Javascript

React: Conheça o Poder dos Hooks

Conheça o que os React Hooks podem fazer e como eles vão facilitar o processo de desenvolvimento da sua próxima aplicação.

Akira Hanashiro • mais de 2 anos atrás

Olá Web Developers! Vamos conhecer o poder dos Hooks do React.

O que são React Hooks?

Até a versão 16.7 era necessário criar classes para utilizar todas as funcionalidades do React em um componente, como estado e lifecycle.

Era possível criar componentes a partir de funções, mas esses componentes apenas recebiam propriedades, não possuindo lifecycles, estados e outras funcionalidades do React.

componentes. A própria comunidade está criando `Hooks` muito interessantes que podemos utilizar em nossos projetos.

Hooks

Vamos conhecer os dois principais `Hooks` , `useState` e `useEffect` .

useState

O `useState` nos permite criar estados em um componente criado a partir de uma função, assim como o `state` presente em componentes criados a partir de classes.

Veja o exemplo da seguinte classe:

Copiar

```
class Treinaweb extends React.Component{
    constructor(props){
        super(props);
        this.state = {
            nome: 'TreinaWeb'
        }
    }

    render(){
        return {
            <div>
                <p>{this.state.nome}</p>
```

Copiar

```
function Treinaweb (props){
    const [nome, setNome] = useState('TreinaWeb');

    render(){
        return {
            <div>
                <p>{nome}</p>
                <button onClick={() => setNome('React')} >CLICK</button>
            </div>
        }
    }
}
```

Bem menos código, não é mesmo? Vamos entender o que o `useState()` está fazendo.

O `useState()` cria uma variável que controlará o estado do componente. Se quiser outra variável execute outro `useState()` .

Para este `Hook` nós passamos o valor inicial do estado, em nosso exemplo, a `String` "TreinaWeb". Como retorno temos a variável com o valor do estado, que demos o nome de `nome` , e uma função que serve unicamente para atualizar o valor desta variável, que demos o nome de `setNome` .

Veja que no clique do botão apenas chamamos esta função `setNome()` , passando o novo valor.

Por estar tudo declarado dentro da própria função, não temos mais a necessidade de utilizar o `this` , que pode ser complicado de ser compreendido no JavaScript para iniciantes, causando erros no código.

useEffect

```
function Treinaweb (props){
    const [produto, setProduto] = useState('Leite');

    useEffect(() => {
        Produtos.buscar(produto);
    })

    render(){
        return {
            <div>
                <p>{produto}</p>
                <button onClick={() => setProduto('Chocolate')} >CLICK</button>
            </div>
        }
    }
}
```

O `useEffect()` recebe como primeiro parâmetro uma função que será executada assim que o componente renderizar. Então é um ótimo lugar para fazer requisições.

Dessa maneira como escrevemos, a função passada ao `useEffect()` será executada sempre que o componente for atualizado.

A beleza deste `Hook` é que podemos definir facilmente quando queremos que esta função seja executada novamente. Basta passar como segundo parâmetro ao `useEffect()` um `Array` com as variáveis que controlarão esse `Hook` .

E se quisermos que a função seja executada apenas uma vez, que é quando o componente é inserido na tela, basta passar um `Array` vazio.

Copiar

```
useEffect(() => {
    Produtos.buscar(produto);
}, [])
```

Você pode executar quantos `useEffects()` quiser, o que nos dá mais controle sobre o que e quando algo deve ser executado.

Reutilização de Hooks

Podemos criar nossos próprios `Hooks`. Basta criar uma função que utilize outros `Hooks` e retorne algum valor. Assim poderemos ter uma lógica fora dos componentes e reutilizá-la onde quisermos. Essas funções também devem ter seu nome iniciado com `use`.

O site https://usehooks.com/ possui vários exemplos de `Hooks` interessantes que podem ser muito úteis em seu projeto, como carregar imagens apenas se estiverem visíveis na tela, criar funcionalidade de desfazer, criar atalhos de teclado, etc. Também é uma ótima fonte para entender melhor os `Hooks` na prática, vendo os códigos disponíveis.

Os `Hooks` nos permitem trabalhar de forma limpa e simples no React, nos dando um grande poder. Com ajuda da comunidade estamos vendo várias possibilidades interessantes.

Já criou um `Hook`? Compartilhe aqui com a gente pelos comentários!

Akira Hanashiro

Graduado em Análise e Desenvolvimento de Sistemas, Pós-graduado em Projetos e Desenvolvimento de Aplicações Web e MBA em Machine Learning. Especializado em Front End e curte desenvolvimento de jogos.

Todos os artigos

Artigos relacionados

Ver todos →

arquivos em seus projetos, evitando que você tenha sempre que escrev...

teremos de diferente no JSX a partir do React 17 e como isso irá influenciar suas aplicaç...

Storybook e como ele pode te ajudar a criar componentes da melhor maneira possível, com in...

Ferramentas

VS Code - Melhores extensões para Front-End - Parte 1

Conheça as melhores extensões disponíveis do VS Code para Front-End e melhore a eficiência dos seus...

Arquitetura de Software

Micro Front-End - Microserviços no seu navegador

Entenda o que são Micro Front-ends, como funcionam, suas vantagens e quando você deve escolher esta...

Desenvolvimento

Qual a diferença entre Framework e Biblioteca?

A confusão entre frameworks e bibliotecas é muito comum. Entenda qual a diferença entre eles e qual...

utilizados no mundo. Conheça melhor essa ferramenta e ent...

que é?

Veja qual é o verdadeiro significado de tecnologia.

Saiba tudo o que precisa para comprar um computador para programar.

Javascript

JavaScript - Comandos do Console

Conheça os comandos do console do navegador e domine essa ferramenta para melhorar seus testes ou ca...

Javascript

JavaScript - Entendendo as partes estranhas

Conheça o motivo por trás de vários comportamentos considerados estranhos que o JavaScript possui.

Javascript

JavaScript - Ordenando Elementos por Cores

Aprenda a ordenar cores e fazer conversões entre os formatos RGB, Hexadecimal e HSL usando JavaScrip...

Av. Paulista, 1765, Conj 71 e 72 - Bela Vista - São Paulo - SP - 01311-200