**useState**

O useState é um Hook do React que permite adicionar o estado do React a componentes funcionais. Antes da introdução dos Hooks, o estado só podia ser usado em componentes de classe. Agora, com o useState, você pode usar estados em componentes funcionais, tornando o código mais limpo e fácil de entender.

**Como usar**

Aqui está um exemplo básico de como usar o useState:

javascript

import React, { useState } from 'react';

function Contador() {

const [contador, setContador] = useState(0);

return (

<div>

<p>Você clicou {contador} vezes</p>

<button onClick={() => setContador(contador + 1)}>Clique aqui</button>

</div>

);

}

Nesse exemplo, useState retorna um par: o valor atual do estado (contador) e uma função que permite atualizá-lo (setContador).

**useEffect**

O useEffect é outro Hook do React que permite realizar efeitos colaterais em componentes funcionais. Efeitos colaterais são operações como buscar dados, modificar diretamente o DOM e executar código que precisa acontecer após cada renderização.

**Como usar**

Aqui está um exemplo básico de como usar o useEffect:

javascript

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

function Temporizador() {

const [segundos, setSegundos] = useState(0);

useEffect(() => {

const intervalId = setInterval(() => {

setSegundos(segundos => segundos + 1);

}, 1000);

return () => clearInterval(intervalId);

}, []);

return <p>{segundos} segundos passados.</p>;

}

Nesse exemplo, o useEffect configura um intervalo que incrementa o estado segundos a cada segundo. O retorno do useEffect é uma função de limpeza que limpa o intervalo quando o componente é desmontado ou quando o efeito é reexecutado.

**Por que são utilizados?**

* useState: É usado para adicionar e gerenciar estado local em componentes funcionais.
* useEffect: É usado para realizar efeitos colaterais como busca de dados, assinaturas e atualizações manuais do DOM.

**Alternativas**

* **Classe de componentes**: Antes dos Hooks, o estado e os efeitos colaterais eram gerenciados em componentes de classe usando this.state, this.setState(), componentDidMount(), componentDidUpdate(), e componentWillUnmount().

javascript

class Contador extends React.Component {

constructor(props) {

super(props);

this.state = { contador: 0 };

}

render() {

return (

<div>

<p>Você clicou {this.state.contador} vezes</p>

<button onClick={() => this.setState({ contador: this.state.contador + 1 })}>

Clique aqui

</button>

</div>

);

}

}

* **Gerenciadores de estado externos**: Em aplicações mais complexas, pode ser útil usar bibliotecas de gerenciamento de estado como Redux, MobX ou Context API do próprio React.