

26. State Lists/Array

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

The image shows a mobile application interface for managing expenses. At the top, there is a purple header containing three input fields: 'Title', 'Amount', and 'Date' (with a date picker icon). Below these fields is a purple button labeled 'Add Expense'. The main content area is dark gray and displays two expense cards. The first card, titled 'Current Bill', shows a date of 'April 2022 23' and an amount of '\$432' with a 'Click Me' button. The second card, titled 'Gas Bill', shows a date of 'April 2022 28' and an amount of '\$215' with a 'Click Me' button.

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

```
src > App.js > [App] > [expenses]
5
6  const App=()=> {
7    const expenses=[
8      {id:'e1',
9        title:'Current Bill',
10       amount:432,
11       date:new Date(2022,3,23)},
12
13     {id:'e2',
14       title:'Gas Bill',
15       amount:215,
16       date:new Date(2022,3,28)},
```

-
- 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 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 statelists
- 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

```
src > components > ExpenseForm > ExpenseForm.js > ExpenseForm > ExpenseForm.js
18 |     setEnteredDate(event.target.value);
19 |   }
20 |   const submitHandler: (event: any) => void
21 |   const submitHandler=(event)=>{
22 |     event.preventDefault();
23 |
24 |     const expenseData={
25 |       title:enteredTitle,
26 |       amount:enteredAmount,
27 |       date:new Date(enteredDate)
28 |     }
29 |     //console.log(expenseData);
30 |     props.onSaveExpenseData(expenseData);
31 |
32 |     setEnteredTitle('');
33 |     setEnteredAmount('');
34 |     setEnteredDate('');
35 |   }
36 |
37 |   return (
38 |     <form onSubmit={submitHandler}>
39 |       <div className='new-expense__controls'>
```

ExpenseForm.js

```
src > components > ExpenseForm > NewExpense.js > NewExpense > onSaveExpenseDataHandler
1 | import React from 'react';
2 | import ExpenseForm from './ExpenseForm';
3 | import './NewExpense.css';
4 |
5 | const NewExpense=(props)=>{
6 |   const onSaveExpenseDataHandler=(enteredExpenseData)=>{
7 |     //console.log(enteredExpenseData);
8 |     const newExpenseData={
9 |       ...enteredExpenseData,
10 |       id:Math.random().toString()
11 |     };
12 |     props.onAddExpense(newExpenseData);
13 |   }
14 |
15 |   return (
16 |     <div className='new-expense'>
17 |       <ExpenseForm onSaveExpenseData={onSaveExpenseDataHandler}></ExpenseForm>
18 |     </div>
19 |   )
20 |
21 | }
22 |
23 |
24 |
25 | export default NewExpense;
```

NewExpense.js

```

src > App.js > [0] App > [0] 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

```

App.js

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

```

src > App.js > [0] App > [0] 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     );

```

```

27
28  const App=()=> {
29      const [expenses,setExpenses]=useState(PredefinedExpenses);
30
31      const AddExpensehandler=(newExpenseData)=>{
32          ⚡ | setExpenses([...expenses,newExpenseData])
33      }
34
35      return (
36
37          <div>
38              <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 VARIABLE 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

```

28  const App=()=> {
29      const [expenses,setExpenses]=useState(PredefinedExpenses);
30
31      const AddExpensehandler=(newExpenseData)=>{
32          ⚡ | setExpenses((prevState)=>{
33              ...return [newExpenseData,...prevState];
34          });
35      }

```

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