Use template

yarn create react-app moveit --template=typescript

yarn start

Código index.tsx

Ao renderizar o DOM, busque o elemento ROOT e dentro dele, jogue o que está App

O App é um componente do React

ReactDOM.render(

  <App />,

  document.getElementById('root')

);

**Componente React**

É uma função que retorna JSX

Permite reaproveitamento de código

Ex: Button()

Obs: Não se pode ter vários componentes em uma função, se precisar colocar 3 botões, coloque uma DIV englobando os 3.

function App() {

  return (

    <h1>

      Hello NLW

    </h1>

  );

}

export default App;

**Propriedades React**

Semelhante a Atributos do HTML

Componente:

interface ButtonProps {

  color: string;

}

export function Button(props: ButtonProps){

  return(

    <button type="button">

      Botão

      <strong>Teste</strong>

    </button>

    )

}

App:

function App() {

  return (

    <div>

      <Button color="red" />

      <Button color="green" />

      <Button color="blue" />

    </div>

  );

}

**Propriedade Children**

Sempre que passamos algo dentro de um componente, ele é children.

Ex: Botão 1 = children

function App() {

  return (

    <div>

      <Button color="red">

        Botão 1

      </Button>

    </div>

  );

}

**Estado**

Armazenar informações que vão variar, ao longo do tempo.

Não são variáveis simples do JS.

import { useState } from 'react';

export function Button(props: ButtonProps){

  const counter = useState(1)

…

}

O useState não retorna apenas a variável em si, mas a variável e uma função para atualizar o valor do conter, então é possível desestruturar

export function Button(props: ButtonProps){

  const [counter, setCounter] = useState(1)

…

}

OBS: O React utiliza IMUTABILIDADE, ele não edita uma variável existente, ele cria um valor novo.

  // Da erro

  function increment() {

    counter++;

  }

  // Ok

  function increment() {

    setCounter(counter + 1);

  }

Dessa forma é possível contar o número de clicks em cada botão de forma isolada, pois o estado é definido dentro de cada componente

import { useState } from 'react';

interface ButtonProps {

  color: string;

  children: string;

}

export function Button(props: ButtonProps){

  const [counter, setCounter] = useState(1)

  function increment() {

    setCounter(counter + 1);

  }

  return(

    <button

      type="button"

      style={{backgroundColor: props.color}}

      onClick={increment}

    >

      {props.children} <strong>{counter}</strong>

    </button>

    )

}

**Incluir código JS dentro do HTML**

Usar chaves

  return(

    <button type="button">

      {props.color}

    </button>

    )

**Incluir objeto de JS**

O objeto também usa chaves

Quando você quer utilizar o objeto dentro do HTML, usar 2 chaves

    <button type="button" style={{backgroundColor: props.color}}>

      Botão

      <strong>Teste</strong>

    </button>