20.working with multiple states(combining multiple states into one approach)

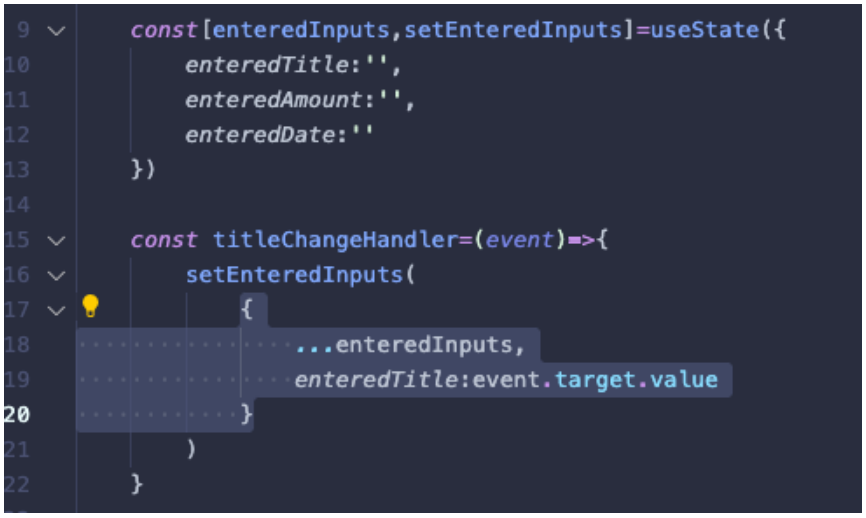
- In pervious lecture we followed the separate state maintenance approach
  - 

```
const ExpenseForm=()=>{
    const[enteredTitle,setEnteredTitle]=useState('');
    const[enteredAmount,setEnteredAmount]=useState('');
    const[enteredDate,setEnteredDate]=useState('');

    const titleChangeHandler=(event)=>{
        setEnteredTitle(event.target.value);
    }

    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">
```

- Now we are going to combine these three into one state and learn how to maintain it
  - We can create a object containing title,amount and date
  - This object should be a state object
  - 

```
const ExpenseForm=()=>{
    //const[enteredTitle,setEnteredTitle]=useState('');
    //const[enteredAmount,setEnteredAmount]=useState('');
    //const[enteredDate,setEnteredDate]=useState('');

    const[enteredInputs,setEnteredInputs]=useState({
        enteredTitle:'',
        enteredAmount:'',
        enteredDate:''
    })



    //const titleChangeHandler=(event)=>{
    //    setEnteredTitle(event.target.value);
    //}

    //const amountChangeHandler=(event)=>{
    //    setEnteredAmount(event.target.value);
    //}
```

- Now we want to handle this whenever each input field changes
- First let us handle the title field
  - 

  - This will called titleChangeHandler
- 
  - Now we have to change only the title inside the titlechangehandler
  - But at the same time we should not lose the other field's data
  - So we are using the spread operator to copy the enteredInputs object and then we are overriding the enteredTitle by the new value
- Similarly for amount and date
  - 

- **BUT THIS IS NOT THE BEST PRACTICE**
  - **RECALL:**
    - **WHEN A STATE'S SET METHOD(HERE SETENTEREDINPUTS) IS CALLED IT ACTUALLY DON'T UPDATE THE VALUE RIGHT AWAY IT JUST SCHEDULES THE UPDTE AND WHEN THE**

COMPONENT IS RE-EVALUATED AT THAT TIME ONLY IT GETS THE UPDATED VALUE

- **SO WHEN WE ARE HAVING MANY STATE UPDATES AT A SINGLE TIME, SOMETIME USING THAT SPREAD OPERATOR(HERE …enteredInputs) MAY GIVE AN OUTDATED VALUE**

- **So the best practice is**
  - **We know that the set function of state accepts an value or even a function**
  - **Here we should use a function inside the setEnteredInput function**

```
29    //const amountChangeHandler=(event)=>{
30        //setEnteredInputs(
31            // {
32                //...enteredInputs,
33                //enteredAmount:event.target.value
34            // }
35        //)
36    //}
37    const amountChangeHandler=(event)=>{
38        setEnteredInputs((prevState)=>{
39            return{
40                ...prevState,
41                enteredAmount:event.target.value
42            }
43        })
44    }
```

  -
  - HERE WE ARE DEFINING A FUNCTION INSIDE THE SETENTEREINPUTS FUNCTION
  - WHEN FUNCTION IS DEFINED INSIDE THE SET FUNCTION, IT PASSES THE PREVIOUS STATE TO THE INNERFUNCTION, SO THAT WE COULD ACCES THE PREVIOUS STATE
  - THE INNER FUNCTION COPIES THE PREVIOUS STATE OBJECT USING SPREAD OPERATOR WHICH WILL NOT LEAD TO ANY INCONSITENCIES UNLIKE COPYING THE OBJECT DIRECTLY
  - **IMPORTANTLY AFTER DOING THIS THEN THE INNER FUNCTION MUST RETURN THE UPDATED OBJECT TO THE SET FUNCTION**

NOTE:
I am going to switch back to separate maintenance approach for this project but both of these approaches are fine