

17. Statefull lists(updating the Expenses list when new expense item is added)

- Until now, in app.js we have a static list

```
App.js  ExpenseItem.js  ExpenseFilter.js  ExpenseFilter.css  NewExpense.js

src > App.js > App > addExpenseHandler
3
4  function App() {
5    const expenses = [
6      { id:"e1", title: "House Rent", amount: 99.0, date: new Date(2023, 7, 29) },
7      { id:"e2", title: "Grocery", amount: 20, date: new Date(2023, 7, 2) },
8      { id:"e3", title: "School fees", amount: 125.5, date: new Date(2023, 7, 18) },
9    ];
10
11    const addExpenseHandler=(expenseData)=>{
12      console.log("in app.js",expenseData);
13    }
14
15    return (
16      <div>
17        <NewExpense onAddExpense={addExpenseHandler} ></NewExpense>
18        <Expenses items={expenses} />
19      </div>
20    );
21  }
22
23  export default App;
24
25
```

- Here we are just console logging the newly entered data from other component
- Instead of console.log we have to update the expenses list so that it should display the new list in our page
- For this, let us have a initial list,

```
App.js  ExpenseItem.js  ExpenseFilter.js  ExpenseFilter.css  NewExpense.js

src > App.js > ...
1  import Expenses from './components/Expenses/Expenses';
2  import NewExpense from './components/NewExpense/NewExpense'
3
4  INITIAL_EXPENSES=[
5    { id:"e1", title: "House Rent", amount: 99.0, date: new Date(2023, 7, 29) },
6    { id:"e2", title: "Grocery", amount: 20, date: new Date(2023, 7, 2) },
7    { id:"e3", title: "School fees", amount: 125.5, date: new Date(2023, 7, 18) },
8  ]
9
10 function App() {
11   const expenses = [
12     { id:"e1", title: "House Rent", amount: 99.0, date: new Date(2023, 7, 29) },
13     { id:"e2", title: "Grocery", amount: 20, date: new Date(2023, 7, 2) },
14     { id:"e3", title: "School fees", amount: 125.5, date: new Date(2023, 7, 18) },
15   ]
16
17   return (
18     <div>
19       <NewExpense onAddExpense={addExpenseHandler} ></NewExpense>
20       <Expenses items={expenses} />
21     </div>
22   );
23 }
24
25 export default App;
```

- Now let us, create a state variable to hold a list,

```
App.js  ExpenseItem.js  ExpenseFilter.js  ExpenseFilter.css  NewExpense.js

src > App.js > App
4
5  let INITIAL_EXPENSES=[
6    { id:"e1", title: "House Rent", amount: 99.0, date: new Date(2023, 7, 29) },
7    { id:"e2", title: "Grocery", amount: 20, date: new Date(2023, 7, 2) },
8    { id:"e3", title: "School fees", amount: 125.5, date: new Date(2023, 7, 18) },
9  ]
10
11  function App() {
12    // const expenses = [
13    //   { id:"e1", title: "House Rent", amount: 99.0, date: new Date(2023, 7, 29) },
14    //   { id:"e2", title: "Grocery", amount: 20, date: new Date(2023, 7, 2) },
15    //   { id:"e3", title: "School fees", amount: 125.5, date: new Date(2023, 7, 18) },
16    // ];
17    const [expenses, setExpenses]=useState(INITIAL_EXPENSES);
18
19    const addExpenseHandler=(expenseData)=>{
20      console.log("in app.js",expenseData);
21    }
22
23    return (
24      <div>
25
26        <NewExpense onAddExpense={addExpenseHandler} ></NewExpense>
27        <Expenses items={expenses} />
28      </div>
29    );
30  }
```

-
- Now instead of logging the newly entered expense item, we have to update the state variable

```
App.js x ExpenseItem.js ExpenseFilter.js ExpenseFilter.css NewExpense.js
src > App.js > App > [e] addExpenseHandler
11 function App() {
12   // const expenses = [
13   //   { id:"e1", title: "House Rent", amount: 99.0, date: new Date(2023, 7, 29) },
14   //   { id:"e2", title: "Grocery", amount: 20, date: new Date(2023, 7, 2) },
15   //   { id:"e3", title: "School fees", amount: 125.5, date: new Date(2023, 7, 18) },
16   // ];
17   const [expenses, setExpenses]=useState(INITIAL_EXPENSES);
18
19   const addExpenseHandler=(expenseData)=>{
20     // console.log("in app.js",expenseData);
21     // setExpenses([expenseData,...expenses]);
22     setExpenses((prevState)=>[expenseData,...prevState]);
23   }
24
25   return (
26     <div>
27       <NewExpense onAddExpense={addExpenseHandler} ></NewExpense>
28       <Expenses items={expenses} />
29     </div>
30   );
31 }
32
33
34 export default App;
35
```

- - As we seen earlier, this is the proper method to update the state variable because react just schedules the state change, so to keep the data shown in frontend consistent we need to use the latest snapshot of the state.
- **NOTE:**
 - It may not work properly if we haven't done this yet

The screenshot shows the Chrome DevTools Console with the 'Console' tab selected. At the top, there's a message from HMR: '[HMR] Waiting for update signal from WDS...' and a link to 'log.js:24'. Below that, a message from 'react-dom.development.js:29840' suggests downloading React DevTools. The main part of the console shows a red warning: 'Warning: Each child in a list should have a unique "key" prop.' It provides a link to 'https://reactjs.org/link/warning-keys' and lists the stack trace: 'at ExpenseItem (http://localhost:3000/static/js/main.chunk.js:96:8:90)', 'at Expenses (http://localhost:3000/static/js/main.chunk.js:1238:97)', 'at div', and 'at App (http://localhost:3000/static/js/main.chunk.js:230:89)'. Below the warning, there are four log entries: 'ExpenseItem Executed' (ExpenseItem.js:14), 'on change executed' (ExpenseForm.js:11), 'ExpenseItem Executed' (ExpenseItem.js:14), and 'ExpenseItem Executed' (ExpenseItem.js:14). The console ends with a blue prompt character '>'.

- So let us add key, while rendering the list

The screenshot shows a VS Code editor with the 'Expenses.js' file open. The file is part of a project named 'react-complete-guide'. The editor shows the following code:

```
17 <div>
18   <Card className="expenses">
19     <ExpenseFilter selected={FilteredYear} onFilterChange={filterChangeHandler} ></ExpenseFilter>
20     {
21       // <ExpenseItem title={props.items[0].title} amount={props.items[0].amount} date={props.items[0].date} />,
22       // <ExpenseItem title={props.items[1].title} amount={props.items[1].amount} date={props.items[1].date} />,
23       // <ExpenseItem title={props.items[2].title} amount={props.items[2].amount} date={props.items[2].date} />
24     }
25
26     {props.items.map((expense)=><ExpenseItem key={expense.id} title={expense.title} amount={expense.amount} date={expense.date} />)}
27
28   </Card>
29 </div>
30
31 </div>
32
33 </div>
34
35 export default Expenses;
```