onClick=(aqi vai uma funcao)

useState é para ter uma variavel flexivel, mas vai ser const no inicio

se tiver useState ter use client

EXERCICIO 1: DUVIDAS

**pq do zero?**

  const [contador, setContador] = useState(0)

se colocar let não funciona, por isso o const de constante

typescript so aceita if else ternário

o recebe é assim no if else do typescript

 if (contador < 0) {

            setContador(0)

        }

setContador recebe 0

EXERCICIO 2: DUVIDAS

**Por que usar o useState aqui?**

**Dicas do GPT:**

 O useState pode começar com **qualquer valor** que você quiser, não só zero.

 Se texto for booleano, true ou false fazem sentido.

 Se quiser armazenar texto, inicialize com string: useState('') ou useState('Algum texto').

Os exercícios prontos estão no meu celular, fotos, disseram que o exercicio 5 é igual aquele dos bottons que fizemos, o exercício 6 é igual o 1 e o 2

**JAVASCRIPT DA ESTRUTURA TERNARIA COMO FUNCIONA:**

let idade = 20;  
let mensagem = idade >= 18 ? "Maior de idade" : "Menor de idade";  
console.log(mensagem); // Saída: Maior de idade

front-end para back-end, dá para separar e ter essa comunicação, mas para evitar que os dados sejam interceptados, cria-se a api para segurança, como o back end é ligado ao banco de dados é possível alguém ter acesso a esses dados nessa comunicação

front >< api >< back

a requisição que é feita entre eles é http, exemplo: get, set, na vida real não usamos as 5 tipos de requisição, quase tudo usa o tipo post

TIPOS: GET, POST, PUT, DELETE, PATCH

O tipo de dado que é comunicado entre eles, o trafego é feito em .json

Quando alguém faz um login, é gerado um token, esse token é validado ali na API

**JSONPlaceholder é um site, entra e clica em “Guide”, vai listar para você vários endpoints, basicamente é a rota que você vai acessar para conseguir os dados, vai devolver o json em um endpoint, ver o guide, as rotas que tem ali vao vir de um banco de dados real sim, e a gente vai pegar esses dados para a nossa aplicação**

**Vamos aprender como fazer a chamada de API no nosso backend, pode ser usado o fat**

**Precisaremos de uma constante que vai receber a variável que vai vir lá da API que vamos usar**

**Segue o exemplo que estará na pasta do Exercicio 5 – API**

**Se te devolver os dados com a URL, é porque ela está correta, vamos para o código agora fazer**

**Precisamos saber também o tipo de HTTP, que vai ser get ou outros, se não tiver nenhum especificado, sempre usar GET**

**Sincrono: vai usar a palavra async, vai de cima para baixo, 1 2 3 4, nesse como é sequencial, caso o 1 demore para carregar, ele já vai carregar o 2, por isso não vai esperar o tempo de processamento, por isso usamos o assíncrona, onde começar a usar a palavra await, porque vai fazer com que tenha que validar primeiro antes que o próximo seja processado, precisamos usar o assincronismo, “espera isso ser resolvido para vc continuar”, falamos isso para o sistema**

**Assincrona: vai ter uma quebra no meio e pode deixar sem sequencia**

**Se vamos fazer toda essa comunicação, vai levar tempo**

**Vamos começar a usar o fetch**

**Vai precisar fazer uma descompactação do json que virá da caixinha da API, tirar o json dela, para fazer isso, assim não está descompactando.**

console.log(resposta)

**Como descompactar os dados está no exercício 1 de api comentado.**

**Como conferir que o código deu certo:**



**Agora que vamos tentar filtrar, vamos precisar tipar os dados primeiro, vamos então para o exercício 2, vamos tipar 1 number e 3 strings no exemplo que estamos usando.**

**Sobre rota:**

**Com o client-side é mais lento, esse problema de piscar é chamado de layout shift, o nome do problema. Por isso que a gente prefere gerar as páginas pelo back-end, deixa so algumas coisas pelo client-side e client server, o cliente server é melhor e mais rápido, está no slide.**

**Se tiver interabilidade para ser feita na tela, precisa ser ser do tipo client-side.**

**Como vamos ter uma tela client-server com interabilidade? Isso que vamos aprender.**

**Por isso, vamos aprender a fazer componentes interabilidade mesmo sendo client-server.**

**Vamos voltar no componente Button e transformar ele em um client.**

**Não dá para fazer side chamando servidor. Só dá para fazer servidor chamando side. Isso serve para não ser a tela toda carregando toda hora, e sim só algumas partes.**

**O caminho da imagem é colocado no source da tag <<<<**

**Hora do Exercicio mesmo**

[**https://fakestoreapi.com/**](https://fakestoreapi.com/)

[**https://fakestoreapi.com/products**](https://fakestoreapi.com/products)

[**https://jsonplaceholder.typicode.com/users**](https://jsonplaceholder.typicode.com/users)

[**https://v2.tailwindcss.com/docs/max-height**](https://v2.tailwindcss.com/docs/max-height)

**https://github.com/AlianeAmaral/Exercicios-Curso-de-React-Nextjs-SENAC**

**quando tiver dando aquele monte de erro na clonagem, executar dentro da pasta my-app**

**npm install --save-dev @types/react @types/react-dom**