React

* Criado pelo **facebook**;
* **Biblioteca JS** para criação de interfaces de usuário;
* Utilizado para construir **SPAs** (Single Page Application);
* Baseado em **componentes**;
* Utiliza o **JSX** para renderizar HTML;
* Aplica o **Virtual DOM** para realizar as alterações de Dom;
* React só atualiza os componentes **que sofreram alterados**.

Criação de um arquivo React:

1. Instalar ***Node.Js***
2. Abrir ***Prompt de Comando*** (Modo administrativo)
3. Digitar ***cd (colar o endereço da pasta)***
4. Digitar ***npx create-react-app (nome do app)***

Processo de criação do app será iniciado.

Execução de um arquivo React:

1. Abrir ***Prompt de Comando*** (Modo administrativo)
2. Digitar ***cd (colar o endereço da pasta)***
3. Digitar ***cd (nome do app)***
4. Digitar ***npm start***

O app React será aberto no navegador.

Entendendo o JSX:

* O **JSX** é como um HTML, porém dentro do código Javascript;
* É a principal maneira de escrever HTML com o REACT;
* Podemos **interpolar variáveis**, inserindo ela entre { };
* É possível também **executar funções** em JSX;
* Inserir **valores em atributos de tags** também é válido em JSX.

**Estrutura básica do React:**

**Entrada:**

import React from 'react'

export default function App(){

  return(

    <>

      <div>

        <h1>Olá Mundo!</h1>

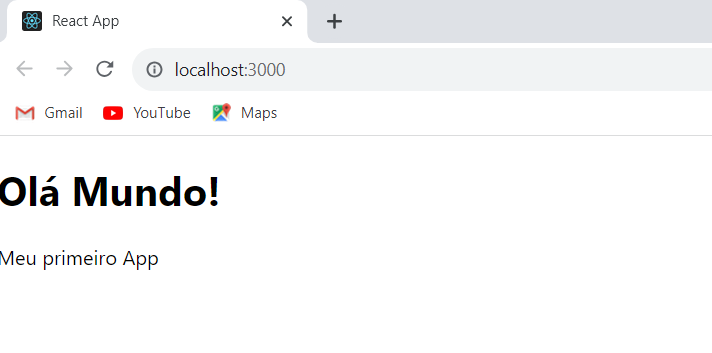
        <p>Meu primeiro App</p>

      </div>

    </>

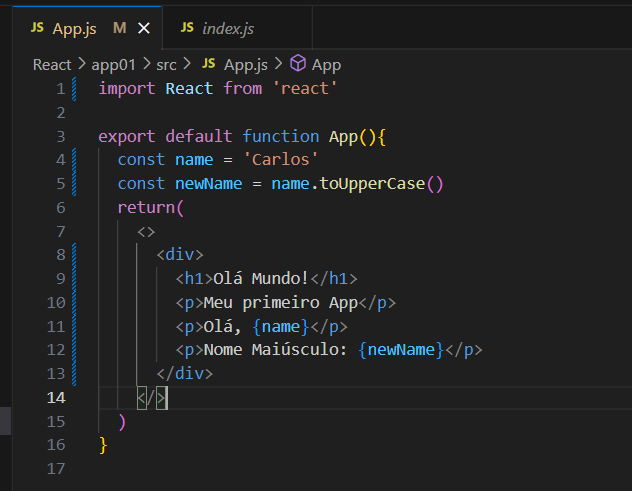
  )

}

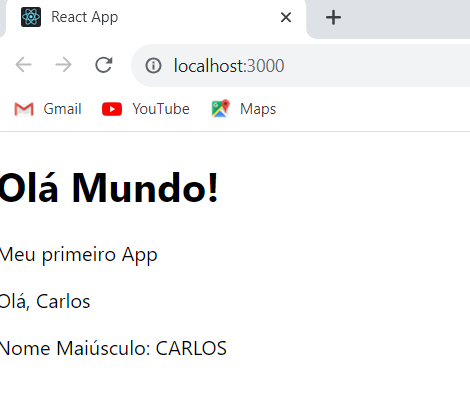
  
**Saída:**

**Interpolação de variáveis:**

**Entrada:**

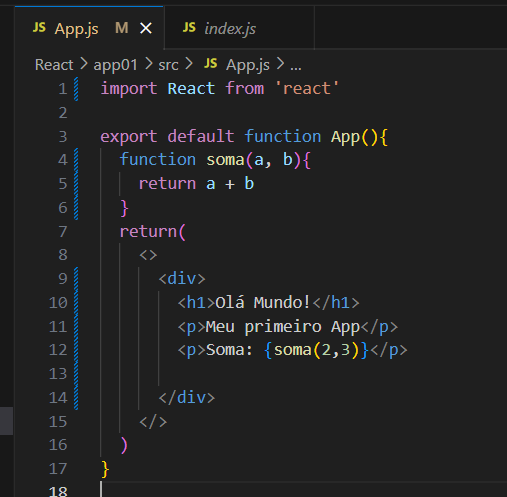
****

**Saída:**

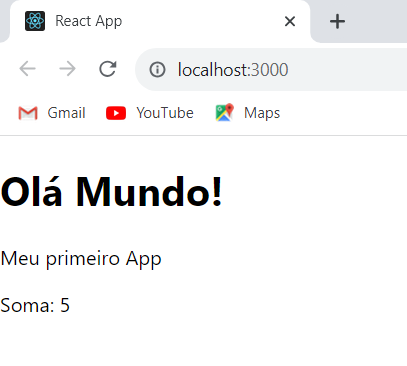
****

**Interpolação de funções:**

**Entrada:**

****

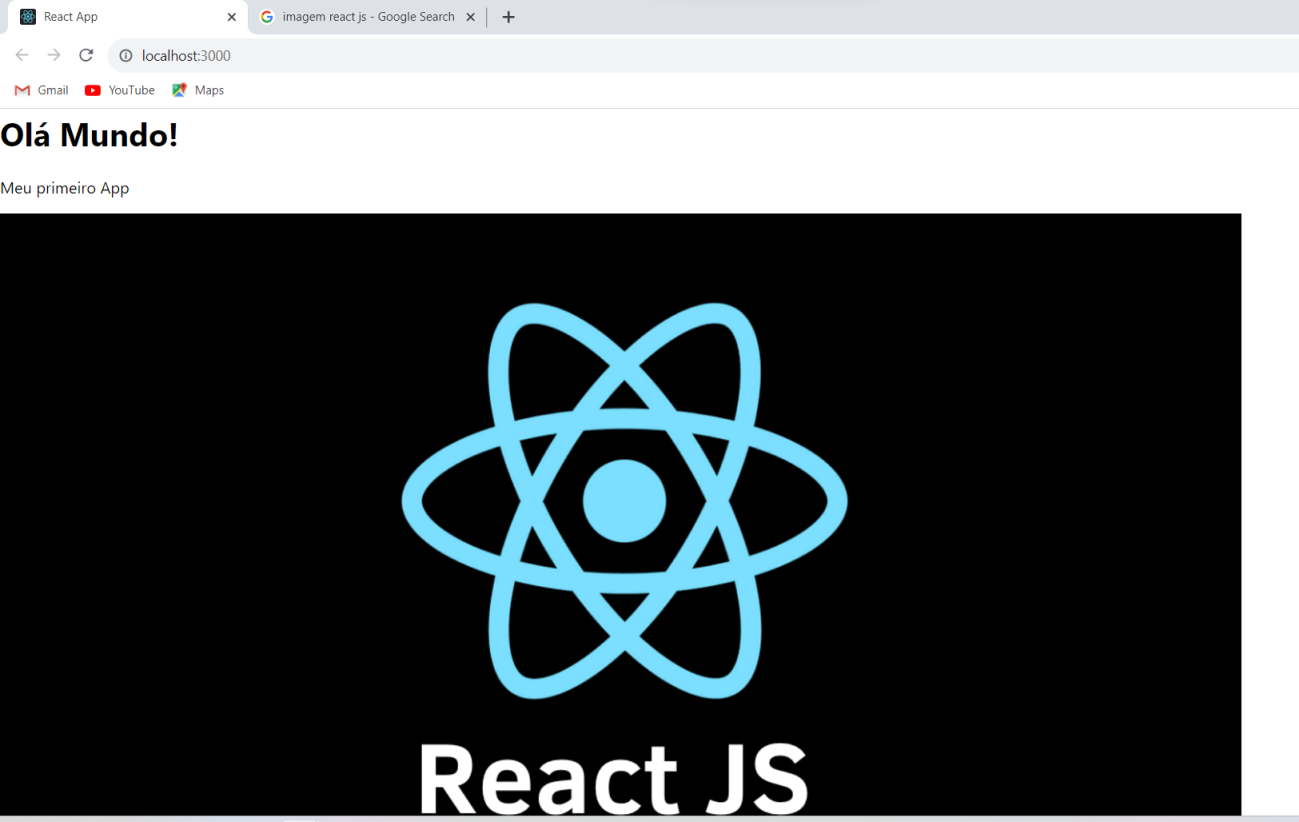
**Saída:**



**Inserindo Imagens da internet:**

**Entrada:**

**Saída:**

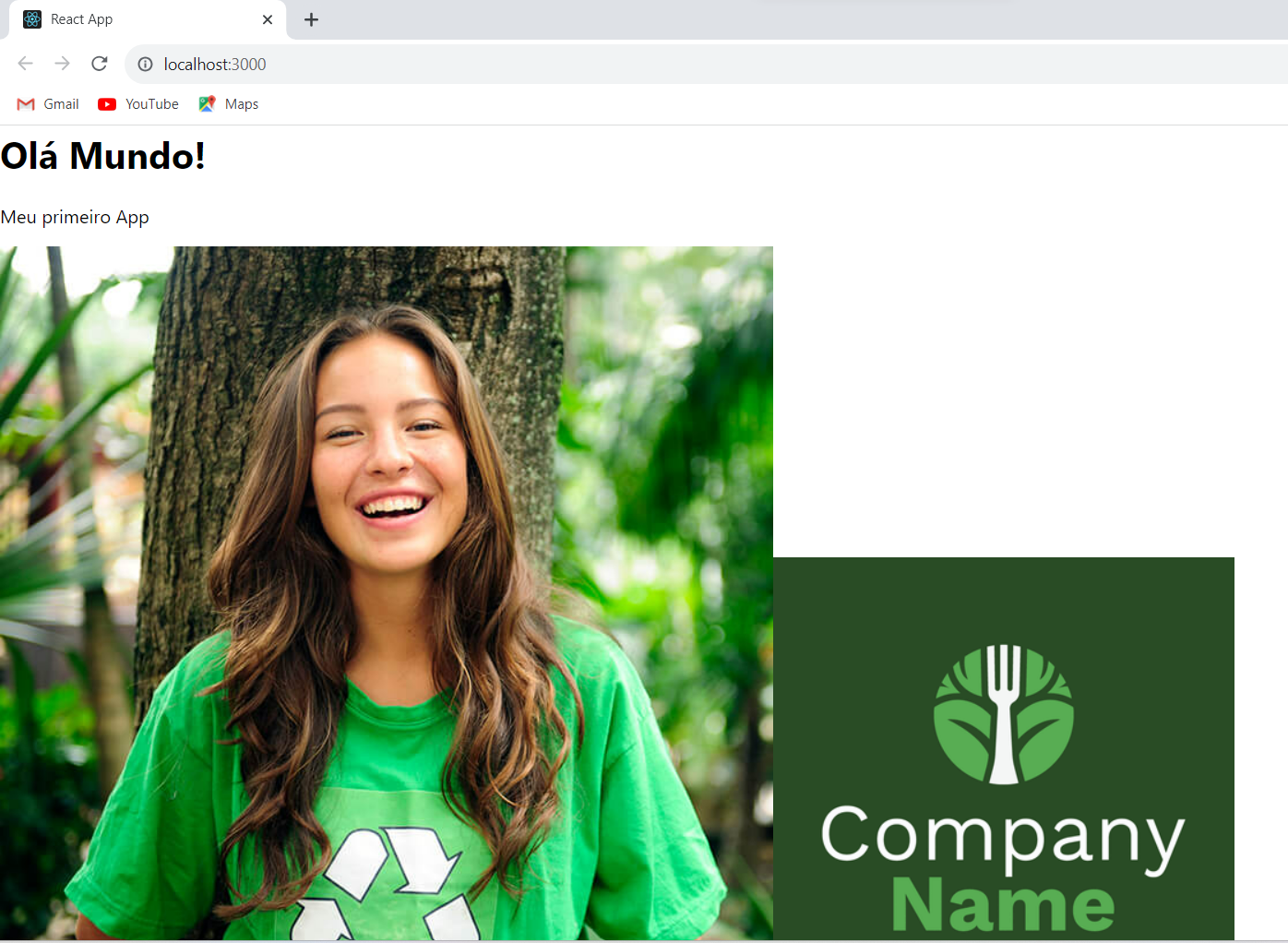


**Inserindo Imagens do computador:**

**Entrada:**

****

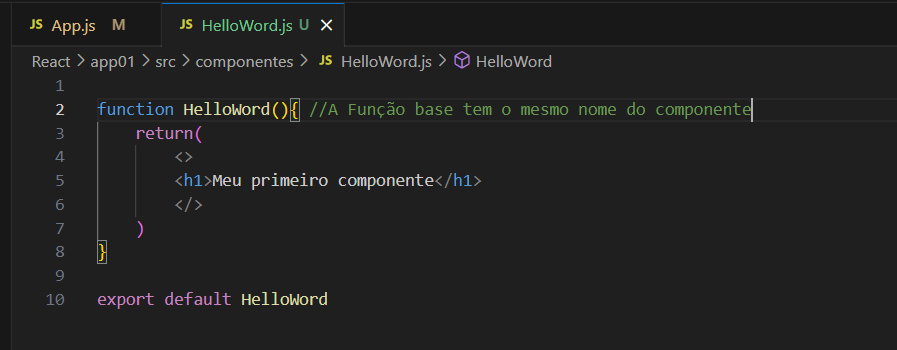
**Saída:**

****

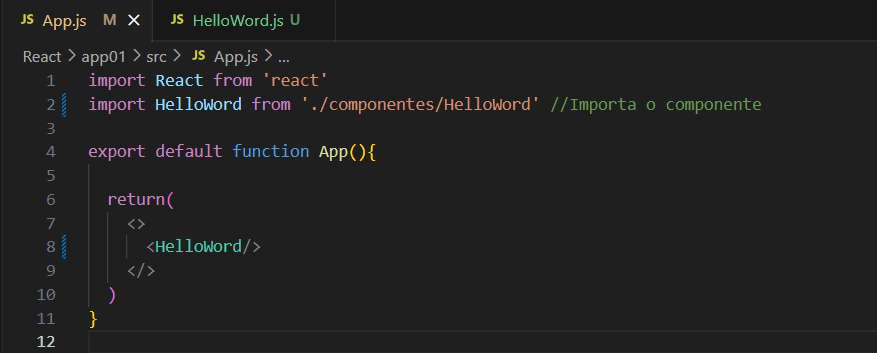
Criando Componentes:

* Permite **dividir a aplicação** em partes;
* Os componentes **renderizam JSX**, assim como App.js (que é um componente);
* Precisamos **criar um arquivo** de componente;
* E **importá-lo** onde precisamos utilizar;
* Normalmente fica em uma pasta chamada **components**.

Estrutura Básica de um componente:



Importação e Utilização do componente:



**Importando um componente dentro do outro:**  
Criar um terceiro componente:

function Frase(){

    return(

        <>

            <p>Este é um componente com uma frase!</p>

        </>

    )

}

export default Frase

Vamos importar o componente **FRASE** ao componente **HelloWord**:

import Frase from './Frase'

function HelloWord(){ //A Função base tem o mesmo nome do componente

    return(

        <>

            <h1>Meu primeiro componente</h1>

            <Frase/>

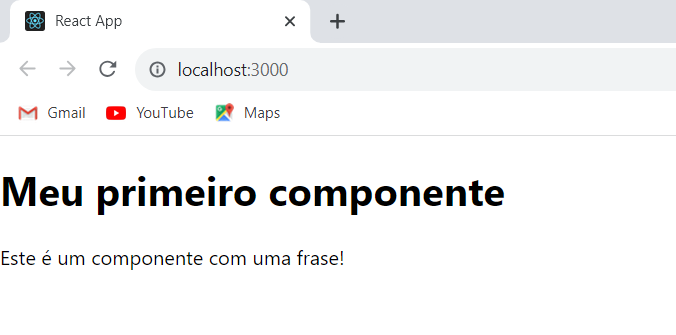
        </>

    )

}

export default HelloWord

**Saída:**



Props:

* As **props** são valores passados para componentes;
* Podemos deixá-los **dinâmicos**;
* Ou seja, **mudando a execução** por causa do valor da prop;
* O valor é **passado como um atributo** na chamada do componente;
* E precisa ser **resgatado dentro de uma propriedade/argumento** chamada props na função de definição do componente;
* As props são **somente de leitura**.

Vamos criar um novo componente que irá receber um valor **props**.

function SayMyName(props){ //Irá receber um valor

    return(

        <div>

            <p>Fala ai {props.nome}, suave?</p>

        </div>

    )

}

export default SayMyName

Fazendo o importe no componente principal, iremos passar o valor pelo props:

import React from 'react'

import SayMyName from './componentes/SayMyName'

export default function App(){

  return(

    <>

      <SayMyName nome='Carlos'/> //Passa o valor

    </>

  )

}

Também podemos passar os valores de forma dinâmica:

import React from 'react'

import SayMyName from './componentes/SayMyName'

export default function App(){

  const nome = 'Maria'

  return(

    <>

      <SayMyName nome='Carlos'/>

      <SayMyName nome='João'/>

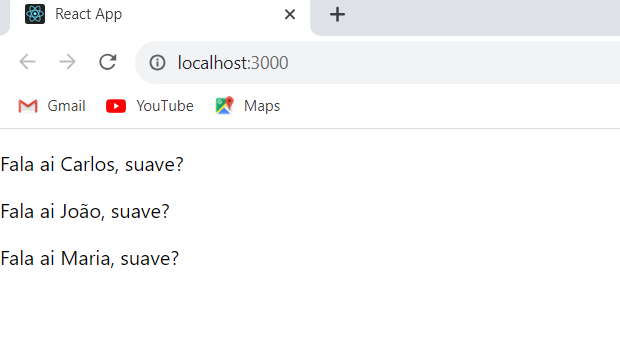
      <SayMyName nome={nome}/>

    </>

  )

}

Saída:

Vamos criar um componente com mais valores props:

function Pessoa(){

    return(

        <div>

            <img src={props.foto} alt="Imagem da pessoa"/>

            <h2>Nome: {props.nome}</h2>

            <p>Idade: {props.idade}</p>

            <p>Profissão: {props.profissao}</p>

        </div>

    )

}

export default Pessoa

Importação no componente principal:

import React from 'react'

import Pessoa from './componentes/Pessoa'

export default function App(){

  return(

    <div>

      <Pessoa

        nome="Carlos"

        idade="28"

        profissao="Programador"

        foto="https://via.placeholder.com/150"

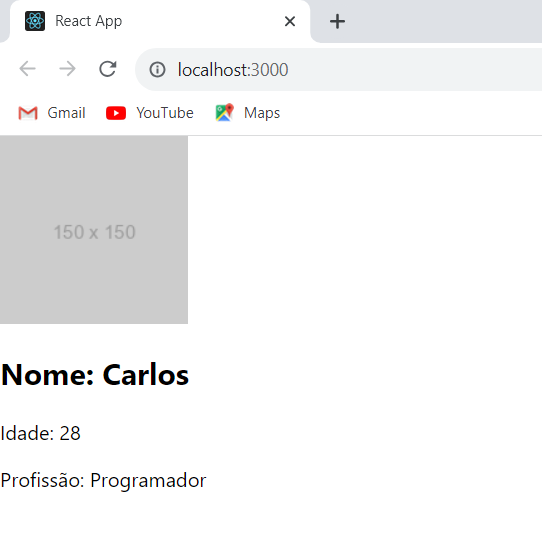
      />

    </div>

  )

}

Saída:



Para não precisarmos escrever ‘props’ várias vezes, podemos fazer: (Destructure)

function Pessoa({nome, idade, profissao, foto}){

    return(

        <div>

            <img src={foto} alt="Imagem da pessoa"/>

            <h2>Nome: {nome}</h2>

            <p>Idade: {idade}</p>

            <p>Profissão: {profissao}</p>

        </div>

    )

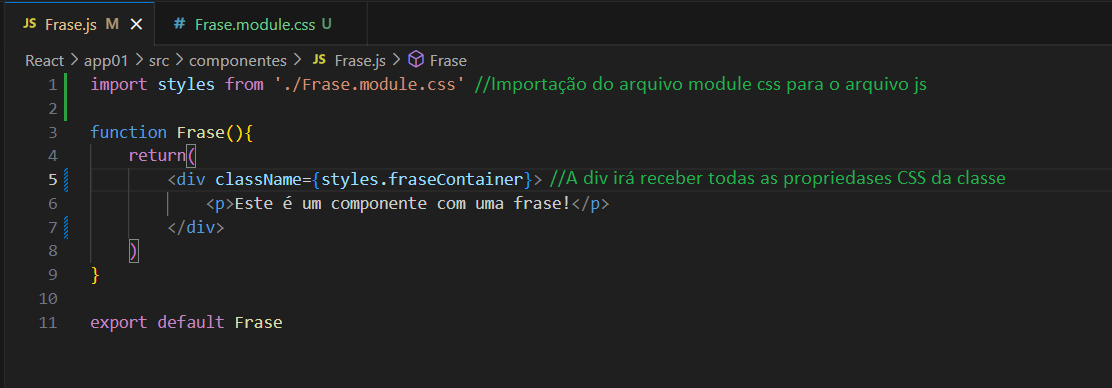
}

export default Pessoa

Adicionando CSS:

* As **CSS** pode ser **adicionado de forma global** na aplicação, por meio do arquivo índex.css por exemplo;
* Porém é possível **estilizar a nível de componentes**;
* Utilizamos o **CSS modules** para isso;
* Basta criar um arquivo como: **Componente.module.css**;
* E chamar esse CSS no componente;

A importação do CSS Modules é semelhante aos componentes.



Fragmentos:

* Os **React Fragments** permitem a criação de um componente sem elemento pai;
* O propósito é **descomplicar os nós do DOM**;
* A sintaxe é **<>** e **</>** , não há um nome para a tag;
* Criamos no próprio JSX;

Avançando em Props:

* Podemos **definir tipos para as props** ,realizando uma espécie de validação;
* Definimos em um objeto chamado **propTypes** no próprio componente;
* E ainda há a possibilidade de **definir um valor padrão**;
* Neste caso utilizamos o objeto **defaultProps**;

Vamos garantir os tipos das propriedades das props passadas pelo componente pai:

import PropTypes from 'prop-types' // Importa o pacote do React

function Item({marca, ano\_lancamento}){

    return(

        <>

            <li>{marca} - {ano\_lancamento}</li>

        </>

    )

}

Item.propTypes = { // Vai garantir que as propriedades passadas sejam do tipo 'string' e 'number' respectivamente.

    marca: PropTypes.string,

    ano\_lancamento: PropTypes.number,

}

export default Item

Podemos adicionar o comando **isRequired**:

Item.propTypes = {

    marca: PropTypes.string.isRequired, // exige que alguma informação seja passada

}

Podemos adicionar um valor padrão para caso nenhum valor seja passado. Para isso utilizamos **defaultProps**:

Item.defaultProps = {

    marca: 'Faltou a marca',

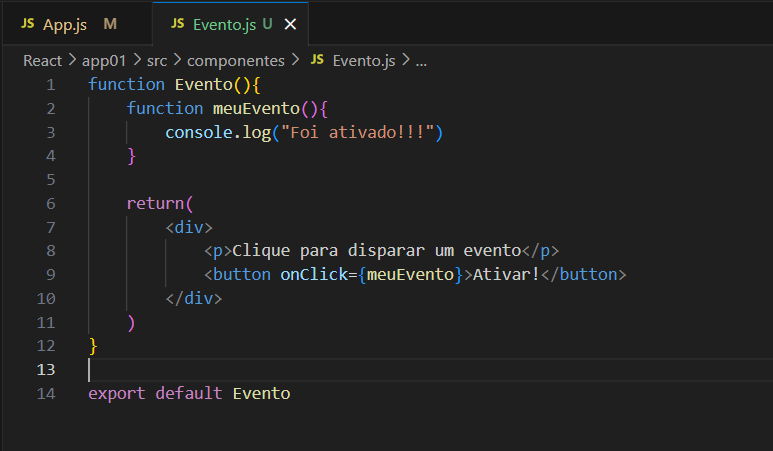
    ano\_lancamento: 0,

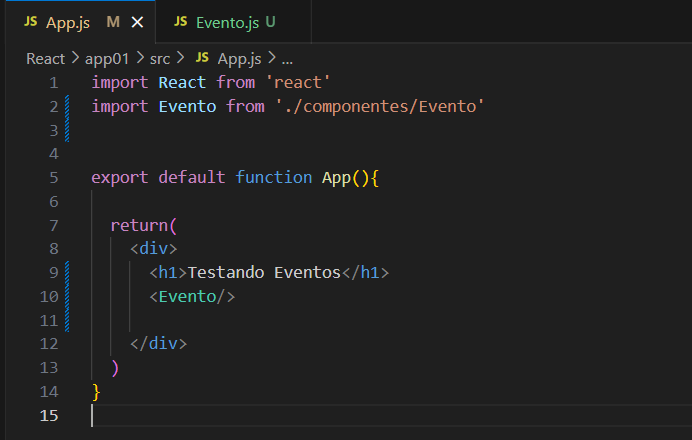
}

Eventos:

* Os **eventos de React** são os mesmos eventos do DOM;
* Ou seja, temos eventos para responder a um click;
* **O evento é atrelado a uma tag** que irá executá-lo;
* Geralmente um **método** é atribuído ao evento;
* Este método deve ser criado no componente;

Vamos criar um novo componente “Evento.js” e ver na prática com funciona:



Agora é só importar no componente pai:

Também podemos trabalhar com propriedades dentro dos eventos:

function Evento({ numero }){    //Recebe a propriedade "numero"

    function meuEvento(){

        console.log(`Foi ativado! ${numero}`)

    }

    return(

        <div>

            <p>Clique para disparar um evento</p>

            <button onClick={meuEvento}>Ativar!</button>

        </div>

    )

}

export default Evento

Componente pai:

import React from 'react'

import Evento from './componentes/Evento'

export default function App(){

  return(

    <div>

      <h1>Testando Eventos</h1>

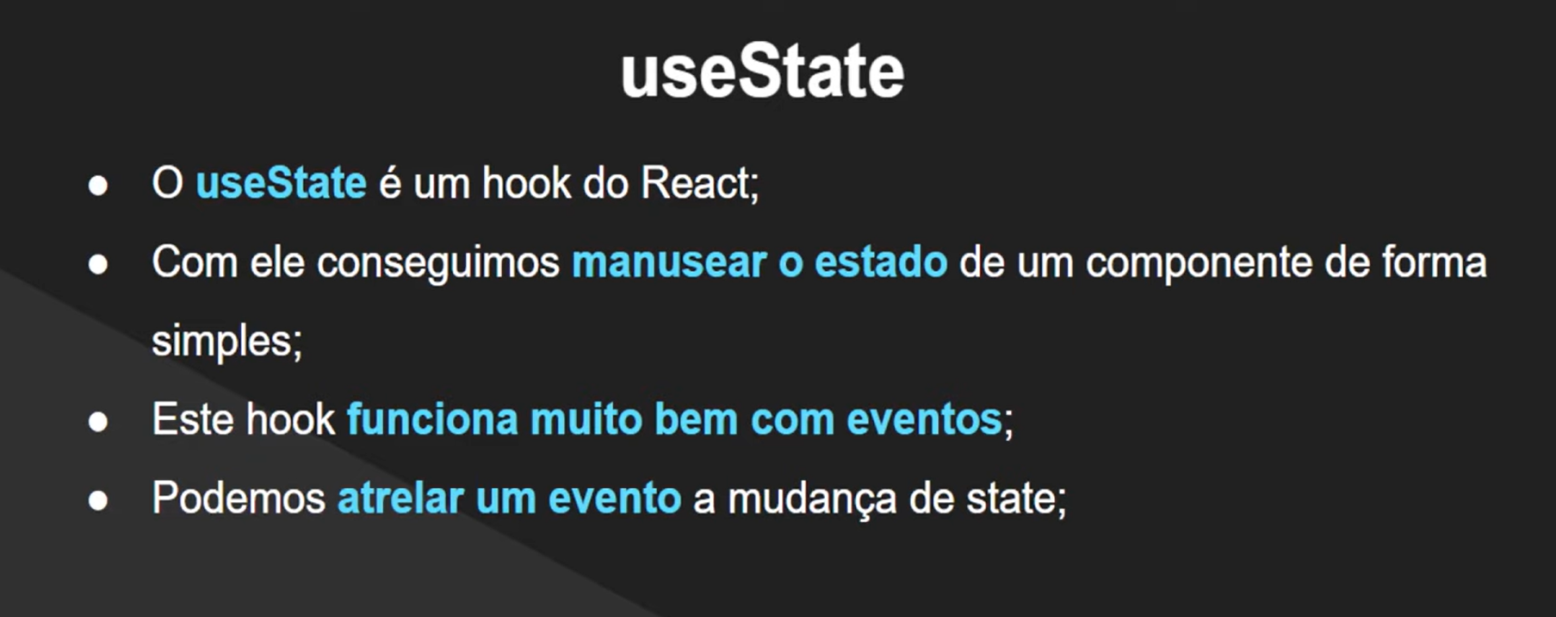
      <Evento numero="1"/> //Passa a propriedade "numero”

      <Evento numero="2"/>

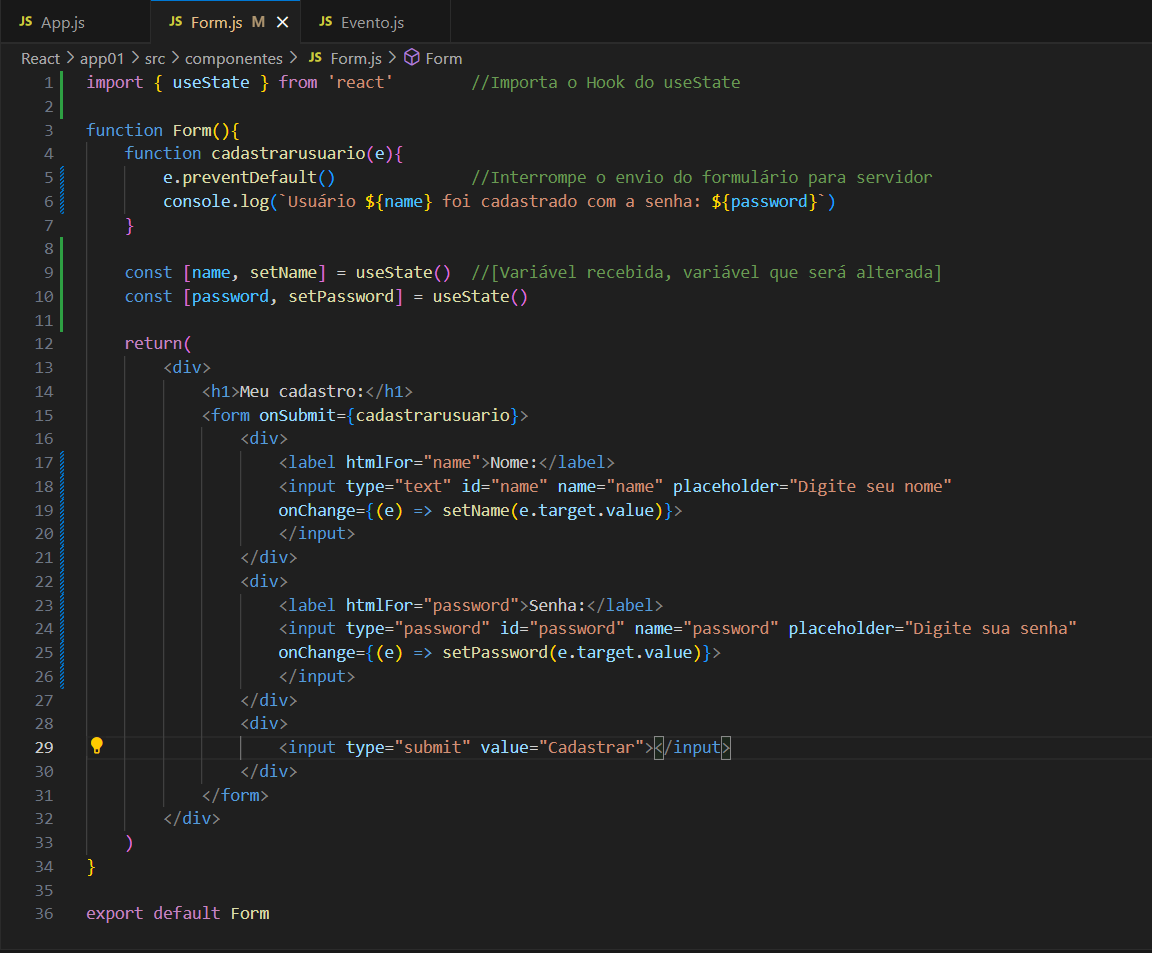
    </div>

  )

}



Vamos trabalhar com useState criando um componente de um formulário:



onChange={(e) => setName(e.target.value)} //Atribui o valor em “setName” e o resgata no useStage em “name”