React State and Events

* Example of event handler:

import React from "react";

import ExpenseDate from "./ExpenseDate";

import Card from "../UI/Card";

import "./ExpenseItem.css";

const ExpenseItem = (props) =>{ //🡸Event handler function

const clickHandler = () =>{

    console.log("clicked");

  }

  return (

    <Card className="expense-item">

      <ExpenseDate date={props.date}/>

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

        <h2>{props.title}</h2>

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

        <button onClick={clickHandler}>Change Ttile</button> //🡸Adding event to button onClick

      </div>

    </Card>

  );

}

export default ExpenseItem;

* Note :
  + React hooks name always starts with usekeyword.
  + We can use react hooks only in component function. On even in the function in components function
* useState() hook:

import React, { useState } from “react”;

const component = (props) =>{

const [Value, setValue] = useState();

…

}

* + useState() hooks create special type of variable. Whose changes lead to call again the component function.
  + useState() hooks return an array with exactly two arguments.
  + 1) value => current state value ; 2) setValue() => updated state value function.

(We can use any name)

* + When we use setValue component function will executed again from start.

import React, {useState}from "react";

import ExpenseDate from "./ExpenseDate";

import Card from "../UI/Card";

import "./ExpenseItem.css";

const ExpenseItem = (props) =>{

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

  // let title = props.title;

  const clickHandler = () =>{

    // title = "Clicked!"           //we cant do this

    setTitle('Updated');

    console.log(title); //this will show old value

  }

  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>

        <button onClick={clickHandler}>Change Ttile</button>

      </div>

    </Card>

  );

}

export default ExpenseItem;

* Using multiple states to store entered value through eventHandler

import React, { useState } from "react";

import "./ExpenseForm.css";

const ExpenseForm = () => {

  const [enteredTitle, setEnteredTitle] = useState(""); //useState('') is empty intially

  const [enteredAmount, setEnteredAmount] = useState("");

  const [enteredDate, setEnteredDate] = useState("");

  const titleChangeHandler = (event) => {

    setEnteredTitle(event.target.value); //event.target.value gives value entered

  };

  const amountChangeHandler = (event) => {

    setEnteredAmount(event.target.value);

  };

  const dateChangeHandler = (event) => {

    setEnteredDate(event.target.value);

  };

  return (

    <form>

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

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

          <label>Title</label>

          <input type="text" onChange={titleChangeHandler} />

        </div>

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

          <label>Amount</label>

          <input

            type="number"

            min="0.01"

            step="0.01"

            onChange={amountChangeHandler}

          />

        </div>

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

          <label>Date</label>

          <input

            type="date"

            min="2019-01-01"

            max="2021-12-31"

            onChange={dateChangeHandler}

          />

        </div>

      </div>

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

        <button type="submit">Add Expense</button>

      </div>

    </form>

  );

};

export default ExpenseForm;

* Using one state for form instead of many ( alternative for above solution ):

Pass object to the state ex: useState ( { } ) instead of string ex. useState(‘’).

import React, { useState } from "react";

import "./ExpenseForm.css";

const ExpenseForm = () => {

  // using multiple state to store values

  // const [enteredTitle, setEnteredTitle] = useState(""); //useState('') is empty intially

  // const [enteredAmount, setEnteredAmount] = useState("");

  // const [enteredDate, setEnteredDate] = useState("");

  // using single state to store values

  const [userInput, setUserInput] = useState({

    enteredTitle: "",

    enteredAmount: "",

    enteredDate: "",

  });

  const titleChangeHandler = (event) => {

    // setEnteredTitle(event.target.value); //event.target.value gives value entered

//method 1

    // setUserInput({

    //   ...userInput,               //spread operator to add all value

    //   enteredTitle : event.target.value,    //override enterTitle value

    // })

//method

    setUserInput((prevState) => {

      return {

        ...prevState,

        enteredTitle: event.target.value,

      };

    });

  };

  const amountChangeHandler = (event) => {

    // setEnteredAmount(event.target.value);

//method 1

    // setUserInput({

    //   ...userInput,

    //   enteredAmount: event.target.value,

    // });

//method 2

    setUserInput((prevState) => {               //use this if state is depends on previous state

      return {

        ...prevState,

        enteredAmount: event.target.value,

      };

    });

  };

  const dateChangeHandler = (event) => {

    // setEnteredDate(event.target.value);

//method 1

    // setUserInput({

    //   ...userInput,

    //   enteredDate: event.target.value,

    // });

//method 2

    setUserInput((prevState) => {

      return {

        ...prevState,

        enteredDate: event.target.value,

      };

    });

  };

  return (

    <form>

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

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

          <label>Title</label>

          <input type="text" onChange={titleChangeHandler} />

        </div>

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

          <label>Amount</label>

          <input

            type="number"

            min="0.01"

            step="0.01"

            onChange={amountChangeHandler}

          />

        </div>

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

          <label>Date</label>

          <input

            type="date"

            min="2019-01-01"

            max="2021-12-31"

            onChange={dateChangeHandler}

          />

        </div>

      </div>

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

        <button type="submit">Add Expense</button>

      </div>

    </form>

  );

};

export default ExpenseForm;

* To stop reload after clicking submit button

const submitHandler = (event) =>{

    event.preventDefault();             //this js is use to stop reloading after clicking sumbit button

  }

  return (

    <form onSubmit={submitHandler}>

       <div>

      </div>

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

        <button type="submit">Add Expense</button>

      </div>

    </form>

  );

};

* When we have to pass data from parent component to child component we use props.

\*Child 🡪 parent

* But when we have to pass data from child to parent use following steps:

Step 1:

To parent component pass function. Ex: onSaveExpenseData = {saveExpenseDataHandler}

const NewExpense = () =>{

    const saveExpenseDataHandler= (enteredExpenseData) =>{

        const expenseData = {

            ...enteredExpenseData,

            id : Math.random().toString()

        }

        console.log(expenseData)

    }

    return(

        <div className = "new-expense">

           <ExpenseForm onSaveExpenseData = {saveExpenseDataHandler}/>

        </div>

    )

}

Step 2:

Now using props use that function to pass data

const submitHandler = (event) => {

    event.preventDefault();

    const expenseData = {

      title: enteredTitle,

      amount: enteredAmount,

      date: new Date(enteredDate),

    };

    props.onSaveExpenseData(expenseData) 🡨use function

    // after submitting value clear all input

    setEnteredTitle('');

    setEnteredAmount('');

    setEnteredDate('');

  };

Note : Lifting state up

We can pass data from child to parent but we can’t pass data from component to sibling component.

So if we want to pass that data first we have to pass data to the comman parent of both component then from that parent we can pass data to its child using props (sibling compoent).

Then pass data from apps to comp2

Using props First pass data from comp1 to Apps using above

We can pass direct data from

component1 to component 2

* Keys to array data:

1. Keys help React identify which items have changed, are added, or are removed. Keys should be given to the elements inside the array to give the elements a stable identity

const numbers = [1, 2, 3, 4, 5];

const listItems = numbers.map((number) =>

<li key={number.toString()}> {number}

</li>

);

1. The best way to pick a key is to use a string that uniquely identifies a list item among its siblings. Most often you would use IDs from your data as keys.
2. When you don’t have stable IDs for rendered items, you may use the item index as a key as a last resort

const todoItems = todos.map((todo, index) =>

// Only do this if items have no stable IDs <li key={index}> {todo.text}

</li>

);