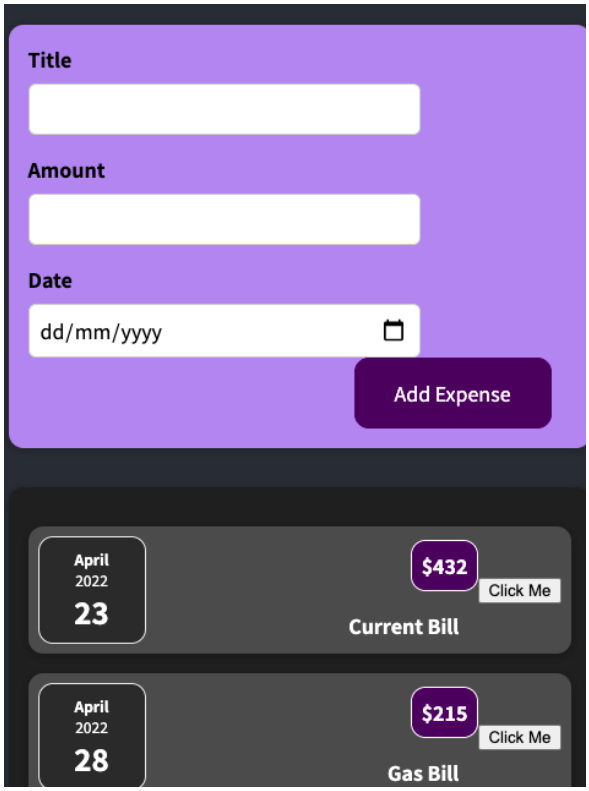
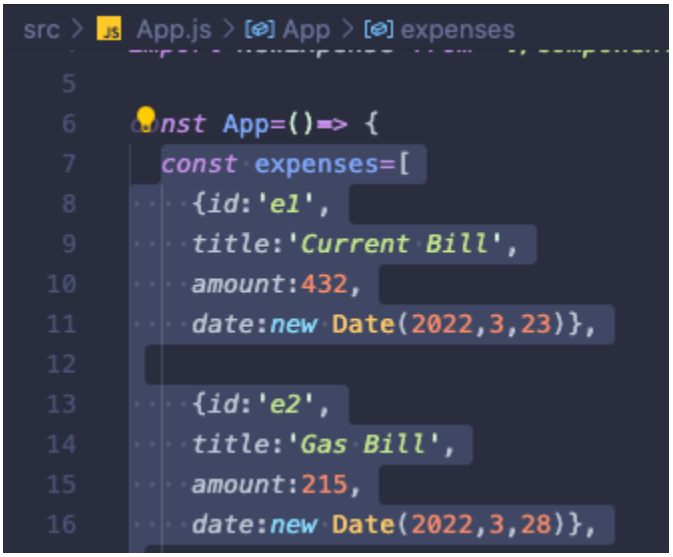
26. State Lists/Array

- Until now we have create a state variable but not a state list
- 

- Our aim:
  - If we add new expense it should be reflected below
- In app.js
  - 

  - Let us move this predefined dummy list outside the App function

- And let us rename it as
  - 
```
src > App.js > App
6   const PredefinedExpenses=[
7     {id:'e1',
8     title:'Current Bill',
9     amount:432,
10    date:new Date(2022,3,23)},
11
12    {id:'e2',
13    title:'Gas Bill',
14    amount:215,
15    date:new Date(2022,3,28)},
16
17    {id:'e3',
18    title:'College Fees',
19    amount:500,
20    date:new Date(2022,3,3)},
21
22    {id:'e4',
23    title:'Fuel Expense',
24    amount:200,
25    date:new Date(2022,3,29)},
26  ]
27
28  const App=()=> {
29    const AddExpensehandler=(newExpenseData)=>{
30        console.log(newExpenseData);
31    }
32
33    return (
```
- Now inside the app let us a create a state list which will have the initial values as predefinedExpenses list
  - 
```
src > App.js > App
22    {id:'e4',
23    title:'Fuel Expense',
24    amount:200,
25    date:new Date(2022,3,29)},
26  ]
27
28  const App=()=> {
29    const [expenses,setExpenses]=useState(PredefinedExpenses);
30
31    const AddExpensehandler=(newExpenseData)=>{
32        console.log(newExpenseData);
33    }
34
35    return (
36
37      <div>
38        <NewExpense onAddExpense={AddExpensehandler}></NewExpense>
39        <Expenses items={expenses}></Expenses>
40      </div>
41    );
42  }
43
44  export default App;
```

- Here we have created a statelist
- We have named it as expenses because we have used that name previously for the predefinedData and we passed it in line 39

- Now let us recall how the newly added expense data comes from expense form to app.js
  - 

ExpenseForm.js
  - 

NewExpense.js

- App.js

```
src > App.js > App > AddExpensehandler
22    {id:'e4',
23    title:'Fuel Expense',
24    amount:200,
25    date:new Date(2022,3,29)},
26  ]
27
28  const App=()=> {
29    const [expenses,setExpenses]=useState(PredefinedExpenses);
30
31    const AddExpensehandler=(newExpenseData)=>{
32        console.log(newExpenseData);
33    }
34
35    return (
36
37      <div>
38        <NewExpense onAddExpense={AddExpensehandler}></NewExpense>
39        <Expenses items={expenses}></Expenses>
40      </div>
41    );
42  }
43
44  export default App;
45
```

- Here instead of printing the entered data let us add it to the expenses list

```
src > App.js > App > AddExpensehandler
22    {id:'e4',
23    title:'Fuel Expense',
24    amount:200,
25    date:new Date(2022,3,29)},
26    ]
27
28    const App=()=> {
29      const [expenses,setExpenses]=useState(PredefinedExpenses);
30
31      const AddExpensehandler=(newExpenseData)=>{
32          setExpenses(newExpenseData,...expenses);
33      }
34
35      return (
36
37        <div>
38          <NewExpense onAddExpense={AddExpensehandler}></NewExpense>
39          <Expenses items={expenses}></Expenses>
40        </div>
41      );
```

```
const App=()=> {
  const [expenses,setExpenses]=useState(PredefinedExpenses);

  const AddExpensehandler=(newExpenseData)=>{
    setExpenses( [...expenses,newExpenseData] )
  }

  return (

    <div>
      <NewExpense onAddExpense={AddExpensehandler}></NewExpense>
```

- **BUT THIS IS NOT THE CORRECT WAY OF DOING THIS, BECAUSE WE ARE COPYING THE OLD VALUE OF THAT LIST AND ALSO ADDING A NEW ONE, WE KNOW THAT THE SET FUNCTION DO NOT CHANGE THE STATE VARAIBLE RIGHT AWAY IT JUST SCHEDULES THE CHANGE**
- **SO WE HAVE TO GET THE PREVIOUS STATE AND COPY IT SO THER WILL BE NO INCONSISTENCY**

- We know that the set function can be used in two ways
  - One is we can directly pass the new value for that state variable, which we have done above
  - Second one is, we could pass or define a function inside the set function, and the inner function receives an parameter by the set function, that parameter is the previous state of that state variable

```
const App=()=> {
  const [expenses,setExpenses]=useState(PredefinedExpenses);

  const AddExpensehandler=(newExpenseData)=>{
    setExpenses((prevState)=>{
      return [newExpenseData,...prevState];
    });
  }
```

- **Here the inner function must return to the setExpense function.**