

ReactJS - Eventos e Hooks

O que vamos ver hoje?

- Eventos
- Hooks
- useState
- useEffect

Eventos

Eventos

- É muito semelhante ao tratamento de eventos em elementos DOM.
- Os eventos React são nomeados usando camelCase.
- Com o JSX você passa uma função ao invés de uma string

Eventos

Por exemplo, o evento HTML definimos assim:

```
1 <button onclick="ativarDesconto()">  
2   Ativar Desconto  
3 </button>
```

Será definido da seguinte forma no React:

```
1 <button onclick={ativarDesconto}>  
2   Ativar Desconto  
3 </button>
```

Eventos

```
1 function App() {  
2   const exhibirAviso = () => {  
3     alert("Cuidado!");  
4   };  
5  
6   return (  
7     <button onClick={exibirAviso}>Exibir aviso</button>  
8   );  
9 }  
10  
11 export default App;
```

Hooks

Hooks

- Foram adicionados ao React na versão 16.8.
- Permitem que componentes de função tenham acesso ao estado e outros recursos do React.
- Os hooks mais utilizados são o useState e o useEffect.

useState

- useState aceita um estado inicial e retorna dois valores:
 - O estado atual
 - Uma função que atualiza o estado
- O estado geralmente se refere aos dados ou propriedades da aplicação que precisam ser rastreados

useState

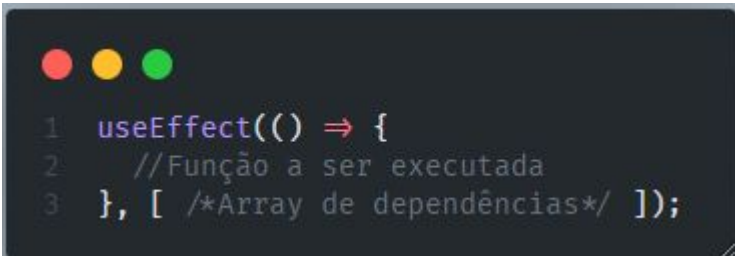
```
1 import React, { useState } from "react";
2
3 function App() {
4   const [color, setColor] = useState("vermelho");
5
6   return (
7     <div>
8       <h1>Minha cor favorita é {color}!</h1>
9       <button type="button"
10         onClick={() => setColor("azul")}> Mudar para azul
11     </button>
12   </div>
13 );
14 }
15 export default App;
```

useEffect

- Permite que você execute efeitos colaterais no seu código.
- Em sua declaração recebe dois parâmetros:
 - Função a ser executada.
 - Array de dependências (pode ser vazio).

useEffect

- Permite que você execute efeitos colaterais no seu código.
- Em sua declaração recebe dois parâmetros:
 - Função a ser executada.
 - Array de dependências (pode ser vazio).



```
1  useEffect(() => {  
2    //Função a ser executada  
3  }, [ /*Array de dependências*/ ]);
```

useEffect

- Caso o array esteja vazio, o useEffect irá disparar somente no carregamento da página.

useEffect + useState

```
1  const [repositoriosList, setRepositoriosList] = useState([]);
2
3  useEffect(() => {
4    async function carregaRepositorios() {
5      const resposta = await fetch(
6        "https://api.github.com/users/rafaelkasper/repos"
7      );
8      const repositorios = await resposta.json();
9      setRepositoriosList(repositorios);
10   }
11   carregaRepositorios();
12 }, []);
```

useEffect + useState + jsx

```
1  const [repositoriosList, setRepositoriosList] = useState([]);
2
3  useEffect(() => {
4    async function carregaRepositorios() {
5      const resposta = await fetch(
6        "https://api.github.com/users/rafaelkasper/repos"
7      );
8      const repositorios = await resposta.json();
9      setRepositoriosList(repositorios);
10   }
11   carregaRepositorios();
12 }, []);
13
14 return (
15   <
16     <ul>
17       {repositoriosList.map((repositorio) => (
18         <li key={repositorio.id}>{repositorio.name}</li>
19       ))}
20     </ul>
21   </
22 );
23 }
```

- 3000TalentosFrontend
- 3000_Talentos
- api_devsnotes
- condicionais_javascript
- crud
- cypress-avancado-udemy
- cypress-basico-udemy
- cypress-intermediario-udemy
- devsnnotes

Resumo

Resumo



- Com `useState` podemos manipular os dados do estado e realizar alterações em tela.
- Por meio do `useEffect` podemos monitorar eventos colaterais e realizar determinadas ações.

Dúvidas?



Programa
3000 TALENTOS TI
Obrigado(a)!