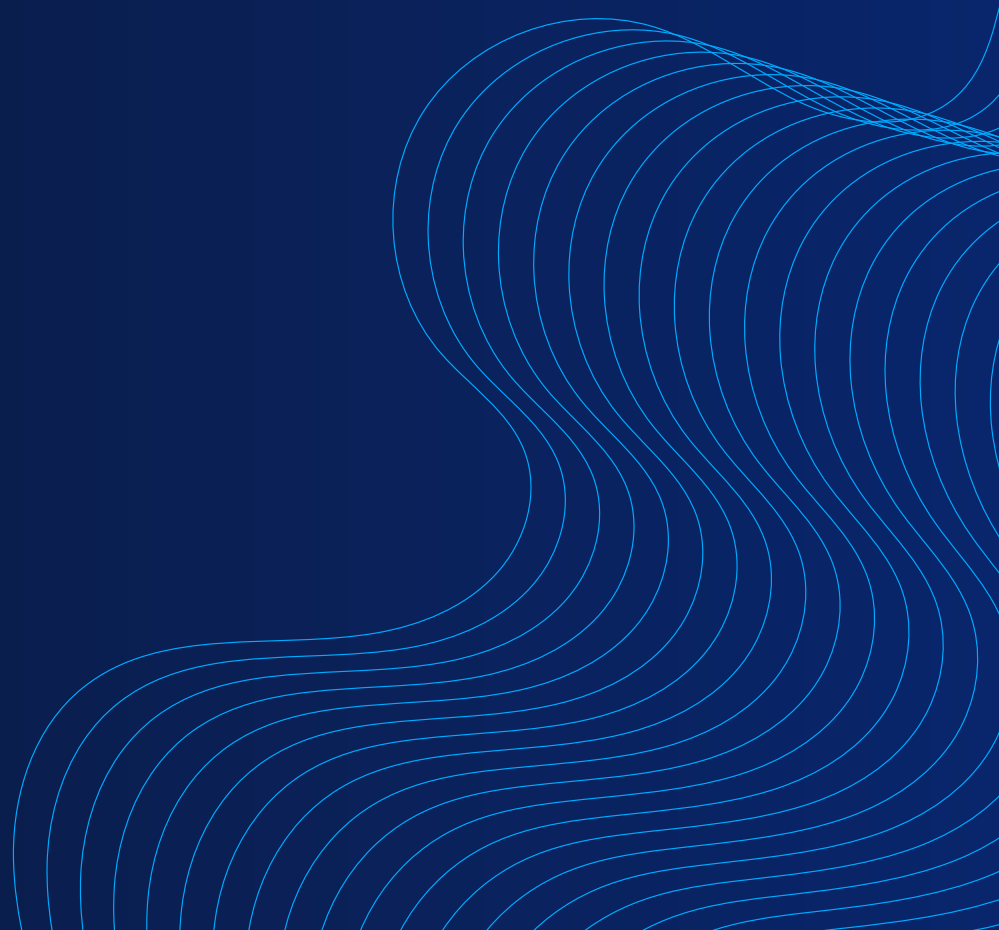




Material de Apoio

O que são
Hooks?



O que são?

Os Hooks são uma funcionalidade do React introduzida na versão 16.8, que permitem o uso de estado e outros recursos do React em componentes funcionais, sem a necessidade de escrever uma classe.

Os Hooks foram desenvolvidos para resolver dois problemas principais:

1. **Reutilização de Lógica:** Antes dos Hooks, a reutilização de lógica entre componentes era complexa e frequentemente envolvia padrões como render props e higher-order components, que podiam aumentar a complexidade do código.
2. **Simplificação do Código:** Componentes funcionais são geralmente mais fáceis de ler e testar do que componentes de classe. Os Hooks permitem que você utilize estado e outros recursos do React diretamente em componentes funcionais, simplificando a estrutura e a manutenção do código.

Principais Hooks

useState

- Permite adicionar estado local a componentes funcionais.



```
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>
  );
}
```

Neste exemplo, contador é uma variável de estado que começa em 0, e setContador é a função que usamos para atualizar esse valor.

Principais Hooks

useEffect

- O hook `useEffect` é usado para realizar efeitos colaterais em componentes funcionais. Neste caso, estamos usando realizar um efeito colateral simples, neste caso, exibir um alerta:



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

const AlertEffectComponent = () => {
  useEffect(() => {
    alert('O componente foi montado!');
  })
  return (
    <div>
      <h1>Componente com Alerta</h1>
    </div>
  );
};

export default AlertEffectComponent;
```

Neste exemplo, `useEffect` configura uma mensagem de alerta quando o componente é montado ou atualizado.

Principais Hooks

useContext

- O useContext permite acessar o valor de um contexto em qualquer componente sem precisar passar propriedades manualmente.



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

const TemaContext = React.createContext('light');

function Botao() {
  const tema = useContext(TemaContext);
  return <button className={tema}>Tema atual: {tema}</button>;
}
```

Aqui, useContext é usado para obter o valor atual do contexto TemaContext.

Outros Hooks Úteis

useReducer

- Útil para gerenciar estados mais complexos.

useMemo

- Memoriza valores calculados para otimizar o desempenho.

useCallback

- Memoriza funções para evitar recriação desnecessária.

useRef


- Cria uma referência mutável que persiste entre renderizações, frequentemente usada para acessar diretamente elementos do DOM.



Hooks Customizados

Você pode criar seus próprios Hooks para reutilizar lógica entre componentes.

Um Hook customizado é apenas uma função que pode usar outros Hooks.



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

function useFetch(url) {
  const [data, setData] = useState(null);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    fetch(url)
      .then(response => response.json())
      .then(data => {
        setData(data);
        setLoading(false);
      });
  }, [url]);

  return { data, loading };
}
```

Introdução aos Hooks

Hooks de Forma Resumida





E aí, curtiu?

Esperamos que esse resumo tenha enriquecido sua perspectiva estratégica para enfrentar os desafios.

Salve esse PDF para consultar sempre que precisar.