

# FRONT-END DESIGN ENGINEERING

[Aula 14]  
[Atualizado em: 11/03/2024]

[React – Props]

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 – PROPS

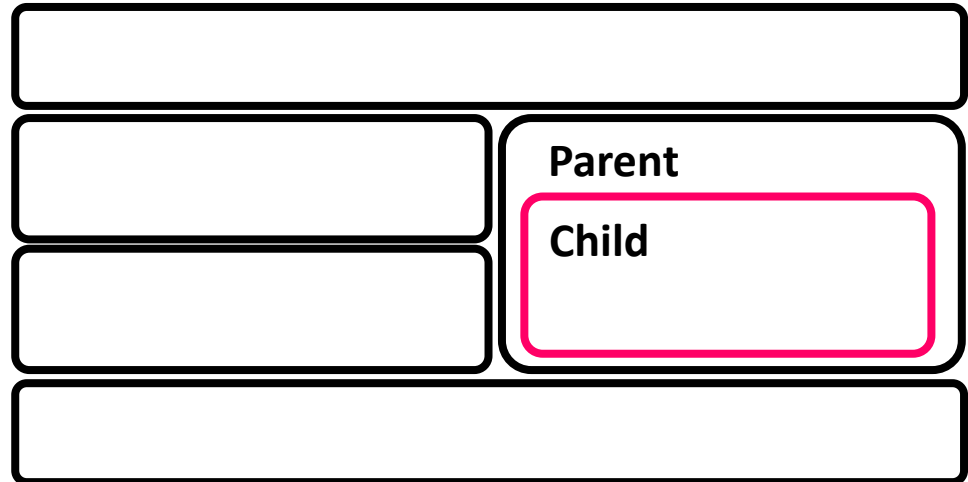
# REACT – PROPS

## ✓ Introdução as Props

- ✓ Props são mecanismos que permitem transmitir dados de um componente pai para um componente filho. São uma forma resumida de dizer **PRO**PERTIES.
- ✓ As Props são o equivalente a argumentos de funções, aceitando variados tipos de dados e funções. O uso de props é uma ótima maneira de tornar seus componentes mais flexíveis e reutilizáveis.

# REACT – PROPS

- ✓ **Introdução as Props**
- ✓ O fluxo de dados do React entre componentes é unidirecional (somente do componente pai para o componente filho)



# REACT – PROPS

## ✓ Vantagens na utilização de Props

### ✓ Reusabilidade de Componentes

- ✓ Props permitem que você crie componentes flexíveis e reutilizáveis. Ao passar diferentes propriedades para o mesmo componente, você pode reutilizá-lo em várias partes do seu aplicativo.

### ✓ Comunicação entre Componentes

- ✓ Props são a principal maneira de passar dados de um componente pai para um componente filho. Isso facilita a comunicação e o compartilhamento de informações entre diferentes partes do seu aplicativo.

### ✓ Configurabilidade

- ✓ Props permitem que os componentes sejam configurados de maneira dinâmica. Você pode ajustar o comportamento de um componente ao passar diferentes valores de propriedades.

# REACT – PROPS

## ✓ Vantagens na utilização de Props

### ✓ Manutenção Simplificada

- ✓ O uso de props torna o código mais modular e fácil de entender. Cada componente é responsável por uma parte específica da lógica ou da interface, o que facilita a manutenção e a solução de problemas.

### ✓ Tipagem e Validação

- ✓ Com TypeScript ou PropTypes, você pode fornecer tipos ou validar as props, garantindo uma maior segurança e prevenindo erros relacionados a tipos de dados incorretos.

### ✓ Consistência de Interface

- ✓ Ao padronizar a passagem de dados por meio de props, você cria uma interface consistente em seu aplicativo, facilitando a compreensão do fluxo de dados.

# REACT – PROPS

## ✓ Vantagens na utilização de Props

### ✓ Testabilidade

- ✓ Ao depender de props, os componentes se tornam mais fáceis de serem testados. Você pode fornecer diferentes props durante os testes para garantir que o componente se comporte conforme o esperado em várias situações.

### ✓ Desacoplamento

- ✓ O uso de props ajuda a manter um alto grau de desacoplamento entre os componentes, o que significa que as mudanças em um componente têm menos impacto em outros componentes.

### ✓ Legibilidade e Manutenibilidade:

- ✓ O código que utiliza props geralmente é mais legível, pois a fonte de dados e as configurações são passadas explicitamente. Isso facilita a compreensão do que um componente faz e como ele se comporta.



# REACT – PROPS

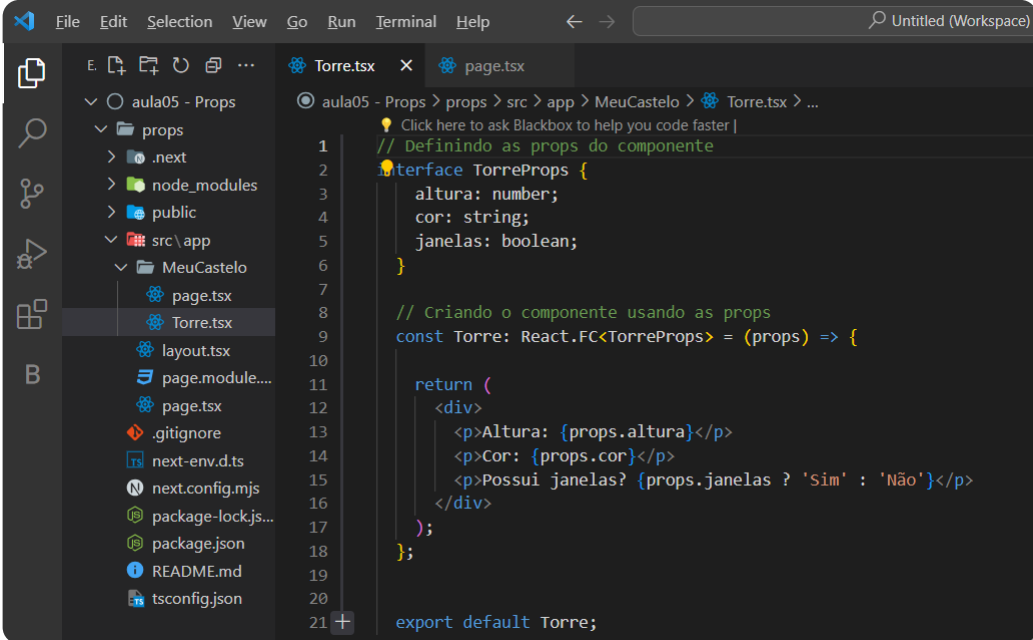
## ✓ Introdução as Props

- ✓ Então, imagina que você está brincando com peças de LEGO para construir um castelo. Cada peça de LEGO é como uma parte do castelo, e elas têm informações importantes, como a cor, o tamanho e a forma.
- ✓ No React, quando você cria componentes (como peças de LEGO), você pode dar a eles informações especiais chamadas "props". Essas props são como as informações que você dá às suas peças de LEGO. Agora, vamos adicionar um toque de TypeScript (TS) para tornar as coisas um pouco mais organizadas. TypeScript é como um assistente que ajuda a garantir que você está usando as peças de LEGO corretamente.
- ✓ Então, ao usar TypeScript com React, você diz exatamente quais informações suas peças de LEGO (ou componentes) podem ter. Por exemplo, se você tem uma peça de LEGO chamada "Torre", você pode dizer que ela tem props como "altura", "cor" e "janelas".

# REACT – PROPS

## ✓ Introdução as Props

- ✓ Exemplo 01 – Crie uma nova aplicação e vamos modificar o código gerado, vamos criar uma pasta chamada **MeuCastelo**, dentro de SRC > APP e na sequência criar um arquivo chamado **Torre.tsx**.



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows the following structure:

- aula05 - Props
  - props
    - .next
    - node\_modules
    - public
    - src\app
      - MeuCastelo
        - page.tsx
        - Torre.tsx
      - layout.tsx
      - page.module....
      - page.tsx
      - .gitignore
      - next-env.d.ts
      - next.config.mjs
      - package-lock.js...
      - package.json
      - README.md
      - tsconfig.json

The code editor shows the following code for **Torre.tsx**:

```
1 // Definindo as props do componente
2
3 interface TorreProps {
4   altura: number;
5   cor: string;
6   janelas: boolean;
7 }
8
9 // Criando o componente usando as props
10 const Torre: React.FC<TorreProps> = (props) => {
11
12   return (
13     <div>
14       <p>Altura: {props.altura}</p>
15       <p>Cor: {props.cor}</p>
16       <p>Possui janelas? {props.janelas ? 'Sim' : 'Não'}</p>
17     </div>
18   );
19 }
20
21 export default Torre;
```

# REACT – PROPS

## ✓ Introdução as Props

### ✓ Exemplo 01 – Analisando linha a linha o arquivo Torre.tsx

```
1  // Definindo as props do componente
2  interface TorreProps {
3      altura: number;
4      cor: string;
5      janelas: boolean;
6  }
7  const Torre: React.FC<TorreProps> = (props) => {    // Criando o componente usando as props
8      return (
9          <div>
10             <p>Altura: {props.altura}</p>
11             <p>Cor: {props.cor}</p>
12             <p>Possui janelas? {props.janelas ? 'Sim' : 'Não'}</p>
13         </div>
14     );
15 };
16 export default Torre;
```

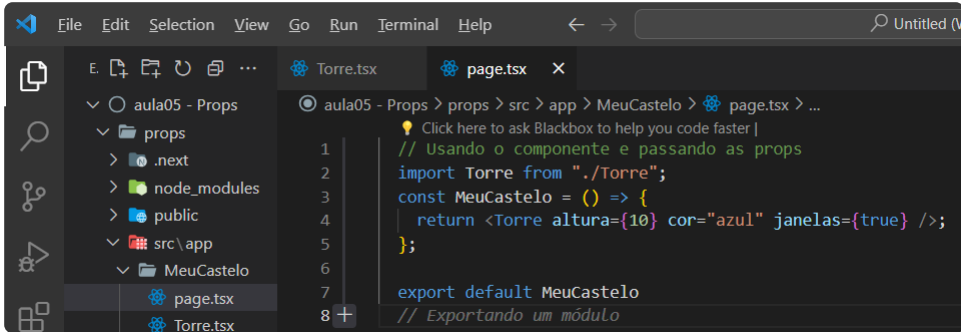
// estamos definindo uma interface chamada TorreProps. Esta interface descreve as propriedades (props) que o componente Torre aceitará. O componente espera receber três props: altura (um número), cor (uma string) e janelas (um valor booleano).

// Recebemos os valores das props altura, cor e janelas e as exibimos em parágrafos.

# REACT – PROPS

## ✓ Introdução as Props

- ✓ Exemplo 01 – Criamos agora um componente chamado **MeuCastelo**, porém, o arquivo o chamaremos de **page.tsx**



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows the following structure:

- aula05 - Props
  - props
    - .next
    - node\_modules
    - public
    - src\app
      - MeuCastelo
        - page.tsx (selected)
        - Torre.tsx

The code editor shows the following code in `page.tsx`:

```
1 // Click here to ask Blackbox to help you code faster |
2 // Usando o componente e passando as props
3 import Torre from "./Torre";
4 const MeuCastelo = () => {
5   | return <Torre altura={10} cor="azul" janelas={true} />;
6 }
7
8 export default MeuCastelo
// Exportando um módulo
```

# REACT – PROPS

## ✓ Introdução as Props

### ✓ Exemplo 01 – Analisando linha a linha o arquivo Torre.tsx

```
1  // Definindo as props do componente
2  import Torre from "./Torre";
3  const MeuCastelo = () => {
4      return <Torre altura={10} cor="azul" janelas={true} />;
5      // Retorna o componente Torre com algumas props específicas. Está passando uma altura de 10 unidades, cor
6      // azul e a propriedade janelas configurada como true. Isso significa que o componente Torre será renderizado
7      // com essas propriedades específicas.
8  };
9
10 export default MeuCastelo
```

# REACT – PROPS

## ✓ Introdução as Props

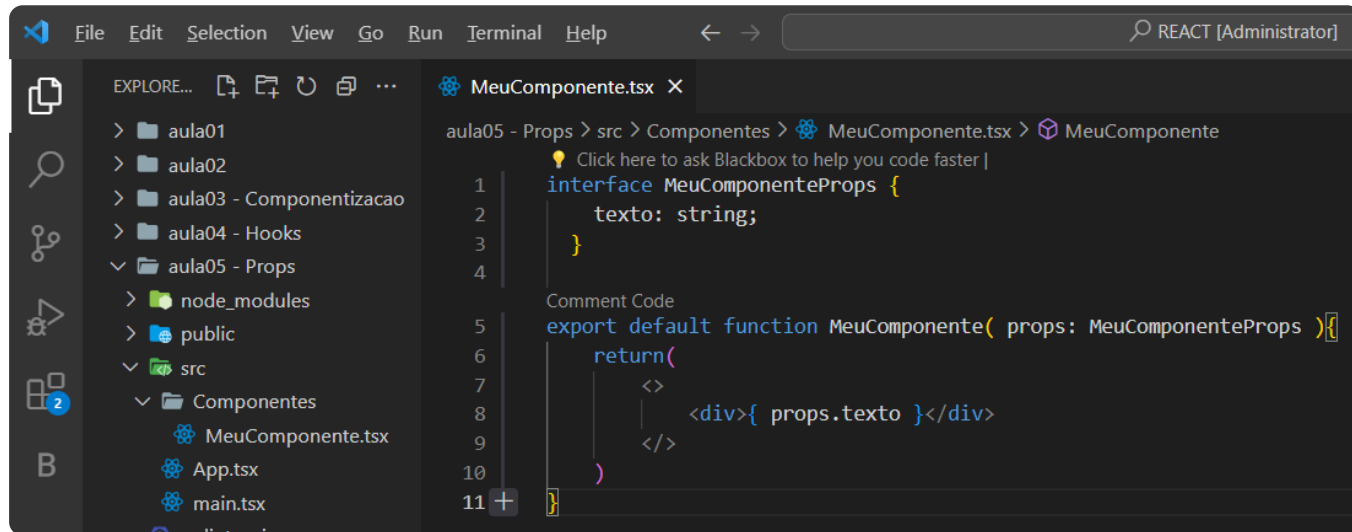
### ✓ Exemplo 01 – Resultado



# REACT – PROPS

## ✓ Introdução as Props

- ✓ Exemplo 02 – Crie uma nova aplicação e vamos modificar o código gerado, vamos criar uma pasta chamada Componentes, dentro de SRC e na sequência criar um arquivo chamado MeuComponente.tsx



The screenshot shows a code editor with a dark theme. On the left, the 'EXPLORE' sidebar shows a file tree with folders 'aula01', 'aula02', 'aula03 - Componentizacao', 'aula04 - Hooks', and 'aula05 - Props'. Under 'aula05 - Props', there is a 'src' folder containing a 'Componentes' folder. Inside 'Componentes', the file 'MeuComponente.tsx' is selected. The main editor area shows the code for 'MeuComponente.tsx'. The code defines an interface 'MeuComponenteProps' with a 'texto' property of type 'string'. It then exports a default function 'MeuComponente' that takes 'props' of type 'MeuComponenteProps' and returns a JSX element: a div with the prop 'props.texto'.

```
1 interface MeuComponenteProps {  
2   texto: string;  
3 }  
4  
5 export default function MeuComponente( props: MeuComponenteProps ) {  
6   return(  
7     <>  
8       <div>{ props.texto }</div>  
9     </>  
10  )  
11 }
```

# REACT – PROPS

## ✓ Introdução as Props

### ✓ Exemplo 02 – Analisando linha a linha o arquivo MeuComponente.tsx

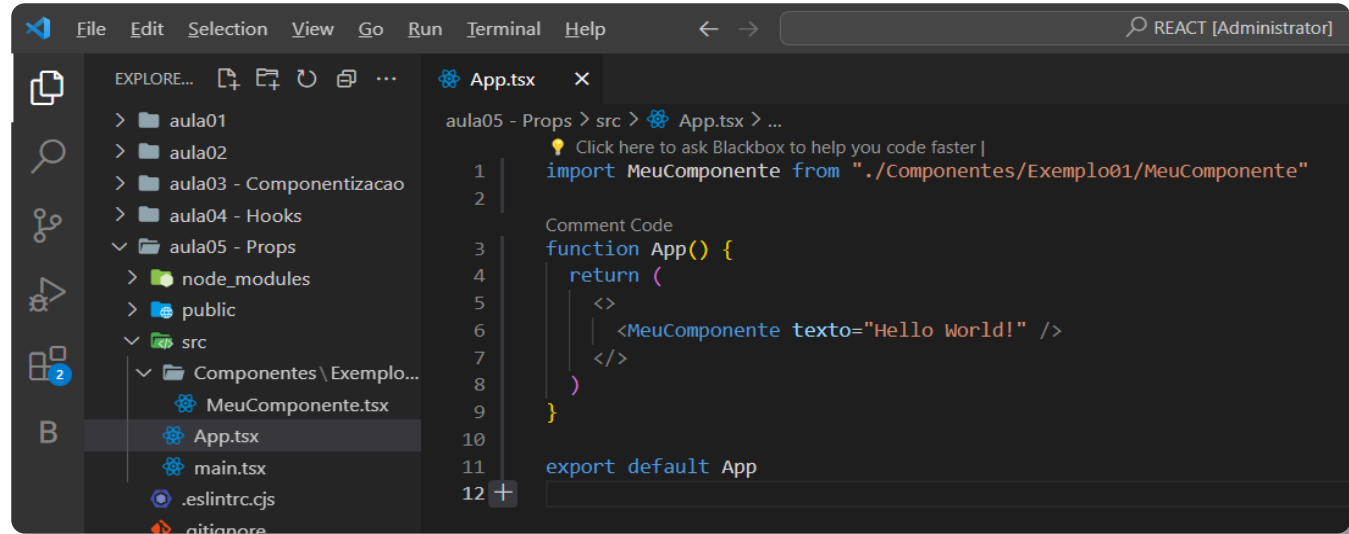
```
1  interface MeuComponenteProps {  
2      texto: string;  
3  }  
4      // Aqui, é definida uma interface chamada MeuComponenteProps que descreve o formato esperado para as propriedades (props) do componente  
5      // MeuComponente. Neste caso, ela espera uma propriedade chamada texto que deve ser uma string.  
6  
7  export default function MeuComponente( props: MeuComponenteProps ){  
8      return (  
9          <>  
10             <div>{ props.texto }</div>  
11             </>  
12         )  
13         // No bloco de retorno da função, o componente renderiza um fragmento (<>...</>) contendo um <div> que exibe o texto passado como propriedade  
14         // (props.texto).  
15     }  
16 }
```



# REACT – PROPS

## ✓ Introdução as Props

- ✓ Exemplo 02 – No arquivo App.tsx, precisamos enviar os valores da propriedade 'texto'

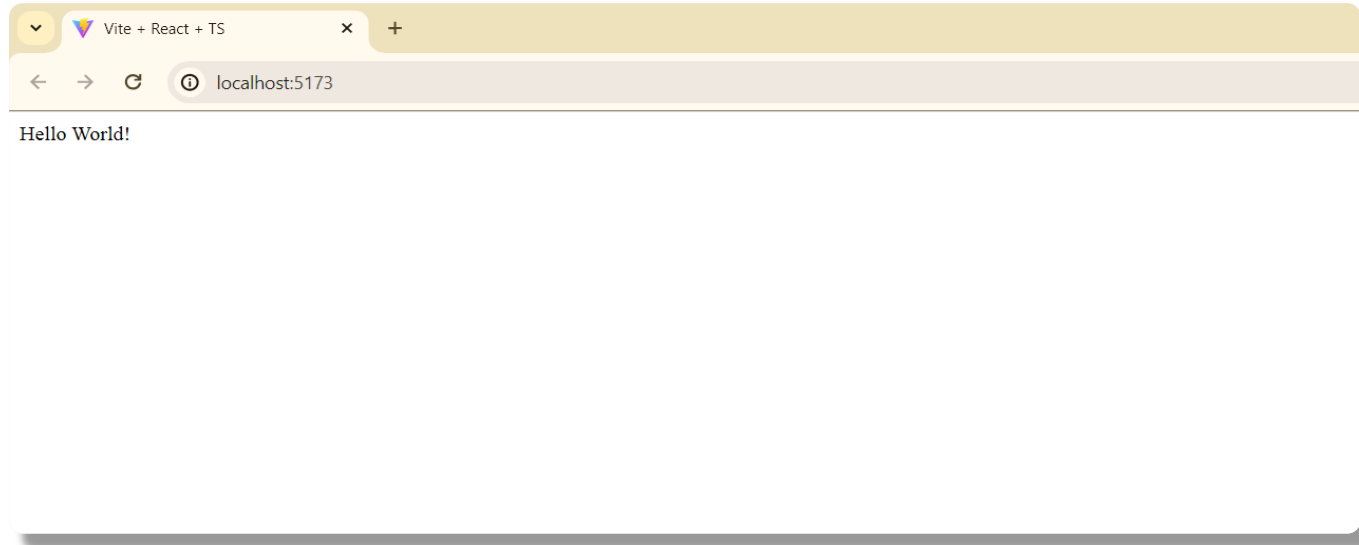


The screenshot shows a code editor with a dark theme. The left sidebar displays a file explorer with a project structure. The main editor area shows the content of App.tsx. The code includes an import statement for MeuComponente, a function App() that returns a JSX element, and an export default statement.

```
File Edit Selection View Go Run Terminal Help
aula05 - Props > src > App.tsx > ...
1 import MeuComponente from "../Componentes/Exemplo01/MeuComponente"
2
3 function App() {
4   return (
5     <>
6       <MeuComponente texto="Hello world!" />
7     </>
8   )
9 }
10
11 export default App
12
```

# REACT – PROPS

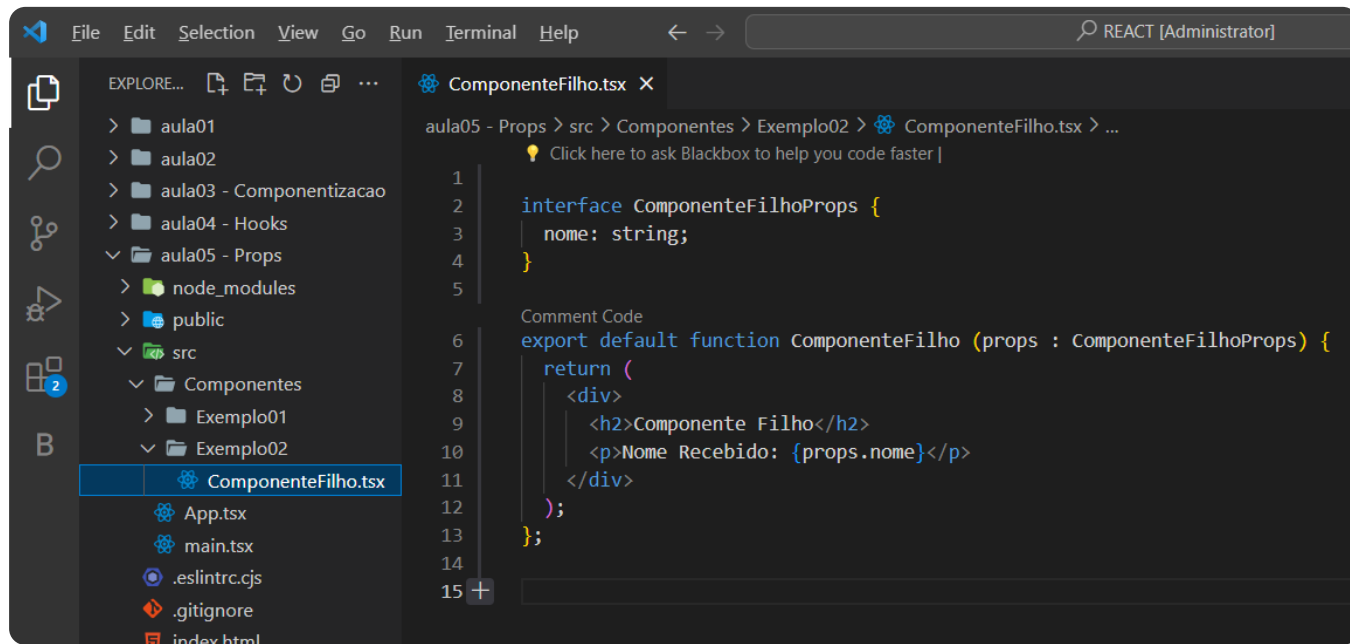
- ✓ **Introdução as Props**
- ✓ Exemplo 02 – Resultado



# REACT – PROPS

## ✓ Introdução as Props

- ✓ Exemplo 03 – Vamos criar um componente chamado ComponenteFilho.tsx



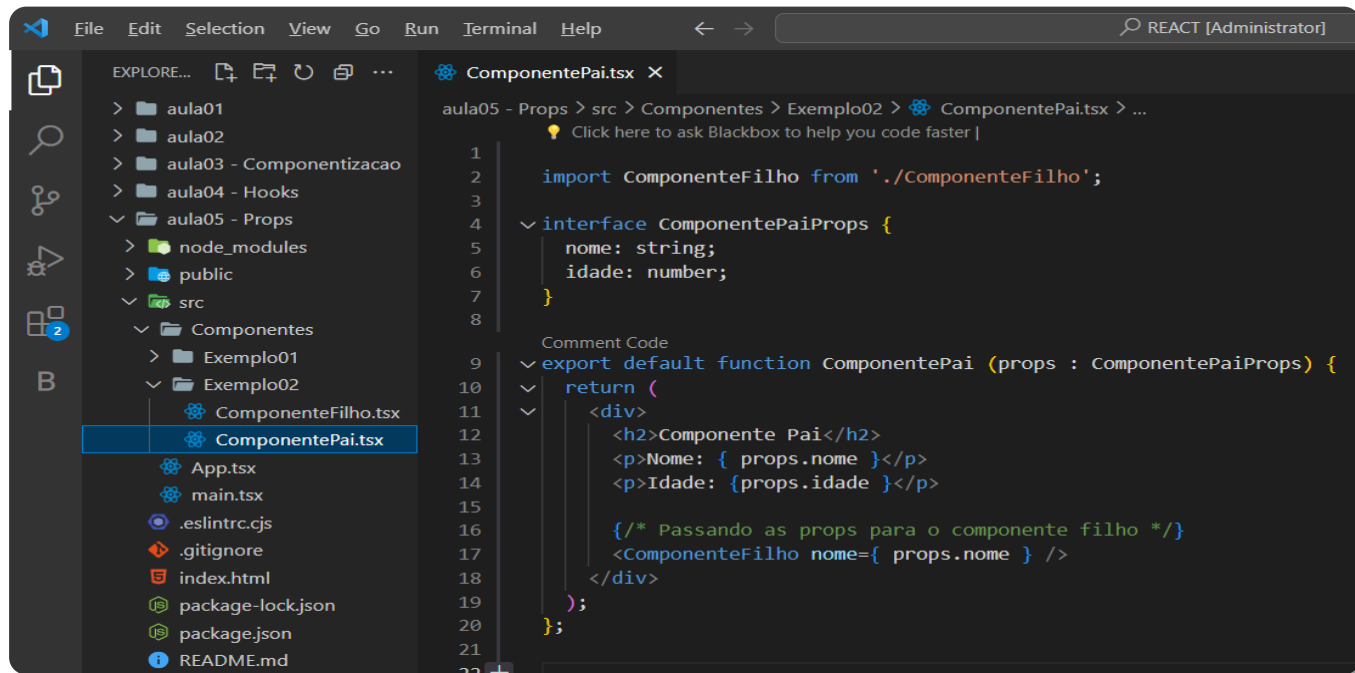
The screenshot shows a code editor with a dark theme. On the left, the 'EXPLORE' sidebar displays a file tree with folders 'aula01' through 'aula05 - Props' and a 'src' folder containing 'Componentes', 'Exemplo01', and 'Exemplo02'. The file 'ComponenteFilho.tsx' is selected under 'Exemplo02'. The main editor area shows the following code:

```
1  
2 interface ComponenteFilhoProps {  
3   nome: string;  
4 }  
5  
6 export default function ComponenteFilho (props : ComponenteFilhoProps) {  
7   return (  
8     <div>  
9       <h2>Componente Filho</h2>  
10      <p>Nome Recebido: {props.nome}</p>  
11    </div>  
12  );  
13 };  
14  
15 +
```

# REACT – PROPS

## ✓ Introdução as Props

- ✓ Exemplo 03 – Vamos, agora, criar um componente chamado ComponentePai.tsx

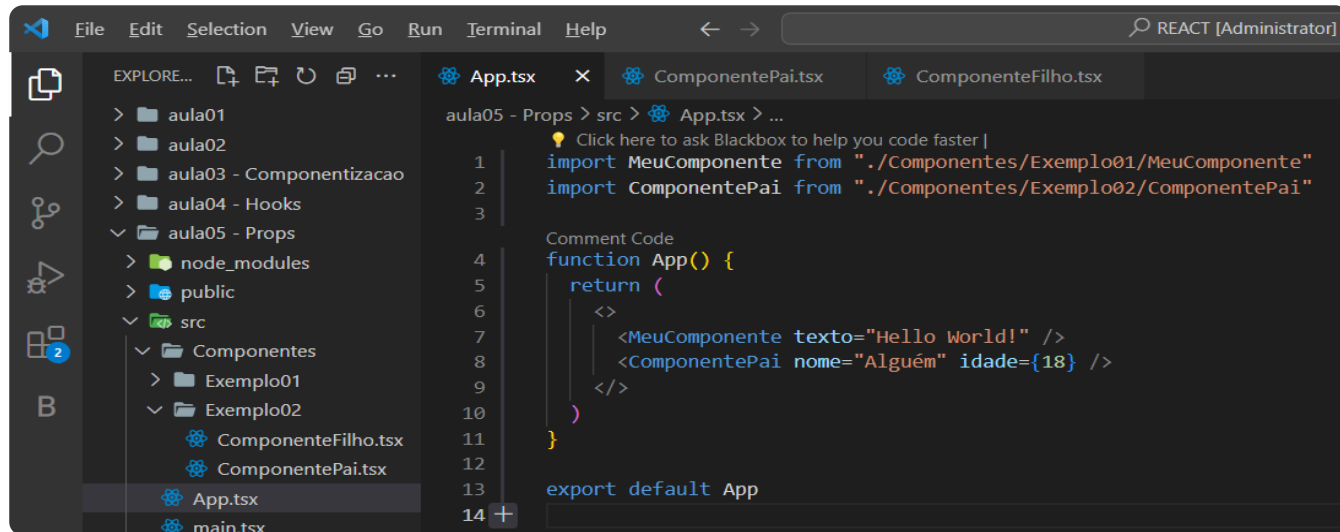


```
1  import ComponenteFilho from './ComponenteFilho';
2
3
4  interface ComponentePaiProps {
5    nome: string;
6    idade: number;
7  }
8
9  export default function ComponentePai (props : ComponentePaiProps) {
10   return (
11     <div>
12       <h2>Componente Pai</h2>
13       <p>Nome: { props.nome }</p>
14       <p>Idade: {props.idade }</p>
15
16       /* Passando as props para o componente filho */
17       <ComponenteFilho nome={ props.nome } />
18     </div>
19   );
20 }
21
22
```

# REACT – PROPS

## ✓ Introdução as Props

- ✓ Exemplo 03 – No arquivo App.tsx, precisamos enviar os valores das propriedades 'nome' e 'idade'



The screenshot shows a VS Code editor window with the following structure:

- EXPLORER:**
  - aula01
  - aula02
  - aula03 - Componentizacao
  - aula04 - Hooks
  - aula05 - Props
    - node\_modules
    - public
    - src
      - Componentes
        - Exemplo01
        - Exemplo02
          - ComponenteFilho.tsx
          - ComponentePai.tsx
      - App.tsx (selected)
      - main.tsx

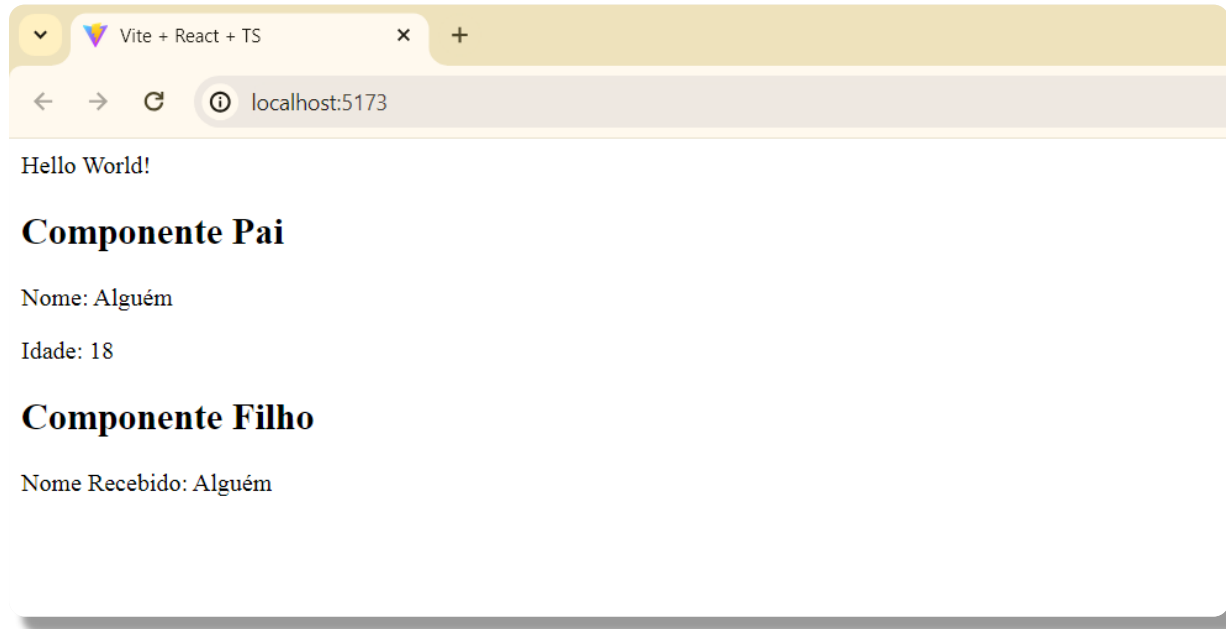
- App.tsx:**

```
1 import MeuComponente from "../Componentes/Exemplo01/MeuComponente"
2 import ComponentePai from "../Componentes/Exemplo02/ComponentePai"
3
4 function App() {
5   return (
6     <>
7       <MeuComponente texto="Hello World!" />
8       <ComponentePai nome="Alguém" idade={18} />
9     </>
10   )
11 }
12
13 export default App
14
```

# REACT – PROPS

## ✓ Introdução as Props

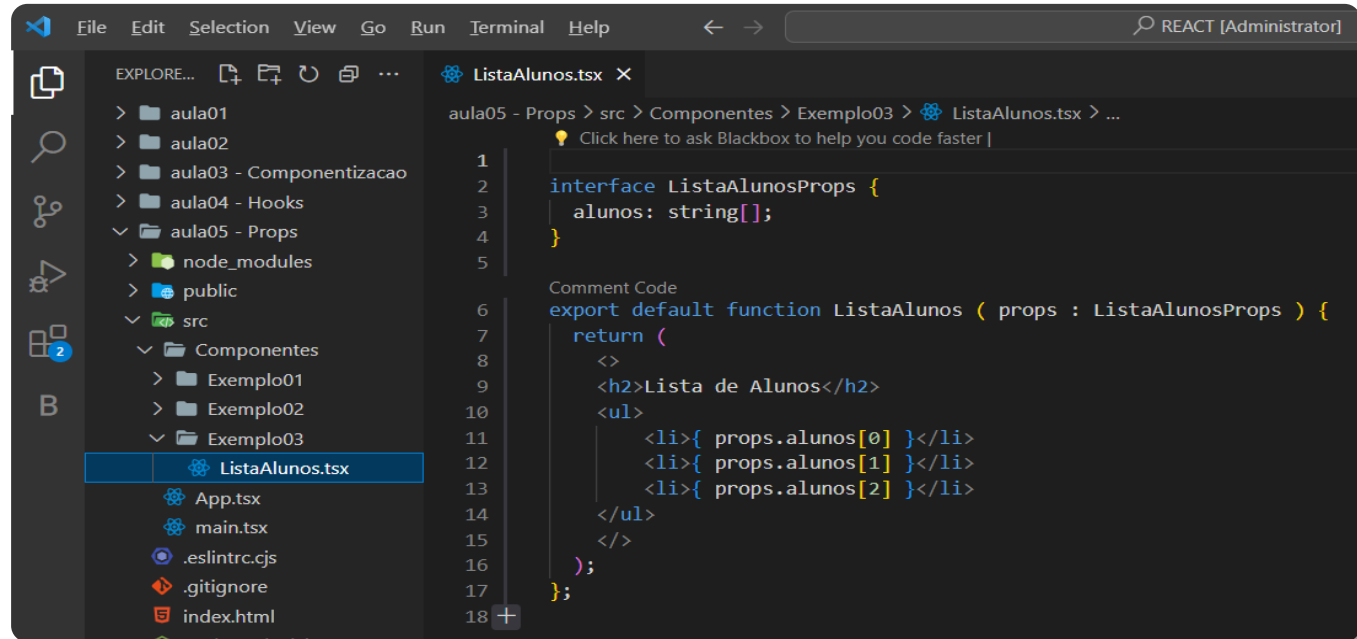
### ✓ Exemplo 03 – Resultado



# REACT – PROPS

## ✓ Introdução as Props

- ✓ Exemplo 04 – Vamos criar um componente chamado ListaAlunos.tsx

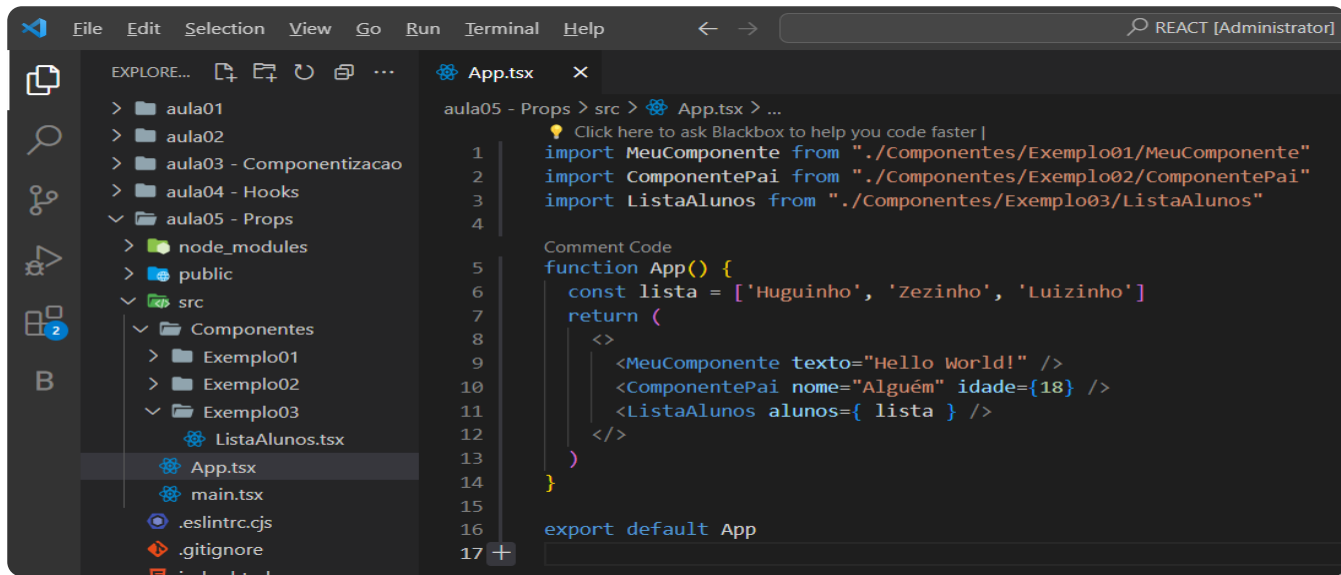


```
1 interface ListaAlunosProps {  
2   alunos: string[];  
3 }  
4  
5  
6 export default function ListaAlunos ( props : ListaAlunosProps ) {  
7   return (  
8     <>  
9     <h2>Lista de Alunos</h2>  
10    <ul>  
11      <li>{ props.alunos[0] }</li>  
12      <li>{ props.alunos[1] }</li>  
13      <li>{ props.alunos[2] }</li>  
14    </ul>  
15    </>  
16  );  
17 }  
18
```

# REACT – PROPS

## ✓ Introdução as Props

- ✓ Exemplo 04 – No arquivo App.tsx, precisamos enviar os valores das propriedades ‘alunos’



```
File Edit Selection View Go Run Terminal Help
EXPLORE...
> aula01
> aula02
> aula03 - Componentizacao
> aula04 - Hooks
> aula05 - Props
  > node_modules
  > public
  > src
    > Componentes
      > Exemplo01
      > Exemplo02
      > Exemplo03
        > ListaAlunos.tsx
        > App.tsx
        > main.tsx
        > .eslintrc.cjs
        > .gitignore
        > index.html

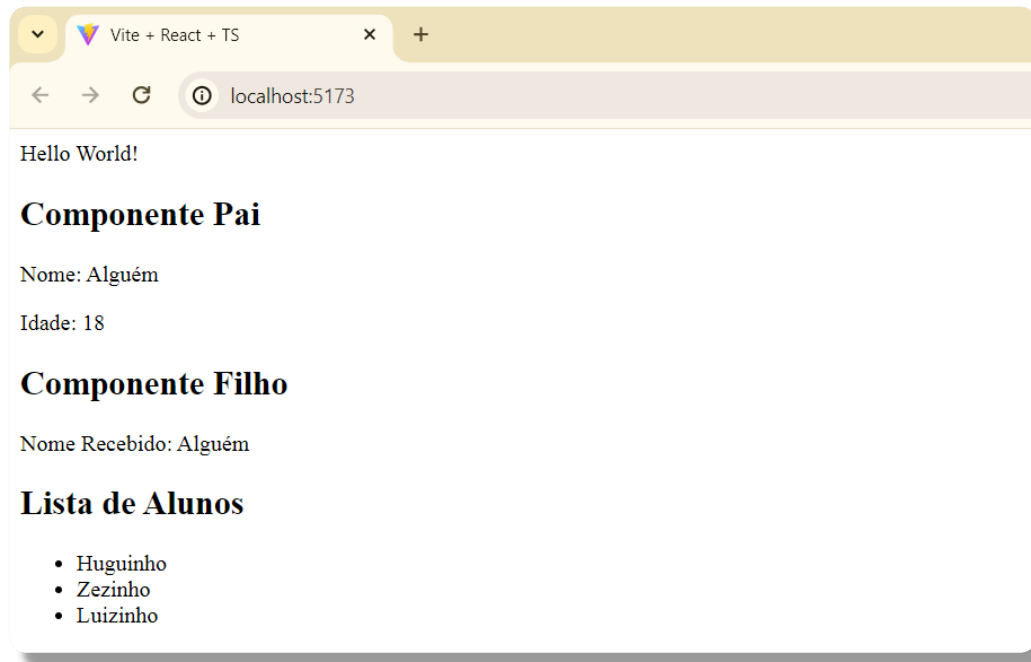
aula05 - Props > src > App.tsx > ...
Click here to ask Blackbox to help you code faster |
1 import MeuComponente from "../Componentes/Exemplo01/MeuComponente"
2 import ComponentePai from "../Componentes/Exemplo02/ComponentePai"
3 import ListaAlunos from "../Componentes/Exemplo03/ListaAlunos"
4
5 function App() {
6   const lista = ['Huguinho', 'Zezinho', 'Luizinho']
7   return (
8     <>
9       <MeuComponente texto="Hello World!" />
10      <ComponentePai nome="Alguém" idade={18} />
11      <ListaAlunos alunos={ lista } />
12    </>
13  )
14 }
15
16 export default App
17
```



# REACT – PROPS

## ✓ Introdução as Props

### ✓ Exemplo 04 – Resultado

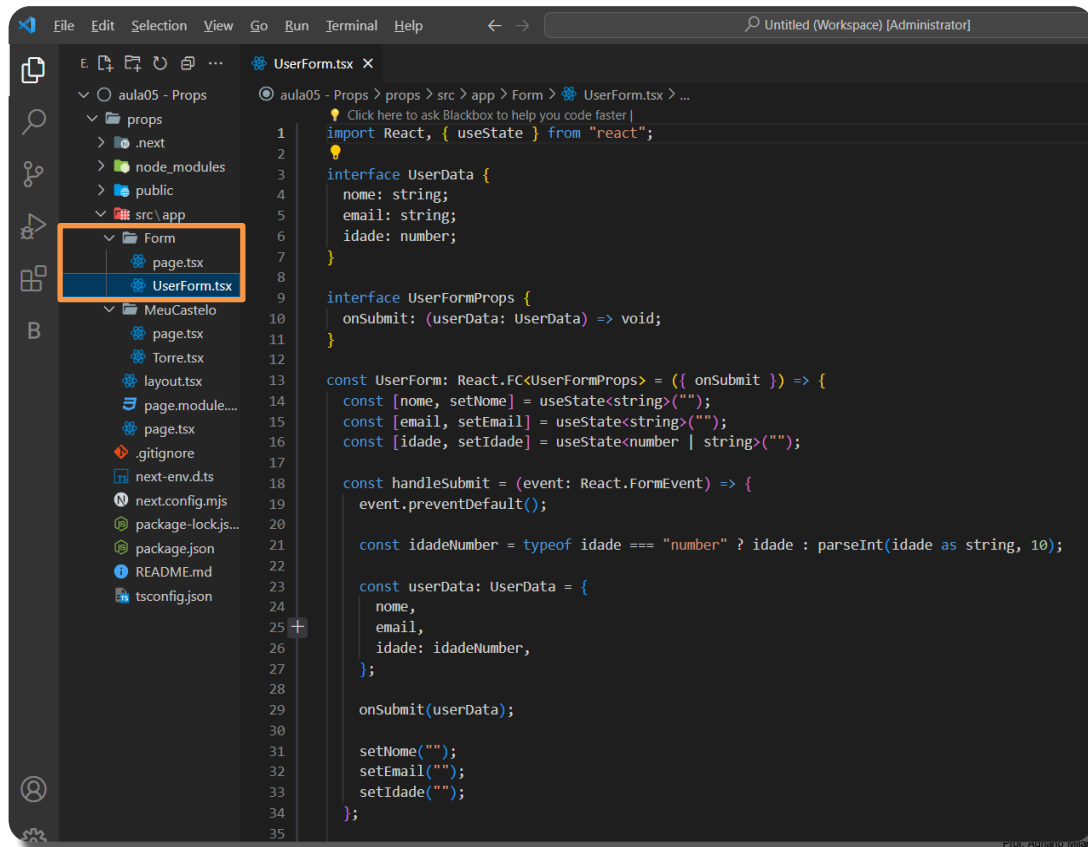


# REACT – PROPS

## ✓ Introdução as Props

- ✓ Exemplo 05 – Vamos criar um formulário e receber e exibir os dados. Vamos criar uma pasta chamada **Form** e criar um arquivo chamado **UserForm.tsx**

Obs.: Neste exemplo foi utilizado o framework NEXT.JS e não o VITE.JS como nos exemplos anteriores.



The screenshot shows a code editor with a sidebar on the left displaying a file tree. The tree structure is as follows:

- aula05 - Props
  - props
    - .next
    - node\_modules
    - public
    - src\app
      - Form
        - page.tsx
        - UserForm.tsx** (highlighted with an orange box)
      - MeuCastelo
        - page.tsx
        - Torre.tsx
      - layout.tsx
      - page.module....
      - page.tsx
      - .gitignore
      - next-env.d.ts
      - next.config.mjs
      - package-lockjs...
      - package.json
      - README.md
      - tsconfig.json

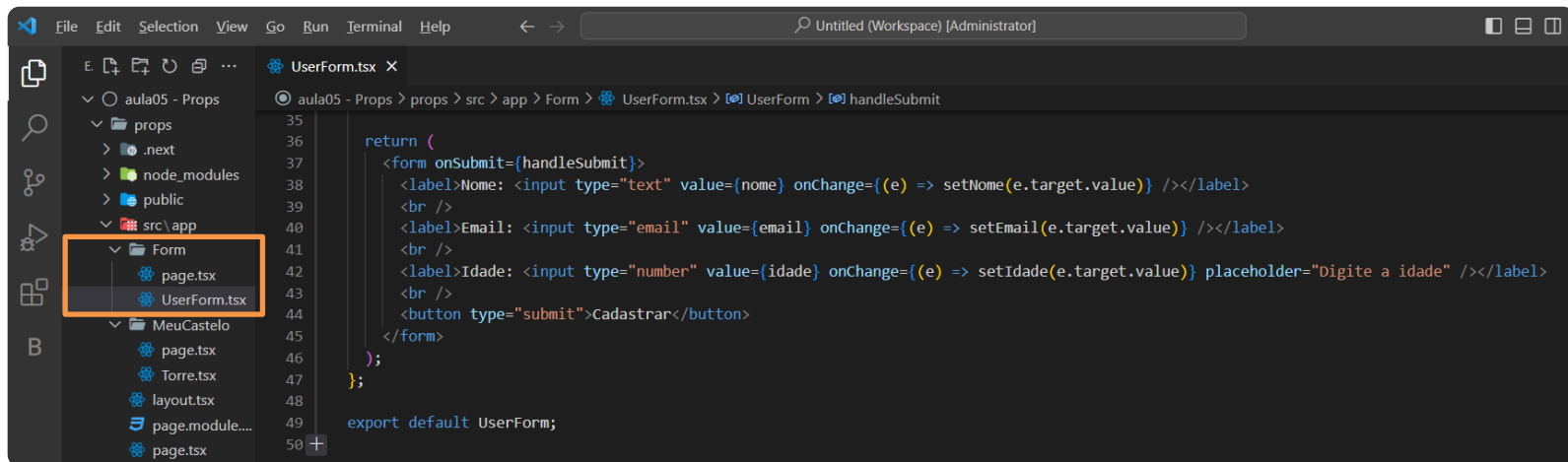
The main editor area displays the content of **UserForm.tsx**:

```
1 import React, { useState } from "react";
2
3 interface UserData {
4   nome: string;
5   email: string;
6   idade: number;
7 }
8
9 interface UserFormProps {
10   onSubmit: (userData: UserData) => void;
11 }
12
13 const UserForm: React.FC<UserFormProps> = ({ onSubmit }) => {
14   const [nome, setNome] = useState<string>("");
15   const [email, setEmail] = useState<string>("");
16   const [idade, setIdade] = useState<number | string>("");
17
18   const handleSubmit = (event: React.FormEvent) => {
19     event.preventDefault();
20
21     const idadeNumber = typeof idade === "number" ? idade : parseInt(idade as string, 10);
22
23     const userData: UserData = {
24       nome,
25       email,
26       idade: idadeNumber,
27     };
28
29     onSubmit(userData);
30
31     setNome("");
32     setEmail("");
33     setIdade("");
34   };
35 }
```

# REACT – PROPS

## ✓ Introdução as Props

- ✓ Exemplo 05 – Vamos criar um formulário e receber e exibir os dados. Vamos criar um arquivo chamado UserForm.tsx (continuação)



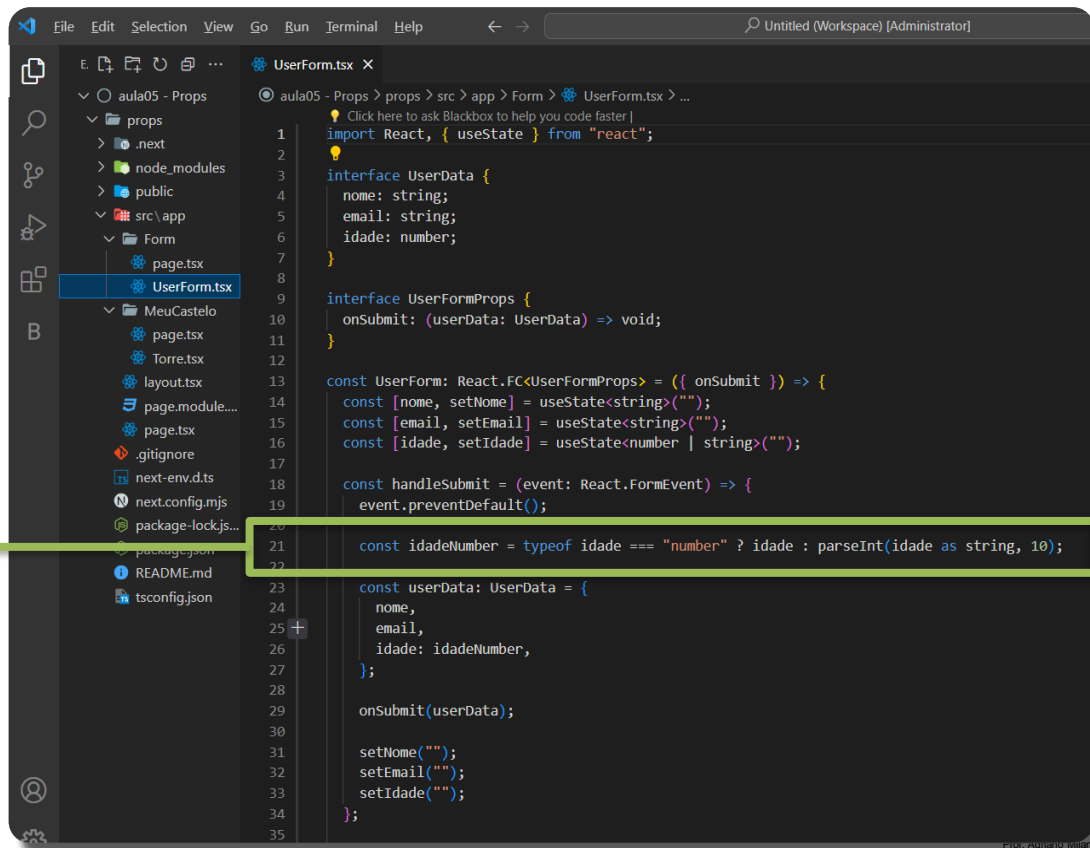
```
35
36
37   return (
38     <form onSubmit={handleSubmit}>
39       <label>Nome: <input type="text" value={nome} onChange={(e) => setName(e.target.value)} /></label>
40       <br />
41       <label>Email: <input type="email" value={email} onChange={(e) => setEmail(e.target.value)} /></label>
42       <br />
43       <label>Idade: <input type="number" value={idade} onChange={(e) => setIdade(e.target.value)} placeholder="Digite a idade" /></label>
44       <br />
45       <button type="submit">Cadastrar</button>
46     </form>
47   );
48
49   export default UserForm;
50
```

# REACT – PROPS

## ✓ Introdução as Props

Este é um operador ternário (?:), que funciona como uma expressão condicional. Se `typeof idade` for "number", a expressão retorna simplesmente `idade`. Caso contrário, a expressão após os dois pontos : é avaliada.

`parseInt(idade as string, 10)`: Isso tenta converter a variável `idade` para um número usando a função `parseInt`. A função `parseInt` aceita dois argumentos: a string a ser convertida e a base numérica para a conversão (neste caso, 10 para base decimal). A expressão `(idade as string)` é usada para indicar explicitamente ao TypeScript que consideramos `idade` como uma string.

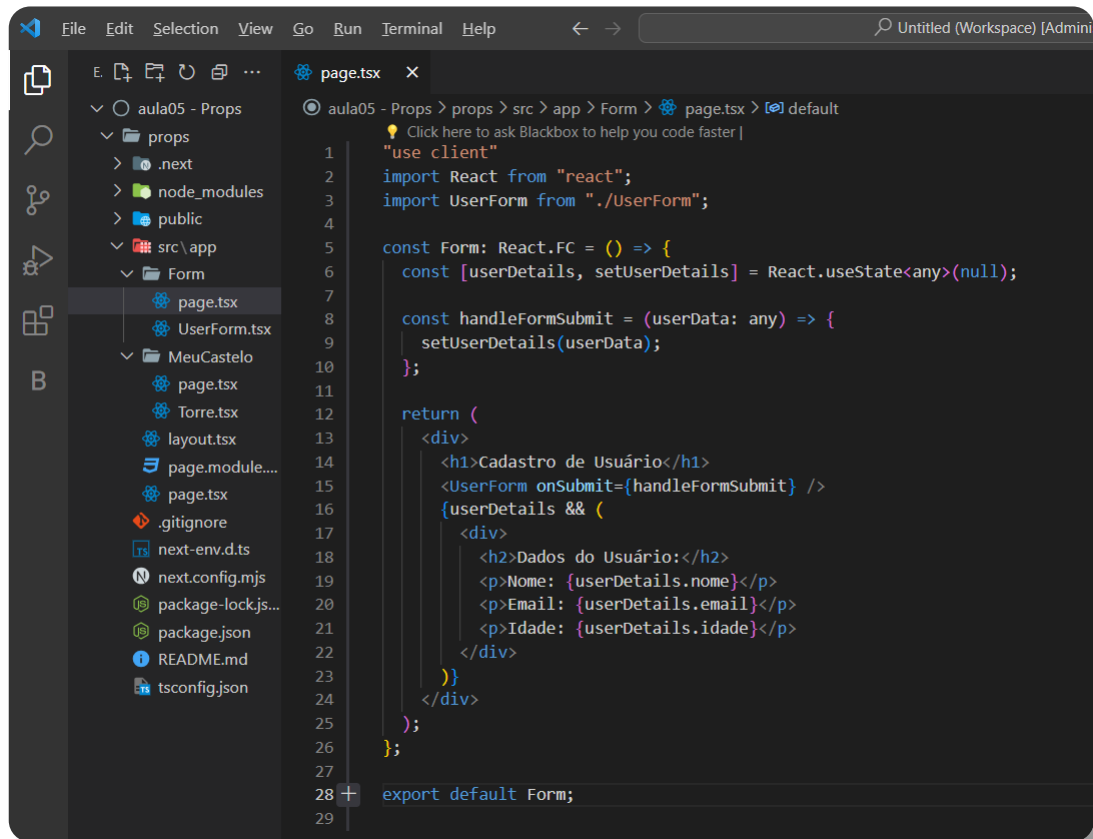


```
1 import React, { useState } from "react";
2
3 interface UserData {
4   nome: string;
5   email: string;
6   idade: number;
7 }
8
9 interface UserFormProps {
10   onSubmit: (userData: UserData) => void;
11 }
12
13 const UserForm: React.FC<UserFormProps> = ({ onSubmit }) => {
14   const [nome, setNome] = useState<string>("");
15   const [email, setEmail] = useState<string>("");
16   const [idade, setIdade] = useState<number | string>("");
17
18   const handleSubmit = (event: React.FormEvent) => {
19     event.preventDefault();
20
21     const idadeNumber = typeof idade === "number" ? idade : parseInt(idade as string, 10);
22
23     const userData: UserData = {
24       nome,
25       email,
26       idade: idadeNumber,
27     };
28
29     onSubmit(userData);
30
31     setNome("");
32     setEmail("");
33     setIdade("");
34   };
35 }
```

# REACT – PROPS

## ✓ Introdução as Props

- ✓ Exemplo 05 – Vamos criar o arquivo **page.tsx**, que receberá o componente **UserForm** e exibirão os dados recebidos.



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 'aula05 - Props' containing a 'Form' folder. The 'Form' folder contains 'page.tsx' and 'UserForm.tsx'. The 'MeuCastelo' folder contains 'page.tsx', 'Torre.tsx', 'layout.tsx', 'page.module...', 'page.tsx', '.gitignore', 'next-env.d.ts', 'next.config.mjs', 'package-lock.js...', 'package.json', 'README.md', and 'tsconfig.json'. The code editor shows the content of 'page.tsx'.

```
1  "use client"
2  import React from "react";
3  import UserForm from "../UserForm";
4
5  const Form: React.FC = () => {
6    const [userDetails, setUserDetails] = React.useState<any>(null);
7
8    const handleFormSubmit = (userData: any) => {
9      setUserDetails(userData);
10   };
11
12   return (
13     <div>
14       <h1>Cadastro de Usuário</h1>
15       <UserForm onSubmit={handleFormSubmit} />
16       {userDetails && (
17         <div>
18           <h2>Dados do Usuário:</h2>
19           <p>Nome: {userDetails.nome}</p>
20           <p>Email: {userDetails.email}</p>
21           <p>Idade: {userDetails.idade}</p>
22         </div>
23       )}
24     </div>
25   );
26 };
27
28 export default Form;
```

# REACT – PROPS

## ✓ Introdução as Props

### ✓ Exemplo 05 – Resultado

#### Estado inicial da aplicação



A screenshot of a web browser window with the title 'Create Next App' and the address bar showing 'localhost:3000/Form'. The page has a heading 'Cadastro de Usuário' and a form with three input fields: 'Nome:', 'Email:', and 'Idade: Digite a idade'. Below the inputs is a 'Cadastrar' button.

#### Recebendo os dados enviados pelo componente UseForm



A screenshot of a web browser window with the title 'Create Next App' and the address bar showing 'localhost:3000/Form'. The page has a heading 'Cadastro de Usuário' and a form with three input fields: 'Nome:', 'Email:', and 'Idade: Digite a idade'. Below the inputs is a 'Cadastrar' button. Below the form, there is a section titled 'Dados do Usuário:' with the following text: 'Nome: Joaozinho', 'Email: joaozinho@algumlugar.com.br', and 'Idade: 23'.

# REACT – PROPS

## ✓ Introdução as Props

### ✓ Exercício 01

- ✓ Suponha que você tenha um componente chamado CardUsuario que exibe informações sobre um usuário, como nome e idade. Crie um novo componente chamado ListaUsuarios que recebe uma lista de usuários como propriedade e renderiza vários CardUsuario com base nessas informações. Certifique-se de passar as propriedades corretas para cada instância do CardUsuario.

# REACT – PROPS

## ✓ Introdução as Props

### ✓ Exercício 02

- ✓ Desenvolva um aplicativo de lista de tarefas em React. Crie dois componentes: ListaTarefas e Tarefa. O componente ListaTarefas receberá uma lista de objetos de tarefa como propriedade. Cada objeto de tarefa deve ter uma descrição e um status (completo ou não completo). O componente ListaTarefas deve renderizar várias instâncias do componente Tarefa, passando as propriedades apropriadas para cada uma delas.
- ✓ O componente Tarefa deve exibir a descrição da tarefa e um indicador visual de seu status. Adicione funcionalidade para alternar o status de uma tarefa (de completo para não completo e vice-versa) quando o usuário clicar na tarefa.



# 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.

# FRONT-END DESIGN ENGINEERING

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.

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.

# FRONT-END DESIGN ENGINEERING

W3C. 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.

\_\_\_\_\_. 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.