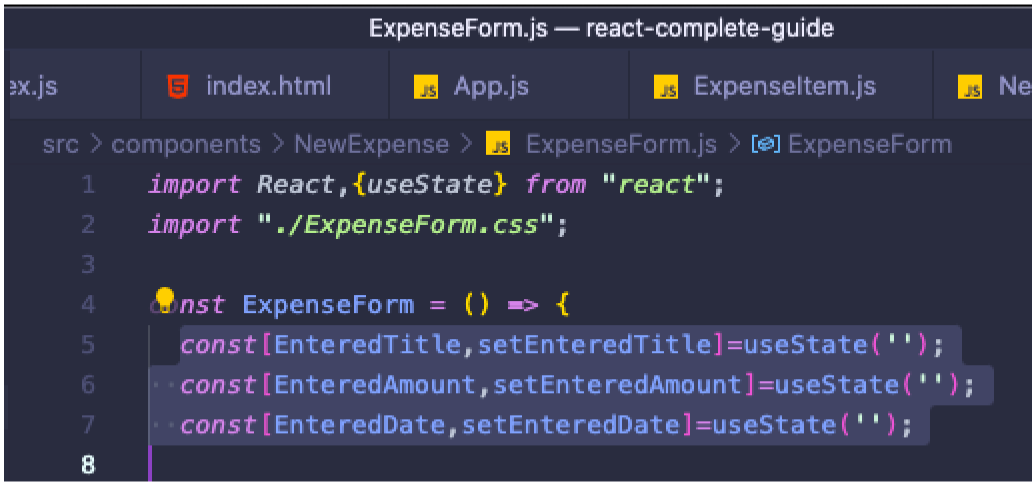
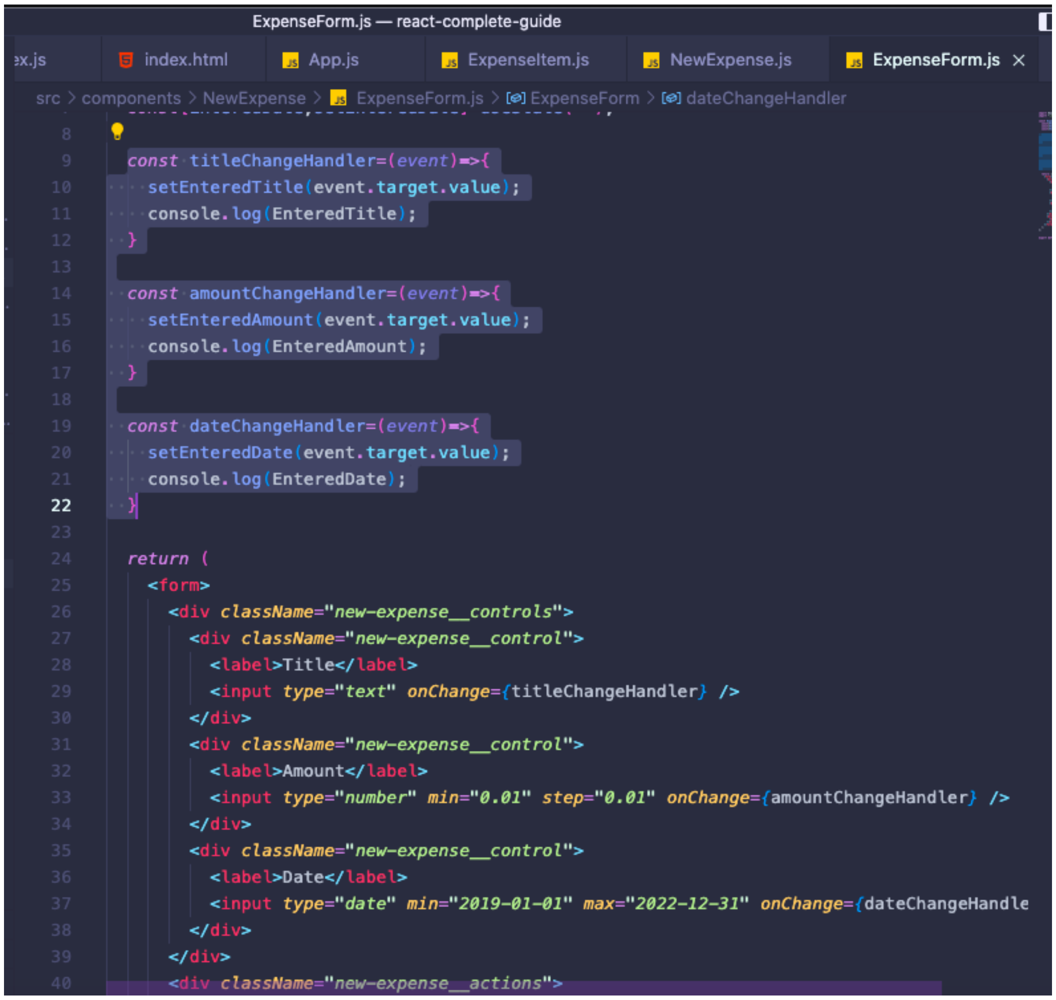
13. Multiple states in a single component

Approach - I (Using separate variables for each state variables**)**

- We are having three state variables
  - 
    ExpenseForm.js — react-complete-guide

    ex.js
index.html
App.js
ExpenseItem.js
Ne
src > components > NewExpense > ExpenseForm.js > ExpenseForm

```
import React,{useState} from "react";
import "./ExpenseForm.css";

const ExpenseForm = () => {
  const[EnteredTitle,setEnteredTitle]=useState('');
  const[EnteredAmount,setEnteredAmount]=useState('');
  const[EnteredDate,setEnteredDate]=useState('');

```
- Now let us add eventlistener for each input fields


ExpenseForm.js — react-complete-guide

ex.js | index.html | App.js | ExpenseItem.js | NewExpense.js | ExpenseForm.js

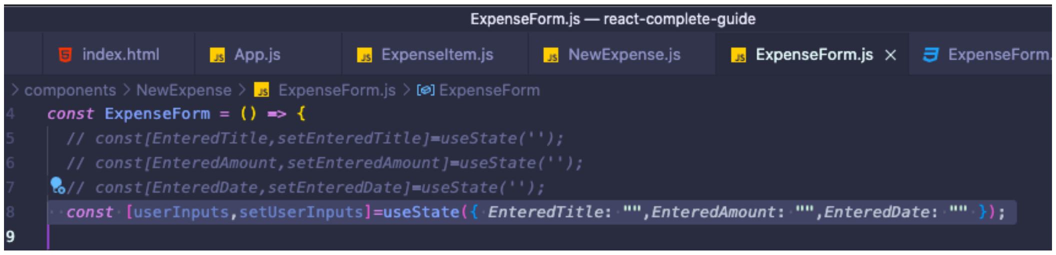
src > components > NewExpense > ExpenseForm.js > ExpenseForm > dateChangeHandler

```
  const titleChangeHandler=(event)=>{
    setEnteredTitle(event.target.value);
    console.log(EnteredTitle);
  }

  const amountChangeHandler=(event)=>{
    setEnteredAmount(event.target.value);
    console.log(EnteredAmount);
  }

  const dateChangeHandler=(event)=>{
    setEnteredDate(event.target.value);
    console.log(EnteredDate);
  }

  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="2022-12-31" onChange={dateChangeHandle
        </div>
      </div>
      <div className="new-expense__actions">
```

- Now we are updating the state variable using the functi setEnteredblablabla

APPROACH - II (using a single object for all state variables)

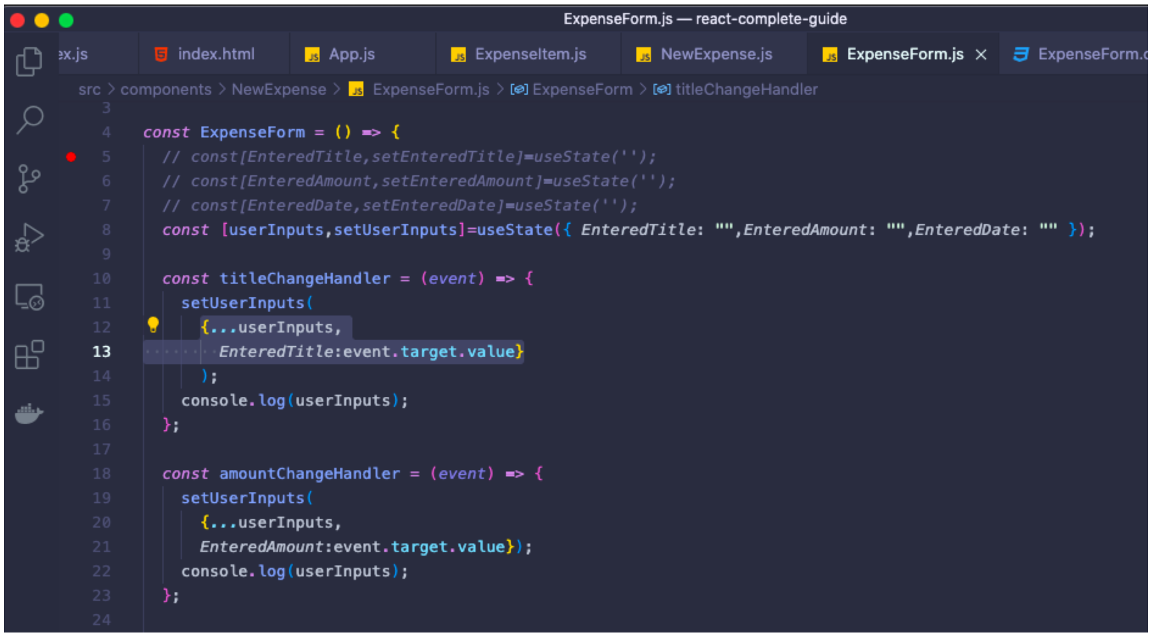
- Instead of using separate variables for all the three state variables, we are just having only one object.


ExpenseForm.js — react-complete-guide

index.html | App.js | ExpenseItem.js | NewExpense.js | ExpenseForm.js | ExpenseForm.

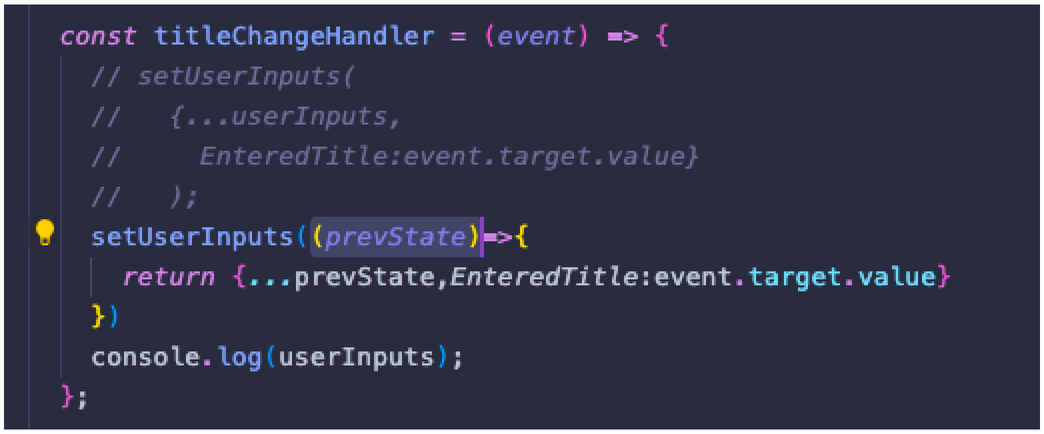
components > NewExpense > ExpenseForm.js > ExpenseForm

```
const ExpenseForm = () => {
  // const[EnteredTitle,setEnteredTitle]=useState('');
  // const[EnteredAmount,setEnteredAmount]=useState('');
  // const[EnteredDate,setEnteredDate]=useState('');
  const [userInputs,setUserInputs]=useState({ EnteredTitle: "",EnteredAmount: "",EnteredDate: "" });
```

- 
  ExpeneForm.js — react-complete-guide

  ex.js | index.html | App.js | ExpenseItem.js | NewExpense.js | ExpenseForm.js | ExpenseForm.c

  src > components > NewExpense > ExpenseForm.js > ExpenseForm > titleChangeHandler

  ```
  const ExpenseForm = () => {
    // const[EnteredTitle,setEnteredTitle]=useState('');
    // const[EnteredAmount,setEnteredAmount]=useState('');
    // const[EnteredDate,setEnteredDate]=useState('');
    const [userInputs,setUserInputs]=useState({ EnteredTitle: "",EnteredAmount: "",EnteredDate: "" });

    const titleChangeHandler = (event) => {
      setUserInputs(
        {...userInputs,
          EnteredTitle:event.target.value}
        );
      console.log(userInputs);
    };

    const amountChangeHandler = (event) => {
      setUserInputs(
        {...userInputs,
        EnteredAmount:event.target.value});
      console.log(userInputs);
    };

  ```
  - Here we are creating a object and by using spread operator(...) we are copying the userInpts object and just overriding the ENteredTitle value in the new object
- **NOTE : But this is not the proper way**
  - **React state, always schedules the state updation, here we have three state variables in a single state object, when there are many state changes, react schedules it, so it may not update instantly.**
  - **In this situation the userInputs object is not up to date, so copying userInput object using spread operator may lead to old data because it may take some more time to update because react schedules state changes, so we should use this below approach**
- 
  ```
  const titleChangeHandler = (event) => {
    // setUserInputs(
    //   {...userInputs,
    //     EnteredTitle:event.target.value}
    //   );
    setUserInputs((prevState)=>{
      return {...prevState,EnteredTitle:event.target.value}
    })
    console.log(userInputs);
  };
  ```
  - Here we are not directly returning the updated value, we are using a function inside the set function, that inner function accepts an argument which is given by the outer set function, that argument is the previous state of the state variable, which is up to date, the value is taken from reacts storage where it schedules the state change.
  -