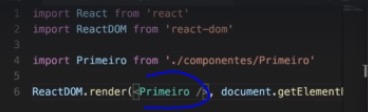
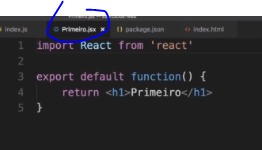
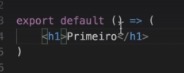


npm i -g create-react-app

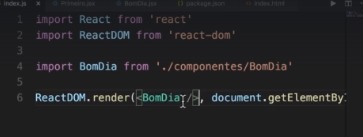
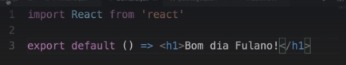
create-react-app exercicios

npm start

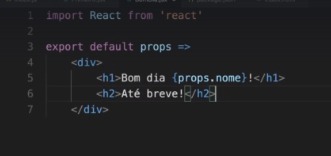
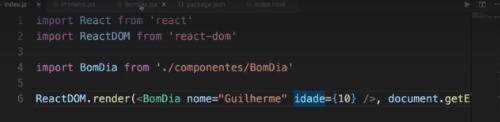
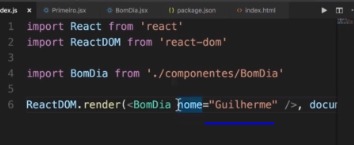
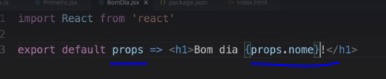




os componentes não podem ter letra minúscula: Primeiro: ok | primeiro: erro



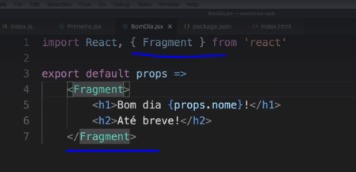




dessa forma, dentro do html, será inserido uma div contendo o h1 e h2.

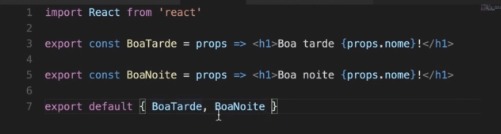
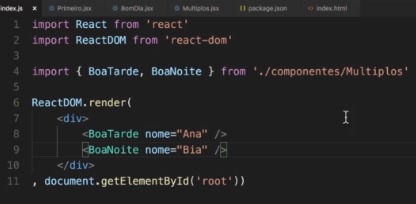
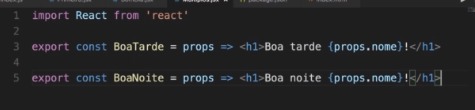
para que não seja criado o elemento <div> no html:

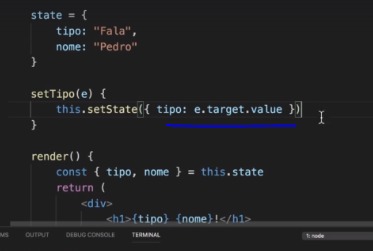
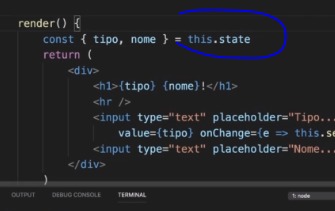
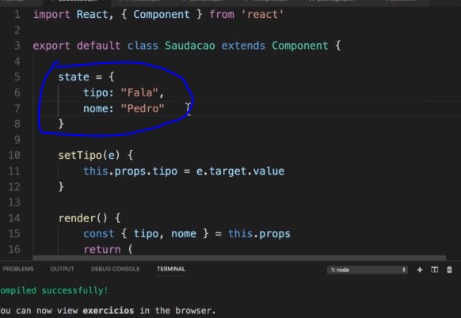
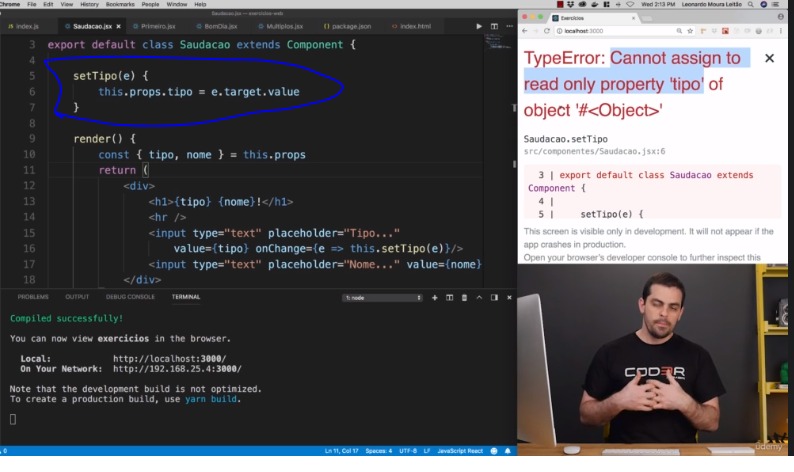
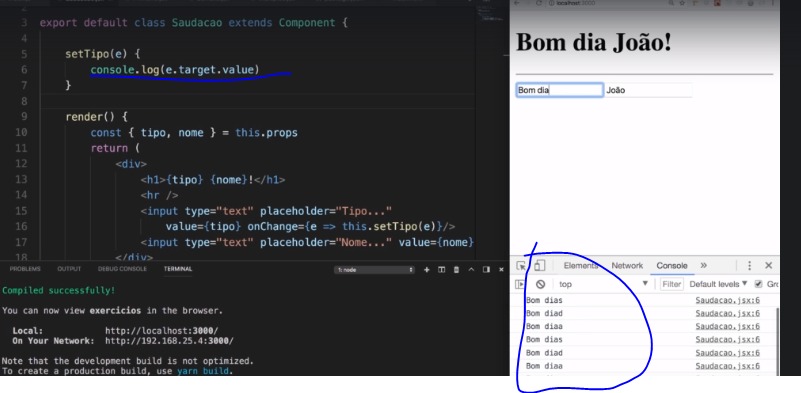
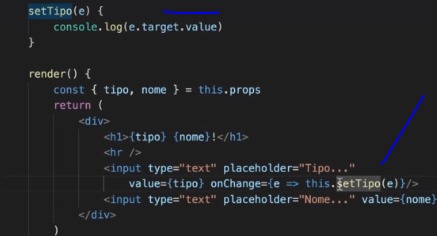
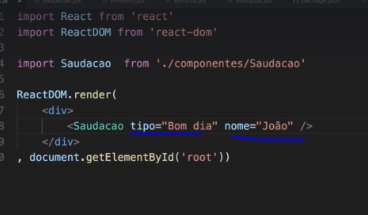
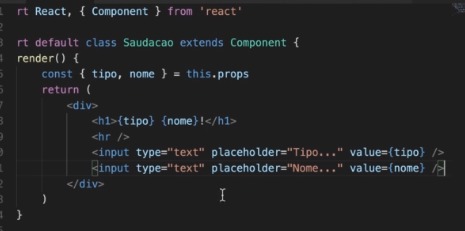
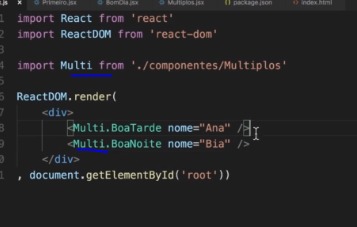




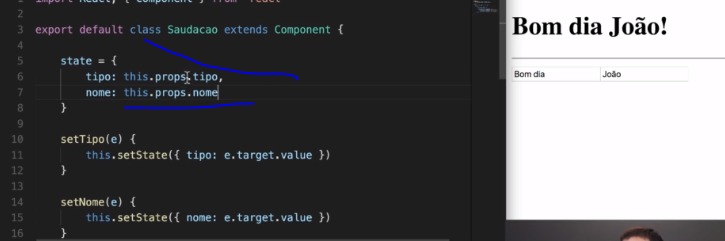
Em array, é preciso colocar o atributo key, para que o React identifique o elemento de forma otimizada dentro da dom.

essa key não pode ser repetida.

A melhor forma é utilizar Fragment ou <div> para envolver vários componentes.



Em React, o estado (alteração do objeto) muda o componente.

O dado mudou: componente atualiza

import React, { Component } from 'react';

export default *class* Saudacao *extends* *Component* {

    state = {

        tipo: *this*.props.tipo,

        nome: *this*.props.nome

    }

*// eslint-disable-next-line no-useless-constructor*

*constructor*(*props*) {

*super*(props)

*// this = esse this dentro do consctructor*

*// representa o componente Saudacao*

*this*.setTipo = *this*.setTipo.bind(*this*)

    }

    setTipo(*e*) {

*this*.setState({

            tipo: e.target.value

        })

    }

    setNome(*e*) {

*this*.setState({

            nome: e.target.value

        })

    }

    render() {

*const* { nome, tipo } = *this*.state

        return (

            <div>

                <h1>{tipo} {nome}</h1>

                <input

*type*="text"

*placeholder*='Tipo'

*value*={tipo}

*onChange*={*this*.setTipo}>

                </input>

                <input

*type*="text"

*placeholder*='Nome'

*value*={nome}

*onChange*={*e* => *this*.setNome(e)}>

                </input>

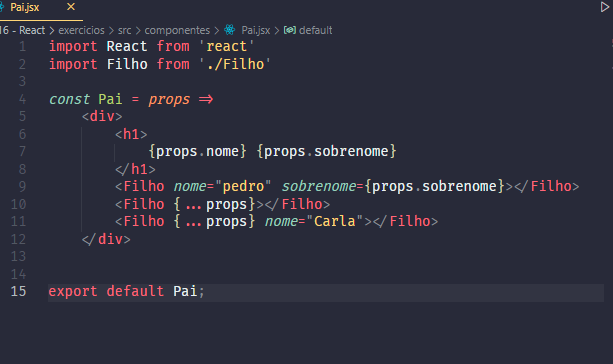
            </div>

        )

    }

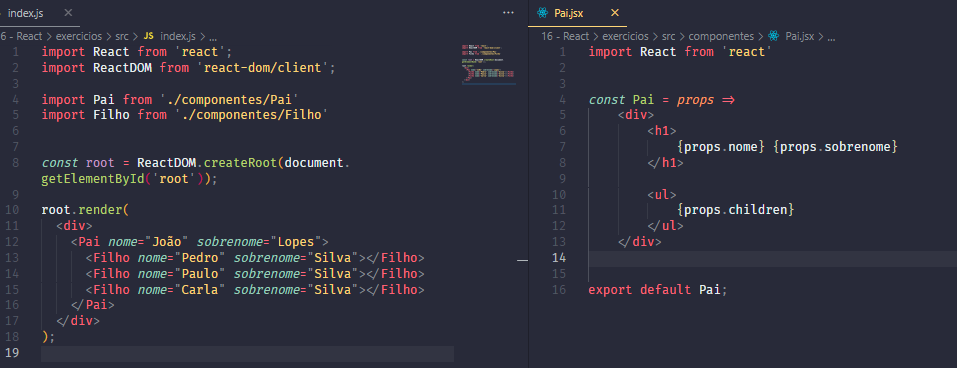
}

// Passando parâmetros do Pai para o elemento Filho



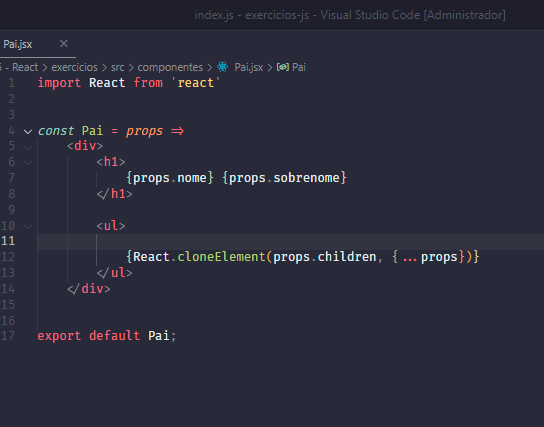
  
Perceba que os Filhos herdaram os parâmetros do elemento Pai

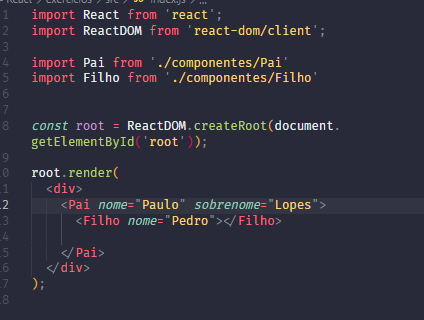
No caso anterior, os Filhos foram criandos dentro do Componente Pai, e para criar os Filhos fora desse componente?

  
Dentro do Componente Pai, é preciso referênciar a área onde serão inseridos os Filhos, através do {props.children}.

E para os Filhos receberem atributos vindos do elemento Pai?

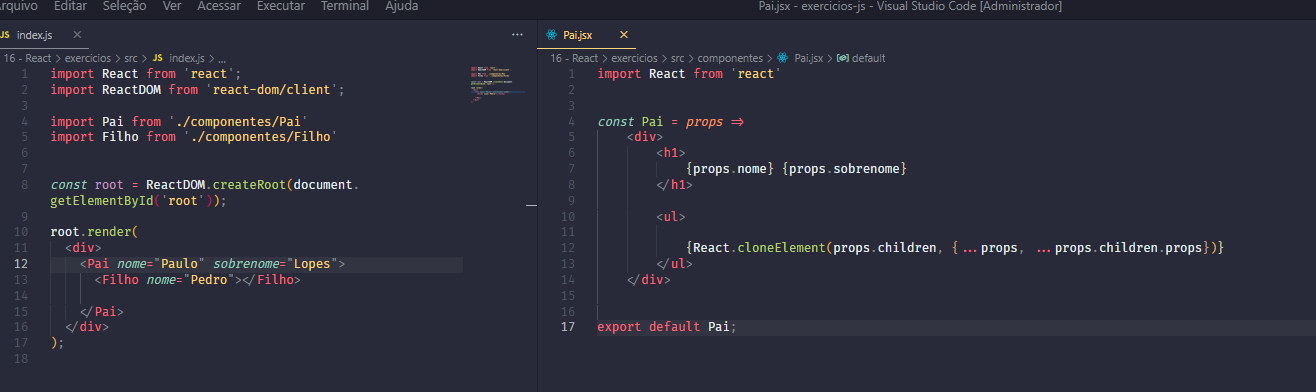
A função cloneElement(), permite clonar um filho vindo do elemento Pai e passar novos parâmetros para ele:





Resultado:  

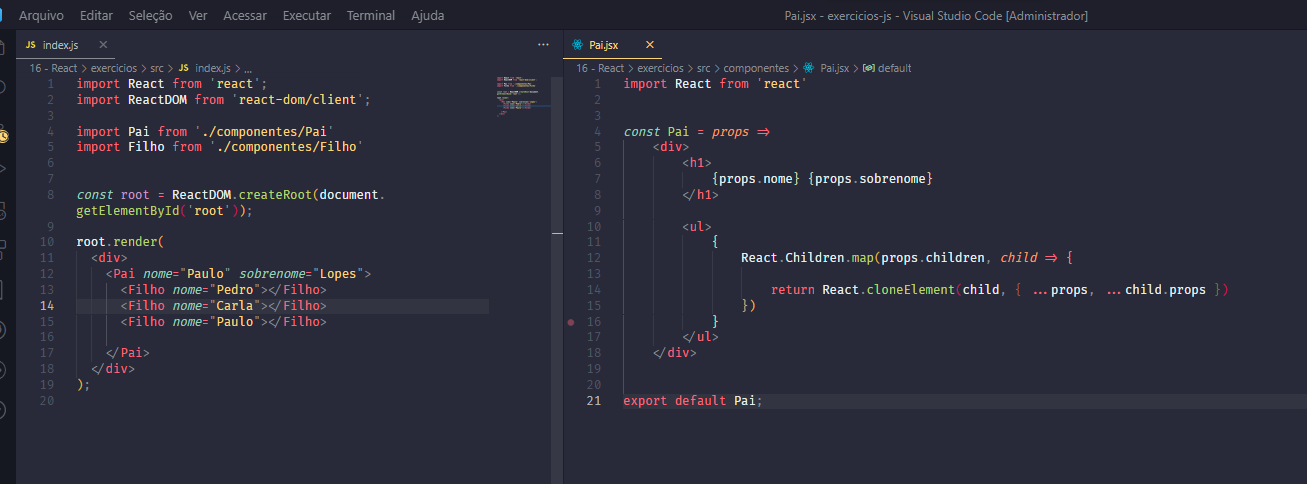

O filho que era Pedro, recebeu os parâmetros do Pai e teve seus parâmetros sobreescritos.

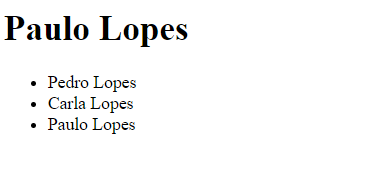
Você também pode, além de passar as propriedades do elementos Pai, passar as propriedades do elemento Filho que será copiado:  


O cloneElement, só clona um elemento por vez. Para clonar múltiplos elementos, podemos utilizar a função Map do React:



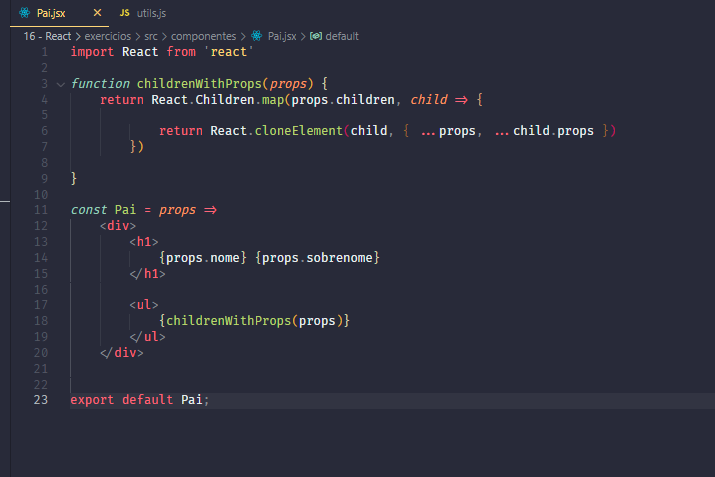
Seleciona todos os filhos: props.children, pega cada um: child e retorna seu clone passando os parâmetros do pai e os parâmetros dos próprios filhos.

RESULTADO:  




Todos herdaram os parâmetros do elemento Pai.

Dando uma refatorada:



Criando arquivo util.js para reutilizar funções:  
