

# FRONT-END DESIGN ENGINEERING

[React – Hooks]

[Aula 14]  
[Atualizado em: 04/03/2024]

Material cedido gentilmente pelo professor Alexandre Carlos ([profalexandre.jesus@fiap.com.br](mailto:profalexandre.jesus@fiap.com.br)) e professor Luís Carlos ([lsilva@fiap.com.br](mailto:lsilva@fiap.com.br)) e atualizado pelo professor Adriano Milanez ([profadriano.milanez@fiap.com.br](mailto:profadriano.milanez@fiap.com.br))



## REACT – HOOKS

REACT – HOOKS

# REACT – HOOKS

## ✓ **Introdução aos Hooks no React com TypeScript**

- ✓ Os Hooks no React representam uma revolução na forma como os desenvolvedores gerenciam o estado e o ciclo de vida em componentes funcionais. Com a adição dos Hooks, é possível extrair lógica de componentes, reutilizar código de forma mais eficiente e simplificar o gerenciamento de estados.

## ✓ **O que são Hooks?**

- ✓ Hooks são funções especiais fornecidas pelo React que permitem adicionar funcionalidades de componente em componentes funcionais. Eles permitem que você use estado e outras características do React em componentes que são funcionalmente equivalentes aos componentes de classe.

# REACT – HOOKS

## ✓ **Por que usar Hooks?**

- ✓ Antes dos Hooks, os componentes funcionais não podiam conter estado local, o que os limitava em termos de funcionalidades. Agora, com Hooks, os componentes funcionais podem ter estados locais, efeitos colaterais e lógica de ciclo de vida, tornando-os tão poderosos quanto os componentes de classe.

# REACT – HOOKS

## ✓ Principais Hooks

- ✓ useState
- ✓ useEffect
- ✓ useContext



## REACT – HOOKS – USEEFFECT

REACT – HOOKS – USEEFFECT

# REACT – HOOKS

## ✓ HOOKS – **useEffect**

- ✓ O **useEffect** é um gancho (hook) em React que permite que você realize efeitos colaterais em componentes funcionais. Você deve utilizar **useEffect** quando precisar realizar operações assíncronas, como chamadas de API, manipulação do DOM, gerenciamento de assinaturas de eventos ou qualquer operação que não seja parte do fluxo de renderização principal. Aqui estão algumas situações em que **useEffect** é útil:

### ✓ Chamadas de API e Requisições Assíncronas:

- ✓ Quando você precisa buscar dados de uma API ou realizar operações assíncronas após a renderização inicial do componente.



# REACT – HOOKS

## ✓ HOOKS – useState

```
import React, { useEffect, useState } from 'react';

const ExampleComponent: React.FC = () => {

  const [data, setData] = useState([]);

  useEffect(() => {

    const fetchData = async () => {

      const response = await fetch('https://api.example.com/data');

      const result = await response.json();

      setData(result);

    };

    fetchData();

  }, []); // O segundo argumento [] significa que este efeito só é executado uma vez após a
montagem do componente.

  return (

    <div>

      {/* Renderiza o conteúdo usando os dados buscados */}

    </div>

  );

};
```

# REACT – HOOKS

- ✓ **HOOKS – useEffect**

- ✓ **Criar e Remover Eventos:**

- ✓ Quando você precisa lidar com eventos do DOM, como adicionar ou remover event listeners.

# REACT – HOOKS

## ✓ HOOKS – useEffect

```
import React, { useEffect } from 'react';

const ExampleComponent: React.FC = () => {

  useEffect(() => {

    const handleClick = () => {

      console.log('Clicou no documento');

    };

    document.addEventListener('click', handleClick);

    return () => {

      document.removeEventListener('click', handleClick);

    };

  }, []); // O segundo argumento [] garante que este efeito é limpo quando o componente é
desmontado.

  return (

    <div>

      {/* Conteúdo do componente */}

    </div>

  );

};
```

# REACT – HOOKS

- ✓ **HOOKS – useEffect**

- ✓ **Atualização de Estado Condicionalmente:**

- ✓ Quando você precisa realizar efeitos somente quando determinadas propriedades ou estado são modificados.

# REACT – HOOKS

## ✓ HOOKS – useEffect

```
import React, { useEffect, useState } from 'react';

const ExampleComponent: React.FC = ({ someProp }) => {

  const [data, setData] = useState([]);

  useEffect(() => {

    // Realiza o efeito somente quando someProp muda

    fetchData(someProp);

  }, [someProp]);

  const fetchData = async (value: string) => {

    const response = await fetch(`https://api.example.com/data/${value}`);

    const result = await response.json();

    setData(result);

  };

  return (

    <div>

      {/* Renderiza o conteúdo usando os dados buscados */}

    </div>

  );

};
```

# REACT – HOOKS

- ✓ **HOOKS – useEffect**

- ✓ **Limpeza de recursos:**

- ✓ Quando você precisa realizar ações de limpeza ou desinscrição ao desmontar o componente.

# REACT – HOOKS

## ✓ HOOKS – useEffect

```
import React, { useEffect } from 'react';

const ExampleComponent: React.FC = () => {

  useEffect(() => {

    // Inicialização do recurso

    return () => {

      // Limpeza ou desinscrição do recurso ao desmontar o componente

    };

  }, []);

  return (

    <div>

      {/* Conteúdo do componente */}

    </div>

  );

};
```

# REACT – HOOKS

## ✓ HOOKS – useEffect

- ✓ Em resumo, você deve utilizar useEffect sempre que precisar realizar operações assíncronas, lidar com efeitos colaterais, ou manipular o ciclo de vida do componente de forma reativa. Isso ajuda a garantir que suas operações ocorram no momento apropriado durante o ciclo de vida do componente funcional em React.



# REACT – HOOKS

## ✓ HOOKS – useEffect

### ✓ Quando você usa useEffect?

- ✓ Você o usa quando precisa executar algum código em momentos específicos do ciclo de vida do seu componente. Pode ser quando o componente é exibido pela primeira vez, sempre que algo no seu componente muda ou quando o componente está prestes a ser removido.

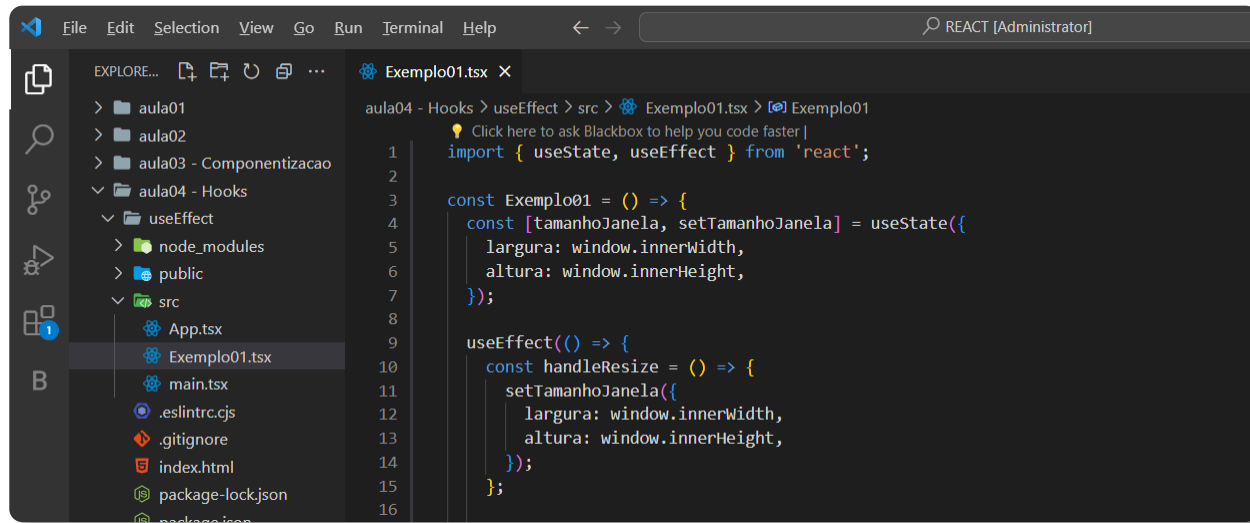
### ✓ Como usar useEffect?

- ✓ Você o coloca dentro do seu componente e passa uma função para ele. Essa função será executada em momentos específicos, dependendo do que você especificar.

# REACT – HOOKS

## ✓ HOOKS – useEffect

### ✓ Exemplo 01 - Código

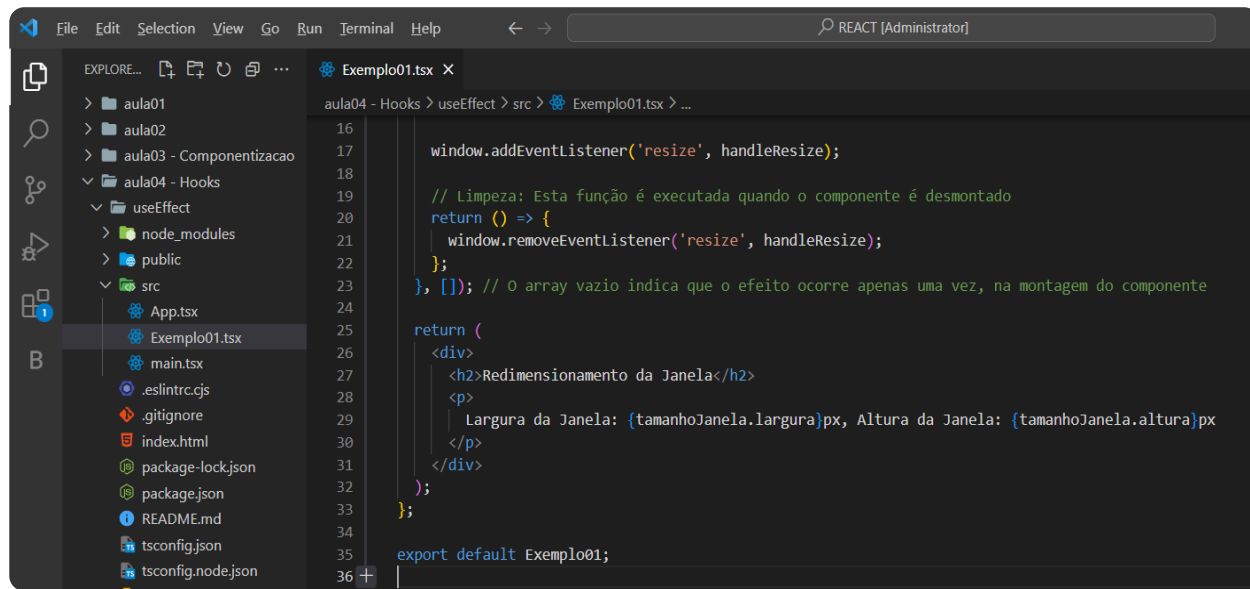


```
File Edit Selection View Go Run Terminal Help
EXPLORER...
  > aula01
  > aula02
  > aula03 - Componentizacao
  > aula04 - Hooks
    > useEffect
      > node_modules
      > public
      > src
        App.tsx
        Exemplo01.tsx
        main.tsx
        .eslintrc.cjs
        .gitignore
        index.html
        package-lock.json
        package.json
Exemplo01.tsx X
aula04 - Hooks > useEffect > src > Exemplo01.tsx > Exemplo01
  Click here to ask Blackbox to help you code faster |
1 import { useState, useEffect } from 'react';
2
3
4 const Exemplo01 = () => {
5   const [tamanhoJanela, setTamanhoJanela] = useState({
6     largura: window.innerWidth,
7     altura: window.innerHeight,
8   });
9
10  useEffect(() => {
11    const handleResize = () => {
12      setTamanhoJanela({
13        largura: window.innerWidth,
14        altura: window.innerHeight,
15      });
16    };
17  });
18 }
```

# REACT – HOOKS

## ✓ HOOKS – useEffect

### ✓ Exemplo 01 - Código



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with a folder named 'aula04 - Hooks' containing a file 'Exemplo01.tsx'. The code editor shows the content of 'Exemplo01.tsx' with the following code:

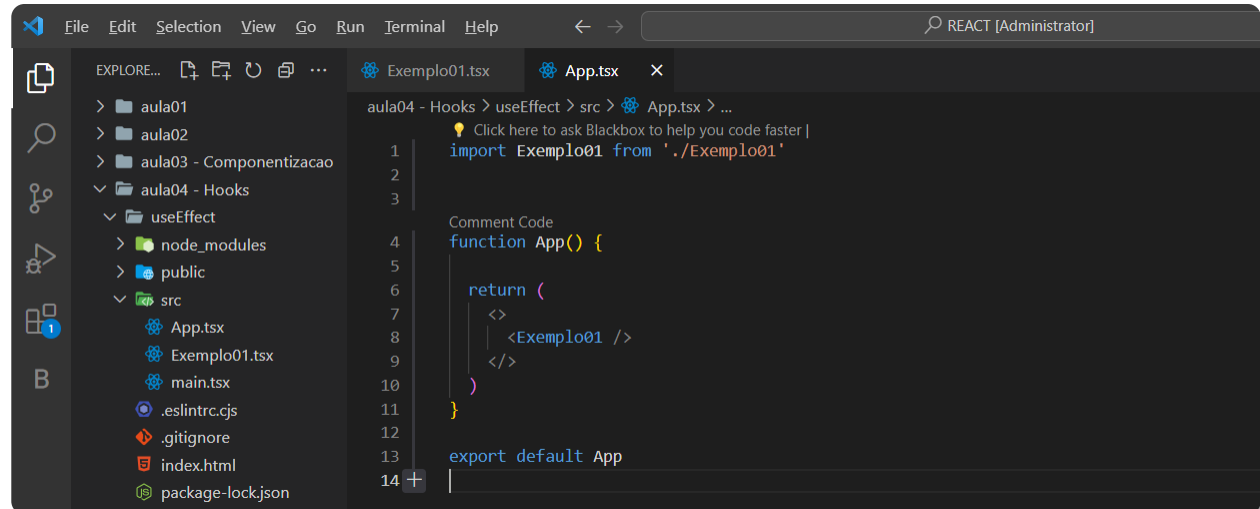
```
16
17 window.addEventListener('resize', handleResize);
18
19 // Limpeza: Esta função é executada quando o componente é desmontado
20 return () => {
21   window.removeEventListener('resize', handleResize);
22 };
23 }, []); // O array vazio indica que o efeito ocorre apenas uma vez, na montagem do componente
24
25 return (
26   <div>
27     <h2>Redimensionamento da Janela</h2>
28     <p>
29       Largura da Janela: {tamanhoJanela.largura}px, Altura da Janela: {tamanhoJanela.altura}px
30     </p>
31   </div>
32 );
33 };
34
35 export default Exemplo01;
36
```

# REACT – HOOKS

## ✓ HOOKS – useEffect

### ✓ Exemplo 01

- ✓ Vamos criar o componente Exemplo01.tsx
- ✓ Agora precisamos importá-lo no App.tsx



The screenshot shows a code editor with the following structure:

- EXPLORER:** A file tree on the left showing a project structure with folders 'aula01', 'aula02', 'aula03 - Componentizacao', and 'aula04 - Hooks'. Under 'aula04 - Hooks', there is a folder 'useEffect' containing 'node\_modules', 'public', and 'src'. The 'src' folder contains 'App.tsx', 'Exemplo01.tsx', and 'main.tsx'. Other files in the project include '.eslintrc.cjs', '.gitignore', 'index.html', and 'package-lock.json'.
- App.tsx:** The active file in the editor. It contains the following code:

```
1 import Exemplo01 from './Exemplo01'
2
3
4 function App() {
5
6   return (
7     <>
8     <Exemplo01 />
9     </>
10  )
11 }
12
13 export default App
14
```

# REACT – HOOKS

## ✓ HOOKS – useEffect

### ✓ Exemplo 01 - Resultado

Estado inicial da aplicação



Após a execução



# REACT – HOOKS

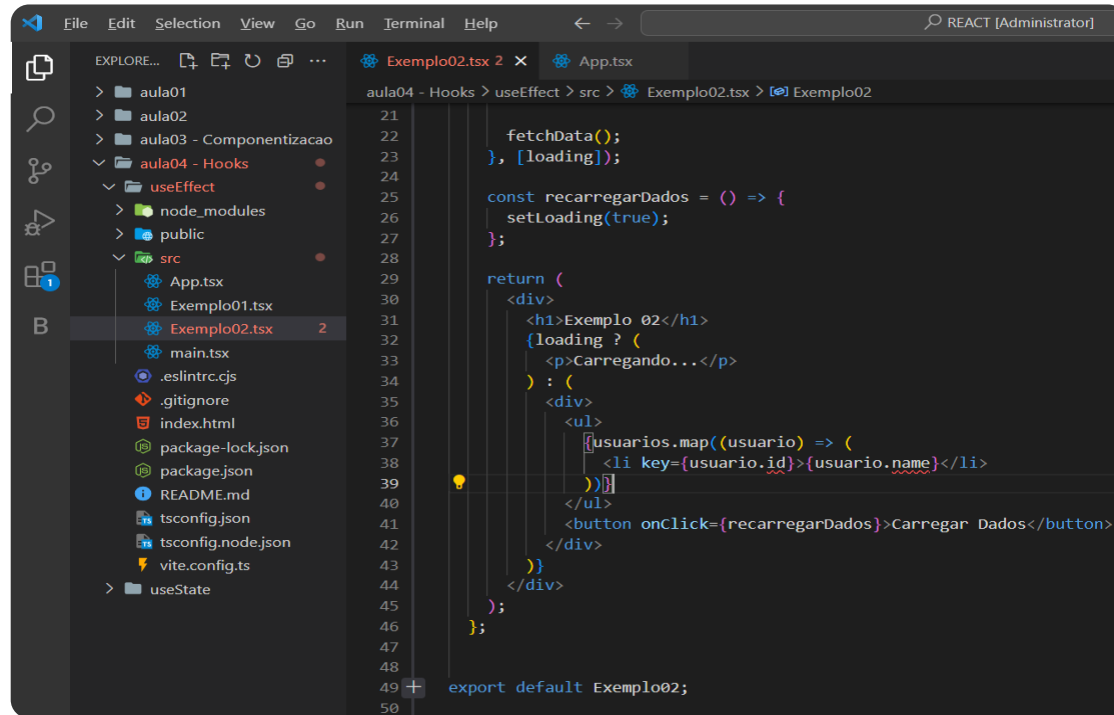
- ✓ **HOOKS – useEffect**
- ✓ Exemplo 02 – Código
- ✓ Vamos criar o componente Exemplo02.tsx

```
File Edit Selection View Go Run Terminal Help
EXPLORE...
  > aula01
  > aula02
  > aula03 - Componentizacao
  > aula04 - Hooks
    > useEffect
      > node_modules
      > public
      > src
        App.tsx
        Exemplo01.tsx
        Exemplo02.tsx 2
        main.tsx
        .eslintrc.cjs
        .gitignore
        index.html
        package-lock.json
        package.json
        README.md
        tsconfig.json
        tsconfig.node.json
Exemplo02.tsx 2 x
aula04 - Hooks > useEffect > src > Exemplo02.tsx > Exemplo02
  Click here to ask Blackbox to help you code faster |
1  import { useState, useEffect } from 'react';
2
3
4  const Exemplo02 = () => {
5    const [usuarios, setUsuarios] = useState([]);
6    const [loading, setLoading] = useState(false);
7
8    useEffect(() => {
9      const fetchData = async () => {
10        if (loading) {
11          try {
12            const response = await fetch('https://jsonplaceholder.typicode.com/users');
13            const data = await response.json();
14            setUsuarios(data);
15          } catch (error) {
16            console.error('Erro ao buscar dados da API:', error);
17          } finally {
18            setLoading(false);
19          }
20        }
21      };
22    });
23  };
24  };
```

# REACT – HOOKS

## ✓ HOOKS – useEffect

### ✓ Exemplo 02 – Código

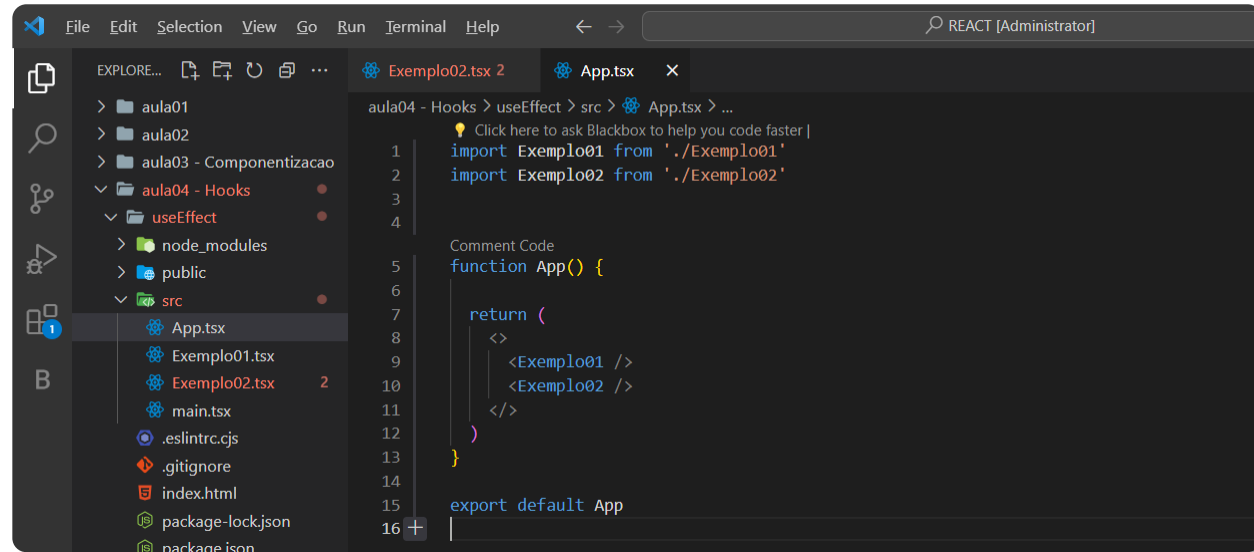


The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with folders 'aula01', 'aula02', 'aula03 - Componentizacao', and 'aula04 - Hooks'. Inside 'aula04 - Hooks', there is a folder 'useEffect' and a file 'Exemplo02.tsx' which is selected. The code editor shows the following code:

```
21      fetchData();
22    }, [loading]);
23  }
24
25  const recarregarDados = () => {
26    setLoading(true);
27  };
28
29  return (
30    <div>
31      <h1>Exemplo 02</h1>
32      {loading ? (
33        <p>Carregando...</p>
34      ) : (
35        <div>
36          <ul>
37            {usuarios.map((usuario) => (
38              <li key={usuario.id}>{usuario.name}</li>
39            ))}
40          </ul>
41          <button onClick={recarregarDados}>Carregar Dados</button>
42        </div>
43      )}
44    </div>
45  );
46
47  };
48
49  export default Exemplo02;
```

# REACT – HOOKS

- ✓ **HOOKS – useEffect**
- ✓ Exemplo 02
  - ✓ Vamos criar o componente Exemplo02.tsx
  - ✓ Agora precisamos importá-lo no App.tsx



```
File Edit Selection View Go Run Terminal Help
EXPLORE...
  > aula01
  > aula02
  > aula03 - Componentizacao
  > aula04 - Hooks
    > useEffect
    > node_modules
    > public
    > src
      App.tsx
      Exemplo01.tsx
      Exemplo02.tsx
      main.tsx
      .eslintrc.cjs
      .gitignore
      index.html
      package-lock.json
      package.json

aula04 - Hooks > useEffect > src > App.tsx > ...
  Click here to ask Blackbox to help you code faster |
  1 import Exemplo01 from './Exemplo01'
  2 import Exemplo02 from './Exemplo02'
  3
  4
  Comment Code
  5 function App() {
  6
  7   return (
  8     <>
  9     <Exemplo01 />
 10     <Exemplo02 />
 11     </>
 12   )
 13 }
 14
 15 export default App
 16 +
```

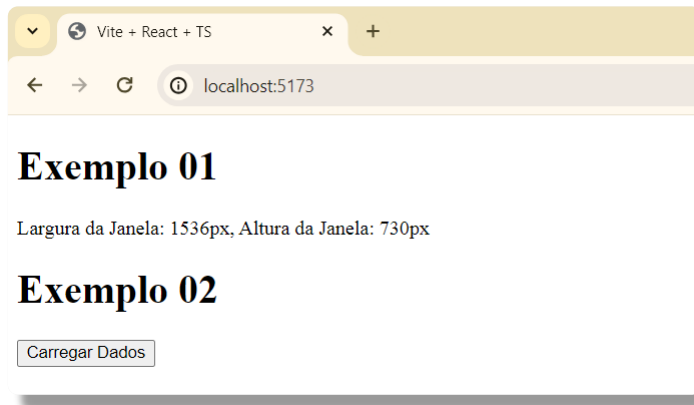


# REACT – HOOKS

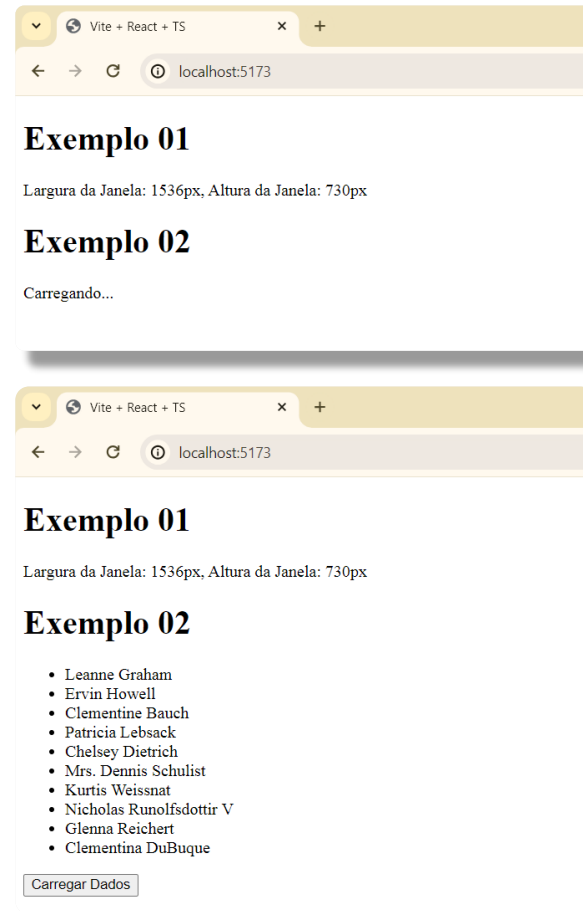
✓ **HOOKS – useEffect**

✓ Exemplo 02 - Resultado

Estado inicial da aplicação



Após a execução



# REACT – HOOKS

## ✓ HOOKS – useEffect

### ✓ Exercício 01

- ✓ Crie um componente React de contador simples. Utilize o estado (useState) para controlar o valor do contador, e o efeito (useEffect) para exibir uma mensagem no console sempre que o contador for incrementado. Adicione um botão que permite ao usuário incrementar o contador.
- ✓ Dica:
  - ✓ Utilize o useState para controlar o estado do contador.
  - ✓ Utilize o useEffect para observar mudanças no contador e exibir uma mensagem no console.
  - ✓ Adicione um botão que, quando clicado, incrementa o contador.

# REACT – HOOKS

## ✓ HOOKS – useEffect

### ✓ Exercício 02

- ✓ Crie um componente de relógio digital que exibe a hora atual. Utilize o estado (useState) para armazenar a hora e o efeito (useEffect) para atualizar a hora a cada segundo.
- ✓ Adicione um botão que permite ao usuário pausar e retomar o relógio.
- ✓ Dica:
  - ✓ Utilize o useState para controlar o estado da hora.
  - ✓ Utilize o useEffect para atualizar a hora a cada segundo.

# FRONT-END DESIGN ENGINEERING

ANTONIO, C. Pro React: Build Complex Front-End Applications in a Composable Way With React. Apress, 2015.

BOSWELL, D; FOUCHER, T. The Art of Readable Code: Simple and Practical Techniques for Writing Better Code. Estados Unidos: O'Reilly Media, 2012.

BRITO, Robin Cris. Android Com Android Studio - Passo A Passo. Editora Ciência Moderna.

BUNA, S. React Succinctly. Estados Unidos: [s.n], 2016. Disponível em: <[www.syncfusion.com/ebooks/reactjs\\_succinctly](http://www.syncfusion.com/ebooks/reactjs_succinctly)>. Acesso em: 12 de janeiro de 2023.

FACEBOOK (2019a). React: Getting Started. React Docs, 2019. Disponível em: <[reactjs.org/docs/react-api.html](https://reactjs.org/docs/react-api.html)>. Acesso em: 13 de janeiro de 2023.

FACEBOOK (2019b). React Without ES6. React Docs, 2019. Disponível em: <[reactjs.org/docs/react-without-es6.html](https://reactjs.org/docs/react-without-es6.html)>. Acesso em: 10 de janeiro de 2023.

FACEBOOK (2019c). React Without JSX. React Docs, 2019. Disponível em: <[reactjs.org/docs/react-without-jsx.html](https://reactjs.org/docs/react-without-jsx.html)>. Acesso em: 10 de janeiro de 2023.

FREEMAN, Eric ROBSON, Elisabeth. Use a Cabeça! Programação em HTML5. Rio de Janeiro: Editora Alta Books, 2014

GACKENHEIMER, C. Introduction to React: Using React to Build scalable and efficient user interfaces.[s.i.]: Apress, 2015.

HUDSON, P. Hacking with React. 2016. Disponível em: <[www.hackingwithreact.com/read/1/3/introduction-to-jsx](http://www.hackingwithreact.com/read/1/3/introduction-to-jsx)>. Acesso em: 13 janeiro de 2023.

KOSTRZEWA, D. Is React.js the Best JavaScript Framework in 2018? 2018. Disponível em: <[hackernoon.com/is-react-js-the-best-javascript-framework-in-2018-264a0eb373c8](https://hackernoon.com/is-react-js-the-best-javascript-framework-in-2018-264a0eb373c8)>. Acesso em: janeiro de 2023.

MARTIN, R. Clean Code: A Handbook of Agile Software Craftsmanship. Estados Unidos: Prentice Hall, 2009.

MDN WEB DOCS. Guia JavaScript. Disponível em <<https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide>>. Acessado em 29 de janeiro de 2023.

NELSON, J. Learn React's Fundamentals Without the Buzzwords? 2018. Disponível em: <[jamesknelson.com/learn-react-fundamentals-sans-buzzwords](https://jamesknelson.com/learn-react-fundamentals-sans-buzzwords)>. Acesso em: 12 janeiro de 2023.

NIELSEN, J. Response Times: The 3 Important Limits. 1993. Disponível em: <[www.nngroup.com/articles/response-times-3-important-limits](http://www.nngroup.com/articles/response-times-3-important-limits)>. Acesso em: 10 janeiro de 2023.

# FRONT-END DESIGN ENGINEERING

O'REILLY, T. What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software. 2005. Disponível em: <[www.oreilly.com/pub/a/web2/archive/what-is-web-20.html#mememap](http://www.oreilly.com/pub/a/web2/archive/what-is-web-20.html#mememap)>. Acesso em: 10 de janeiro de 2023.

PANDIT, N. What Is ReactJS and Why Should We Use It? 2018. Disponível em: <[www.c-sharpcorner.com/article/what-and-why-reactjs](http://www.c-sharpcorner.com/article/what-and-why-reactjs)>. Acesso em: 12 de janeiro de 2023.

RAUSCHMAYER, A. Speaking JavaScript: An In-Depth Guide for Programmers. Estados Unidos: O'Reilly Media, 2014.

REACTIVA. O arquivo package-lock.json. Disponível em: <<https://nodejs.reativa.dev/0020-package-lock-json/index>>. Acessado em 13 de janeiro de 2023.

\_\_\_\_\_. O guia do package.json. Disponível em: <<https://nodejs.reativa.dev/0019-package-json/index>>. Acessado em 13 de janeiro de 2023.

RICOY, L. Desmitificando React: Uma Reflexão para Iniciantes. 2018. Disponível em: <[medium.com/trainingcenter/desmitificando-react-uma-reflex%C3%A3o-para-iniciantes-a57af90b6114](https://medium.com/trainingcenter/desmitificando-react-uma-reflex%C3%A3o-para-iniciantes-a57af90b6114)>. Acesso em: 13 janeiro de 2023.

SILVA, Maurício Samy. Ajax com jQuery: requisições Ajax com a simplicidade de jQuery. São Paulo: Novatec Editora, 2009.

\_\_\_\_\_. Construindo Sites com CSS e XHTML. Sites Controlados por Folhas de Estilo em Cascata. São Paulo: Novatec, 2010.

\_\_\_\_\_. CSS3 - Desenvolva aplicações web profissionais com o uso dos poderosos recursos de estilização das CSS. São Paulo: Novatec Editora, 2010.

STACKOVERFLOW. Most Popular Technologies: Web Frameworks. Developer Survey Results, StackOverflow, 2019. Disponível em: <[insights.stackoverflow.com/survey/2019#technology](https://insights.stackoverflow.com/survey/2019#technology)>. Acesso em: 13 de janeiro de 2023.

W3C. HTML5 - A linguagem de marcação que revolucionou a web. São Paulo: Novatec Editora, 2010.

\_\_\_\_\_. A vocabulary and associated APIs for HTML and XHTML. Disponível em <<https://www.w3.org/TR/2018/SPSD-html5-20180327/>>. Acessado em 28 de abril de 2020, às 20h53min.

\_\_\_\_\_. Cascading Style Sheets, level 1. Disponível em <<https://www.w3.org/TR/2018/SPSD-CSS1-20180913/>>. Acessado em 28 de abril de 2020, às 21h58min.

\_\_\_\_\_. Cascading Style Sheets, level 2 Revision 2. Disponível em <<https://www.w3.org/TR/2016/WD-CSS22-20160412/>>. Acessado em 28 de abril de 2020, às 22h17min.

# FRONT-END DESIGN ENGINEERING

W3C. Cascading Style Sheets, level 2. Disponível em <<https://www.w3.org/TR/2008/REC-CSS2-20080411/>>. Acessado em 28 de abril de 2020, às 22h03min.

\_\_\_\_\_. Cascading Style Sheets, level 3. Disponível em <<https://www.w3.org/TR/css-syntax-3/>>. Acessado em 28 de abril de 2020, às 22h18min.

\_\_\_\_\_. HTML 3.2 Reference Specification. Disponível em <<https://www.w3.org/TR/2018/SPSD-html32-20180315/>>. Acessado em 28 de abril de 2020, às 19h37min.

\_\_\_\_\_. HTML 4.0 Specification. Disponível em <<https://www.w3.org/TR/1998/REC-html40-19980424/>>. Acessado em 28 de abril de 2020, às 19h53min.

\_\_\_\_\_. HTML 4.01 Specification. Disponível em <<https://www.w3.org/TR/2018/SPSD-html401-20180327/>>. Acessado em 28 de abril de 2020, às 20h04min.

\_\_\_\_\_. Cascading Style Sheets, level 2 Revision 1. Disponível em <<https://www.w3.org/TR/CSS2/>>. Acessado em 28 de abril de 2020, às 22h13min.

WIKIPEDIA. JavaScript. Disponível em <<https://pt.wikipedia.org/wiki/JavaScript>>. Acessado em 29 de abril de 2020, às 10h.