



# **Desenvolvimento Front-End**

React

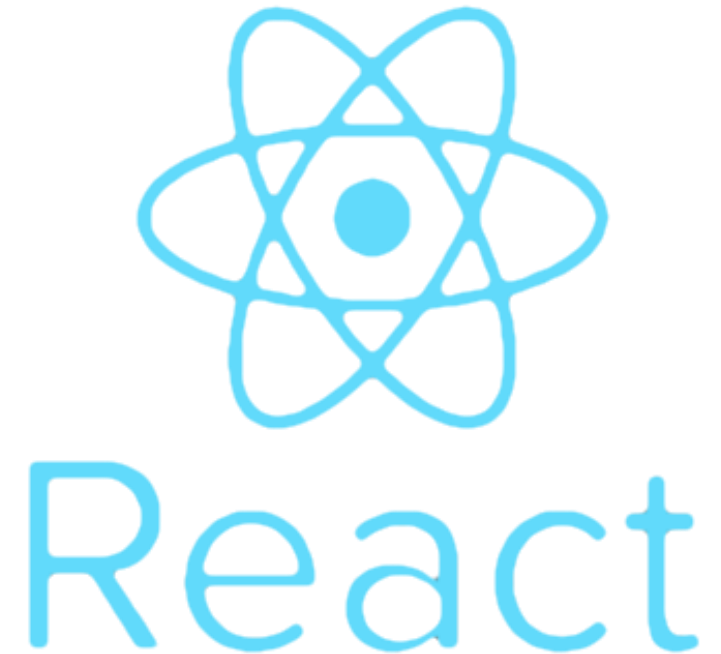


## Criando novos componentes.

As aplicações React são compostas por *componentes*. Um componente é uma parte da IU (interface do usuário) que possui sua própria lógica e aparência. Um componente pode ser tão pequeno quanto um botão, ou tão grande quanto uma página inteira.

Componentes do React são funções JavaScript que retornam marcação (markup):

Fonte: <https://pt-br.react.dev/learn>



# ***Criar um projeto React***

**npx create-react-app componentes**

**cd componentes**

**npm start**



## Criando um componente Botão

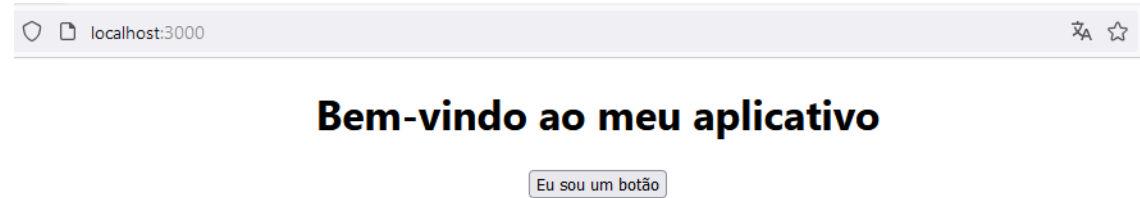
Lembre da estrutura de pastas da aula anterior.

```
my-app/
|
├── node_modules/
├── public/
│   ├── favicon.ico
│   ├── index.html
│   └── manifest.json
├── src/
│   ├── assets/
│   │   └── images/    // (Imagens e outros arquivos estáticos)
│   ├── components/   // (Componentes reutilizáveis da aplicação)
│   │   └── Header.js
│   ├── App.js
│   ├── App.css
│   ├── index.js
│   └── index.css
├── .gitignore
├── package.json
├── package-lock.json
└── README.md
```

# Criando um componente Botão

Lembre da estrutura de pastas da aula anterior.

```
function Primeiro_Botao() {  
  return (  
    <button>Eu sou um botão</button>  
  );  
}  
  
export default Primeiro_Botao;
```



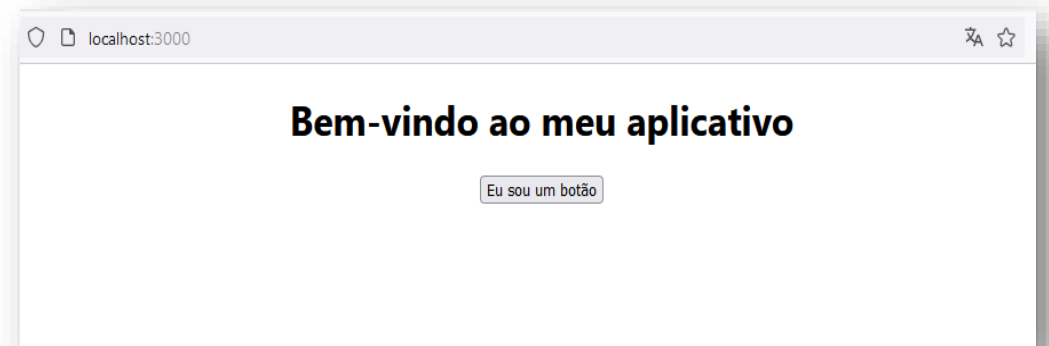

# Chamando o componente Botão no App.js

Lembre da estrutura de pastas da aula anterior.

```
import './App.css';
import Primeiro_Botao from './componentes/botao/Primeiro_Botao';

function App() {
  return (
    <div className="App">
      <h1>Bem-vindo ao meu aplicativo</h1>
      <Primeiro_Botao/>
    </div>
  );
}

export default App;
```



# Atenção!

```
import './App.css';
import Primeiro_Botao from './componentes/botao/Primeiro_Botao';

function App() {
  return (
    <div className="App">
      <h1>Bem-vindo ao meu aplicativo</h1>
      <Primeiro_Botao/>
    </div>
  );
}

export default App;
```



Repare que `<Primeiro_Botao />` começa com letra maiúscula. É dessa forma que você identifica um componente React. Os nomes dos componentes React sempre devem começar com letra maiúscula, enquanto as tags HTML devem ser em minúsculas.



# Aplicando um estilo para o componente Botão no App.js

Criar o arquivo  
botao.css

```
button {  
  background-color: #2106ac; /* Cor de fundo azul do botão */  
  color: white; /* Cor do texto branca */  
  border: none; /* Remove as bordas padrão */  
  padding: 12px 24px; /* Adiciona espaçamento interno */  
  font-size: 16px; /* Define o tamanho da fonte do texto */  
  font-weight: bold; /* Deixa o texto em negrito */  
  border-radius: 8px; /* Arredonda as bordas do botão */  
  cursor: pointer; /* Mostra o cursor de 'mãozinha' ao passar sobre o botão */  
  transition: background-color 0.3s ease; /* Anima a mudança de cor do fundo de forma suave */  
}  
  
/* Estilo para quando o botão é focado (clicado ou acessado via teclado) */  
button:focus {  
  outline: none; /* Remove a linha de contorno padrão que aparece ao focar */  
  box-shadow: 0 0 10px rgba(68, 170, 237, 0.5); /* Adiciona um brilho ao redor do botão */  
}  
  
/* Efeito hover (ao passar o mouse sobre o botão) */  
button:hover {  
  background-color: #2980b9; /* Altera a cor de fundo ao passar o mouse */  
}  
  
/* Estilo para quando o botão é pressionado */  
button:active {  
  background-color: #1f8d8d; /* Muda a cor para um tom de azul ainda mais escuro ao clicar */  
  transform: scale(0.98); /* Faz um leve efeito de "apertar" o botão ao pressionar */  
}
```

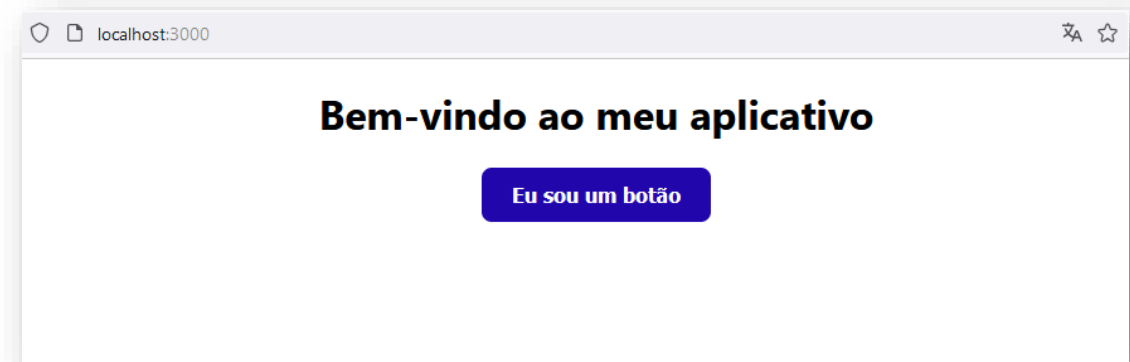
## Aplicando um estilo para o componente Botão no App.js

```
import './botao.css';
```



```
function Primeiro_Botao() {  
  return (  
    <button>Eu sou um botão</button>  
  );  
}
```

```
export default Primeiro_Botao;
```

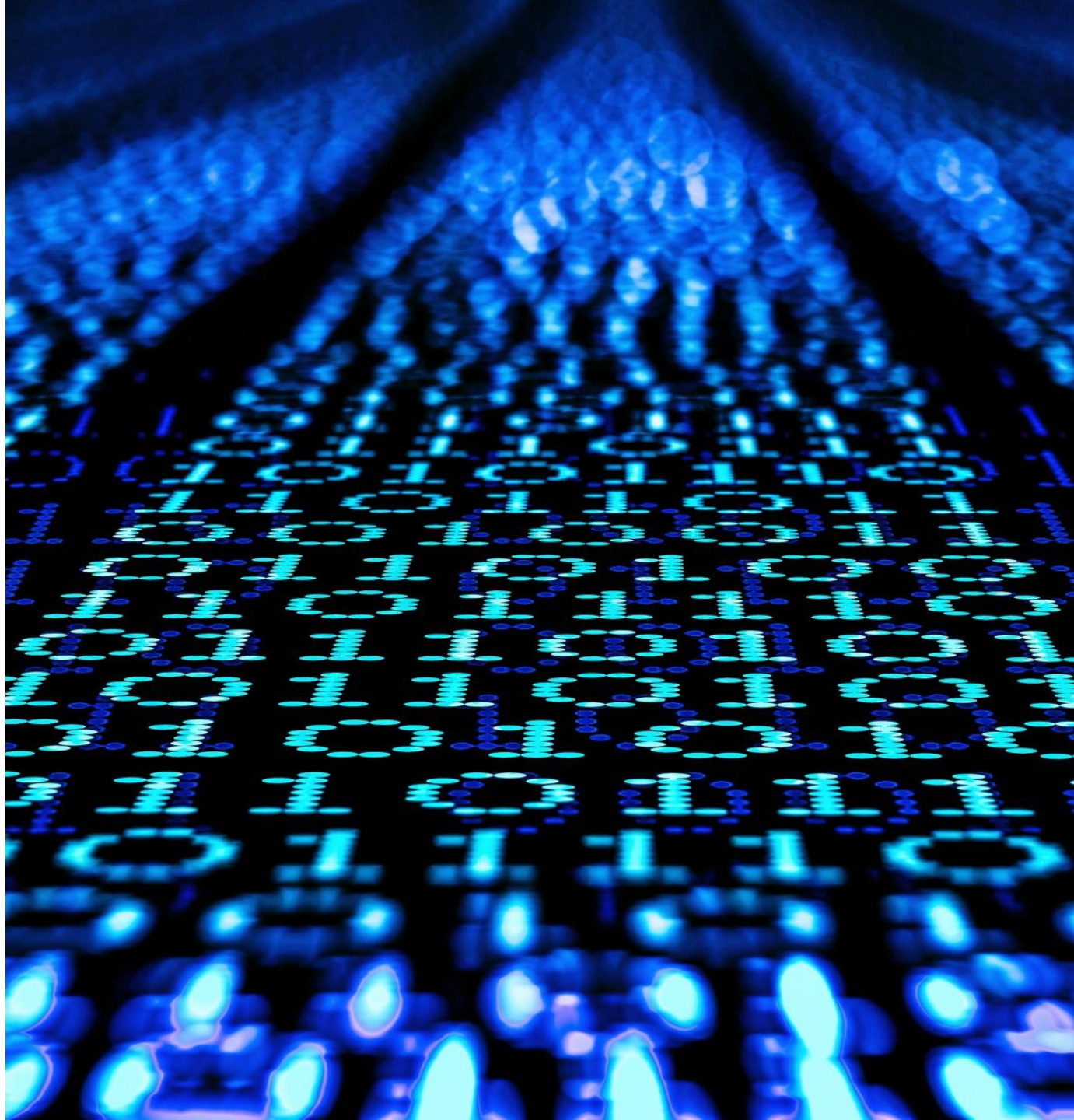


---

## Trabalhando com dados


JSX permite que você coloque marcação dentro do JavaScript.

As chaves permitem que você insira expressões JavaScript. Isso é útil para incorporar variáveis do seu código e exibi-las para o usuário. Por exemplo, isso irá exibir `user.name`:



## Criando o componente Texto\_01

Foi criada uma variável e passado um valor para essa variável. Após é passada a expressão entre chaves.

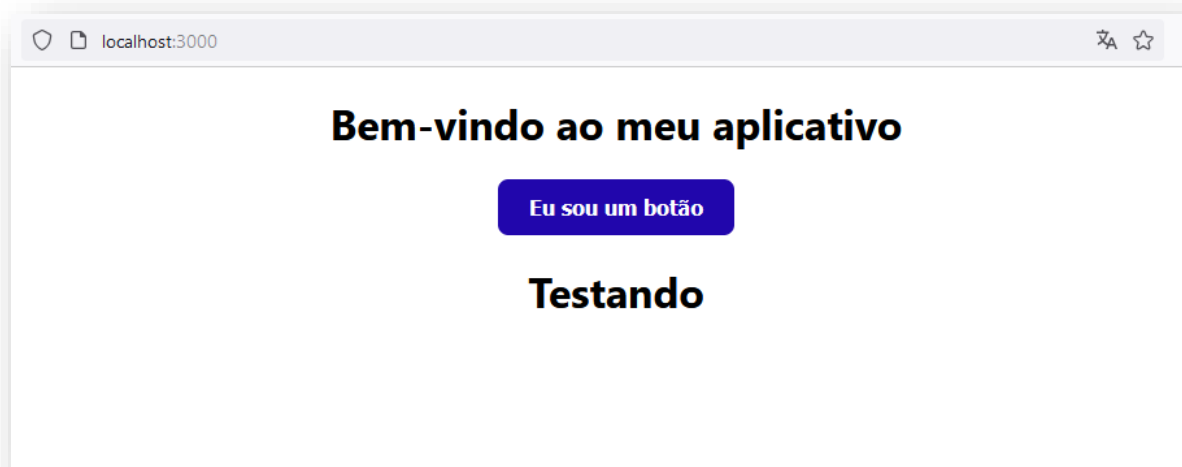
```
let valor;  
let user;  
valor = 'Testando';  
function Texto_01() {  
  return(  
    <h1>  
      {valor}   
    </h1>  
  );  
}  
export default Texto_01;
```

## Chamando o componente Texto\_01 no App.js

```
import './App.css';
import Primeiro_Botao from './componentes/botao/Primeiro_Botao';
import Texto_01 from './componentes/dados/Texto_01';

function App() {
  return (
    <div className="App">
      <h1>Bem-vindo ao meu aplicativo</h1>
      <Primeiro_Botao/>
      <Texto_01/>
    </div>
  );
}

export default App;
```





# Atividade

Crie dois novos elementos baseados nos elementos Primeiro\_Botao e Texto\_01, convertendo suas funções para o formato de ***arrow function***.

Elabore um texto explicando as vantagens da utilização da *arrow function*.





## Trabalhando com listas.

Note que `<li>` possui um atributo `key`. Para cada item em uma lista, deve-se passar uma string ou um número que identifica unicamente esse item entre seus irmãos. Normalmente, uma chave deve vir dos seus dados, como um ID de banco de dados. O React utiliza essas chaves para entender as mudanças que ocorrem se você posteriormente inserir, excluir ou reordenar os itens.

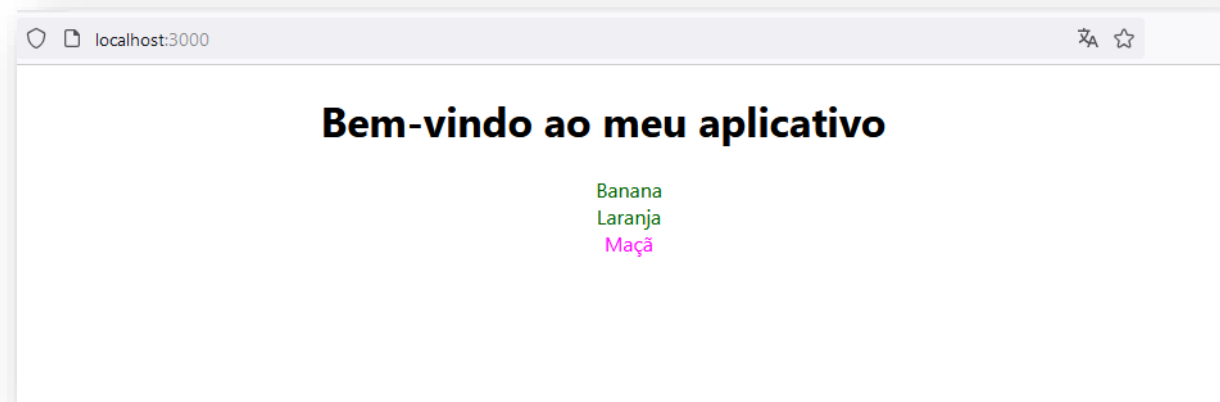
```
const frutas = [
  { title: 'Banana', Fruta: false, id: 1 },
  { title: 'Laranja', Fruta: false, id: 2 },
  { title: 'Maçã', Fruta: true, id: 3 },
];

export default function Lista_Frutas() {
  const Lista = frutas.map(product =>
    <li
      key={product.id}
      style={{color: product.Fruta ? 'magenta' : 'darkgreen' }}
    >
      {product.title}
    </li>
  );

  return (
    <ul>{Lista}</ul>
  );
}
```

## Chamando o componente lista no App.js

```
import './App.css';  
import Lista_Frutas from './componentes/lista/Lista';  
  
function App() {  
  return (  
    <div className="App">  
      <h1>Bem-vindo ao meu aplicativo</h1>  
      <Lista_Frutas/>  
    </div>  
  );  
}  
  
export default App;
```





## Trabalhando com eventos

É possível responder a eventos declarando funções de event handler dentro dos seus componentes:

Note como `onClick={handleClick}` não tem parênteses no final! Não chame a função de manipulador de evento: você só precisa passá-la. O React chamará seu manipulador de evento quando o usuário clicar no botão.

```
const Botao_02 = () => {  
  function handleClick() {  
    alert('Você clicou no botão!');  
  }  
  
  return (  
    <button onClick={handleClick}>  
      Clique aqui  
    </button>  
  );  
}  
  
export default Botao_02;
```

## Chamando o componente Botao\_02 no App.js

```
import './App.css';
import Botao_02 from './componentes/botao_02/Botao_02';

function App() {
  return (
    <div className="App">
      <h1>Bem-vindo ao meu aplicativo</h1>
      <Botao_02/>
    </div>
  );
}

export default App;
```



# Exercício

Elabore uma aplicação utilizando React que contenha **quatro botões**. Cada botão deve executar uma ação ao abrir uma **tela de alerta**, como descrito a seguir:

- **Botão 1:** Abrir uma tela de alerta com a mensagem: **“ETEC”**.
- **Botão 2:** Abrir uma tela de alerta com a mensagem: **“Técnico em Informática”**, com os caracteres da mensagem em **vermelho**.
- **Botão 3:** Abrir uma tela de alerta com a mensagem: **"Curso de Desenvolvimento de Sistemas"**, com a mensagem centralizada e em **negrito**.
- **Botão 4:** Abrir uma tela de alerta com a mensagem: **"Bem-vindo ao React!"**, com a mensagem em **itálico** e em um **tamanho maior** de fonte.

Deixe os botões no App.js organizados em duas colunas / duas linhas.