Projeto ReactJS com TypeScript: Estado e Ciclo de Vida dos Componentes

Passo 1: Configurar o Projeto com Vite e TypeScript

1. Instale o Node.js e npm:

Certifique-se de que você tem o Node.js e o npm instalados. Você pode baixar a versão LTS do Node.js no site oficial.

2. Crie um Novo Projeto com Vite e TypeScript:

- Abra o terminal ou o prompt de comando.

- Execute o comando abaixo para criar um novo projeto com Vite e TypeScript. Substitua 'meu-projeto' pelo nome do seu projeto.

```
npm create vite@latest meu-projeto -- --template react-ts
```

3. Navegue até o Diretório do Projeto:

- Entre no diretório do seu projeto recém-criado.

```
cd meu-projeto
```

4. Instale as Dependências:

- Execute o comando abaixo para instalar todas as dependências necessárias.

```
npm install
```

5. Inicie o Servidor de Desenvolvimento:

- Após a instalação das dependências, inicie o servidor de desenvolvimento.

npm run dev

- O Vite iniciará um servidor de desenvolvimento e fornecerá um link (geralmente http://localhost:3000) para visualizar seu projeto no navegador.

Passo 2: Implementação do Componente com Estado e Ciclo de Vida

Agora que o projeto está configurado, vamos criar um componente que exemplifique o uso de estado e ciclo de vida em um componente de classe.

1. Modifique o arquivo 'src/App.tsx':

Abra o arquivo 'src/App.tsx' e modifique o código para incluir um componente de classe que usa estado e métodos de ciclo de vida:

```
import React from 'react';
import './App.css';

// Definindo a interface para o estado do componente
interface ClockState {
  date: Date;
}

class Clock extends React.Component<{}, ClockState> {
  private timerID?: NodeJS.Timeout;
```

Projeto ReactJS com TypeScript: Estado e Ciclo de Vida dos Componentes

```
constructor(props: {}) {
  super(props);
  // Inicializando o estado
  this.state = { date: new Date() };
}

// Método chamado imediatamente após o componente ser montado no DOM
componentDidMount() {
  this.timerID = setInterval(() => this.tick(), 1000);
}

// Método chamado imediatamente antes do componente ser desmontado e destruído
componentWillUnmount() {
  clearInterval(this.timerID);
}

// Método que atualiza o estado com a hora atual
tick() {
  this.setState({ date: new Date() });
}

render() {
  return (
```

```
      <div>
        <h1>Hora Atual:</h1>
        <h2>{this.state.date.toLocaleTimeString()}</h2>
      </div>
    );
  }
}

function App() {
  return (
    <div className='App'>
      <Clock />
    </div>
  );
}

export default App;
```

Explicação do Código

- ClockState Interface: Define o tipo do estado que o componente Clock irá gerenciar. Neste caso, é apenas uma propriedade date do tipo Date.

- Clock Componente de Classe:

  - Estado: Inicializado no construtor com a hora atual.

Projeto ReactJS com TypeScript: Estado e Ciclo de Vida dos Componentes

- Ciclo de Vida:

  - componentDidMount: Este método é chamado assim que o componente é montado no DOM. Aqui, ele define um temporizador que chama o método tick a cada segundo.

  - componentWillUnmount: Este método é chamado imediatamente antes do componente ser desmontado. Aqui, o temporizador é limpo para evitar atualizações no estado de um componente desmontado.

  - tick: Este método atualiza o estado com a hora atual, o que faz com que o componente seja re-renderizado.

- App Componente: O componente principal (App) renderiza o componente Clock, que mostra a hora atual e atualiza a cada segundo.

Estrutura do Projeto

O projeto deve ter a seguinte estrutura após a modificação:

meu-projeto/

??? node_modules/

??? public/

?   ??? vite.svg

??? src/

?   ??? App.css

?   ??? App.tsx

?   ??? index.css

?   ??? main.tsx

?   ??? vite-env.d.ts

??? .gitignore

Projeto ReactJS com TypeScript: Estado e Ciclo de Vida dos Componentes

??? index.html

??? package.json

??? tsconfig.json

??? vite.config.ts

Conclusão

Esse exemplo simples demonstra como usar o estado (state) e os métodos de ciclo de vida em um componente de classe React com TypeScript. O componente Clock exibe a hora atual e a atualiza a cada segundo, enquanto gerencia corretamente o ciclo de vida do componente para garantir que o temporizador seja limpo quando o componente for desmontado.