React the Complete Guide

Para crear un proyecto utilizamos los comandos:

**npx create-react-app my-app**

**cd my-app**

**npm start**

**SEC-4: REACT STATE & WORKING with Events**

**WORKING with “STATE”**

Primero importamos el **useState**:

import {useState} from "react";

Luego dentro del **componente**, **pero por fuera del return** lo llamamos. El primer elemento es el valor actual con el que lo inicializamos, y el segundo elemento es una function que actualiza el primer elemento:

function Expenseitem(props) {

  const [title, setTitle] = useState(props.title);

  const clickHandled = () => {

    setTitle("Updated");

  }

  return (

    <Card className="expense-item">

      <ExpenseDate date={props.date}/>

      <div className="expense-item\_\_description">

        <h2>{title}</h2>

        <div className="expense-item\_\_price">${props.amount}</div>

      </div>

      <button onClick={clickHandled}>Change title</button>

    </Card>

  );

}

**ADDING TWO-WAY BINDING**

En la función para la etiqueta **form** seteamos al final los valores nuevamente en un string vacío, así al enviarse el formulario se puede volver a completar sin tener que borrar uno mismo la información puesta con anterioridad. Y en los input le pasamos al value el primer elemento del useState:

const [enteredTitle, setEnteredTitle] = useState("");

const submitHandler = (e) => {

        e.preventDefault();

        const expenseData = {

            title: enteredTitle,

            amount: Number(enteredAmount),

            date: new Date(enteredDate),

        }

        console.log(expenseData);

        setEnteredTitle("");

        setEnteredAmount("");

        setEnteredDate("");

    }

return (

      <form onSubmit={submitHandler}>

        <div className="new-expense\_\_controls">

          <div className="new-expense\_\_control">

              <label>Title:</label>

              <input

                type="text"

                value={enteredTitle}

                onChange={titleChangeHandler}/>

          </div>

        </div>

**SEC-9: WORKING with FRAGMENTS, PORTALS & REFS**

**REACT FRAGMENTS**

Se utiliza como un wrapper, es decir, no renderiza ningún elemento HTML del DOM, solo rellena el requerimiento de JSX para renderizar un solo componente.

Podemos hacerlo de dos formas:

* Poniendo una etiqueta vacía.

return (

    <>

      <AddUser onAddUser={addUserHandler}/>

      <UserList users={usersList}/>

    </>

  );

* O utilizando **React.Fragment**:

return (

    <React.Fragment>

      <AddUser onAddUser={addUserHandler}/>

      <UserList users={usersList}/>

    </React.Fragment>

  );

**WORKING with PORTALS**

Para utilizar los **portals** primero debemos ir a la carpeta **public/index.html** y escribir los div con un id para identificarlos: agregué el div backdrop-root y overlay-root.

<body>

    <noscript>You need to enable JavaScript to run this app.</noscript>

    <div id="backdrop-root"></div>

    <div id="overlay-root"></div>

    <div id="root"></div>

</body>

Luego, en el componente ErrorModal, creamos dos nuevas funciones, una que contiene el backdrop y otro el modal:

function Backdrop(props) {

  return <div className="backdrop" onClick={props.onConfirm} />;

}

function ModalOverlay(props) {

  return (

    <Card className={"modal"}>

      <header className="header">

        <h2>{props.title}</h2>

      </header>

      <div className="content">

        <p>{props.message}</p>

      </div>

      <footer className="actions">

        <Button onClick={props.onConfirm}>Close</Button>

      </footer>

    </Card>

  );

}

A continuación importamos **ReactDOM from “react-dom”** y lo utilizamos para crear un portal en la función de ErrorModal, mediante el método **createPortal()**, el cual recibe dos parámetros, el primero es el React node que queremos renderizar, por ejemplo **<Backdrop/>**, y el segundo parámetro es un apuntador al conteiner en el DOM real donde estos elementos deberían ser renderizados, el cual escribimos en el index.html.

function ErrorModal(props) {

  return (

    <React.Fragment>

      {ReactDOM.createPortal(

        <Backdrop onConfirm={props.onConfirm} />,

        document.getElementById("backdrop-root")

      )}

      {ReactDOM.createPortal(

        <ModalOverlay

          title={props.title}

          message={props.message}

          onConfirm={props.onConfirm}

        />,

        document.getElementById("overlay-root")

      )}

    </React.Fragment>

  );

}

**WORKING with REFS**

Lo utilizamos para conectar un elemento HTML con el prop **ref**. Primero importamos el **useRef**:

import { useRef } from "react";

Luego dentro de la función creamos una constante que va a ser la prop de **ref** que le pasemos y luego dentro del elemento html pasamos esa prop:

function AddUser(props) {

  const nameImputRef = useRef();

  const ageImputRef = useRef();

...

<input

   type="text"

   id="username"

   ref={nameImputRef}

/>

<input

   type="number"

    id="age"

    ref={ageImputRef}

/>

Ahora con esta conexión nosotros podemos acceder al valor que tenga ese input cuando el usuario lo escriba, por lo que podemos eliminar los setState. Todo el código quedaría de la siguiente manera:

import { useState, useRef } from "react";

import "./AddUser.css";

// Components

import Card from "../UI/Card";

import Button from "../UI/Button";

import ErrorModal from "../UI/ErrorModal";

import Wrapper from "../Helpers/Wrapper";

function AddUser(props) {

  const nameImputRef = useRef();

  const ageImputRef = useRef();

  const [error, setError] = useState();

  const addUserHandler = (e) => {

    e.preventDefault();

    const enteredName = nameImputRef.current.value;

    const enteredUserAge = ageImputRef.current.value;

    if (enteredName.trim().length === 0 || enteredUserAge.trim().length === 0) {

      setError({title: "Invalid input", message: "Please enter a valid name and age"})

      return;

    }

    if (Number(enteredUserAge) < 1) {

      setError({title: "Invalid age", message: "Please enter a valid age greater than 0"})

      return;

    }

    props.onAddUser(enteredName, enteredUserAge);

    nameImputRef.current.value = "";

    ageImputRef.current.value = "";

  };

  const errorHandler = () => {

    setError(null);

  }

  return (

    <Wrapper>

      {error && <ErrorModal title={error.title} message={error.message} onConfirm={errorHandler}/> }

      <Card className={"input"}>

        <form onSubmit={addUserHandler}>

          <label htmlFor="username">Username</label>

          <input

            type="text"

            id="username"

            ref={nameImputRef}

          />

          <label htmlFor="age">Age (Years)</label>

          <input

            type="number"

            id="age"

            ref={ageImputRef}

          />

          <Button type="submit">Add User</Button>

        </form>

      </Card>

    </Wrapper>

  );

}

export default AddUser;