Hooks

1. O que são Hooks?

Hooks são funções que permitem "enganchar" (hook into) recursos do React, como estado e ciclo de vida, diretamente em componentes funcionais. Eles promovem uma abordagem mais funcional e declarativa, simplificando a lógica e melhorando a reutilização de código. Conhecer e dominar os principais hooks é essencial para qualquer desenvolvedor React, pois eles são fundamentais para escrever código limpo e eficiente.

Os hooks mais básicos e importantes são **useState** e **useEffect**, mas existem vários outros que atendem a diferentes necessidades.

1. **useState**: Adiciona estado local a um componente funcional.
2. **useEffect**: Permite realizar efeitos colaterais, como fetch de dados ou manipulações do DOM.
3. **useContext**: Consome contextos para compartilhar estado global entre componentes.
4. **useReducer**: Gerencia estados mais complexos de forma previsível usando funções redutoras.
5. **useMemo** e **useCallback**: Otimizam performance memorizando valores e funções.

2. useState

O useState é um hook que permite adicionar estado a componentes funcionais no React. Ele retorna um par de valores: o **estado atual** e **uma função** que permite atualizá-lo.

Podemos utilizar o useState sempre que precisamos armazenar e atualizar valores que mudam ao longo do tempo dentro de um componente. É ideal para gerenciar estados simples como inputs de formulário, visibilidade de elementos e seleção de itens.

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

function App() {
  const [isVisivel, setIsVisivel] = useState(false);

  const toggleVisibilidade = () => {
    setIsVisivel(!isVisivel);
  };

  return (
    <div>
      <button onClick={toggleVisibilidade}>
        {isVisivel ? 'Esconder' : 'Mostrar'} Mensagem
      </button>
      {isVisivel && <p>Esta é a mensagem!</p>}
    </div>
  );
}

export default App;
```

3. useEffect

O useEffect é um hook que permite realizar efeitos colaterais em componentes funcionais. Exemplos de efeitos colaterais incluem buscar dados e atualizar o DOM. O useEffect aceita dois argumentos: **uma função** que contém o efeito e **uma lista** de dependências que determina quando o efeito deve ser reexecutado.

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

function App() {
  const [usuarios, setUsuarios] = useState([]);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    async function fetchData() {
      const response = await
fetch('https://jsonplaceholder.typicode.com/users');
      const data = await response.json();
      setUsuarios(data);
```

```
      setLoading(false);
    }

    fetchData();
  }, []); // Executa apenas uma vez, após o primeiro render

  if (loading) {
    return <div>Carregando...</div>;
  }

  return (
    <div>
      <h1>Lista de Usuários</h1>
      <ul>
        {usuarios.map(user => (
          <li key={user.id}>{user.name}</li>
        ))}
      </ul>
    </div>
  );
}

export default App;
```