REACT

Uma biblioteca JavaScript para criar interfaces de usuário

- **Declarativo**

React faz com que a criação de UIs interativas seja uma tarefa fácil. Crie views simples para cada estado na sua aplicação, e o React irá atualizar e renderizar de forma eficiente apenas os componentes necessários na medida em que os dados mudam.

Views declarativas fazem com que seu código seja mais previsível e simples de depurar.

- **Baseado em componentes**

Crie componentes encapsulados que gerenciam seu próprio estado e então, combine-os para criar UIs complexas.

Como a lógica do componente é escrita em JavaScript e não em templates, você pode facilmente passar diversos tipos de dados ao longo da sua aplicação e ainda manter o estado fora do DOM.

O React também pode ser renderizado no servidor, usando Node, e ser usado para criar aplicações mobile, através do React Native.

1. INSTALAÇÃO

- Adicionar react à um site
- Criar um novo react app
- CDN links

Tanto React como ReactDOM estão disponíveis através de CDN.

```
<script crossorigin src="https://unpkg.com/react@17/umd/react.development.js"></script>
<script crossorigin src="https://unpkg.com/react-dom@17/umd/react-dom.development.js"></script>
```

As versões acima devem ser utilizadas apenas para desenvolvimento e não são adequadas para o ambiente de produção. Versões reduzidas e otimizadas para produção estão disponíveis em:

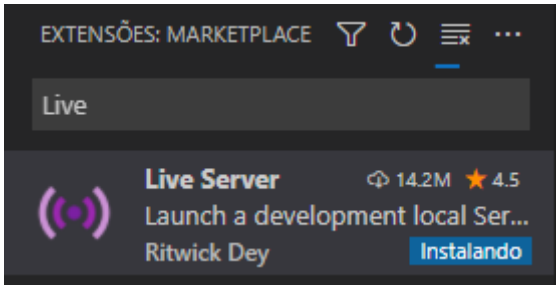
```
<script crossorigin src="https://unpkg.com/react@17/umd/react.production.min.js"></script>
<script crossorigin src="https://unpkg.com/react-dom@17/umd/react-dom.production.min.js"></script>
```

Para carregar uma versão específica do `react` e `react-dom`, substitua `17` com o número da versão que você deseja.

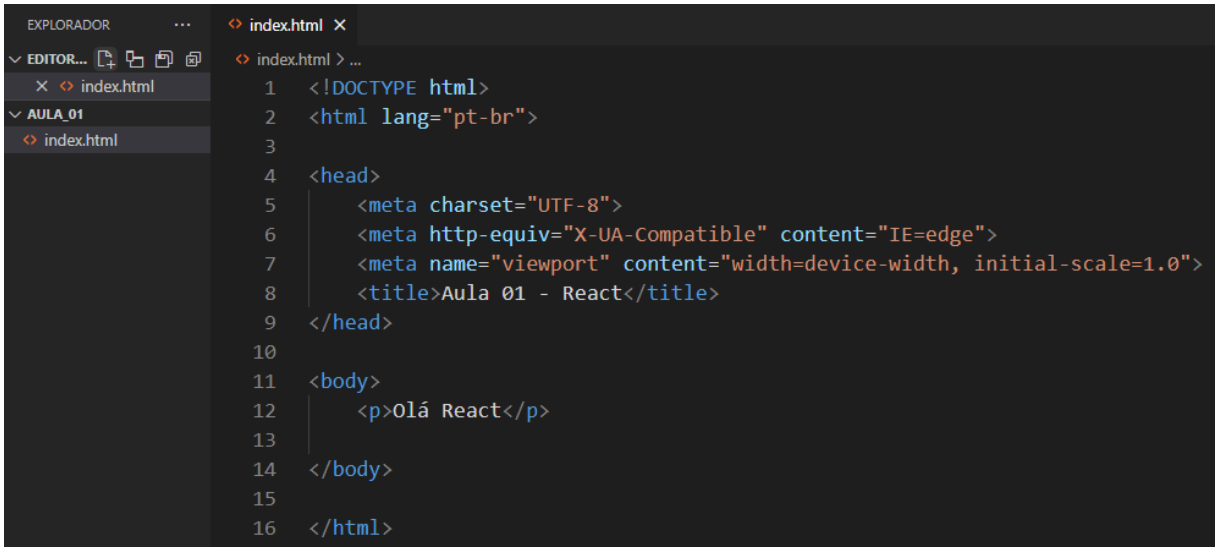
- **React com JSX (*)**

```
<script src="https://unpkg.com/babel-standalone@6/babel.min.js"></script>
```

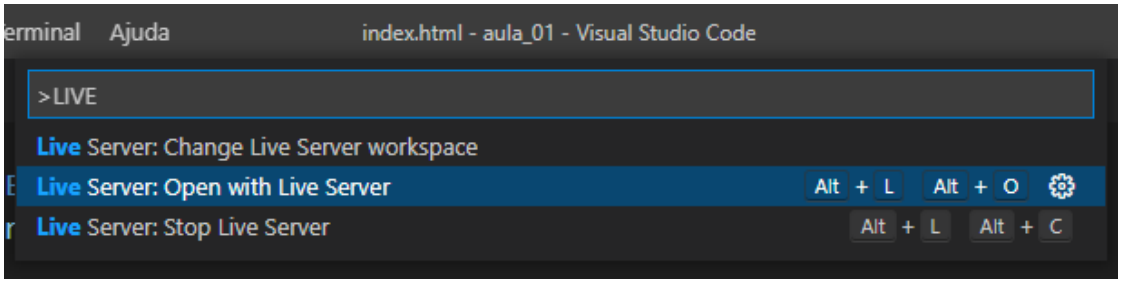
No Visual Studio Code ➔ extensão: Live Server

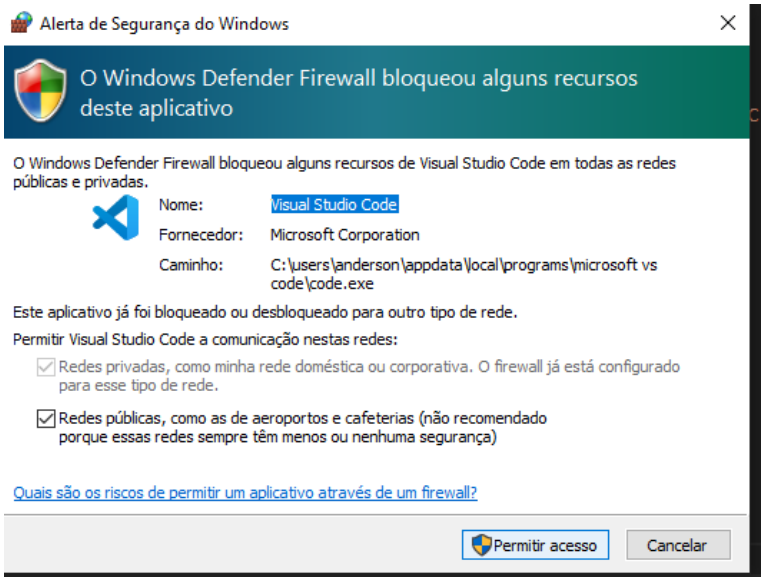


Crie uma pasta para os projetos com React e um arquivo inicial index.html.

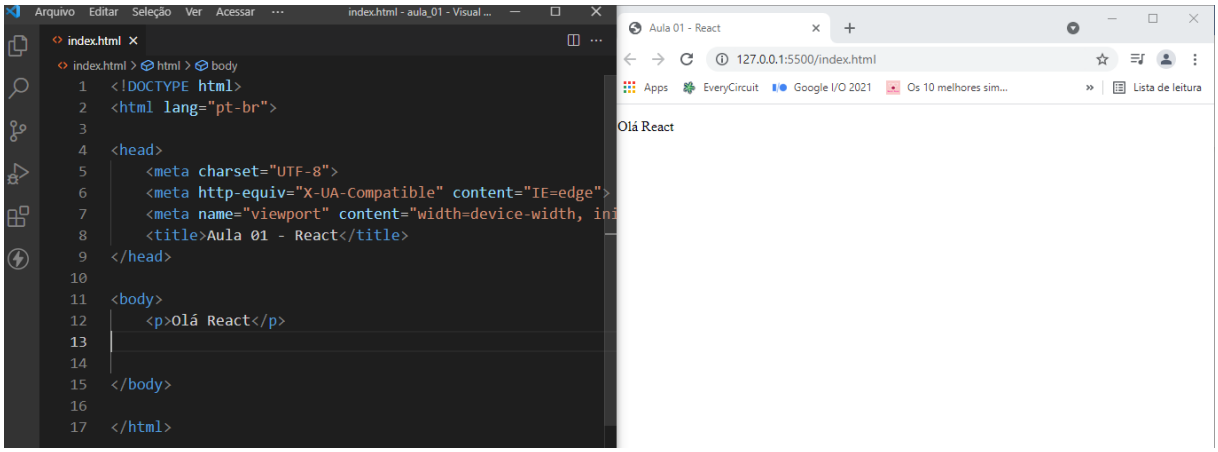


Tecle CTRL+SHIFT+P



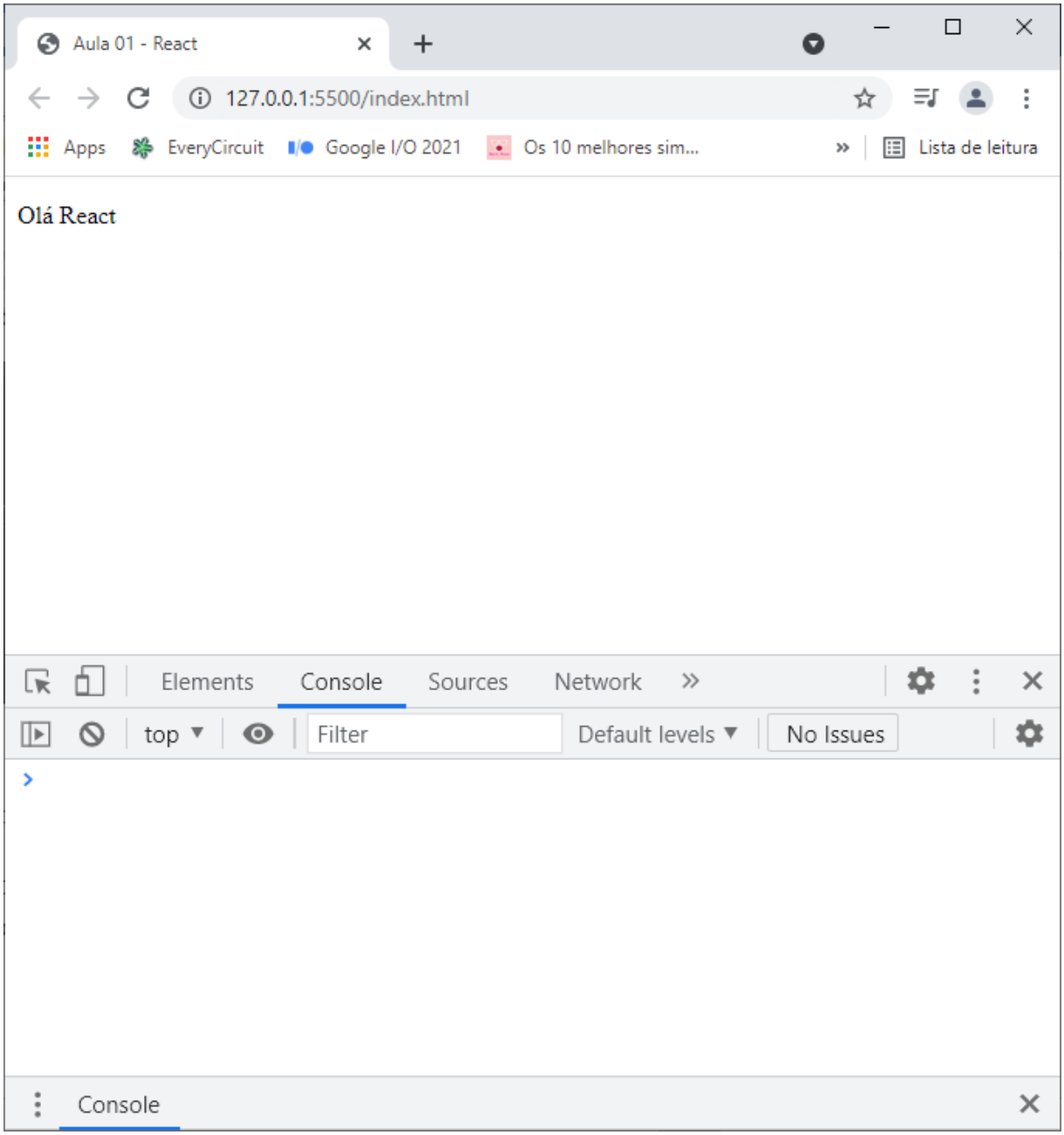


Agora todas as alterações no código serão vistas em tempo real no navegador.



2. PRIMEIRO EXEMPLO COM REACT.JS E JSX

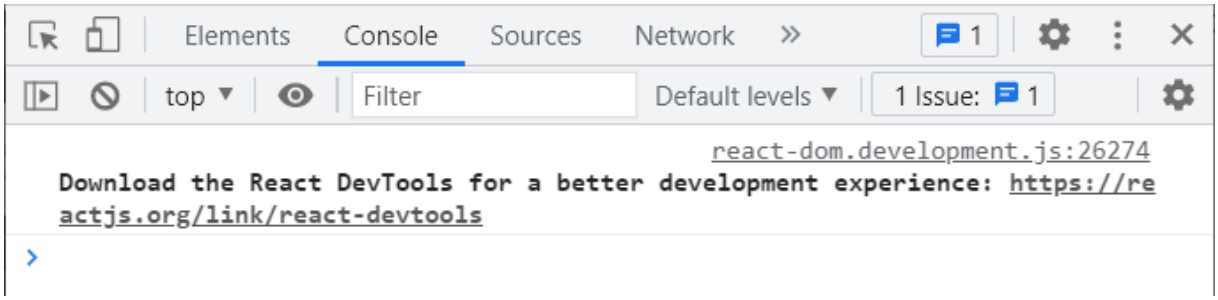
Antes de iniciarmos, deixe aberto em sua janela do navegador o console para verificarmos as mensagens e debugs do código.



Vamos utilizar as CDNs de desenvolvimento.

```
11  <body>
12      <p>Olá React</p>
13
14      <script crossorigin src="https://unpkg.com/react@17/umd/react.development.js"></script>
15      <script crossorigin src="https://unpkg.com/react-dom@17/umd/react-dom.development.js"></script>
16  </body>
```

Repare que o console já identifica o uso de React.



Insira um div com id=”root” que será a raiz de toda a nossa aplicação.

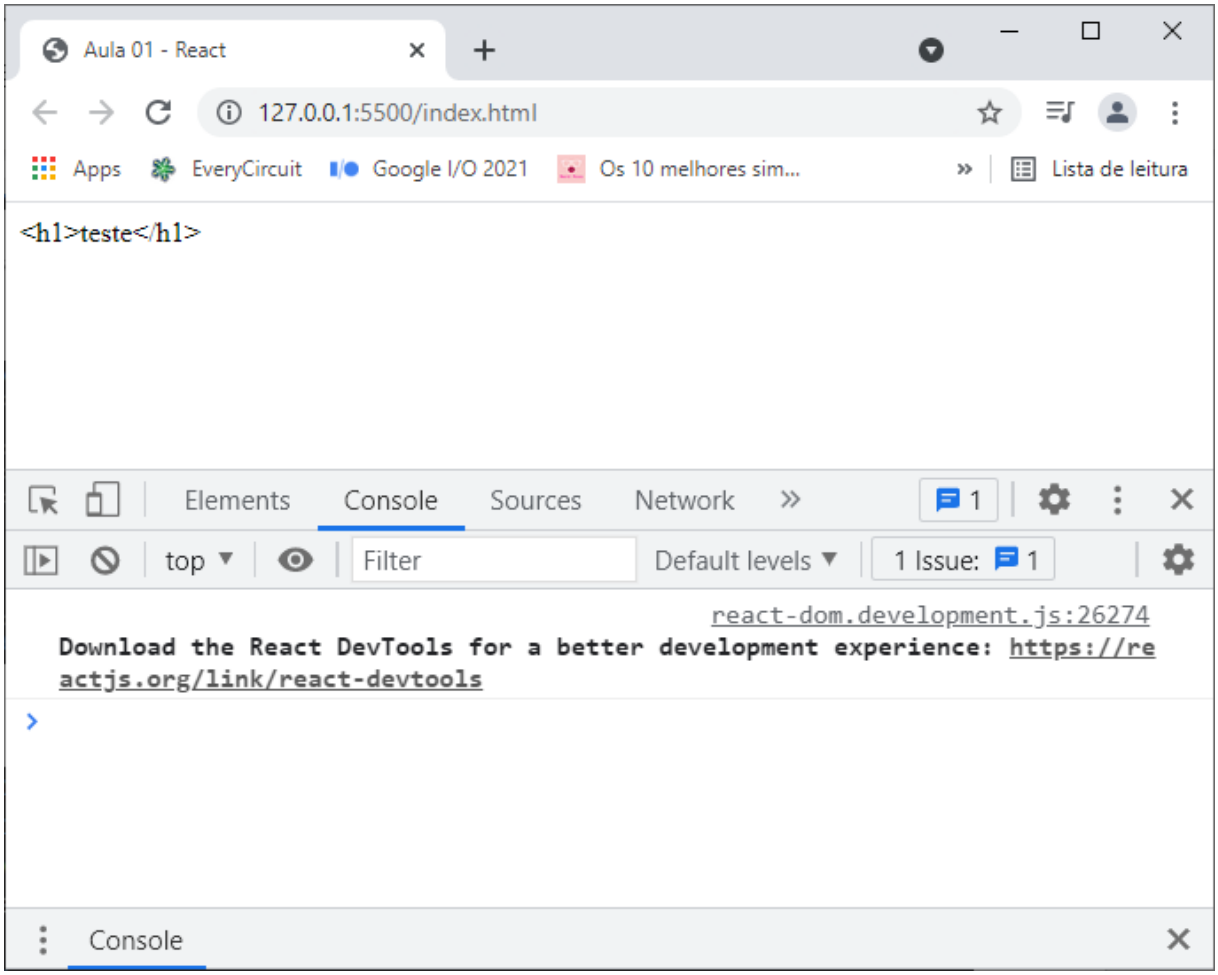
```
<body>
    <div id="root"></div>

    <script crossorigin src="https://unpkg.com/react@17/umd/react.development.js"></script>
    <script crossorigin src="https://unpkg.com/react-dom@17/umd/react-dom.development.js"></script>
</body>
```

Vamos executar um pequeno teste criando um script básico que irá chamar o React e inserir uma string na div.

```
<body>
    <div id="root"></div>

    <script crossorigin src="https://unpkg.com/react@17/umd/react.development.js"></script>
    <script crossorigin src="https://unpkg.com/react-dom@17/umd/react-dom.development.js"></script>
    <script>
        const root = document.getElementById('root');
        //A seguir uma expressão do react para apresentar conteúdo
        ReactDOM.render('teste', root)

    </script>
</body>
```

Altere a string e veja que o conteúdo não é renderizado como html.

```
<script>
    const root = document.getElementById('root');
    //A seguir uma expressão do react para apresentar conteúdo
    ReactDOM.render('<h1>teste</h1>', root)

</script>
```



Para renderizar o html na string, vamos utilizar o JSX. Precisamos inserir a CDN para o Babel.

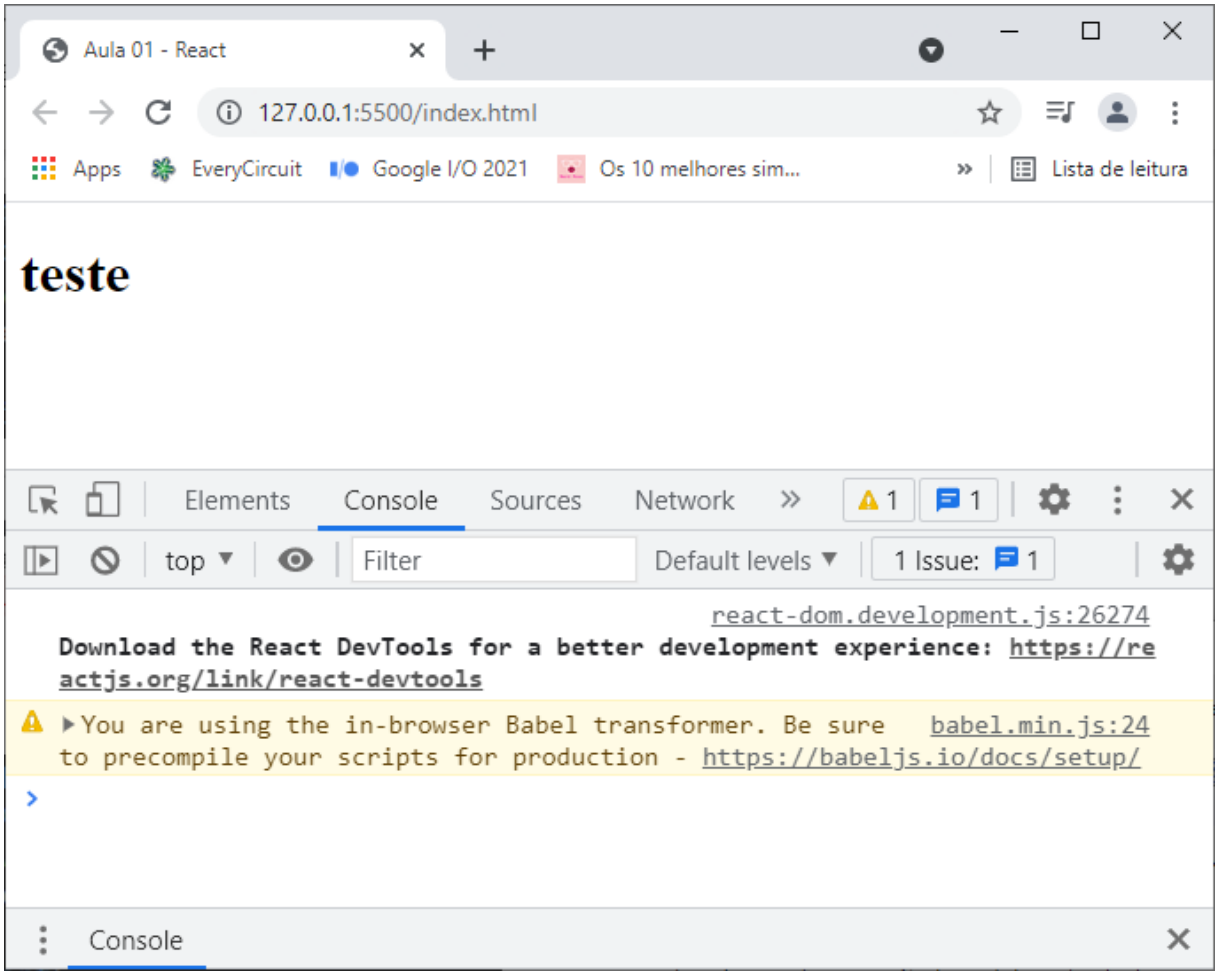
```
<body>
    <div id="root"></div>

    <script crossorigin src="https://unpkg.com/react@17/umd/react.development.js"></script>
    <script crossorigin src="https://unpkg.com/react-dom@17/umd/react-dom.development.js"></script>
    <script src="https://unpkg.com/babel-standalone@6/babel.min.js"></script>
    <script>
        const root = document.getElementById('root');
        //A seguir uma expressão do react para apresentar conteúdo
        ReactDOM.render('<h1>teste</h1>', root)

    </script>
</body>
```

Retire as aspas da string e veja que ainda o problema persiste.

```
<script>
    const root = document.getElementById('root');
    //A seguir uma expressão do react para apresentar conteúdo
    ReactDOM.render(<h1>teste</h1>, root)

</script>
```

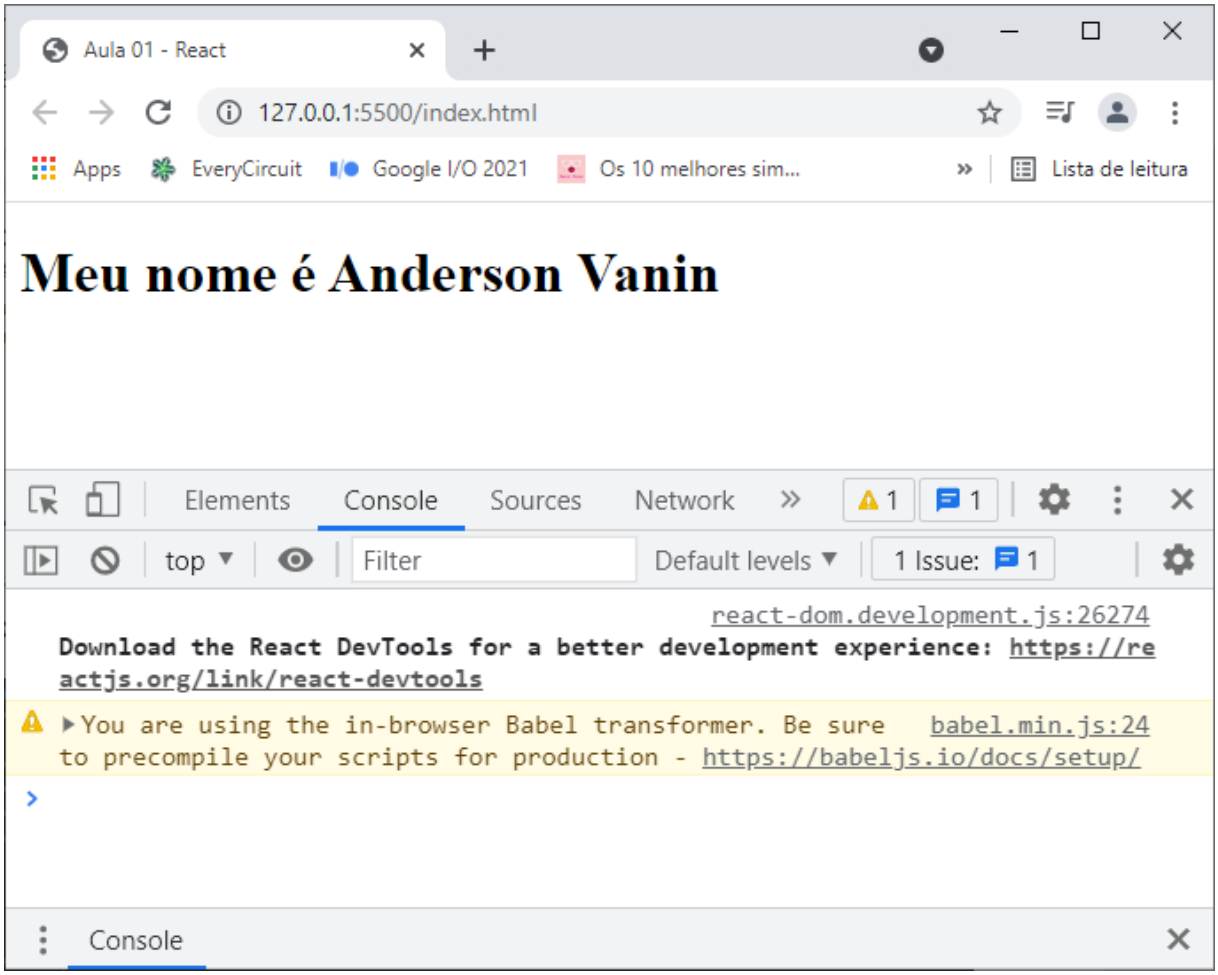
Altere a tag script

```
<script type="text/babel">
    const root = document.getElementById('root');
    //A seguir uma expressão do react para apresentar conteúdo
    ReactDOM.render(<h1>teste</h1>, root)

</script>
```



Para complementar este exemplo vamos ver como funciona a passagem de valores de uma variável para a exibição na tela pelo DOM.

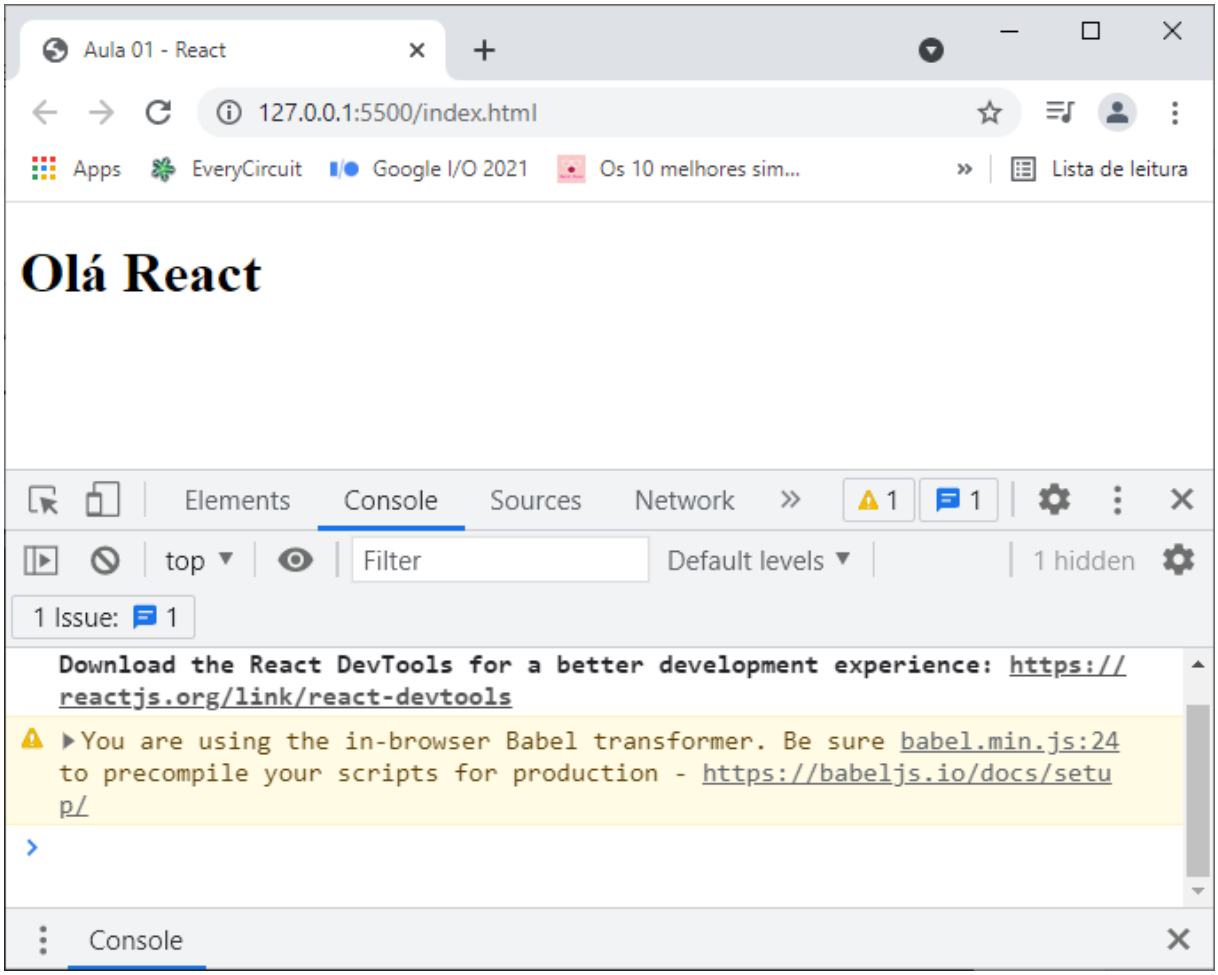
```
11  <body>
12      <div id="root"></div>
13
14      <script crossorigin src="https://unpkg.com/react@17/umd/react.development.js"></script>
15      <script crossorigin src="https://unpkg.com/react-dom@17/umd/react-dom.development.js"></script>
16      <script src="https://unpkg.com/babel-standalone@6/babel.min.js"></script>
17      <script type="text/babel">
18          const root = document.getElementById('root');
19          let nome = 'Anderson Vanin';
20          //A seguir uma expressão do react para apresentar conteúdo
21          ReactDOM.render(<h1>Meu nome é {nome}</h1>, root);
22      </script>
23  </body>
```



3. ENTENDENDO SOBRE FUNCTION COMPONENT

Criação de componentes

```
<body>
    <div id="root"></div>

    <script crossorigin src="https://unpkg.com/react@17/umd/react.development.js"></script>
    <script crossorigin src="https://unpkg.com/react-dom@17/umd/react-dom.development.js"></script>
    <script src="https://unpkg.com/babel-standalone@6/babel.min.js"></script>
    <script type="text/babel">
        const root = document.getElementById('root');
        function MeuComponente() {
            return (
                <div>
                    <h1>Olá React</h1>
                </div>
            );
        }

        ReactDOM.render(<MeuComponente />, root);
    </script>
</body>
```



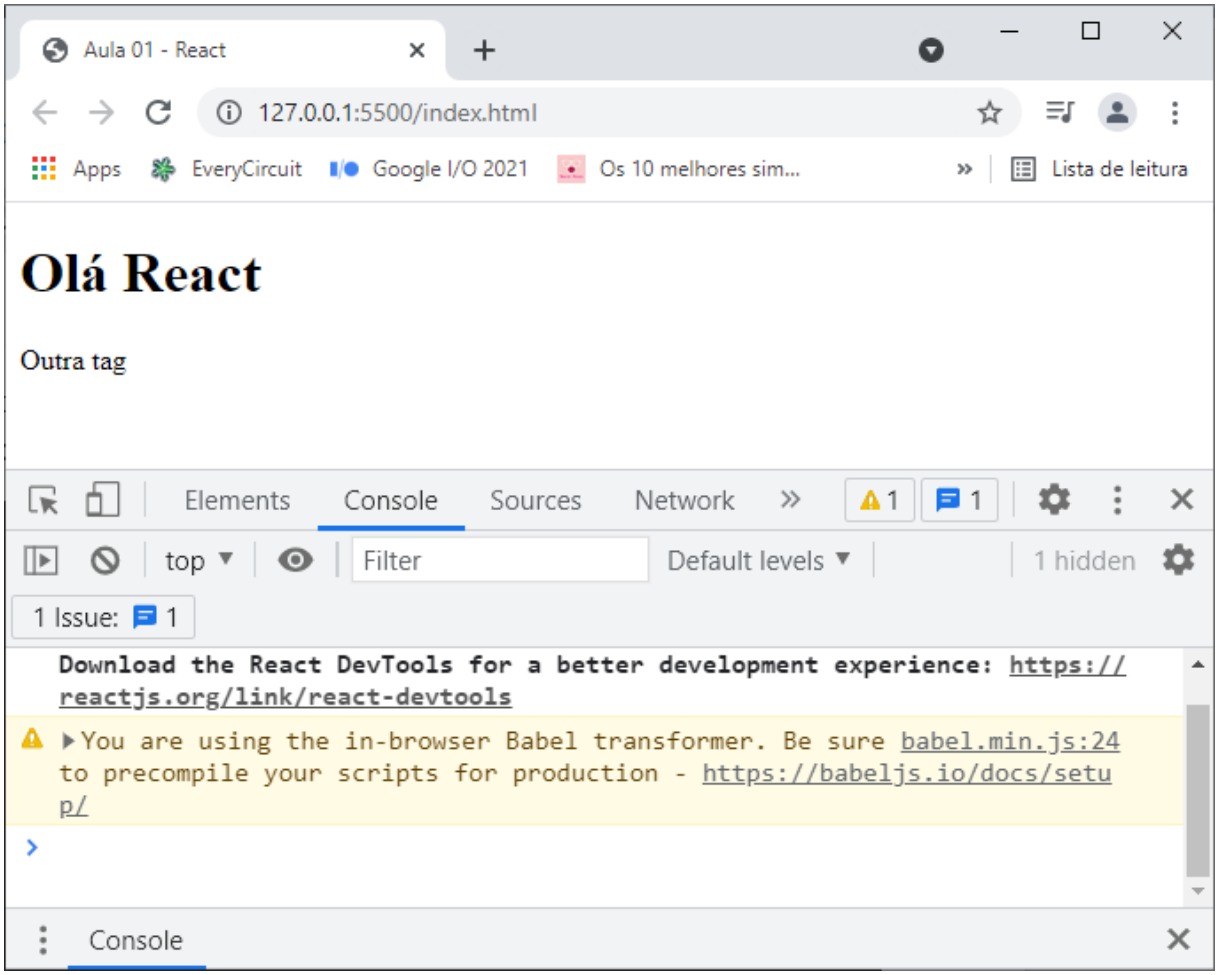
Veja o que ocorre quando tentamos renderizar mais uma tag html removendo a div dentro da função.

```
17      <script type="text/babel">
18          const root = document.getElementById('root');
19          function MeuComponente() {
20              return (
21
22                  <h1>Olá React</h1>
23                  <p>Outra tag</p>
24
25              );
26          }
27
28          ReactDOM.render(<MeuComponente />, root);
29      </script>
```

```
Uncaught SyntaxError: Inline Babel script: Adjacent JSX elements must be wrapped in an enclosing tag (7:20)
  5 |
  6 |                     <h1>Olá React</h1>
> 7 |                     <p>Outra tag</p>
    |                     ^
  8 |
```

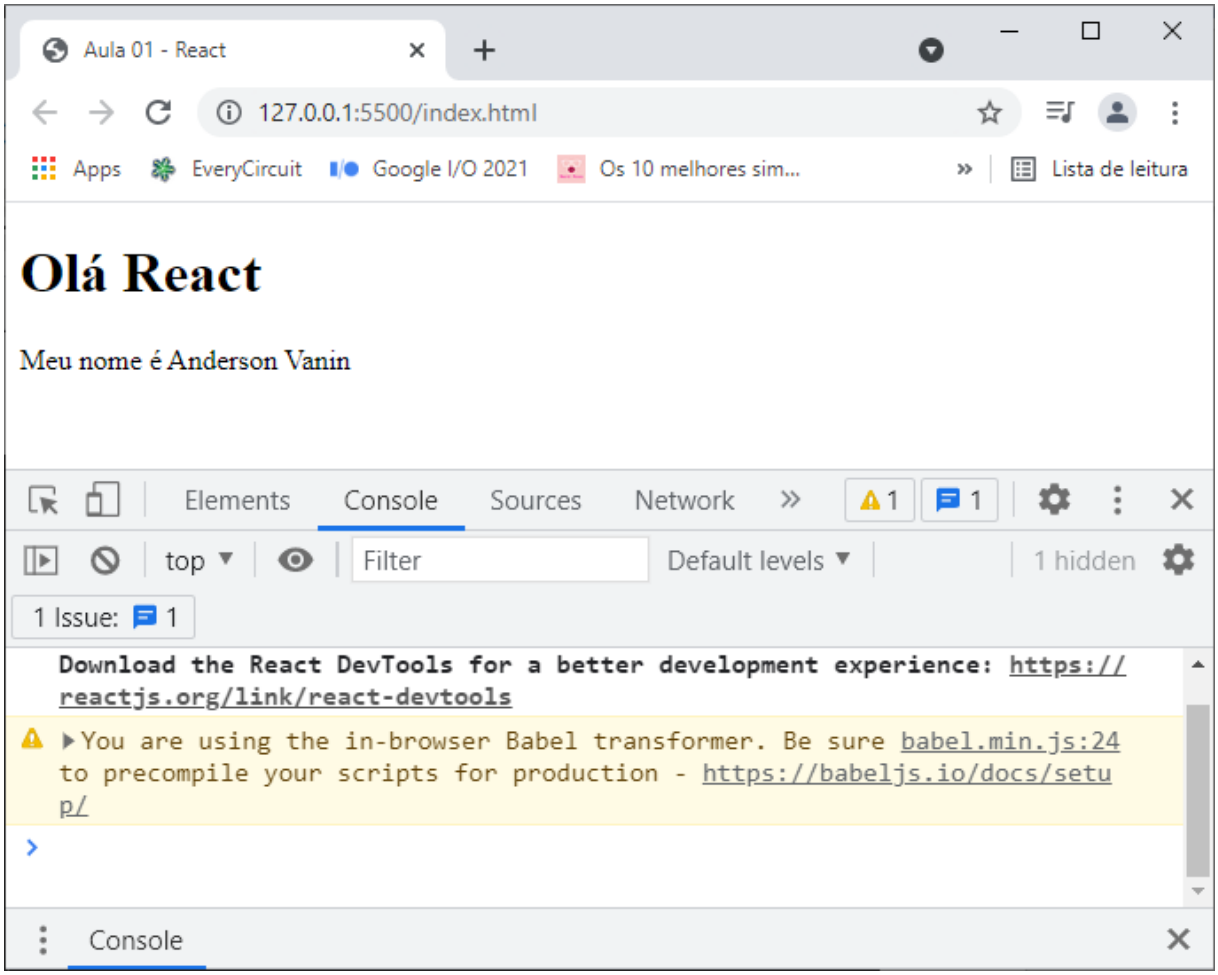
Portanto é necessário que todos os elementos a serem renderizados estejam contidos dentro de um único elemento.

```
<script type="text/babel">
    const root = document.getElementById('root');
    function MeuComponente() {
        return (
            <div>
                <h1>Olá React</h1>
                <p>Outra tag</p>
            </div>
        );
    }

    ReactDOM.render(<MeuComponente />, root);
</script>
```



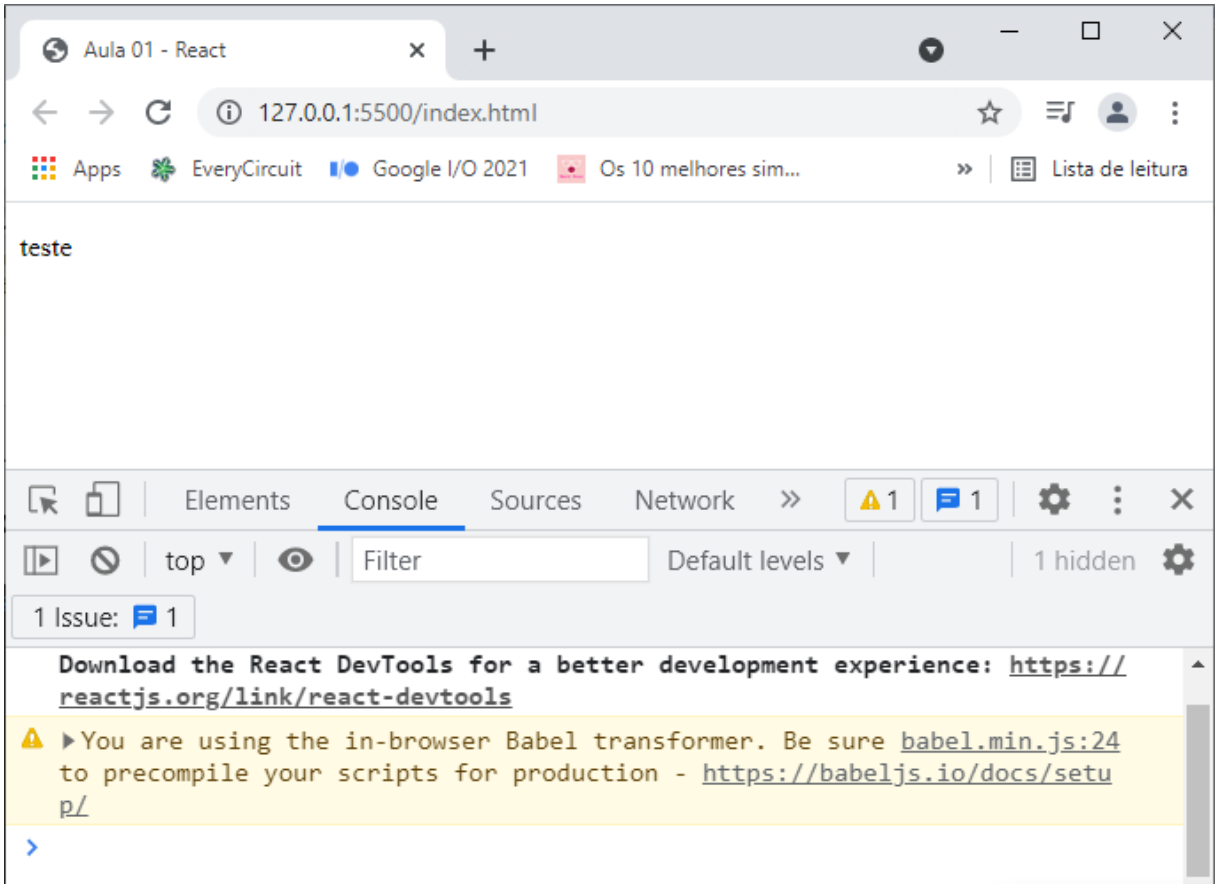
4. VARIÁVEIS INTERNAS DE UMA FUNÇÃO

```
17      <script type="text/babel">
18          const root = document.getElementById('root');
19          function MeuComponente() {
20              var nome = 'Anderson Vanin';
21              return (
22                  <div>
23                      <h1>Olá React</h1>
24                      <p>Meu nome é {nome}</p>
25                  </div>
26              );
27          }
28
29          ReactDOM.render(<MeuComponente />, root);
30      </script>
```



5. EXEMPLO DE UM CLASS COMPONENT

```
17      <script type="text/babel">
18          const root = document.getElementById('root');
19
20          class MeuComponente extends React.Component {
21              render() {
22                  return (
23                      <div>
24                          <p>teste</p>
25                      </div>
26                  )
27              }
28          }
29
30          ReactDOM.render(<MeuComponente />, root);
31      </script>
```

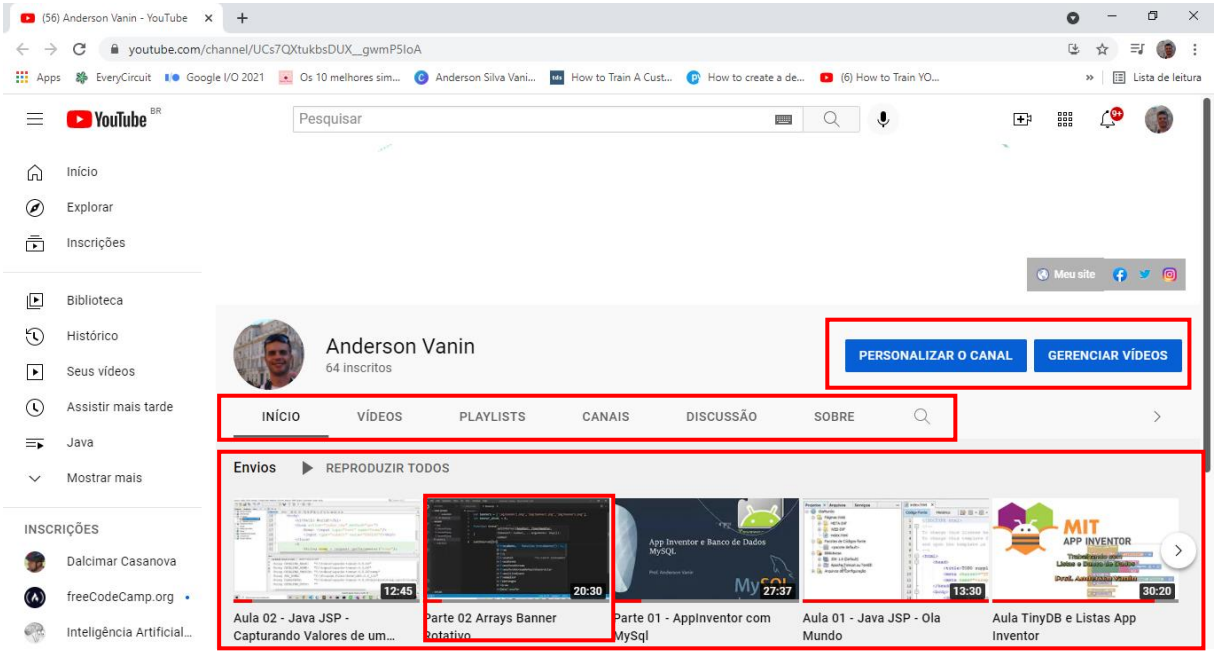


6. INCORPORAR UM COMPONENTE DENTRO DE OUTRO COMPONENTE

Para exemplificar, vamos criar um componente A e um B, e incorporá-los dentro de um componente C.

```
<script type="text/babel">
    const root = document.getElementById('root');

    class CompA extends React.Component {
        render() {
            return (
                <h1>Componente A</h1>
            )
        }
    }
    class CompB extends React.Component {
        render() {
            return (
                <h1>Componente B</h1>
            )
        }
    }
    class MeuComponente extends React.Component {
        render() {
            return (
                <div>
                    <CompA />
                    <hr />
                    <CompB />
                </div>
            )
        }
    }

    ReactDOM.render(<MeuComponente />, root);
</script>
```

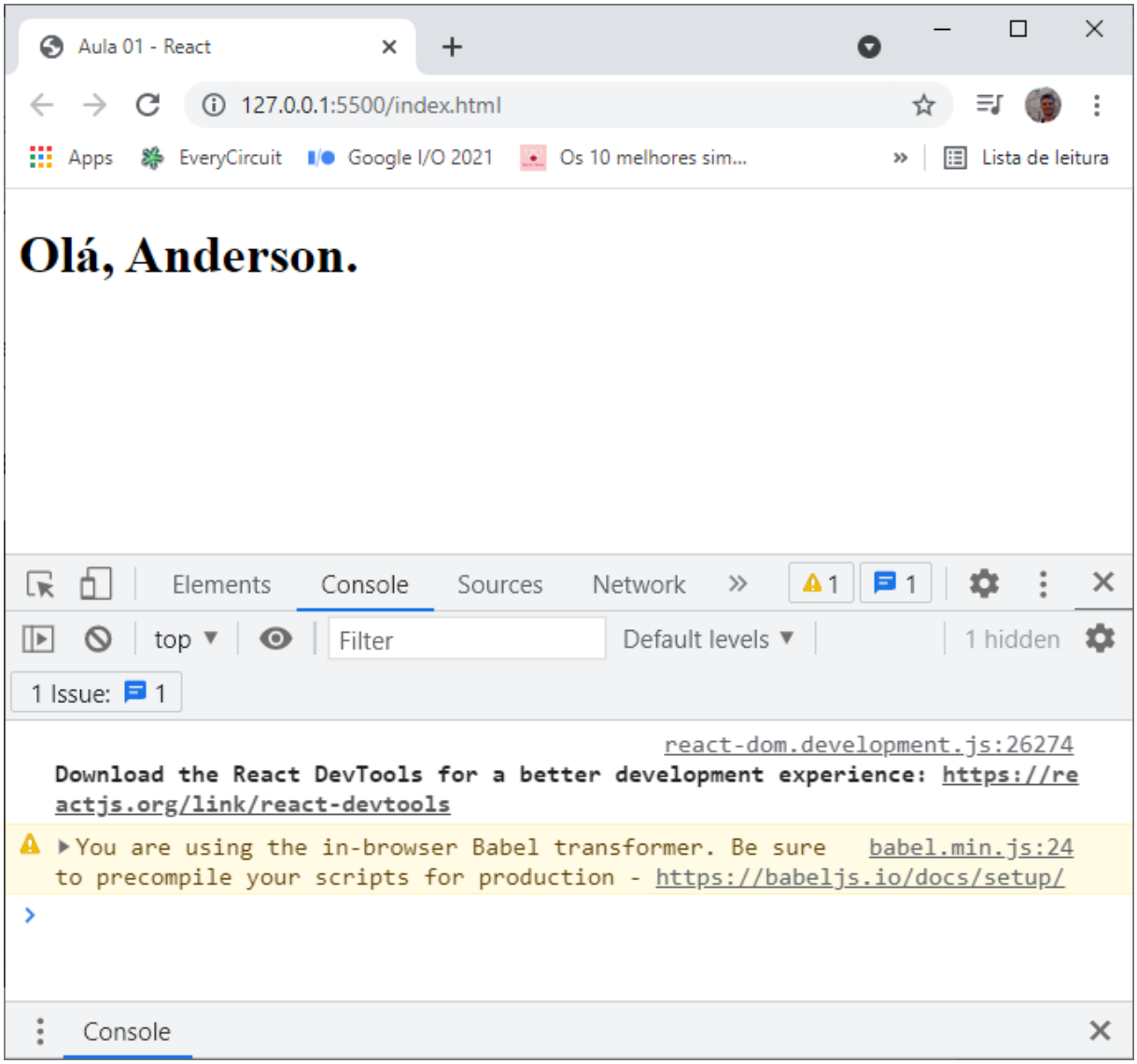
De forma geral para o entendimento de onde e como se utilizam estes conceitos, considere a seguinte página:



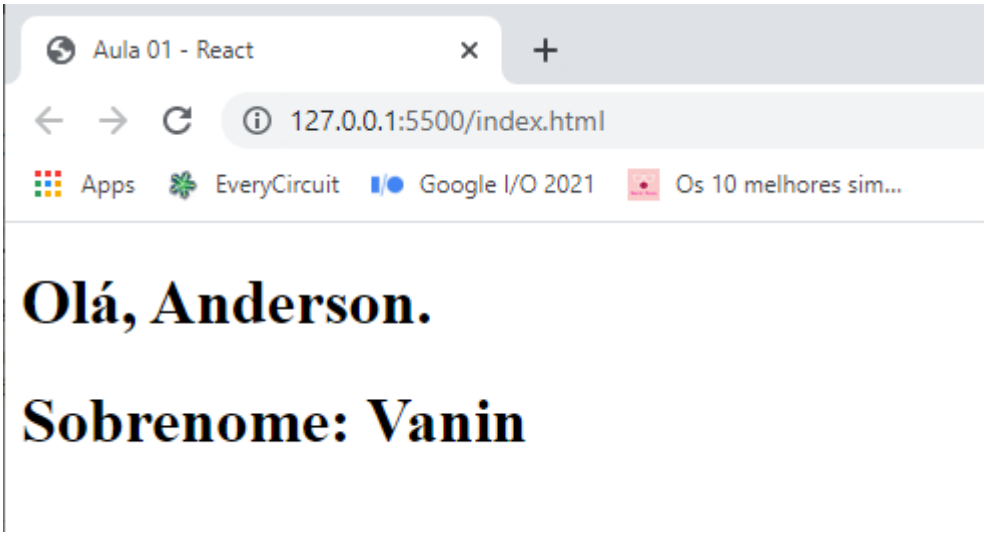
Considere cada área realçada em vermelho como sendo um componente e que é possível termos um componente dentro de outro.

7. COMPONENT PROPS

```
17    <script type="text/babel">
18        function CompA(props) {
19            return (
20                <div>
21                    <h1>Olá, {props.nome}.</h1>
22                </div>
23            )
24        }
25
26        const root = document.getElementById('root');
27
28        ReactDOM.render(<CompA nome='Anderson' />, root);
29    </script>
```




```
<script type="text/babel">
    function CompA(props) {
        return (
            <div>
                <h1>Olá, {props.nome}.</h1>
                <h1>Sobrenome: {props.sobrenome}</h1>
            </div>
        )
    }

    const root = document.getElementById('root');

    ReactDOM.render(<CompA nome='Anderson' sobrenome='Vanin' />, root);
</script>
```



Vamos ver agora o mesmo exemplo utilizando uma classe

```
class CompB extends React.Component {
    render() {
        return (
            <div>
                <p>Olá novamente, {this.props.nome}</p>
            </div>
        )
    }
}

const root = document.getElementById('root');

ReactDOM.render(<CompB nome='Anderson' sobrenome='Vanin' />, root);
```