Front End

Class 6

August 18, 2021

Agenda

- Kahoot
- Code walkthrough with slides
- Studio

Rendering Lists of Data

Branch lists

Remove the code that hardcodes the expenses array and instead display a number of ExpenseItem's based on the number of expenses. Array of JSX items.

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map

```
<ExpenseItem
  title={expenses[0].title}
  amount={expenses[0].amount}
  date={expenses[0].date}
/>
<ExpenseItem
  title={expenses[1].title}
  amount={expenses[1].amount}
  date={expenses[1].date}
/>
<ExpenseItem
  title={expenses[2].title}
  amount={expenses[2].amount}
  date={expenses[2].date}
/>
<ExpenseItem
  title={expenses[3].title}
  amount={expenses[3].amount}
  date={expenses[3].date}
/>
```

```
{props.expenses.map((expense) => {
  return (
    <ExpenseItem
      title={expense.title}
      amount={expense.amount}
      date={expense.date}
    />
  );
})}
```

Updating Expenses

Just updating expenses will not update the ui, we need to use state.

```
const [expenses, setExpenses] = useState(DUMMY_EXPENSES);

const addExpenseHandler = (expense) => {
  setExpenses([expense, ...expenses]);      You, 2 weeks ago • initial
};
return (
  <div className="App">
    <NewExpense onAddExpense={addExpenseHandler} />
    <Expenses expenses={expenses} />
  </div>
);
};
```

Updating Expenses

rc > App.js > App > DUMMY_EXPENSES

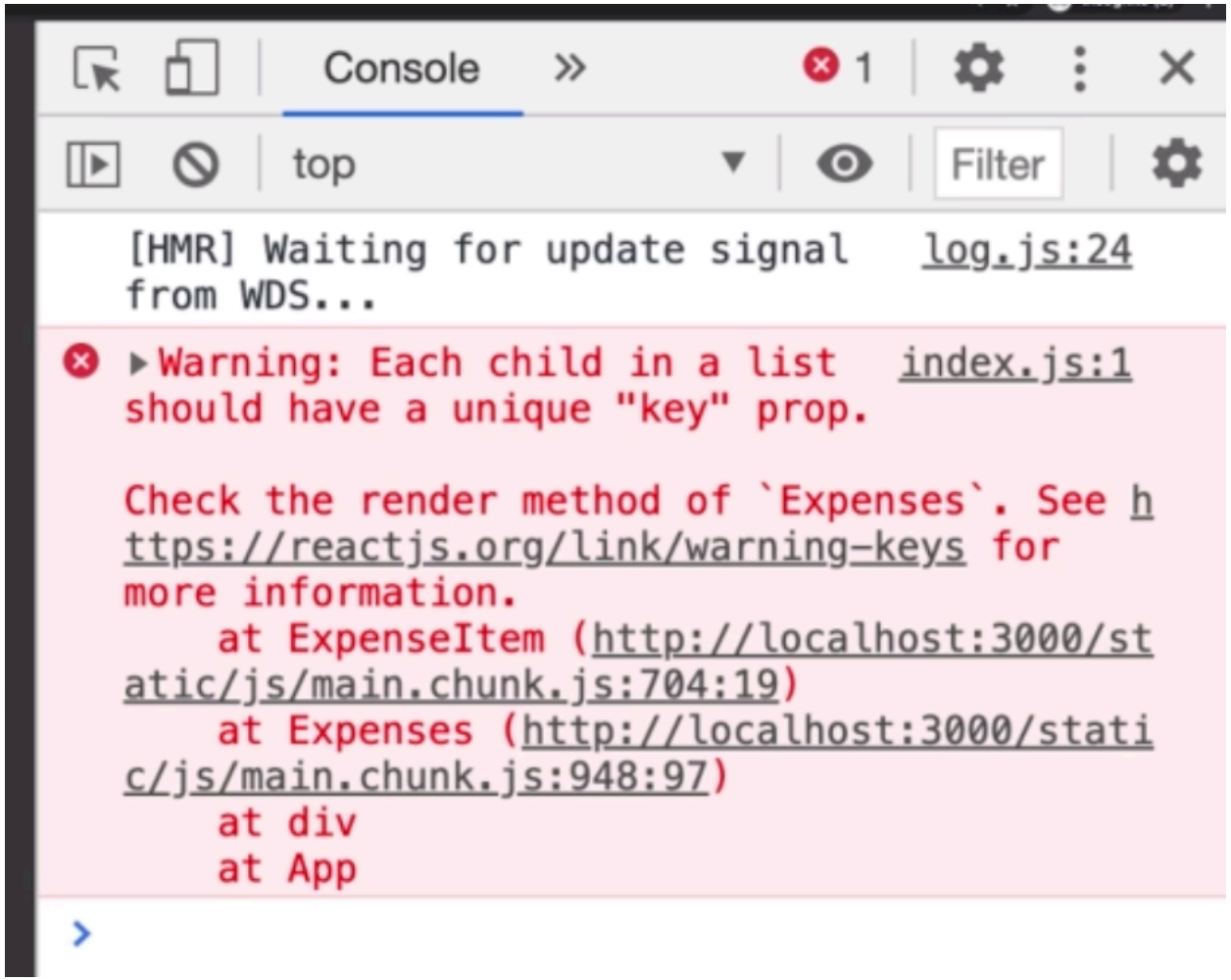
You, 7 minutes ago | 1 author (You)

```
import logo from "./logo.svg";
import "./App.css";
import Expenses from "./components/Expenses/Expenses";

import NewExpense from "./components/NewExpense/NewExpense";
import { useState } from "react";

const App = () => {
  const DUMMY_EXPENSES = [
    {
      id: "e1",      You, 3 weeks ago • added
      title: "Toilet Paper",
      amount: 94.12,
      date: new Date(2020, 7, 14),
    },
    { id: "e2", title: "New TV", amount: 799.49, date: new Date(2021, 2, 12) },
    {
      id: "e3",
      title: "Car Insurance",
      amount: 294.67,
      date: new Date(2021, 2, 28),
    },
    {
      id: "e4",
      title: "New Desk (Wooden)",
      amount: 450,
      date: new Date(2021, 5, 12),
    },
  ];
```

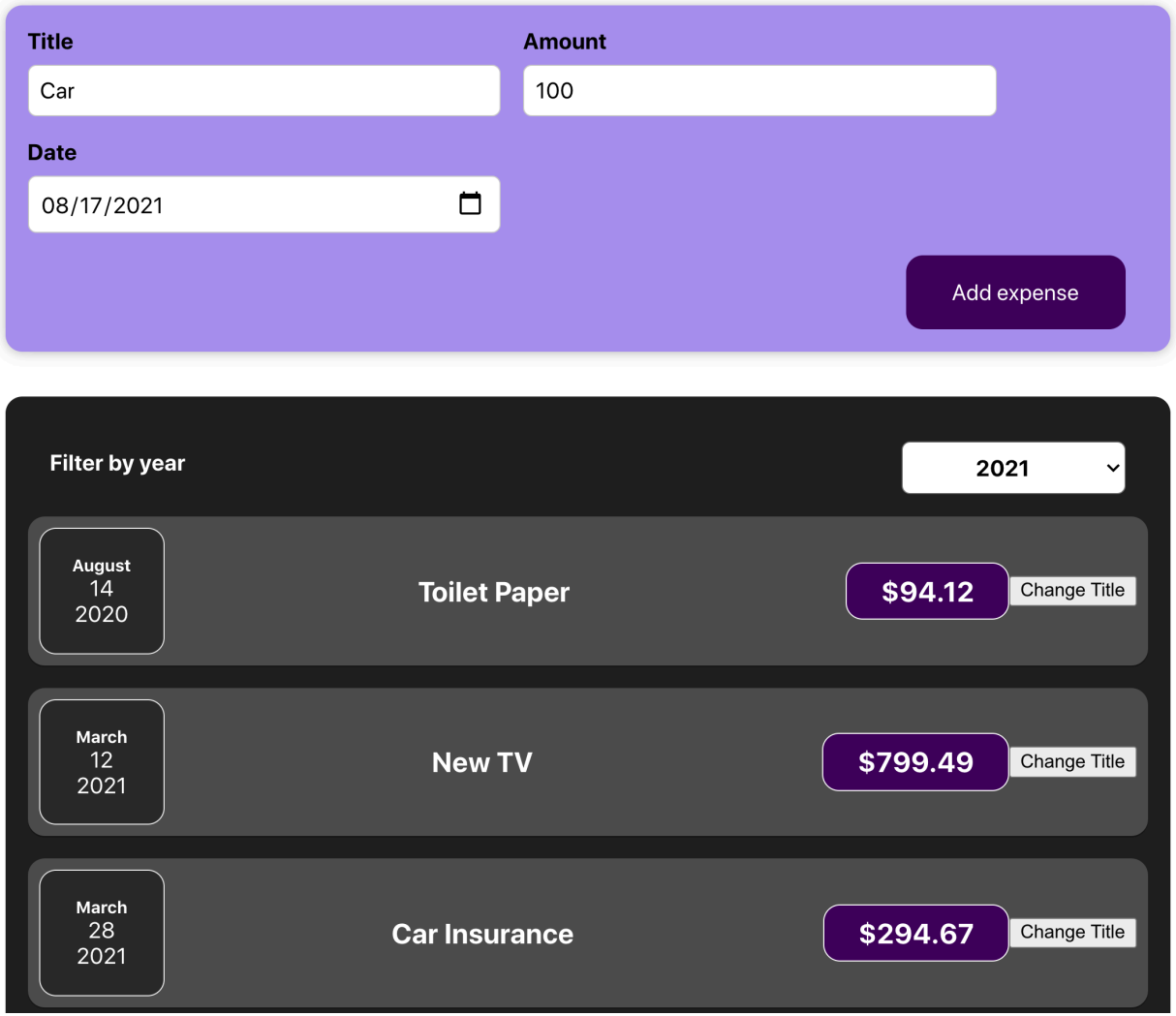
Updating Expenses

```
  const [expenses, setExpenses] = useState(DUMMY_EXPENSES);

  const addExpenseHandler = (expense) => {
    setExpenses((prevExpenses) => {
      return [expense, ...prevExpenses];
    });
  };
  return (
    <div className="App">
      <NewExpense onAddExpense={addExpenseHandler} />
      <Expenses expenses={expenses} />
    </div>
  );
};

export default App;
```

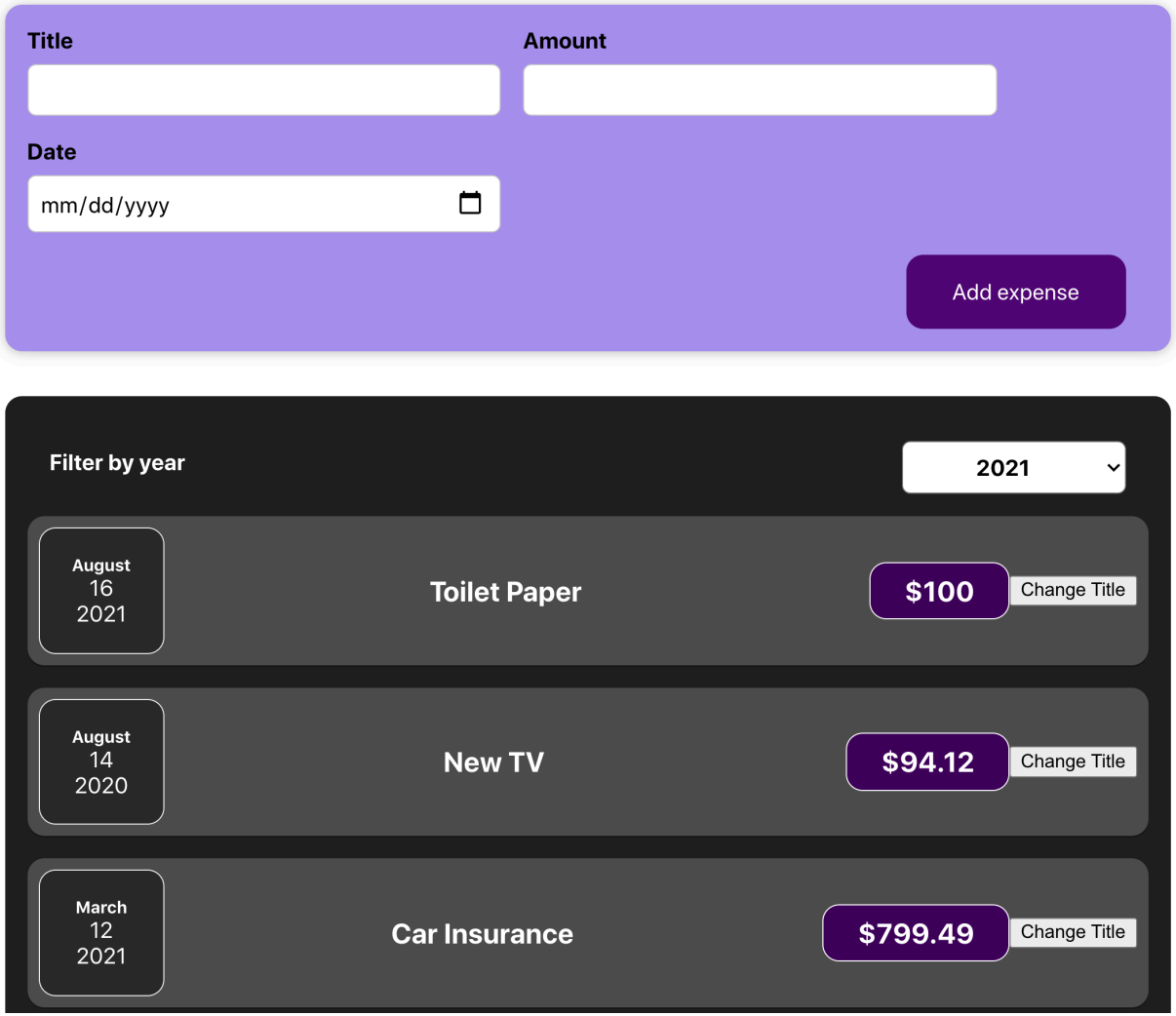
Error


Console
1
top
Filter
[HMR] Waiting for update signal from WDS...
log.js:24
Warning: Each child in a list should have a unique "key" prop.
index.js:1
Check the render method of `Expenses`. See https://reactjs.org/link/warning-keys for more information.
at ExpenseItem (http://localhost:3000/static/js/main.chunk.js:704:19)
at Expenses (http://localhost:3000/static/js/main.chunk.js:948:97)
at div
at App

Screenshot

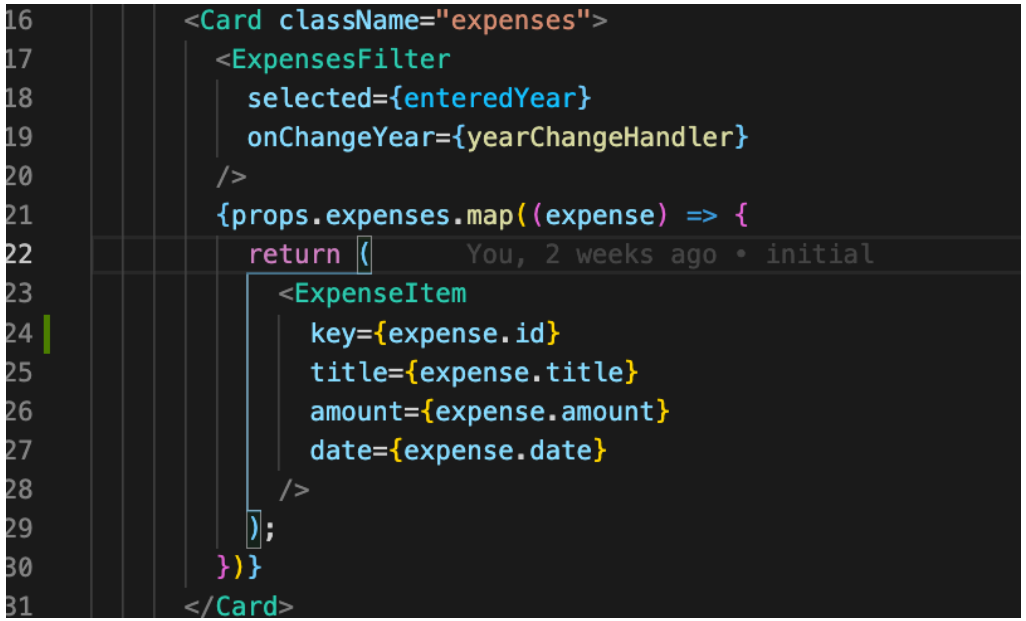

Title
Car
Amount
100
Date
08/17/2021
Add expense
Filter by year
2021
August
14
2020
Toilet Paper
$94.12
Change Title
March
12
2021
New TV
$799.49
Change Title
March
28
2021
Car Insurance
$294.67
Change Title

Screenshot

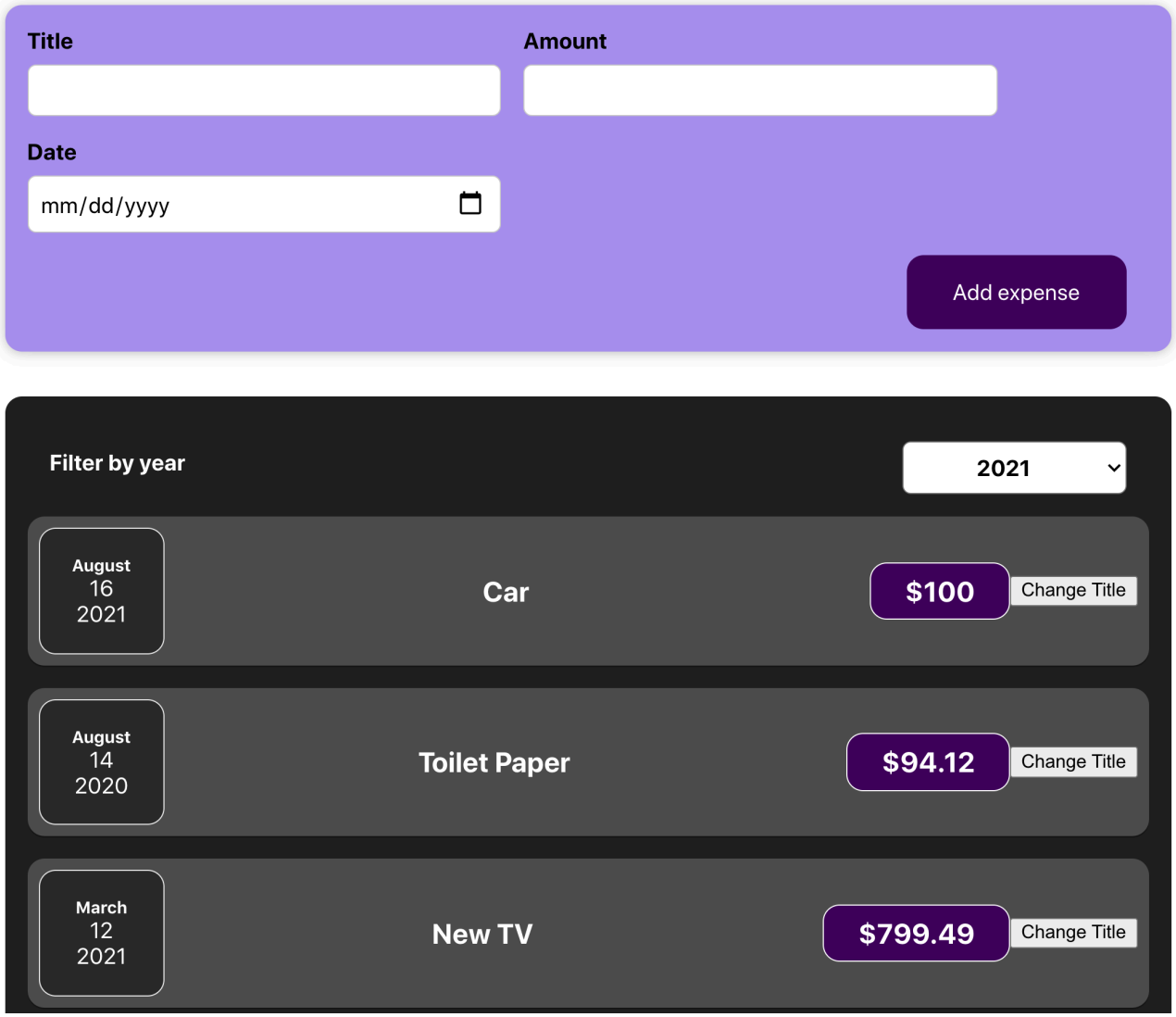

Title
Amount
Date
mm/dd/yyyy
Add expense
Filter by year
2021
August
16
2021
Toilet Paper
$100
Change Title
August
14
2020
New TV
$94.12
Change Title
March
12
2021
Car Insurance
$799.49
Change Title

Keys

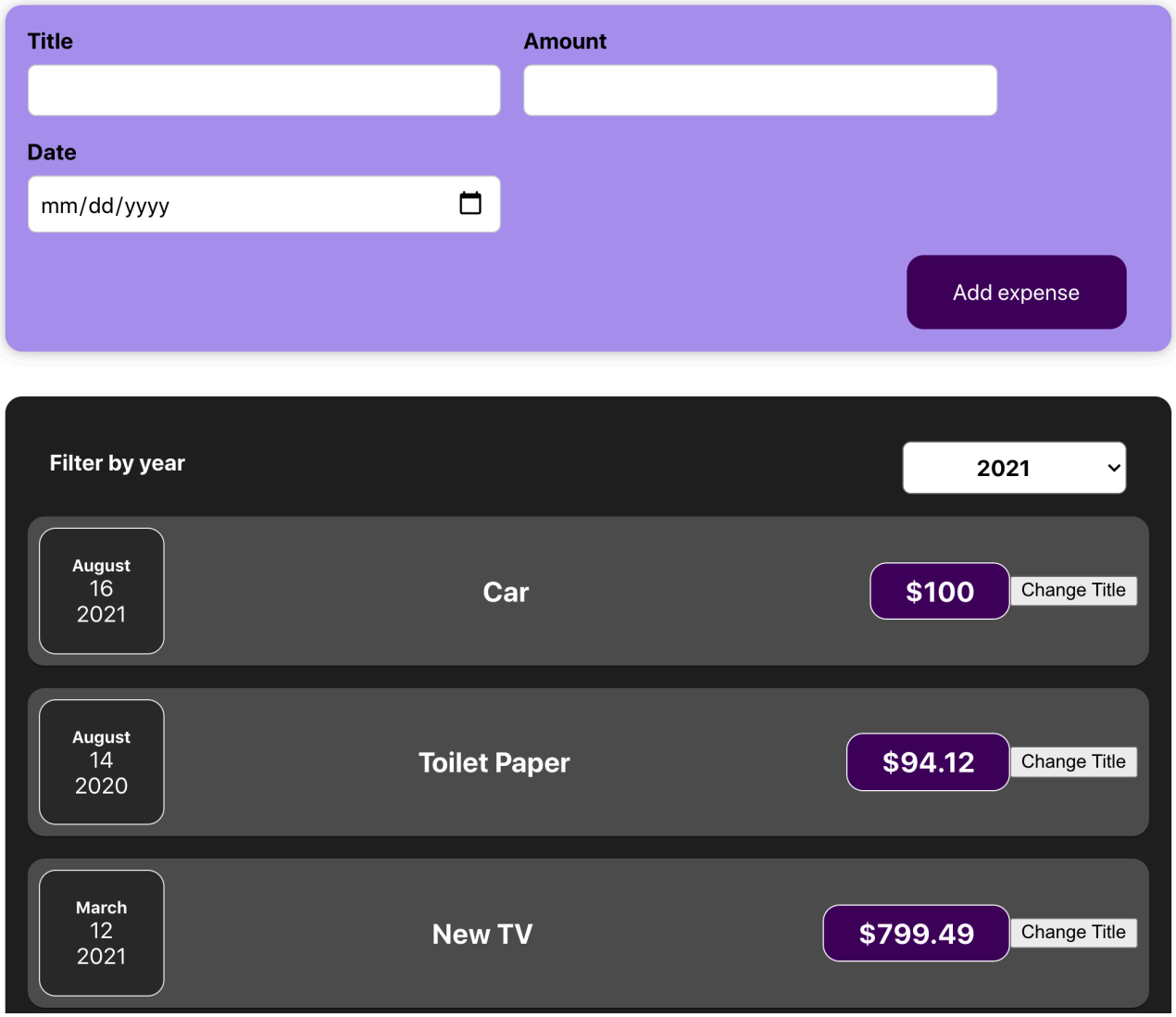
Adding new items means updating an item or adding an item updates all items and can also be buggy with respect to state changes getting lost.
To address that problem that is because it checks the length of items.
Special prop called key that can be added to any component.
Unique item per list item. In our case it will be id.
Can use index to automatically be added.



Screenshot


Title
Amount
Date
mm/dd/yyyy
Add expense
Filter by year
2021
August
16
2021
Car
$100
Change Title
August
14
2020
Toilet Paper
$94.12
Change Title
March
12
2021
New TV
$799.49
Change Title

Screenshot


Title
Amount
Date
mm/dd/yyyy
Add expense
Filter by year
2021
August
16
2021
Car
$100
Change Title
August
14
2020
Toilet Paper
$94.12
Change Title
March
12
2021
New TV
$799.49
Change Title

Assignment 3 : Working with lists

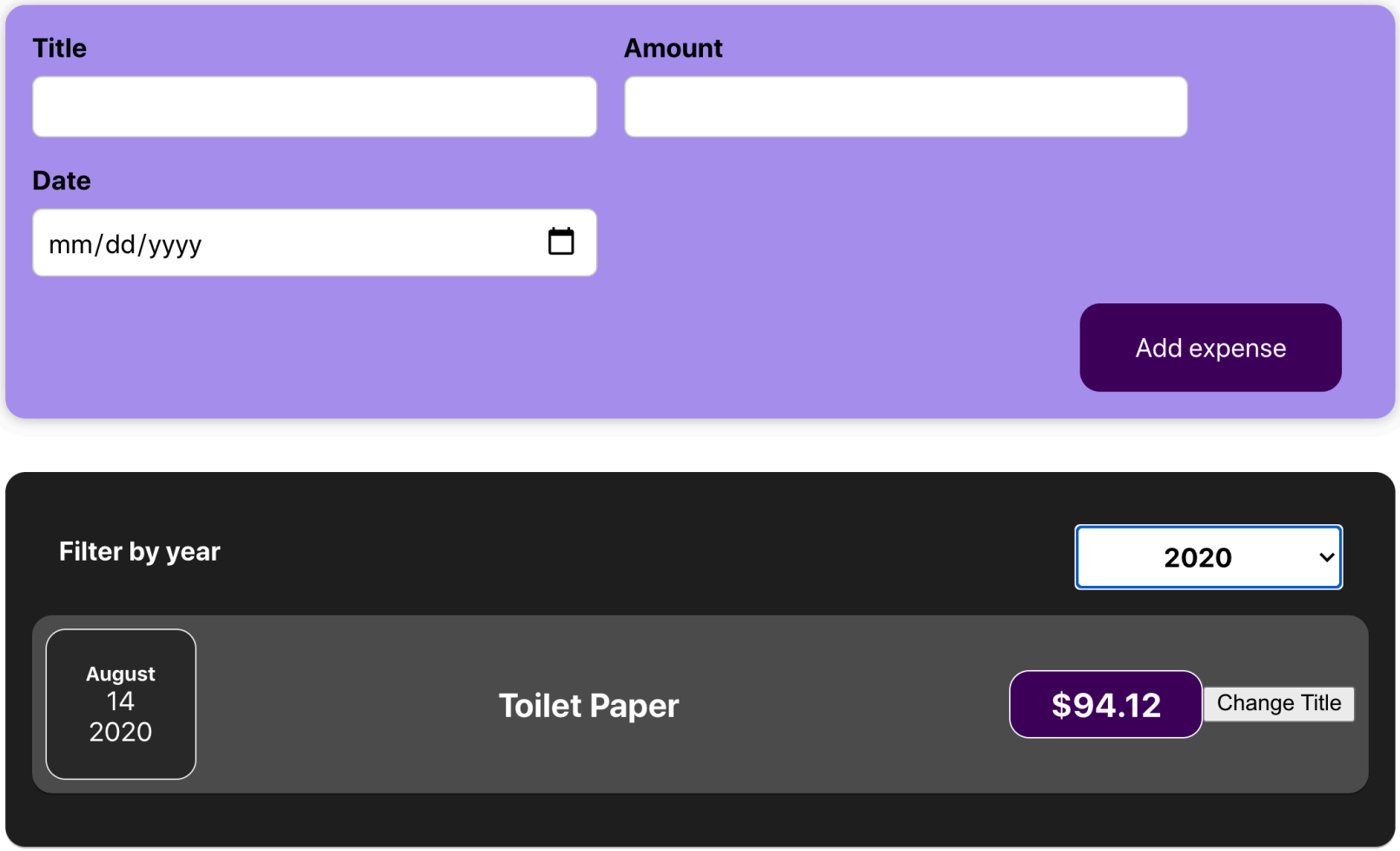
Added new branch called assignment3

Challenge is to make filter work. List is filtered correctly so when year is picked display only those items
Use Array.filter method.
Don’t change overall expenses array but derive new array that is a subset of the array based on filter.
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/filter

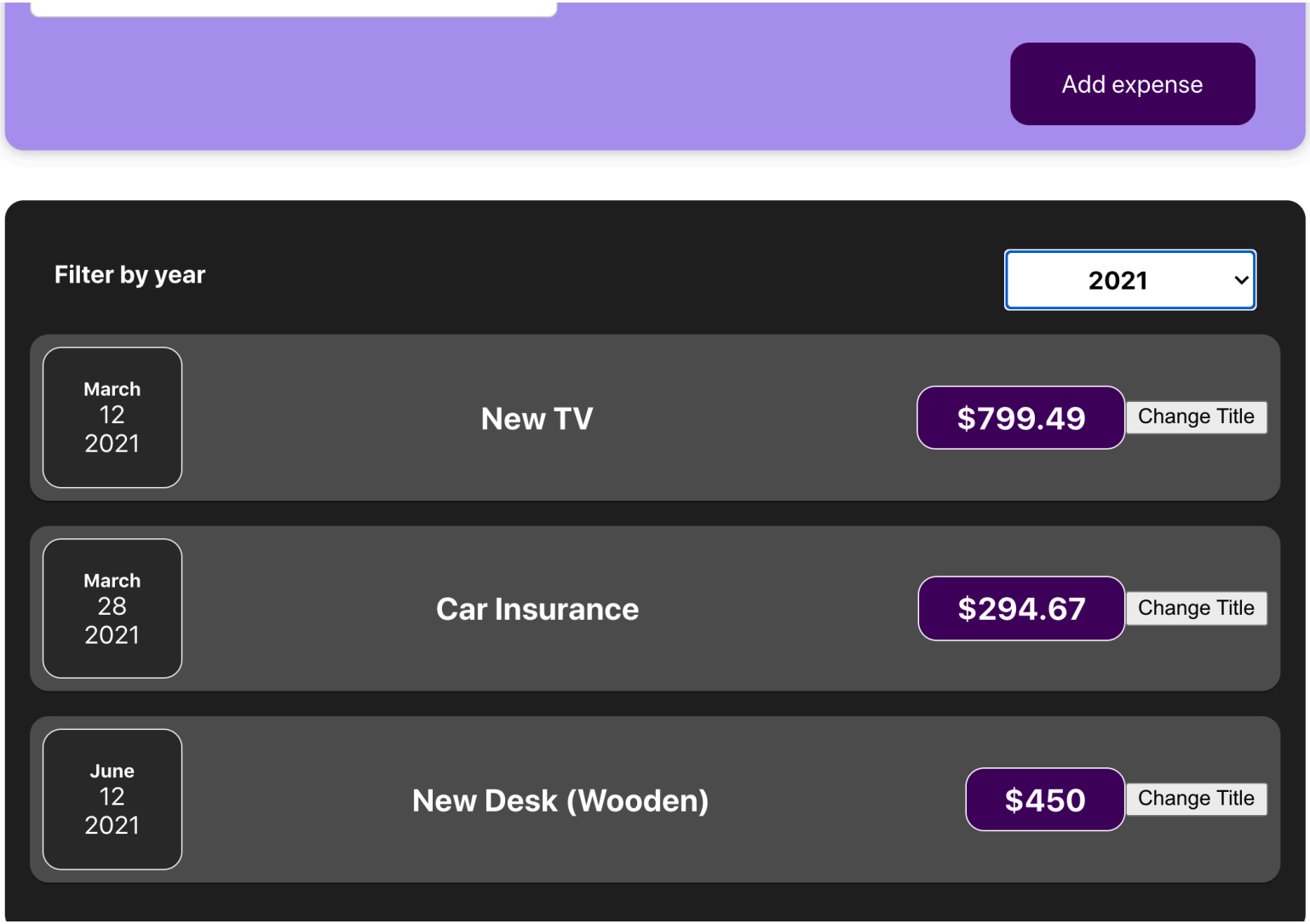
Assignment 3 : Working with lists

```
  const filteredExpenses = props.expenses.filter((expense) => {
    return expense.date.getFullYear().toString() === enteredYear;
  });
  return (
    <div>
      <Card className="expenses">
        <ExpensesFilter      You, 2 weeks ago • solution
          selected={enteredYear}
          onChangeYear={yearChangeHandler}
        />
        {filteredExpenses.map((expense) => {
          console.log("year is ", enteredYear);
          return (
            <ExpenseItem
              key={expense.id}
              title={expense.title}
              amount={expense.amount}
              date={expense.date}
            />
          );
        })}
      </Card>
    </div>
  );
};

export default Expenses;
```

Assignment 3 : Working with lists



Assignment 3 : Working with lists



Outputting Conditional Content – new branch class6

Conditional output is rendering different content under different conditions.
For example, if there is no data for a year, we can show some message.

Can use ternary expression.

```
/>

{filteredExpenses.length === 0 ? (
  <p>No expenses found.</p>
) : (
  filteredExpenses.map((expense) => (
    <ExpenseItem
      key={expense.id}
      title={expense.title}
      amount={expense.amount}
      date={expense.date}
    />
  ))
```

Outputting Conditional Content – new branch class6

Trick in JS – if you use && operator then second par is returned when first condition is true.

```
/>
{filteredExpenses.length === 0 && <p>No expenses found.</p>}
{filteredExpenses.length > 0 &&
  filteredExpenses.map((expense) => (
    <ExpenseItem
      key={expense.id}
      title={expense.title}
      amount={expense.amount}
      date={expense.date}
    />
  ))}
```

Outputting Conditional Content – new branch class6

Instead add a variable called expensesContent.

```
let expensesContent = <p>No expenses found.</p>;
if (filteredExpenses.length > 0) {
  expensesContent = filteredExpenses.map((expense) => (
    <ExpenseItem
      key={expense.id}
      title={expense.title}
      amount={expense.amount}
      date={expense.date}
    />
  ));
}
return (
  <div>
    <Card className="expenses">
      <ExpensesFilter      You, 2 weeks ago • solution
        selected={enteredYear}
        onChangeYear={yearChangeHandler}
      />
      {expensesContent}
    </Card>
  </div>
);
};
```

Conditional Return Statement

Creating ExpensesList component.

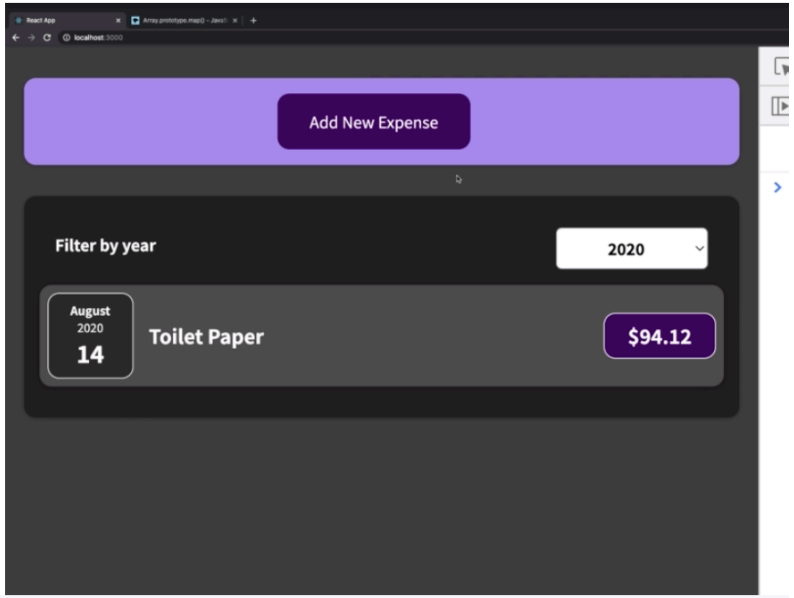
```
import "./ExpensesList.css";
import ExpenseItem from "./ExpenseItem";
const ExpensesList = (props) => {
  if (props.expenses.length === 0) {
    return <h2 className="expenses-list__fallback">Found no expenses</h2>;
  }

  return (
    <ul className="expenses-list">      You, seconds ago • added expenseslis
      {props.expenses.map((expense) => (
        <ExpenseItem
          key={expense.id}
          title={expense.title}
          amount={expense.amount}
          date={expense.date}
        />
      ))}
    </ul>
  );
};

export default ExpensesList;
```

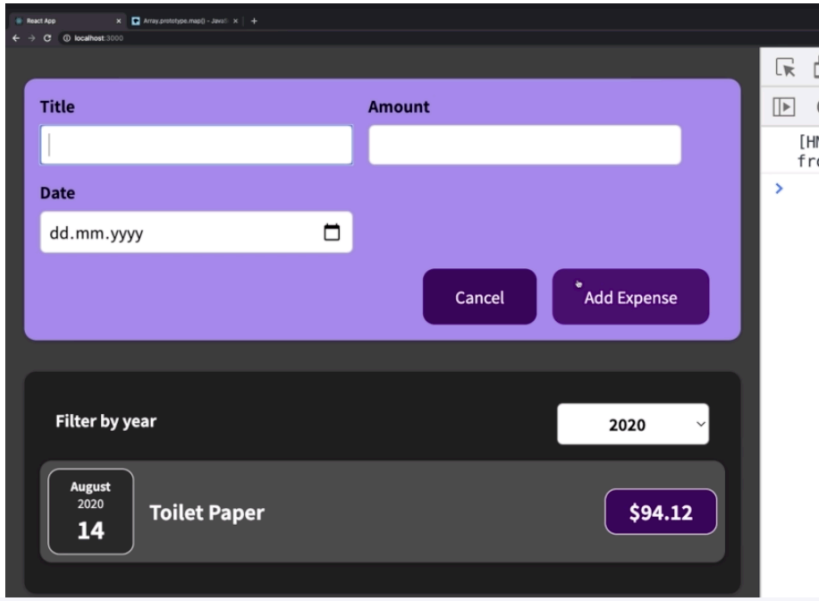
Conditional Return Statement

```jsx
import React, { useState } from "react";
import "./Expenses.css";
import Card from "../UI/Card";
import ExpensesFilter from "./ExpensesFilter";
import ExpensesList from "./ExpensesList";
const Expenses = (props) => {
  const [enteredYear, setEnteredYear] = useState("2021");
  const yearChangeHandler = (selectedYear) => {
    console.log("Expenses.js");
    console.log(selectedYear);
    setEnteredYear(selectedYear);
  };

  const filteredExpenses = props.expenses.filter((expense) => {
    return expense.date.getFullYear().toString() === enteredYear;
  });

  return (
    <div>
      <Card className="expenses">
        <ExpensesFilter
          selected={enteredYear}
          onChangeYear={yearChangeHandler}
        />
        <ExpensesList expenses={filteredExpenses} />
      </Card>
    </div>
  );
};

export default Expenses;
```

Assignment # 4

Goal: Form is always shown. Button for Add New Expense. Form shows when button is clicked. Add cancel button for the form as well. Created branch for assignment4
State to control whether button should be shown or form.





Assignment # 4

```jsx
import ExpenseForm from "./ExpenseForm";
import "./NewExpense.css";
import React, { useState } from "react";
const NewExpense = (props) => {
  const [showForm, setShowForm] = useState(false);

  const showFormHandler = () => {
    setShowForm(true);
  };

  const unshowFormHandler = () => {
    setShowForm(false);      You, 2 minutes ago • assignment 4
  };
  const saveExpenseDataHandler = (enteredExpenseData) => {
    const expenseData = {
      ...enteredExpenseData,
      id: Math.random().toString(),
    };
    props.onAddExpense(expenseData);
    setShowForm(false);
  };
  return (
    <div className="new-expense">
      {!showForm && <button onClick={showFormHandler}>Add New Expense</button>}
      {showForm && (
        <ExpenseForm
          onSaveExpenseData={saveExpenseDataHandler}
          onCancel={unshowFormHandler}
        />
      )}
    </div>
  );
};

export default NewExpense;
```

```jsx
<div className="new-expense__actions">
  <button type="button" onClick={props.onCancel}>
    Cancel
  </button>
  <button type="submit">Add expense</button>
</div>
```

Adding a Chart – new branch called chart

https://www.statisticshowto.com/probability-and-statistics/descriptive-statistics/bar-chart-bar-graph-examples/#WhatisBar

Create Chart and ChartBar

Chart will contain ChartBar

12 ChartBars for 12 months. Can make it more flexible with props.

Adding a Chart – new branch called chart

```
import ChartBar from "./ChartBar";
import "./Chart.css";

const Chart = (props) => {
  return (
    <div className="chart">
      {props.dataPoints.map((dataPoint) => (
        <ChartBar
          key={dataPoint.label}
          value={dataPoint.value}
          maxValue={null}
          label={dataPoint.label}
        />
      ))}
    </div>
  );
};

export default Chart;
```

Adding a Chart – new branch called chart

Style needs js object

```
import "./ChartBar.css";

const ChartBar = (props) => {
  let barFillHeight = "0%";

  if (props.max > 0) {
    barFillHeight = Math.round((props.value / props.maxValue) * 100) + "%";
  }
  return (
    <div className="chart-bar">
      <div className="chart-bar__inner">
        <div
          className="chart-bar__fill"
          style={{ height: barFillHeight }}
        ></div>
      </div>
      <div className="chart-bar__label">{props.label}</div>
    </div>
  );
};

export default ChartBar;
```

Adding a Chart – new branch called chart

Using Chart and ChartBar – new Component called ExpensesChart that takes the filteredExpenses (expenses). Loop through expenses and get month of expense and add it up for a month. Index of month is 0 based.

for in versus for of
https://bitsofco.de/for-in-vs-for-of/
https://medium.com/swlh/javascripts-for-vs-for-in-vs-for-of-in-depth-a589feb88bdd

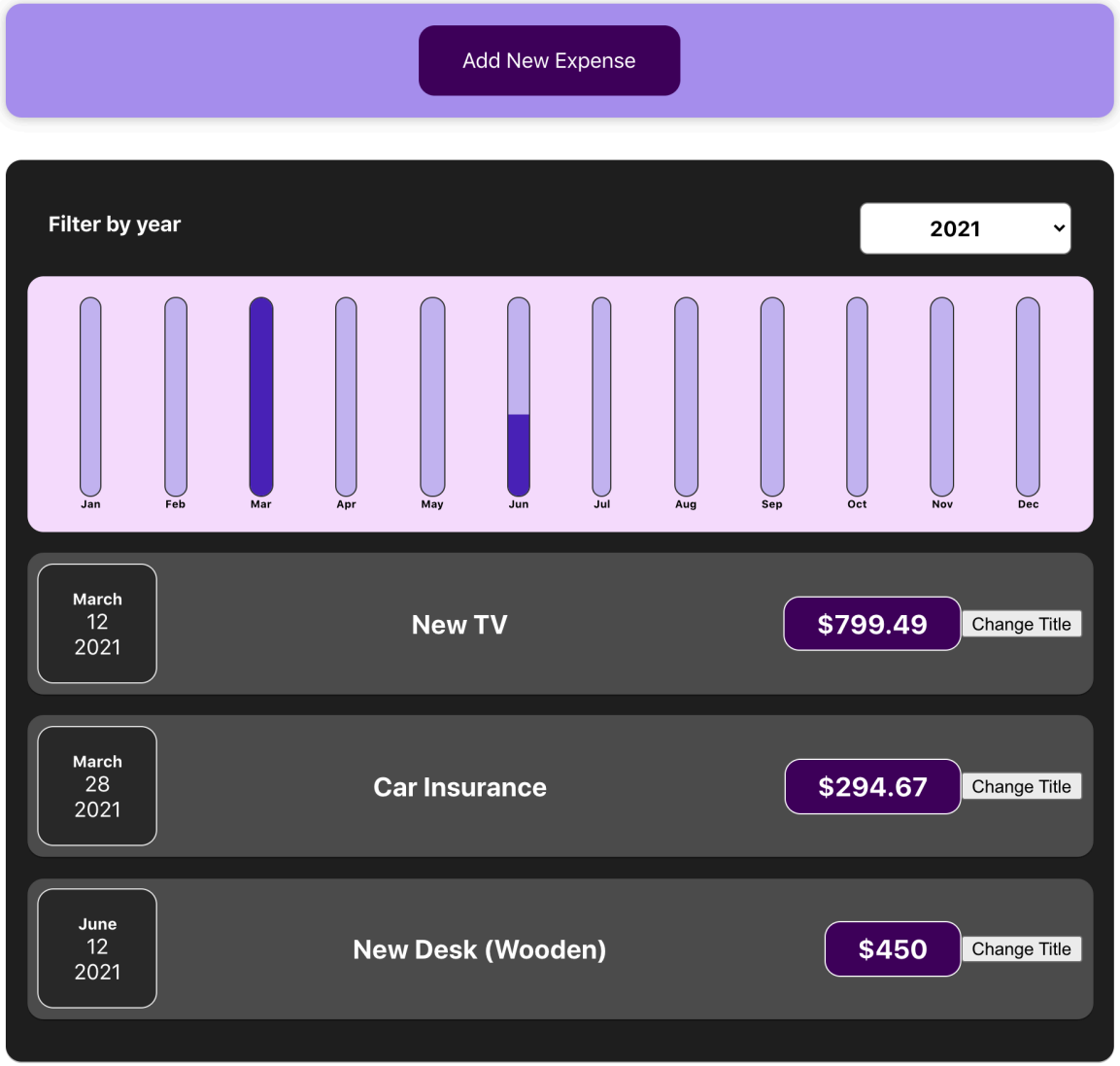
Adding a Chart – new branch called chart

```
import Chart from "../Chart/Chart";
const ExpensesChart = (props) => {
  const chartDataPoints = [
    { label: "Jan", value: 0 },
    { label: "Feb", value: 0 },
    { label: "Mar", value: 0 },
    { label: "Apr", value: 0 },
    { label: "May", value: 0 },
    { label: "Jun", value: 0 },
    { label: "Jul", value: 0 },
    { label: "Aug", value: 0 },
    { label: "Sep", value: 0 },
    { label: "Oct", value: 0 },
    { label: "Nov", value: 0 },
    { label: "Dec", value: 0 },
  ];
  for (const expense of props.expenses) {
    const expenseMonth = expense.date.getMonth();
    chartDataPoints[expenseMonth].value += expense.amount;
  }

  return <Chart dataPoints={chartDataPoints} />;
};

export default ExpensesChart;
```

Adding a Chart – new branch called chart

```jsx
import ChartBar from "./ChartBar";
import "./Chart.css";

const Chart = (props) => {
  const dataPointValues = props.dataPoints.map((dataPoint) => dataPoint.value);
  const totalMaximum = Math.max(...dataPointValues);
  return (
    <div className="chart">
      {props.dataPoints.map((dataPoint) => (
        <ChartBar
          key={dataPoint.label}
          value={dataPoint.value}
          maxValue={totalMaximum}
          label={dataPoint.label}
        />
      ))}
    </div>
  );
};

export default Chart;
```

Adding a Chart – new branch called chart



Quiz

Good job!

That's the correct choice!

Question 1:

What does this code snippet do?

```
1 | someArray.map((element) => <p>{element}</p>)
```

- ○ It's a special React syntax that tells React to loop through a list of elements and output some JSX code.
- ◉ It transforms an array (someArray) into a new array full of JSX elements (which can be output by React)
- ○ It tells React to output this paragraph conditionally.

Quiz

Question 2:

Why should you add the special "key" prop to list JSX elements?

- ○ It's required for React to be able to loop through the elements
- ○ It's required for the map() method to work correctly.
- ◉ It's required for React to correctly identify and update (if needed) the list elements.

Quiz

Question 3:

What's true about outputting conditional content in React components?

- ○ You need a special conditional content syntax in your JSX code.
- ○ You need to create multiple, alternative components from which React then chooses.
- ◉ You can work with regular ternary expressions of if checks to output or return different results in / from your component.