Projeto ReactJS com TypeScript: Renderização Condicional e Listas

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 de Renderização Condicional e Listas

Agora que o projeto está configurado, vamos criar um componente que exemplifique a renderização condicional e o uso de listas.

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

Abra o arquivo 'src/App.tsx' e modifique o código para incluir exemplos de renderização condicional e listas:

```
import React, { useState } from 'react';
import './App.css';

// Definindo a interface para os itens da lista
interface Item {
  id: number;
  name: string;
}

// Componente de Lista que renderiza itens com base em uma condição
const ItemList: React.FC = () => {
```

Projeto ReactJS com TypeScript: Renderização Condicional e Listas

```
const [items, setItems] = useState<Item[]>([
  { id: 1, name: 'Apple' },
  { id: 2, name: 'Banana' },
  { id: 3, name: 'Cherry' },
]);

const [showItems, setShowItems] = useState<boolean>(true);

return (
  <div>
    <h1>Lista de Itens</h1>
    <button onClick={() => setShowItems(!showItems)}>
      {showItems ? 'Esconder Itens' : 'Mostrar Itens'}
    </button>

    {/* Renderização Condicional */}
    {showItems ? (
      <ul>
        {/* Renderizando a lista de itens */}
        {items.map((item) => (
          <li key={item.id}>{item.name}</li>
        ))}
      </ul>
    ) : (
```

```
        <p>A lista está oculta</p>
      )}
    </div>
  );
};

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

export default App;
```

Explicação do Código

- Item Interface: Define o tipo de dados que cada item da lista terá. Neste caso, um id do tipo number e um name do tipo string.
- ItemList Componente Funcional:
  - Estado (state): Utiliza o hook useState para gerenciar dois estados:
    - items: uma lista de objetos do tipo Item.
    - showItems: um booleano que controla se a lista de itens será mostrada ou ocultada.

Projeto ReactJS com TypeScript: Renderização Condicional e Listas

- Renderização Condicional: Baseada no estado showItems, o componente decide se deve mostrar a lista de itens ou uma mensagem indicando que a lista está oculta.

- Renderização de Lista: Utiliza o método map para renderizar cada item da lista como um <li> dentro de um <ul>. Cada item tem uma chave (key) única, que é o id do item.

- App Componente: O componente principal (App) simplesmente renderiza o componente ItemList.

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

??? index.html

??? package.json

??? tsconfig.json

Projeto ReactJS com TypeScript: Renderização Condicional e Listas

??? vite.config.ts

Conclusão

Esse exemplo simples demonstra como utilizar a renderização condicional e como renderizar listas em um componente React com TypeScript. A aplicação permite que o usuário mostre ou oculte uma lista de itens, exemplificando o uso de renderização condicional e a importância de utilizar chaves únicas ao renderizar listas de elementos.