29. Outputting conditional contents

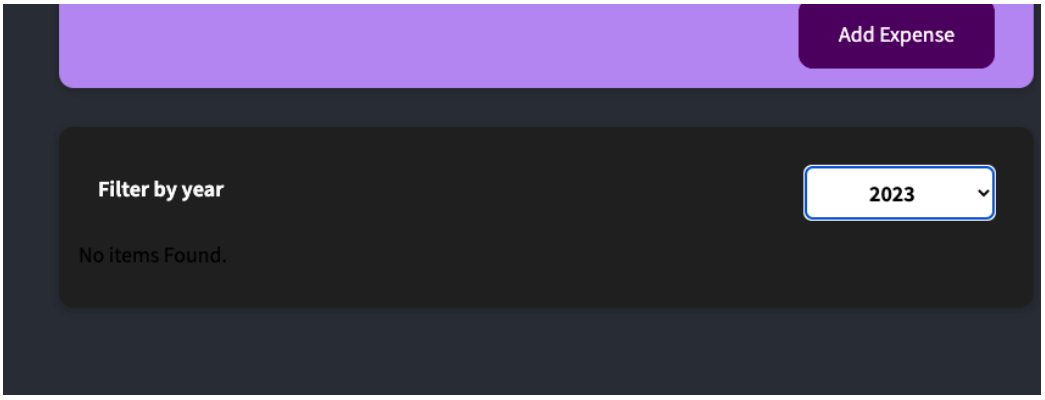
- In our app, suppose if we have no expenses for the pentered year then we don't want to leave those pages blank, instead we could display 'no items found'

- How can we do that?
  - Can we use if-else?
    - no , because inside JSX code, if and for conditions or not allowed because it is too long and its syntax has several curly braces inised it which is not allowed
  - Then how?
    - We could use ternary operator

-

```
src > components > Expenses > Expenses.js > Expenses
11
12    const changeFilterhandler=(selectedYear)=>{
13      setFilteredYear(selectedYear);
14    }
15    const filteredExpenses=props.items.filter((expense)=>{
16      return expense.date.getFullYear().toString() === filteredYear;
17    });
18
19    return (
20
21      <Card className="expenses">
22
23        <ExpenseFilter onChangeFilter={changeFilterhandler} selected={filteredYear} ></ExpenseFilter>
24        {filteredExpenses.length===0 ? (<p>No items Found.</p>) : (filteredExpenses.map((singleexpense)=>
25        <ExpenseItem key={singleexpense.id} title={singleexpense.title} amount={singleexpense.amount} date={singleexpens
26        }
27
28
29
30      </Card>
31    );
32  }
33
34  export default Expenses;
```

  - Here we have used ternary operator



- This ternary operator looks a bit long, we can use another trick
- 

```
src > components > Expenses > Expenses.js > Expenses
18
19   return (
20
21     <Card className="expenses">
22
23       <ExpenseFilter onChangeFilter={changeFilterhandler} selected={filteredYear} ></ExpenseFilter>
24       {filteredExpenses.length===0 && (<p>No items Found.</p>)}
25       {filteredExpenses.length>0 && (filteredExpenses.map((singleexpense)=>
26         <ExpenseItem key={singleexpense.id} title={singleexpense.title} amount={singleexpense.amount} date={singleexpense.
27       }
28
29
30
31
```

  - If we do like this
    - {condition && contents to be rendered}
  - This also works perfectly

We have another approach

- Here if you notice the jsx code is too complex and long
- So we can do this
- 

```
src > components > Expenses > Expenses.js > Expenses
15     const filteredExpenses=props.items.filter((expense)=>{
16       return expense.date.getFullYear().toString() === filteredYear;
17     });
18
19     let expenseContent=<p>No items Found.</p>
20     if(filteredExpenses.length>0){
21       expenseContent=filteredExpenses.map((singleexpense)=>
22                     <ExpenseItem
23                       key={singleexpense.id}
24                       title={singleexpense.title}
25                       amount={singleexpense.amount}
26                       date={singleexpense.date}>
27                     </ExpenseItem>)
28     }
29
30     return (
31
32       <Card className="expenses">
33         <ExpenseFilter onChangeFilter={changeFilterhandler} selected={filteredYear} ></ExpenseFilter>
34
35           let expenseContent: JSX.Element
36         {expenseContent}
37       </Card>
38     );
39   }
```

  - **<u>NOTE: We can use jsx like codes(eg. <p>) outside the return statement like this by assigning it to a variable</u>**

- After assigning the value of expensecontent based on the condition, we have used it in line 36 inside the return statement, so that it will be rendered.
- If the filtered year changes, because it is a state variable, it re-evaluates that component fully, so the expenseContent will be updated and kept consistent

-