npm create vite@latest

npm i

npm run dev

<http://127.0.0.1:5173/>

trocar de pasta: cd.

Rafc =para criar um componente;

Trocar a porta

package json:

Scripts:  "dev": "vite --port 3000",

/////////////////////////////////////////////////////

const events = () => {

const handleMyevent = (e) => { console.log(e)}

return(

<div>

<div>

<button onClick={handleMyevent}>CLIQUE AQUI</button>

</div>

</div>

/////////////////////////////////////////////////////////////////////////////////

const FIRSTCOMPONENT = () => {

return (

<div>

<h2>Meu primeiro componente</h2>

</div>

);

}

export default FIRSTCOMPONENT

///////////////////////////////////

app.jsx

import { useState } from 'react'

import reactLogo from './assets/react.svg'

import viteLogo from '/vite.svg'

import './App.css'

import FIRSTCOMPONENT from './components/firtscomponents'

function App() {

const [count, setCount] = useState(0)

return (

<div>

<h1> <FIRSTCOMPONENT/></h1>

<p className="teste">TEXTO CLASS</p>

</div>

)

}

export default App

////RENDER – LIST, //PREVIUS STATE

import { useState } from "react"

export const ListRender = () => {

    const [list] = useState(["Paulo", "Mateus", "Marcos", "Lucas"]);

    const [users, setUsers] = useState([{id: 1, name: "Matheus", age:31},

  {id: 2, name: "Lucas", age:28},

  {id:3, name: "João", age: 30 }

  ])

  const deleterandom = () => {

const random = Math.floor(Math.random() \* 4)

setUsers((prevUsers) =>{

 return prevUsers.filter((user) => random !== user.id)

 //Esta função é responsável por remover aleatoriamente

 //um usuário do array users. Ela faz o seguinte:

//Gera um número aleatório entre 0 e 2 usando Math.random()

//e Math.floor().

//Usa a função setUsers para atualizar o estado users. Ela recebe uma função que recebe prevUsers (os usuários atuais) como argumento.

//Dentro da função, usamos o método filter() para retornar um novo array de usuários que não inclui o usuário com o id igual ao número aleatório gerado.

})

  }

  return (

    <div>

<ul>

{list.map((item, i)=>(

    <li key={i}>{item}</li>

))}

</ul>

{users.map((user) => (

<li key = {user.id}>{user.age}-{user.name}</li>

))}

<button onClick={deleterandom}>DELETE UM USUARIO</button>

<ul>

</ul>

    </div>

  );

};

export default ListRender

CONDICONAL   
  
import { useState } from "react"

const Condicional = () => {

const [x] =  useState(false)

CONDICONAL COM USE STATE

import { useState } from "react"

const Condicional = () => {

const [x] =  useState(false)

const [tasks, setTasks] = useState([{id: 1, titulo1: "LISTA", lista: "1-FEIJÃO,\n 2 ARROZ"}])

//adcionar no  array

const adcionararray = () =>{

    setTasks(prevTask =>[...prevTask, {id:2, titulo2: "NOVOT", lista: "novalista"}]

    )

    const deletearray = () => {

        const random = Math.floor(Math.random \* 8)

    setTasks((prevTask)=> {

        return prevTask.filter((task) => random !== task.id )

    })

    }

};

console.log('atual estado: ', tasks)

  return (

  //condição usestate usando true e false,

    <div>

<h1>ISSO VAI APARECER</h1>

{x &&<p>Se x for true, sim</p>}

{!x &&<p>Se x for false, sim</p>}

//condição use state usando uma lista como estado e uma função para adciona-la

{tasks.length >= 1 ? (

<ul>{tasks.map((task) => (<li key={task.id}>

{task.lista}

</li>))}</ul>

) : (

  <div> é diferente de 1</div>

)}

<button onClick={adcionararray}>CLICA PARA ADCIONAR</button>

    </div>

  );

};

export default Condicional

PROPS

É passar um elemento da classe pai par classe filho, por exemplo; pode ser feito em desestrutura: em app.jsx na tag da componente PROPS é colocad: os alias com valores em desestrutura, depois:

import React from 'react'

//o destructing é feito ao colocar ao passar em chaves nos objetos de const PROPS

export const PROPS = ({brand, km}) => {

  return (

    <div>

<ul>

<li>MARCA: {brand}</li>

<li>KM: {km}</li>

</ul>

    </div>

  )

}

export default PROPS

RESULTADO NO SITE

* MARCA: BMW
* KM: 1000

SECRET WORD

É interessante para CSS preencher alinhamento do item no app.css

.App {

display: flex;

justify-content: center;

align-items: center;

text-align: center;

}

E para o index.css é interessante preencher elementos gerais do body.

html, body{

height: 100%;

pegando a página toda

}

body{

font-family:  Helvetica;

margin: 0%; zerando as margens

padding: 0%;

background: linear-gradient(180deg,rgba(9,35,175,1)0%, //180graus começando no topo e finalizando na cor abaixo

rgba(0,0,0,1)100%

);

}