

Executar os processos de codificação, manutenção e documentação de aplicativos computacionais para desktop

SENAC PE

Thiago Nogueira

Instrutor de Educação Profissional

React: Introdução aos Hooks

Introdução aos Hooks

O que são Hooks

- Antes de nos aprofundarmos nos Hooks, vamos primeiro falar sobre **componentes funcionais** e **componentes de classe** no React.
- Os componentes de classe eram usados quando você precisava fazer coisas como:
 - Acompanhar dados que mudam ao longo do tempo (chamado de **estado**).
 - Executar determinado código quando um componente aparece na tela ou quando algo muda (chamados de **efeitos colaterais**).
- Por outro lado, os componentes funcionais eram mais simples. Eles não tinham efeitos colaterais ou de estado e eram usados apenas para exibir dados estáticos.
- No entanto, conforme os desenvolvedores começaram a construir aplicativos mais complexos, eles precisaram de efeitos colaterais e de estado, mesmo em componentes funcionais simples. Foi aí que os Hooks entraram.

Introdução aos Hooks

O que são Hooks

- Hooks são funções especiais que permitem que você adicione funcionalidades extras a componentes funcionais. Eles permitem que você faça coisas como:
- Gerenciar **estado** (dados que podem mudar ao longo do tempo, como um contador ou uma entrada de formulário).
- Lide com **efeitos colaterais** (como buscar dados de uma API, configurar temporizadores ou interagir com o navegador).
- Os hooks permitem escrever componentes de uma forma mais simples e organizada usando funções em vez de classes.

Introdução aos Hooks

Por que o React criou os Hooks?

- Digamos que você esteja construindo um aplicativo simples com um botão que aumenta um número na tela cada vez que você clica nele. Esse número é um dado que muda ao longo do tempo e é chamado de **estado** no React.
- Antes de Hooks serem criados, você teria que escrever esse componente de botão usando uma classe para gerenciar o estado, e isso tornava as coisas um pouco mais difíceis de entender, especialmente para iniciantes.
- Componentes de classe envolvem mais configuração, palavras-chave extras e regras diferentes que você tem que seguir.
- Por exemplo, em um componente de classe, você tinha que usar a palavra-chave **this** para se referir ao seu componente, o que poderia facilmente confundir novos desenvolvedores.
- Mas com Hooks, o React nos deu uma maneira mais fácil e limpa de gerenciar coisas como estado e efeitos colaterais em componentes funcionais, que são muito mais simples de escrever.

Introdução aos Hooks

Como funcionam?

Ele permite coisas como:

- Acompanhe as alterações nos seus dados (usando **state**).
- Execute determinado código quando seu componente for carregado ou atualizado (como buscar dados ou definir temporizadores).

Existem diferentes tipos de hooks para diferentes trabalhos, mas os três mais importantes são:

- **useState** : Isso ajuda seu componente a lembrar e atualizar dados alterados, como entradas do usuário ou cliques em botões.
- **useEffect** : Isso ajuda você a executar algum código quando seu componente é criado, atualizado ou removido. É útil para coisas como buscar dados do servidor ou limpar recursos como timers.
- **useContext** : Isso ajuda seu componente a compartilhar dados entre diferentes partes do seu aplicativo sem passar propriedades manualmente em cada nível.

Introdução aos Hooks

Resumo

- Os hooks permitem que você adicione mais funções aos seus componentes funcionais, como gerenciar o estado e lidar com efeitos colaterais.
- Os hooks tornam seu código mais simples, limpo e fácil de entender, especialmente para iniciantes.
- Com Hooks, você pode fazer tudo o que faria em componentes de classe, mas de uma forma mais direta.
- Há três tipos principais:
- `useState` (para gerir o estado),
- `useEffect` (para lidar com efeitos colaterais) e
- `useContext` (para compartilhar dados entre componentes).

Hook useState no React

useState

O que é?

- No React, **useState** é um Hook especial que permite adicionar algo chamado **estado** aos seus componentes funcionais. Mas o que é exatamente estado?
- Pense no estado como uma maneira do seu componente "lembrar" das coisas.
- Por exemplo, imagine que você está construindo um aplicativo de contador simples. O número do contador mudará toda vez que você clicar em um botão. Esse número é o estado do seu componente porque ele continua mudando, mas o componente precisa "lembrar" do valor mais recente.
- Sem estado, seu componente esqueceria o número toda vez que fosse renderizado novamente e ficaria preso exibindo o mesmo valor.
- O **useState** permite que seu componente lembre e atualize valores ao longo do tempo.

useState

Como funciona?

Primeiro, no topo do arquivo, **importamos** o useState do React.

Dentro do componente funcional você **cria a variável de estado**.

- **count**: armazena o valor atual
- **setCount**: função para atualizar o valor de count
- **useState(0)**: informa que count deve começar em 0

Dentro return do componente, **exibimos o valor atual** de count (usando chaves {count}) na tela.

O botão usa o evento **onClick** para chamar a função **setCount**, **atualizando o estado**.
Toda vez que você clicar no botão, **setCount(count + 1)** atualizará a contagem adicionando 1 ao valor atual.

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

```
function ExampleComponent() {  
  // 1. Crie uma variável de estado e uma função para atualizá-la  
  const [count, setCount] = useState(0);
```

```
  return (  
    <div>  
      <p>The current count is: {count}</p>  
      <button onClick={() => setCount(count + 1)}>Increase Count</button>  
    </div>  
  );  
}
```

```
export default ExampleComponent;
```

useState

O que realmente faz?

Vamos analisar mais de perto o que está acontecendo:

- **Estado inicial:** quando o componente é carregado pela primeira vez, `useState(0)` atribui à variável `count` seu valor inicial, que é 0 neste caso.
- **Mudanças de estado:** Cada vez que você clica no botão, a função `setCount` atualiza o estado. O React então atualiza o componente com o novo valor.
- **Re-renderização:** sempre que o estado muda, o React re-renderiza automaticamente o componente para mostrar o valor atualizado de `count`.

É assim que seu componente "lembra" coisas como o número de vezes que um botão foi clicado — por meio do estado.

useState

Adendos

- **O estado é lembrado somente dentro do componente:** O estado criado com `useState` existe somente no componente onde você o declara. Cada componente pode ter seu próprio estado que não interfere com outros componentes.
- **O estado aciona a nova renderização:** sempre que você usar a função do `useState` (como `setCount`), o React atualizará o componente com o novo valor e o renderizará novamente na tela.
- **State pode ser qualquer tipo:** O valor em `useState` não precisa ser apenas um número. Você pode usar `useState` com strings, booleanos, objetos, arrays, qualquer coisa que possa mudar ao longo do tempo.

useState

Exemplos

Entrada de formulário de rastreamento

- **name** é o estado que armazena a entrada do usuário.
- **setName** atualiza o estado toda vez que o usuário digita algo novo no campo de entrada.



```
const [name, setName] = useState('');

<input
  type="text"
  value={name}
  onChange={(e) => setName(e.target.value)}
/>
<p>Your name is: {name}</p>
```

useState

Exemplos

Alternando a visibilidade

- **isOpen** é o estado que monitora se o menu está aberto.
- **setIsOpen** alterna o estado entre verdadeiro e falso sempre que o botão é clicado.



```
const [isOpen, setIsOpen] = useState(false);

<button onClick={() => setIsOpen(!isOpen)}>
  {isOpen ? 'Close Menu' : 'Open Menu'}
</button>
{isOpen && <div>This is the menu!</div>}
```

Exercícios

Exercícios

Instruções


- Realizar o exercício da aula 18
- Documentar o que foi feito no README.md
- A documentação deve ter a explicação de cada bloco de código do novo componente criado bem como a explicação do que é o novo componente
- Enviar o exercício roteiro da aula 18 para um repositório no GitHub
- Anexar link do repo na atividade do Teams


*“Ensinar é impregnar
de sentido o que
fazemos a cada
instante”*


Paulo Freire

Obrigado!

Thiago Nogueira

 [linkedin.com/tdn](https://www.linkedin.com/tdn)

 thiago.nogueira@pe.senac.br

 (81) 9 9627-0419