REACT HOOKS & ROUTER DOM

Introdução aos React Hooks

Os React Hooks são uma adição poderosa ao React que permitem que você use o estado e outros recursos do React sem escrever uma classe. Eles foram introduzidos no React 16.8 e oferecem uma maneira mais simples e concisa de gerenciar o estado e o ciclo de vida dos componentes.

Agora, vamos dar uma olhada nos exemplos práticos dos principais hooks do React:

1. useState

O `useState` é um dos hooks mais básicos e essenciais. Ele permite adicionar o estado a um componente de função.

```
import React, { useState } from 'react';

function Counter() {
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>Você clicou {count} vezes</p>
```

```
      <button onClick={() => setCount(count + 1)}>
        Clique aqui
      </button>
    </div>
  );
}
```

2. useEffect

O `useEffect` é usado para realizar operações secundárias em componentes de função, como efeitos colaterais (por exemplo, chamadas de API, manipulação de DOM).

```
import React, { useState, useEffect } from 'react';

function DataFetching() {
  const [data, setData] = useState(null);

  useEffect(() => {
    const fetchData = async () => {
      const response = await fetch('https://api.example.co
m/data');
      const jsonData = await response.json();
      setData(jsonData);
    };

    fetchData();
  }, []); // O segundo parâmetro vazio indica que este efei
to só é executado uma vez

  return (
    <div>
      {data ? (
        <p>Dados carregados: {data}</p>
      ) : (
        <p>Carregando dados...</p>
      )}
    </div>
```

```
  );
}
```

3. useContext

O `useContext` é usado para acessar o contexto em componentes de função.

```
import React, { useContext } from 'react';
import MyContext from './MyContext';

function MyComponent() {
  const value = useContext(MyContext);
  return <p>Valor do contexto: {value}</p>;
}
```

Introdução ao React Router DOM

O React Router DOM é uma biblioteca que possibilita a navegação declarativa e dinâmica em aplicações React. Ele permite que você defina diferentes rotas para diferentes partes do seu aplicativo e renderize componentes específicos com base na URL atual. Isso é crucial para construir aplicações de página única (SPA) e aplicativos da web com várias páginas.

Instalação do React Router DOM

Para começar, você precisa instalar o React Router DOM no seu projeto. Você pode fazer isso executando o seguinte comando no terminal, dentro do diretório do seu projeto:

```
npm install react-router-dom
```

Configuração do arquivo de rotas

Agora, vamos criar um arquivo separado para configurar e agrupar as rotas da sua aplicação. Vamos chamá-lo de `routes.js`:

```
import React from 'react';
import { BrowserRouter as Router, Route, Switch } from 'rea
ct-router-dom';
import Home from './components/Home';
import About from './components/About';
import Contact from './components/Contact';
import NotFound from './components/NotFound';

function Routes() {
  return (
    <Router>
      <Switch>
        <Route path="/" exact component={Home} />
        <Route path="/about" component={About} />
        <Route path="/contact" component={Contact} />
        <Route component={NotFound} />
      </Switch>
    </Router>
  );
}

export default Routes;
```

Integração do Context Provider no componente principal

Agora, vamos integrar o provedor de contexto do React Router DOM no componente principal (`App.js`). Isso é necessário para que os componentes da sua aplicação possam acessar informações sobre a navegação atual, como o histórico de navegação e os parâmetros da URL.

```
import React from 'react';
import { BrowserRouter as Router } from 'react-router-dom';
import Routes from './routes';

function App() {
  return (
    <Router>
      <Routes />
```

```
    </Router>
  );
}

export default App;
```

Utilizando o arquivo de configuração de rotas na sua aplicação

Finalmente, você pode utilizar o arquivo de configuração de rotas (`routes.js`) na sua aplicação. Por exemplo, você pode importá-lo e renderizá-lo em qualquer componente onde deseja que as rotas sejam exibidas:

```
import React from 'react';
import Routes from '../routes';

function Main() {
  return (
    <div>
      <h1>Minha Aplicação</h1>
      <Routes />
    </div>
  );
}

export default Main;
```

Dessa forma, você separa a lógica de roteamento da sua aplicação em um arquivo dedicado (`routes.js`), facilitando a manutenção e organização do seu código. O provedor de contexto do React Router DOM (`BrowserRouter`) é integrado no componente principal (`App.js`), garantindo que todas as rotas tenham acesso ao contexto de navegação.