>Olá, React!</h1> <button onClick={() => setCount((count) => count + 1)}> Contador: {count} </button>  
</div> ); } export default App;
```

Salve o arquivo e veja as mudanças refletidas no navegador.

dia 16 video 2



Conclusão

Nesta aula, configuramos nosso ambiente de desenvolvimento [React](#) usando o Vite e criamos nosso primeiro projeto. Exploramos a estrutura básica do projeto e editamos o componente principal. Nas próximas aulas, vamos aprofundar no desenvolvimento com [React](#), aprendendo a criar componentes, gerenciar estados e muito mais.

Espero vocês na próxima aula!

1.2. Criando Componentes

Estrutura do Projeto [React](#)


Olá pessoal, tudo bem com vocês? Vamos continuar nosso desenvolvimento com [React](#), utilizando nosso primeiro projeto da aula 1 para explicar algumas coisas importantes. Vamos entender a estrutura do nosso projeto [React](#) e aprender a criar componentes.

Estrutura do Projeto

Nosso projeto [React](#) tem a seguinte estrutura básica:

- **public/index.html**: Contém a estrutura HTML principal do projeto. Aqui temos uma `div` com o ID `root`, onde nosso [React](#) vai injetar os componentes.
- **src/main.jsx**: Arquivo principal que inicializa o [React](#). Ele importa o componente `App` e renderiza dentro da `div` com ID `root`:

jsx

 Copiar código

```
import React from 'react'; import ReactDOM from 'react-dom'; import App from './App.jsx'; ReactDOM.render(<App />, document.getElementById('root'));
```

- **src/App.jsx**: O componente principal da aplicação.

Criando o Primeiro Componente


Vamos criar nosso primeiro componente [React](#). No arquivo `App.jsx`, podemos definir nosso componente principal.

1. Abrindo o arquivo `App.jsx`:

- Abra o arquivo `src/App.jsx`.
- Apague o conteúdo existente para criar um componente do zero.

2. Definindo o Componente:

jsx

 Copiar código

```
import React from 'react'; const App = () => { return ( <div> <h1>Olá, React!</h1> </div> ); }; export default App;
```

Explicação do Código

- **Função do Componente:**
 - Definimos uma função chamada `App` que retorna um elemento JSX. Este elemento é o que será renderizado na tela.
 - Usamos a sintaxe de arrow function para definir a função do componente.
- **JSX:**
 - JSX é uma extensão de sintaxe do JavaScript que permite escrever HTML dentro do JavaScript.
 - No retorno da função, temos um elemento `div` com um `h1` dentro.

Adicionando Mais Componentes

Podemos criar e usar outros componentes dentro do componente `App`.

1. Criando um Componente `Hello`:

- Dentro do mesmo arquivo `App.jsx`, vamos criar outro componente chamado `Hello`:

jsx

 Copiar código

```
const Hello = () => { return ( <h1>Olá, Mundo!</h1> ); };
```

2. Usando o Componente `Hello` no Componente `App`:

- Para usar o componente `Hello` dentro do componente `App`, basta incluí-lo como um elemento JSX:

jsx

 Copiar código

```
const App = () => { return ( <div> <Hello /> </div> ); };
```

Importância de Componentes em Letra Maiúscula

- No [React](#), os nomes dos componentes devem começar com letra maiúscula. Isso ajuda o [React](#) a distinguir entre elementos HTML nativos e componentes [React](#).

Salvando e Visualizando as Alterações

1. Salvar e Verificar no Navegador:

- Salve o arquivo `App.jsx`.
- No navegador, você deve ver a mensagem "Olá, Mundo!" renderizada na tela.

dia 16 video 3



Conclusão

Aprendemos a criar e usar componentes em [React](#), entendendo a importância da estrutura de pastas e arquivos. Componentes são a base do [React](#) e nos permitem criar interfaces de usuário modulares e reutilizáveis.