Žinoma! useState yra vienas iš pagrindinių React hook'ų, naudojamų funkciniuose komponentuose valdyti būseną. Štai pagrindinė informacija apie useState:

**Kas yra useState?**

useState leidžia jums pridėti būseną į funkcinius komponentus. Kai naudojate useState, jūs deklaruojate būsenos kintamąjį ir funkciją, kuri leidžia atnaujinti tą būseną.

**Kaip naudoti useState?**

Štai paprastas pavyzdys, kaip naudoti useState:

import React, { useState } from 'react';

function Counter() {

// Deklaruojame naują būsenos kintamąjį, kurio pradinė reikšmė yra 0

const [count, setCount] = useState(0);

return (

<div>

<p>Jūs paspaudėte {count} kartus</p>

<button onClick={() => setCount(count + 1)}>

Paspausk mane

</button>

</div>

);

}

export default Counter;

**Paaiškinimas:**

1. **Importavimas**: Pirmiausia importuojame useState iš React.
2. **Būsenos deklaravimas**: Naudojame useState funkciją, kad deklaruotume būsenos kintamąjį count ir funkciją setCount, kuri atnaujina count. Pradinė count reikšmė yra 0.
3. **Būsenos naudojimas**: Komponente rodome count reikšmę ir turime mygtuką, kuris, paspaudus, padidina count reikšmę vienu vienetu, naudojant setCount.

**Ką dar reikia žinoti?**

* **Pradinė reikšmė**: useState priima vieną argumentą – pradinę būsenos reikšmę.
* **Atnaujinimo funkcija**: Funkcija, kurią grąžina useState, naudojama atnaujinti būseną. Ji priima naują būsenos reikšmę kaip argumentą.
* **Re-render**: Kai atnaujinate būseną, komponentas bus perrenderintas su nauja būsenos reikšme.

useState yra labai galingas ir leidžia jums lengvai valdyti būseną funkciniuose komponentuose. Jei turite daugiau klausimų ar reikia papildomų paaiškinimų, drąsiai klauskite!

Žinoma! props (trumpinys nuo "properties") yra mechanizmas, kurį React naudoja perduoti duomenis iš vieno komponento į kitą. Tai leidžia komponentams būti dinamiškiems ir pakartotinai naudojamiems.

**Kas yra props?**

props yra objektas, kuris perduodamas į komponentą kaip argumentas. Jis gali turėti bet kokius duomenis, kuriuos norite perduoti iš tėvinio komponento į vaiką.

**Kaip naudoti props?**

Štai paprastas pavyzdys, kaip naudoti props:

1. **Tėvinis komponentas**:

import React from 'react';

import Greeting from './Greeting';

function App() {

return (

<div>

<Greeting name="Jonas" />

</div>

);

}

export default App;

1. **Vaiko komponentas**:

import React from 'react';

function Greeting(props) {

return <h1>Sveikas, {props.name}!</h1>;

}

export default Greeting;

**Paaiškinimas:**

1. **Tėvinis komponentas (App)**:
   * Importuoja vaiko komponentą (Greeting).
   * Naudoja Greeting komponentą ir perduoda jam name prop'ą su reikšme "Jonas".
2. **Vaiko komponentas (Greeting)**:
   * Priima props kaip argumentą.
   * Naudoja props.name reikšmę, kad atvaizduotų pasveikinimo žinutę.

**Ką dar reikia žinoti?**

* **Read-only**: props yra tik skaitomi. Vaiko komponentas negali jų keisti.
* **Destruktūrizacija**: Galite naudoti destruktūrizaciją, kad ištrauktumėte props reikšmes tiesiogiai:

function Greeting({ name }) {

return <h1>Sveikas, {name}!</h1>;

}

* **Numatytosios reikšmės**: Galite nustatyti numatytąsias props reikšmes, jei jos nėra perduodamos:

Greeting.defaultProps = {

name: 'Nežinomas'

};

props leidžia jums kurti lankstesnius ir pakartotinai naudojamus komponentus, perduodant jiems reikalingus duomenis iš tėvinių komponentų. Jei turite daugiau klausimų ar reikia papildomų paaiškinimų, drąsiai klauskite!

Žinoma! useEffect yra vienas iš pagrindinių React hook'ų, naudojamas funkciniuose komponentuose atlikti šalutinius veiksmus. Štai pagrindinė informacija apie useEffect:

**Kas yra useEffect?**

useEffect leidžia jums atlikti šalutinius veiksmus funkciniuose komponentuose. Tai gali būti duomenų užklausos, prenumeratos, DOM manipuliacijos ir kiti veiksmai, kurie turi įvykti po komponento atvaizdavimo.

**Kaip naudoti useEffect?**

Štai paprastas pavyzdys, kaip naudoti useEffect:

import React, { useEffect, useState } from 'react';

function ExampleComponent() {

const [count, setCount] = useState(0);

useEffect(() => {

document.title = `Paspaudėte ${count} kartus`;

}, [count]);

return (

<div>

<p>Jūs paspaudėte {count} kartus</p>

<button onClick={() => setCount(count + 1)}>

Paspausk mane

</button>

</div>

);

}

export default ExampleComponent;

**Paaiškinimas:**

1. **Importavimas**: Pirmiausia importuojame useEffect ir useState iš React.
2. **Būsenos deklaravimas**: Naudojame useState, kad deklaruotume būsenos kintamąjį count ir funkciją setCount.
3. **useEffect naudojimas**:
   * useEffect priima dvi argumentus: funkciją, kuri vykdoma po kiekvieno atvaizdavimo, ir priklausomybių masyvą.
   * Funkcija atnaujina dokumento pavadinimą su dabartine count reikšme.
   * Priklausomybių masyvas [count] nurodo, kad useEffect turėtų būti vykdomas tik tada, kai count pasikeičia.

**Ką dar reikia žinoti?**

* **Priklausomybių masyvas**: Jei nenurodysite priklausomybių masyvo, useEffect bus vykdomas po kiekvieno atvaizdavimo. Jei nurodysite tuščią masyvą [], useEffect bus vykdomas tik vieną kartą po pirmojo atvaizdavimo.
* **Šalutinių veiksmų valymas**: Jei jūsų useEffect funkcija grąžina funkciją, ši grąžinta funkcija bus vykdoma valant šalutinį veiksmą prieš kitą useEffect vykdymą arba komponentui išmontuojant:

useEffect(() => {

const subscription = someAPI.subscribe();

return () => {

subscription.unsubscribe();

};

}, []);

useEffect yra labai galingas ir leidžia jums lengvai valdyti šalutinius veiksmus funkciniuose komponentuose. Jei turite daugiau klausimų ar reikia papildomų paaiškinimų, drąsiai klauskite!

useParams yra React Router Hook, kuris leidžia jums pasiekti URL parametrus jūsų komponentuose. Tai ypač naudinga, kai norite gauti dinamiškus duomenis iš URL, pvz., vartotojo ID ar produkto ID.

Štai paprastas pavyzdys, kaip naudoti useParams:

import React from 'react';

import { useParams } from 'react-router-dom';

const UserProfile = () => {

const { userId } = useParams();

return (

<div>

<h1>User Profile</h1>

<p>User ID: {userId}</p>

</div>

);

};

export default UserProfile;

Šiame pavyzdyje useParams hook'as iš URL ištraukia userId parametrą ir jį naudoja UserProfile komponente. Pavyzdžiui, jei URL yra /users/123, userId bus 123.

Ar norėtumėte sužinoti daugiau apie React Router ar kitus React aspektus?

useForm yra populiarus hook'as, naudojamas formų valdymui React programose. Vienas iš dažniausiai naudojamų bibliotekų, kuri siūlo useForm hook'ą, yra react-hook-form. Ši biblioteka padeda lengvai valdyti formų būseną, validaciją ir pateikimą.

Štai paprastas pavyzdys, kaip naudoti useForm su react-hook-form:

1. Pirmiausia, įdiekite biblioteką:

npm install react-hook-form

1. Tada galite naudoti useForm savo komponente:

import React from 'react';

import { useForm } from 'react-hook-form';

const MyForm = () => {

const { register, handleSubmit, formState: { errors } } = useForm();

const onSubmit = data => {

console.log(data);

};

return (

<form onSubmit={handleSubmit(onSubmit)}>

<div>

<label htmlFor="name">Name</label>

<input id="name" {...register('name', { required: true })} />

{errors.name && <span>This field is required</span>}

</div>

<div>

<label htmlFor="email">Email</label>

<input id="email" {...register('email', { required: true })} />

{errors.email && <span>This field is required</span>}

</div>

<button type="submit">Submit</button>

</form>

);

};

export default MyForm;

Šiame pavyzdyje useForm hook'as naudojamas formos valdymui. register funkcija priskiria formos laukus, handleSubmit tvarko formos pateikimą, o formState: { errors } leidžia pasiekti validacijos klaidas.

Žinoma! Štai kaip naudoti useNavigate kabliuką React Router:

* **Importavimas**: Pirmiausia reikia importuoti useNavigate iš react-router-dom.
* **Naudojimas**: Iškvietus useNavigate, gausite navigacijos funkciją, kurią galite naudoti norėdami pereiti į skirtingus maršrutus.

Štai paprastas pavyzdys:

import { useNavigate } from 'react-router-dom';

function MyComponent() {

const navigate = useNavigate();

const goToHome = () => {

navigate('/home');

};

return (

<button onClick={goToHome}>

Eiti į pagrindinį puslapį

</button>

);

}

Šis kodas nukreips į /home maršrutą, kai bus paspaustas mygtukas.

Ar norėtumėte daugiau detalių ar turite kokių nors konkrečių klausimų apie useNavigate naudojimą?

function App() {

const [count, setCount] = useState(0)

function increment() {

setCount(prevCount => prevCount + 1)

}

return (

<div>

<h1>{count}</h1>

<button onClick={increment}>Increment</button>

</div>

)

}

import React, {useState} from "react"

function App() {

const [count, setCount] = useState(0)

const [answer, setAnswer] = useState("Yes")

function increment() {

setCount(prevCount => prevCount + 1)

}

function decrement() {

setCount(prevCount => prevCount - 1)

}

return (

<div>

<h1>{count}</h1>

<button onClick={increment}>Increment</button>

<button onClick={decrement}>Decrement</button>

</div>

)

}

export default App

// Side effects?

// Network request

// Manual DOM manipulation

// Event listeners or timeouts and intervals