Introdução ao React.JS

Othon Oliveira

SENAC - PROA

Sumário

Minha Imagem

Figura: Componentes React

O que são eventos?

Um simples click no botão

Handle event

Você pode responder a eventos declarando funções de manipulador de eventos (handle event) dentro de seus componentes:

O que é um Hook?

Traduzindo ao pé da letra o termo

Gancho

Hook: Um dispositivo curvado, metálico, na forma de um gancho; Hook também pode ser um termo usado nas lutas de boxe, gancho de esquerda (ou direita): a left/rigth hook

O que é um Hook?

Traduzindo ao pé da letra o termo

Gancho

Hook: Um dispositivo curvado, metálico, na forma de um gancho; Hook também pode ser um termo usado nas lutas de boxe, gancho de esquerda (ou direita): a left/rigth hook

Hook em React

Os hooks (ganchos) em React são funções que permitem "conectar-se" ao estado do React e aos recursos do ciclo de vida a partir de componentes.

O que é um Hook?

Traduzindo ao pé da letra o termo

Gancho

Hook: Um dispositivo curvado, metálico, na forma de um gancho; Hook também pode ser um termo usado nas lutas de boxe, gancho de esquerda (ou direita): a left/rigth hook

Hook em React

Os hooks (ganchos) em React são funções que permitem "conectar-se" ao estado do React e aos recursos do ciclo de vida a partir de componentes.

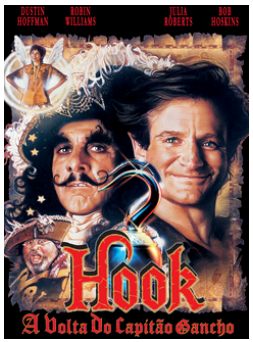

Figura: A volta do Capitão Gancho

Tipos de hooks

Existem dois tipos principais

useState

Este hook (gancho) permite adicionar estado aos seus componentes funcionais. Ele pega um valor de estado inicial e retorna um array com o valor do estado atual e uma função para atualizar o estado.

Tipos de hooks

Existem dois tipos principais

useState

Este hook (gancho) permite adicionar estado aos seus componentes funcionais. Ele pega um valor de estado inicial e retorna um array com o valor do estado atual e uma função para atualizar o estado.

useEffect

Este gancho permite que você execute efeitos "colaterais", como busca de dados, quando seu componente é montado ou atualizado.

Quem são os manipuladores?

["algumaCoisa", "setAlgumaCoisa"]

Na primeira vez que o botão for exibido, a contagem será 0 porque você passou 0 para useState(). Quando você quiser mudar de estado, chame setContar() e passe o novo valor para ele. Clicar neste botão aumentará o contador:

```
1 function MeuBotao() {
2   const [contar, setContar] = useState(0);
3
4   function manipulaClick() {
5     // ...
6   }
7 }
```

Mundando o estado de um componente

Olhando com mais detalhe o useState(0)

Na primeira vez que o botão for exibido, a contagem será 0 porque você passou 0 para useState(). Quando você quiser mudar de estado, chame setContar() e passe o novo valor para ele. Clicar neste botão aumentará o contador:

```
1  function MeuBotao() {
2    const [contar, setContar] = useState(0); // zera o contador
3
4    function manipulaClick() {
5      setContar(contar + 1); //atualiza o contador "contar"
6    }
7
8    return (
9      <button onClick={manipulaClick}> // chama a funcao
10       Clicou {contar} vezes
11       // apos clicar a const contar sera setada (atualizada)
12     </button>
13   );
14 }
```

Classes

O que seria?

```
class Carro {
  constructor(marca, modelo) {
    this.marca = marca;
    this.modelo = modelo;
  }

  getInfo() {
    return `${this.marca} ${this.modelo}`;
  }
}

const meuCarro = new Carro('Toyota', 'Corolla');
console.log(meuCarro.getInfo()); // Saída: Toyota Corolla
```

Classes

O que seria?

```
class CarroEsportivo extends Carro {
  constructor(marca, modelo, velocidadeMax) {
    super(marca, modelo);
    this.velocidadeMax = velocidadeMax;
  }

  getInfo() {
    return `${super.getInfo()} - Velocidade Máxima: ${this.vel
  }
}

const meuCarroEsportivo = new CarroEsportivo('Ferrari', '458 I
console.log(meuCarroEsportivo.getInfo()); // Saída: Ferrari 45
```

Se não existissem os hooks, como seria?

Ahh, “saudade dos hooks” !!

```
1
2  class Treinamento extends React.Component {
3    constructor(props) { // Detalhes no arquivo comandos.txt
4      super(props);
5      this.state = {          // Estado inicial
6        nome: 'Treinamento' // propriedade "nome" inicializada
7      };
8    }
9
10   render() {
11     return (
12       <div>
13         <p>{this.state.nome}</p> {/* Exibe o valor da propried
14         <button onClick={() => this.setState({ nome: 'React' }
15       </div>
16     );
17   }
18 }
```

Explicando a classe "Treinamento"

1. class Treinamento extends React.Component ... : Define uma classe chamada Treinamento que estende a classe React.Component.
2. constructor(props) ... : O método constructor é um método especial chamado quando um objeto é criado com a classe. Ele inicializa o estado do componente. Neste caso, a propriedade nome é inicializada como 'Treinamento'.
3. $< p > this.state.nome < /p >$: Exibe o valor da propriedade nome do estado. No início, isso será 'Treinamento'.
4. $< buttononClick = () => this.setState(nome :'' React'') > CLICK < /button >$: Quando o botão é clicado, ele chama a função setState (método encapsulado), que atualiza a propriedade nome do estado para 'React'. Após clicar no botão, o parágrafo será atualizado para mostrar 'React' em vez de 'Treinamento'.

Usando Hooks ao invés de classes

Seja o seguinte componente hook do exemplo anterior

```
1  import React, { useState } from 'react';
2
3  const Treinamento = () => {
4    const [nome, setNome] = useState('Treinamento'); // Iniciali
5
6    return (
7      <div>
8        <p>{nome}</p> {/* Exibe o valor do estado 'nome' */}
9        <button onClick={() => setNome('React')}>CLICK</button>
10     </div>
11   );
12 };
13
14 export default Treinamento;
```