React

Site:Geração TechImpresso por:JOÃO VITOR DE MELO FREITASCurso:Formação em Desenvolvedor Web - OnlineData:quinta-feira, 18 jul. 2024, 22:39

Livro: React

Índice

1. Introdução aos Hooks e useState

- 1.1. Hooks e useState
- 1.2. Vídeo da Aula
- 1.3. Variação do useState e Introdução ao useRef
- 1.4. Vídeo da Aula

1. Introdução aos Hooks e useState

Aula de Hoje: Introdução aos Hooks e useState no React

Bem-vindos a mais uma aula! Hoje vamos aprender sobre Hooks no <u>React</u>, com foco no hook useState. Hooks são funções especiais que permitem usar recursos do <u>React</u>, como estado e ciclo de vida, dentro de componentes funcionais.

O que será visto na aula de hoje:

Introdução aos Hooks

O que são Hooks:

- Hooks são funções que permitem usar os recursos do React, como estado e ciclo de vida, em componentes funcionais.
- Eles foram introducidos para tornar a funcionalidade dos componentes mais robusta e para permitir o uso de estados em componentes funcionais, que antes era possível apenas nos componentes de classe.

Importância dos Hooks para a funcionalidade de componentes funcionais:

- Simplificam a lógica do estado.
- Eliminam a necessidade de componentes de classe.
- Tornam o código mais limpo e fácil de entender.

Utilizando o Hook useState

Criando Estados:

- O hook useState permite declarar variáveis de estado em componentes funcionais.
- A estrutura do useState usa desestruturação de arrays para definir uma variável de estado e uma função para atualizála

Modificando Estados:

- A função modificadora de estado permite atualizar o valor da variável de estado.
- A atualização do estado deve ser feita de forma correta em resposta a eventos, garantindo a reatividade do componente.

Exemplo Prático: Componente de Contador

Vamos criar um componente de contador para demonstrar o uso do useState.

1.1. Hooks e useState

1. Configuração Inicial:

• Abra o seu projeto React e crie um novo arquivo chamado Contador, jex na pasta components.

2. Implementação do Componente Contador:

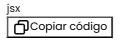
o Adicione o seguinte código ao arquivo Contador.jsx:

```
jsx
Copiar código
```

```
import React, { useState } from 'react'; const Contador = () => { // Declarar uma variável de estado chamada
"contador" e uma função para atualizá-la const [contador, setContador] = useState(0); // Função para incrementar
o contador const incrementar = () => { setContador(contador + 1); }; return ( <div> <h1>Contador: {contador}</h1>
<br/><button onClick={incrementar}>Incrementar</button> </div> ); }; export default Contador;
```

3. Integrar o Componente no App:

- Abra o arquivo App. jsx e importe o componente Contador.
- o Adicione o componente Contador do JSX do App:



```
import React from 'react'; import Contador from './components/Contador'; const App = () => { return ( <div> <Contador /> </div> ); }; export default App;
```

Objetivos da Aula:

1. Compreender o conceito de Hooks no React:

- o Entender a importância dos Hooks para a manipulação de estados e efeitos colaterais.
- o Conhecer a sintaxe e a utilização básica do useState.

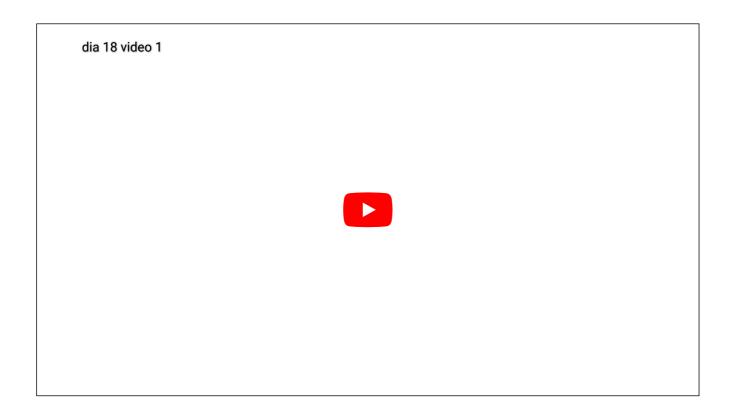
2. Prática com useState:

- o Implementar um estado simples em um componente funcional.
- o Aprender a modificar o estado em resposta a eventos de usuário.
- o Compreender as boas práticas ao trabalhar com estados em componentes React.

Conclusão

Nesta aula, exploramos como utilizar o hook useState para gerenciar estados em componentes funcionais do <u>React</u>, tornandoos mais dinâmicos e interativos. Nos vemos na próxima aula!

1.2. Vídeo da Aula



1.3. Variação do useState e Introdução ao useRef

Revisão do useState e Introdução ao useRef

Olá pessoal, tudo bem com vocês? Hoje, vamos continuar com mais um exemplo prático de uso do useState e apresentar o hook useRef. Vamos criar um componente de input para mostrar como podemos gerenciar estados e referências de elementos DOM em componentes funcionais do React.

O que será visto na aula de hoje:

Utilizando o useState

Criando Estados:

• O hook useState permite declarar variáveis de estado em componentes funcionais.

Modificando Estados:

• A função modificadora de estado permite atualizar o valor da variável de estado em resposta a eventos.

Exemplo Prático: Componente de Input com useState

Vamos criar um componente de input para demonstrar o uso do useState.

1. Criando o Componente Input:

- Na pasta components, crie um arquivo chamado Input.jsx.
- o Adicione o seguinte código ao arquivo Input.jsx:

```
jsx
Copiar código
```

```
import React, { useState } from 'react'; const Input = () => { const [nome, setNome] = useState(''); const
handleChange = (e) => { setNome(e.target.value); }; return ( <div> <input type="text" value={nome} onChange=
{handleChange} placeholder="Digite seu nome" /> <button onClick={() => console.log(nome)}>Mostrar Nome</button>
<Renderizou: {nome}</p> </div> ); }; export default Input;
```

2. Integrar o Componente no App:

- Abra o arquivo App. jsx e importe o componente Input.
- Adicione o componente Input ao JSX do App:

```
jsx
Copiar código
```

```
import React from 'react'; import Input from './components/Input'; const App = () => { return ( <div> <Input /> </div> ); }; export default App;
```

Utilizando o Hook useRef

O useRef é útil para acessar diretamente elementos DOM e não força re-renderizações quando atualizado.

Exemplo Prático: Componente de Input com useRef

Vamos modificar o componente de input para usar o useRef.

1. Modificando o Componente Input para usar useRef:

• No arquivo Input.jsx, modifique o código para usar useRef:



```
import React, { useRef } from 'react'; const Input = () => { const nomeRef = useRef(null); const as-underscore: <--pre & event (/** comment: use--space-trigger (\"true\") --> */ <- (st: str) {
```

console.log(nomeRef.current.value); }; return (<div> <input type="text" ref={nomeRef} placeholder="Digite seu
nome" /> <button onClick={mostrarNome}>Mostrar Nome</button> </div>); }; export default Input;

Explicação dos Conceitos

useState:

- Declaramos uma variável de estado nome e uma função setNome para atualizá-la.
- Usamos useState para inicializar o estado com uma string vazia.

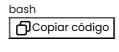
useRef:

- Declaramos uma referência nomeRef e inicializamos com useRef(null).
- Usamos nomeRef para acessar o valor do input diretamente sem forçar uma re-renderização.

Testando a Aplicação

1. Executando a Aplicação:

• No terminal, execute o comando:



npm run dev

• Abra o navegador e acesse http://localhost:3000 para ver a aplicação rodando.

Conclusão

Nesta aula, exploramos como utilizar os hooks useState e useRef para gerenciar estados e referências em componentes funcionais do React, tornando-os mais dinâmicos e interativos. Nos vemos na próxima aula!

1.4. Vídeo da Aula

