



PUC Minas
DIRETORIA DE
EDUCAÇÃO CONTINUADA

Pós Graduação *Lato Sensu*

Pós Graduação

Desenvolvimento Web Full Stack

Disciplina

Frameworks front end: React (FFR)

Professor

Samuel Martins

samuelmartins.sw@gmail.com

No capítulo anterior...

- Ciclos de vida de um componente;
- Comunicação entre componentes;
- Escrevendo rotas *client-side* com React Router.

No capítulo de hoje...

- React hooks;
- Trabalhando com APIs externas;
- Formulários.

React Hooks

Ciclos de vida de um *class component*

- Lógica espalhada entre os métodos de ciclos de vida;
- Dificuldade em reaproveitar lógica de negócio que utilizam estado;
- Obrigatoriedade em utilizar *class components* para ter acesso aos métodos de ciclo de vida.

Exemplo - botão com loading

```
1 import React, { Component } from 'react';
2
3 export class ButtonLoading extends Component {
4   state = {
5     loading: false
6   }
7
8   setLoadingState = () => {
9     this.setState({
10       loading: true
11     })
12
13     document.title = 'Application is loading'
14   }
15
16   render() {
17     return (
18       <button onClick={setLoadingState}>{this.state.loading ? 'Loading...' : 'Login'}</button>
19     )
20   }
21 }
```

Exemplo - tela de conta do usuário

```
1 import React, { Component } from 'react';
2
3 export class UserScreen extends Component {
4   state = {
5     loading: false,
6     fullName: 'Samuel Martins'
7   }
8
9   setLoadingState = () => {
10     this.setState({
11       loading: true
12     })
13
14     document.title = 'Application is loading'
15   }
16
17   render() {
18     return (
19       <>
20         <h1>{this.state.loading ? 'Screen loading...' : 'User screen'}</h1>
21         <form action="/user/udate" onSubmit={setLoadingState}>
22           <input type="text" value={this.state.fullName} />
23           <button onClick={setLoadingState}>{this.state.loading ? 'Loading...' : 'Update'}</button>
24         </form>
25       </>
26     )
27   }
28 }
```

Class components

- Contextos totalmente diferentes;
- Lógicas idênticas;

React hooks

- Possibilidade de adicionar estados e ciclos de vida em *function components*;
- Melhor reaproveitamento de lógica entre os componentes;
 - Possibilidade de criar um hook customizado com estado embutido.

React Hooks

```
1 export class Products extends Components {
2   state = {
3     items: []
4   }
5
6   handleResizeEvent = () => console.log('resized!');
7
8   async componentWillMount() {
9     const externalItems = await ProductService.fetchItems();
10    this.setState({ items: externalItems })
11  }
12
13  render() {
14    return this.state.items.map(item => <span>{item.title}</span>)
15  }
16 }
```

React Hooks

```
1 export const Products = () => {
2   const [items, setItems] = useState([])
3
4   useEffect(() => {
5     const fetchItems = async () => {
6       const externalItems = await ProductsService.fetchItems();
7       setItems(externalItems)
8     }
9
10    fetchItems()
11  })
12
13  return items.map(item => item => <span>{item.title}</span>)
14 }
```

useState()

- Recebe como parâmetro o valor inicial do estado desejado;
- Retorna um array com duas variáveis: valor do estado e método para alterar o estado definido;

useState()



```
1 export class UserScreen extends Component {  
2     state = {  
3         loading: false,  
4     }  
5 }
```

useState()



```
1 export const UserScreen ( ) => {  
2   const [loading, setLoading] = useState(false)  
3 }
```

useState()



```
1 export const UserScreen ( ) => {  
2   const [loading, setLoading] = useState(false)  
3   const [pageTitle, setPageTitle] = useState('')  
4 }
```

useState()

```
1 export const UserScreen () => {
2   const [loading, setLoading] = useState(false)
3   const [pageTitle, setPageTitle] = useState('')
4
5   return (
6     <h1>{pageTitle}</h1>
7     <button onClick={() => setLoading(true)}>{loading ? 'Loading...' : 'Login'}</button>
8   )
9 }
```



useEffect()

- Utilizado para tratar efeitos colaterais no componente
 - Atualização de propriedades;
 - Atualização de estados;
 - Re-renderização do componente;
- Recebe como parâmetro uma função para tratar os efeitos colaterais e um array com valores a serem “monitorados”;
- Array pode ser passado como vazio para monitorar todas as alterações;



```
1 useEffect(() => {  
2   //componentWillMount, componentDidUpdate  
3   console.log('some side effect was released')  
4 }, [stateValue])
```

```
1 class Exemplo extends React.Component {
2   constructor() {
3     this.state = { count: 0 };
4   }
5
6   componentDidMount() {
7     document.title = `Você clicou ${this.state.count} vezes`;
8   }
9
10  componentDidUpdate() {
11    document.title = `Você clicou ${this.state.count} vezes`;
12  }
13
14  render() {
15    return (
16      <div>
17        <p>Você clicou {this.state.count} vezes</p>
18        <button onClick={() => this.setState({ count: this.state.count + 1 })}>
19          Click me
20        </button>
21      </div>
22    );
23  }
24 }
```



```
1 function Example() {
2   const [count, setCount] = useState(0);
3
4   useEffect(() => {
5     document.title = `Você clicou ${count} vezes`;
6   });
7
8   return (
9     <div>
10      <p>Você clicou {count} vezes</p>
11      <button onClick={() => setCount(count + 1)}>
12        Clique aqui
13      </button>
14    </div>
15  );
16 }
```

useEffect()

- Possibilidade de tratar o ciclo “componentWillUnmount” dentro do useEffect, retornando uma função de execução.



```
1 class FriendStatus extends React.Component {  
2   constructor(props) {  
3     this.state = { isOnline: null };  
4   }  
5  
6   componentDidMount() {  
7     SomeService.subscribeToSomeData();  
8   }  
9  
10  componentWillUnmount() {  
11    SomeService.unsubscribeToSomeData();  
12  }  
13 }
```



```
1 const FriendStatus = () => {  
2   const [isOnline, setIsOnline] = useState(false)  
3   useEffect(() => {  
4     SomeService.subscribeToSomeData();  
5  
6     return () => SomeService.unsubscribeToSomeData();  
7   })  
8 }
```

Reaproveitamento de lógica com hooks customizados

- Hooks customizados possuem states embutidos;
- Esses states “embutidos” podem ser reaproveitados em qualquer componente.

Reaproveitamento de lógica com hooks customizados

```
1 const MyCustomPage = () => {
2   const [width, setWidth] = useState(window.innerWidth);
3   const handleWindowWidth = () => setWidth(window.innerWidth)
4
5   useEffect(() => {
6     window.addEventListener('resize', handleWindowWidth)
7
8     return () => window.removeEventListener('resize', handleWindowWidth)
9   })
10
11   return <h1>Window width is {width}!</h1>
12 };
```

Hook customizado

```
1 import { useEffect, useState } from 'react'
2
3 export function useWindowWidth() {
4     const [width, setWidth] = useState(window.innerWidth);
5     const handleWindowWidth = () => setWidth(window.innerWidth)
6
7     useEffect(() => {
8         window.addEventListener('resize', handleWindowWidth)
9
10        return () => window.removeEventListener('resize', handleWindowWidth)
11    })
12
13    return width
14 }
```

Hook customizado



```
1 const MyCustomPage = () => {  
2   const width = useWindowWidth()  
3   return <h1>Window width is {width}!</h1>  
4 }
```

Integrando com APIs externas

Integrando com APIs externas

- Aplicações desenvolvidas com React “puro” devem ser **SOMENTE** front-end;
 - Lógicas do browser;
 - Interação com usuário;
 - Lógicas de negócio em geral.
- Todos os exemplos que utilizamos foram com dados fixos. Mas e se esses dados vierem de um serviço externo? 🤔

Integração com APIs externas

- No javascript temos o conceito de “Serviço” para fazer requisições em APIs;
- Nada mais é do que uma classe “comum”, com métodos estáticos que podem ser chamados sem uma instância;
- Permite descentralizar lógica de chamada de APIs dos componentes;
- Evite fazer requisições diretas na API no mesmo arquivo do componente. Sempre tenha um serviço para isso.



```
1 import axios from "axios";
2
3 export class MoviesService {
4     static API_KEY = "d416af5d4faee64e25ab001d87aab5c3";
5
6     static _withBaseUrl(path) {
7         return `https://api.themoviedb.org/3/${path}?api_key=${
8             MoviesService.API_KEY
9         }`;
10    }
11
12    static getPopularMovies() {
13        return axios(MoviesService._withBaseUrl("movie/popular"));
14    }
15 }
```



```
1 import React, { useState, useEffect } from "react";
2 import { MoviesService } from "../services/MoviesService";
3
4 export const MoviesList = () => {
5   const [movies, setMovies] = useState({ data: { results: [] } });
6
7   const requestMovies = async () => {
8     const moviesResult = await MoviesService.getPopularMovies();
9     setMovies(moviesResult);
10  };
11
12  useEffect(() => {
13    requestMovies();
14  }, []);
15
16  return (
17    <ul>
18      {movies.data.results.map(movie => (
19        <li>{movie.title}</li>
20      ))}
21    </ul>
22  );
23 };
```