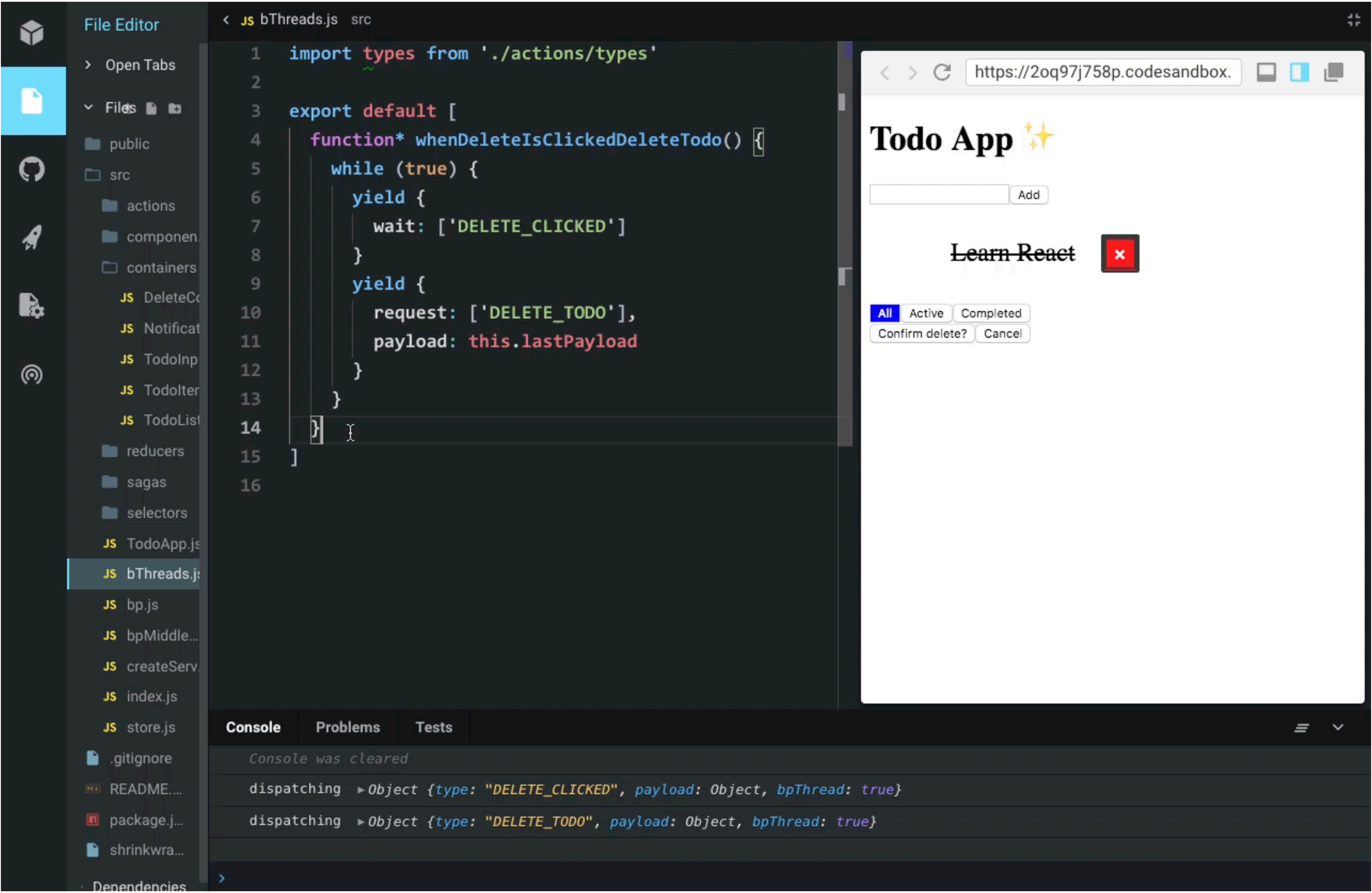
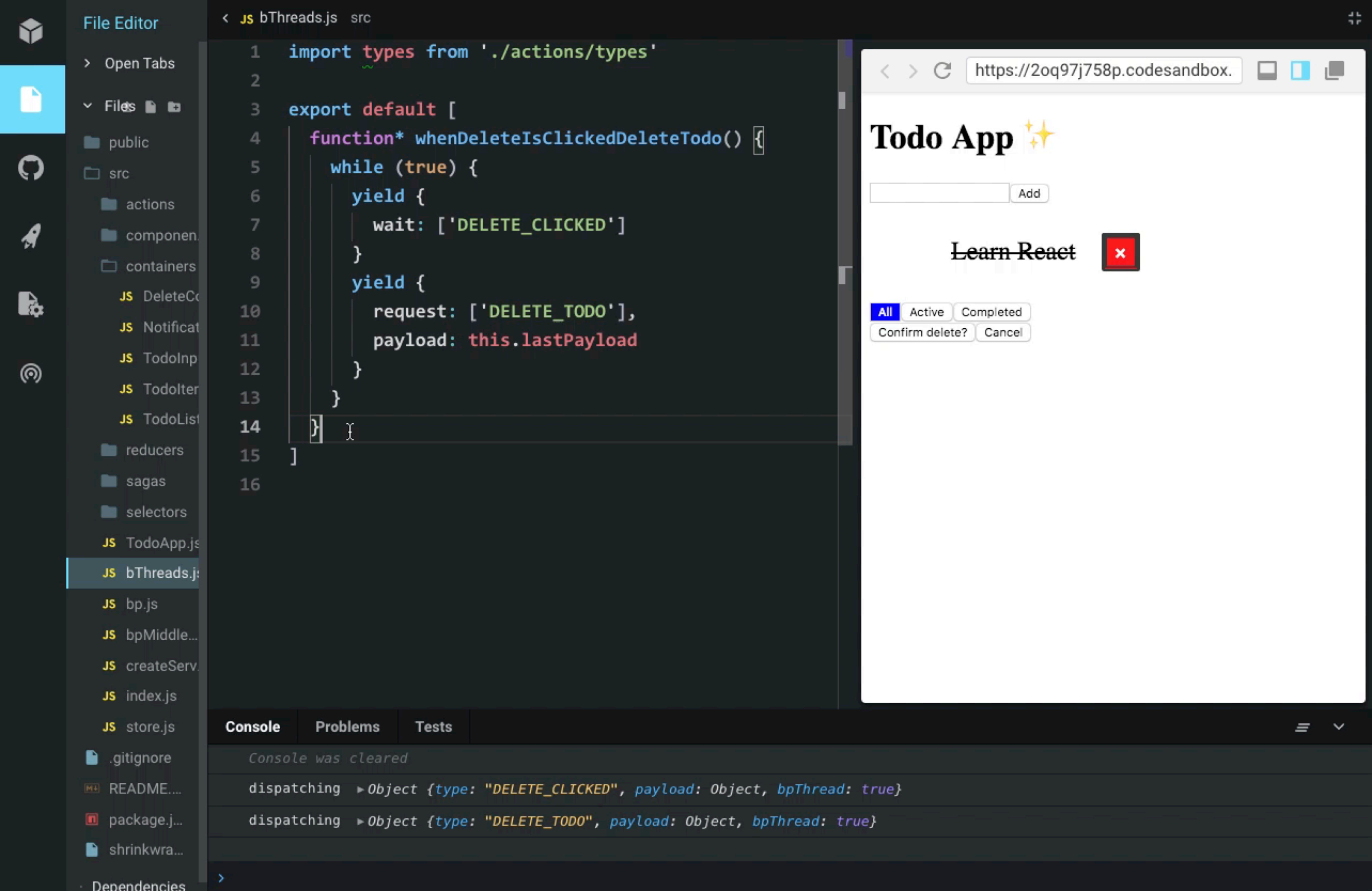
When delete is clicked, block delete until confirmation



```javascript
import types from './actions/types'

export default [
  function* whenDeleteIsClickedDeleteTodo() {
    while (true) {
      yield {
        wait: ['DELETE_CLICKED']
      }
      yield {
        request: ['DELETE_TODO'],
        payload: this.lastPayload
      }
    }
  }
]
```

```
Console was cleared
dispatching ▸Object {type: "DELETE_CLICKED", payload: Object, bpThread: true}
dispatching ▸Object {type: "DELETE_TODO", payload: Object, bpThread: true}
```



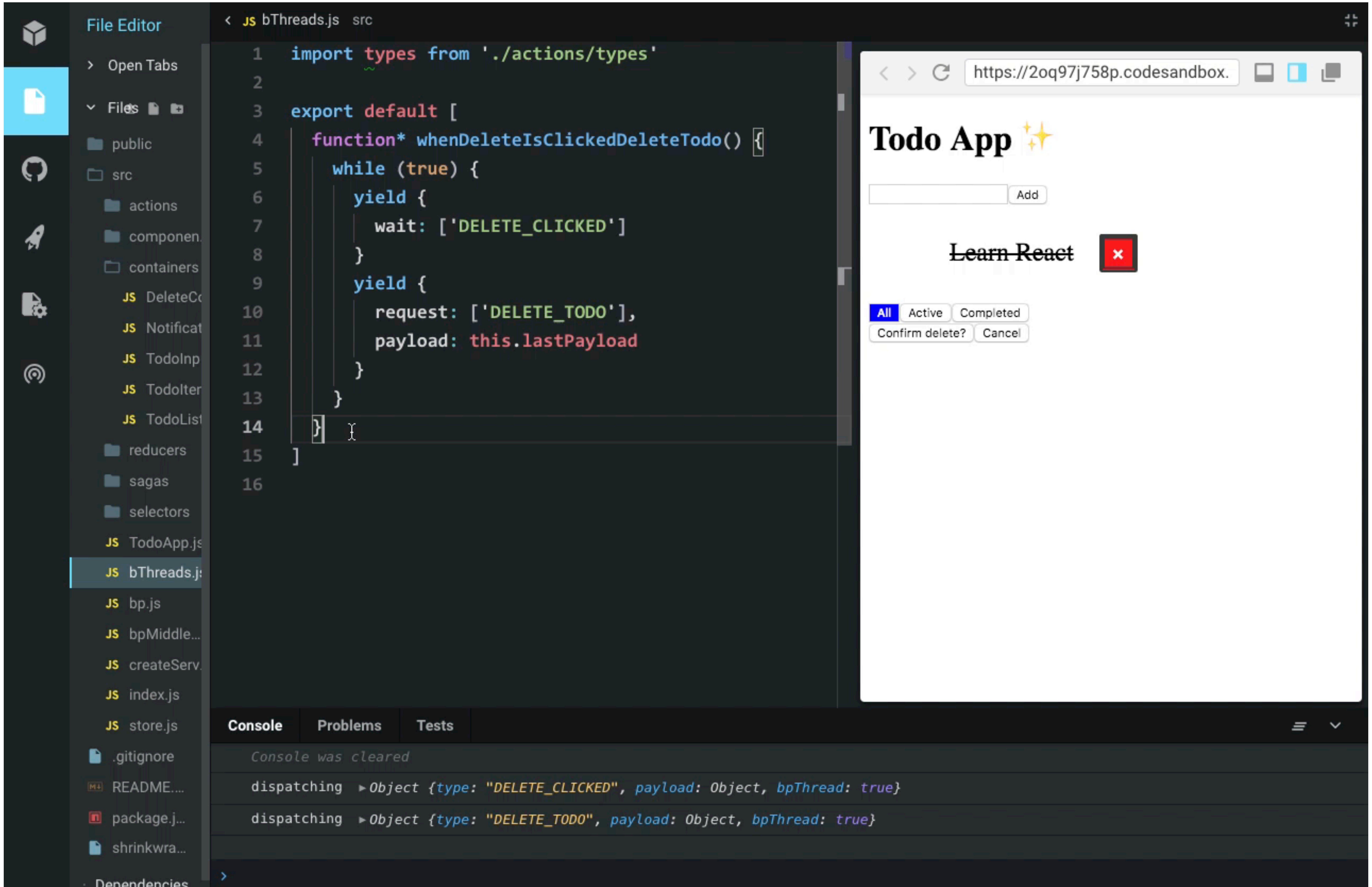
```
import types from './actions/types'

export default [
  function* whenDeleteIsClickedDeleteTodo() {
    while (true) {
      yield {
        wait: ['DELETE_CLICKED']
      }
      yield {
        request: ['DELETE_TODO'],
        payload: this.lastPayload
      }
    }
  }
]
```

Todo App ✨

~~Learn React~~ ×

```
Console was cleared
dispatching ▸Object {type: "DELETE_CLICKED", payload: Object, bpThread: true}
dispatching ▸Object {type: "DELETE_TODO", payload: Object, bpThread: true}
```

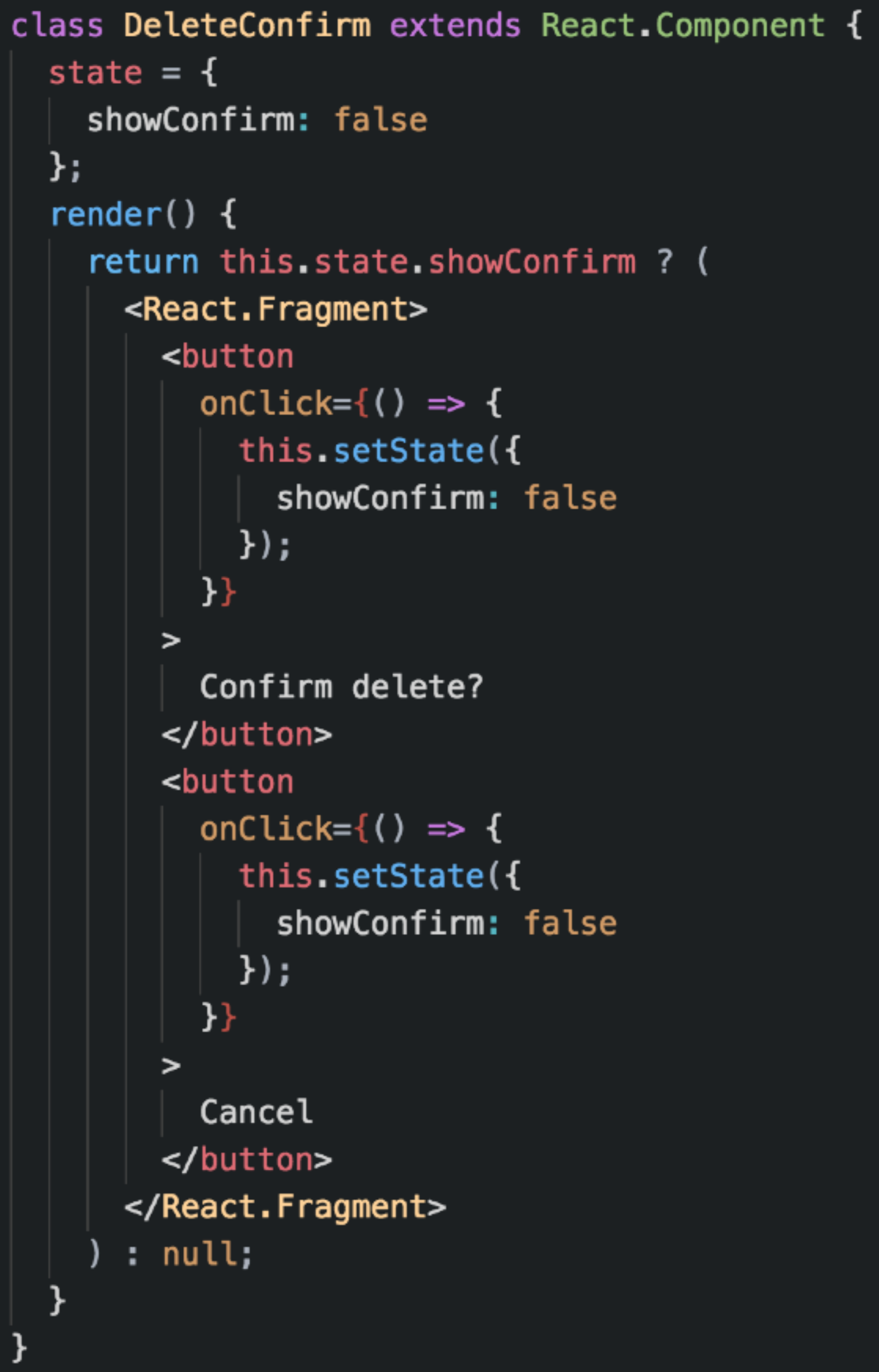
When delete is clicked, block delete until confirmation



```
import types from './actions/types'

export default [
  function* whenDeleteIsClickedDeleteTodo() {
    while (true) {
      yield {
        wait: ['DELETE_CLICKED']
      }
      yield {
        request: ['DELETE_TODO'],
        payload: this.lastPayload
      }
    }
  }
]
```

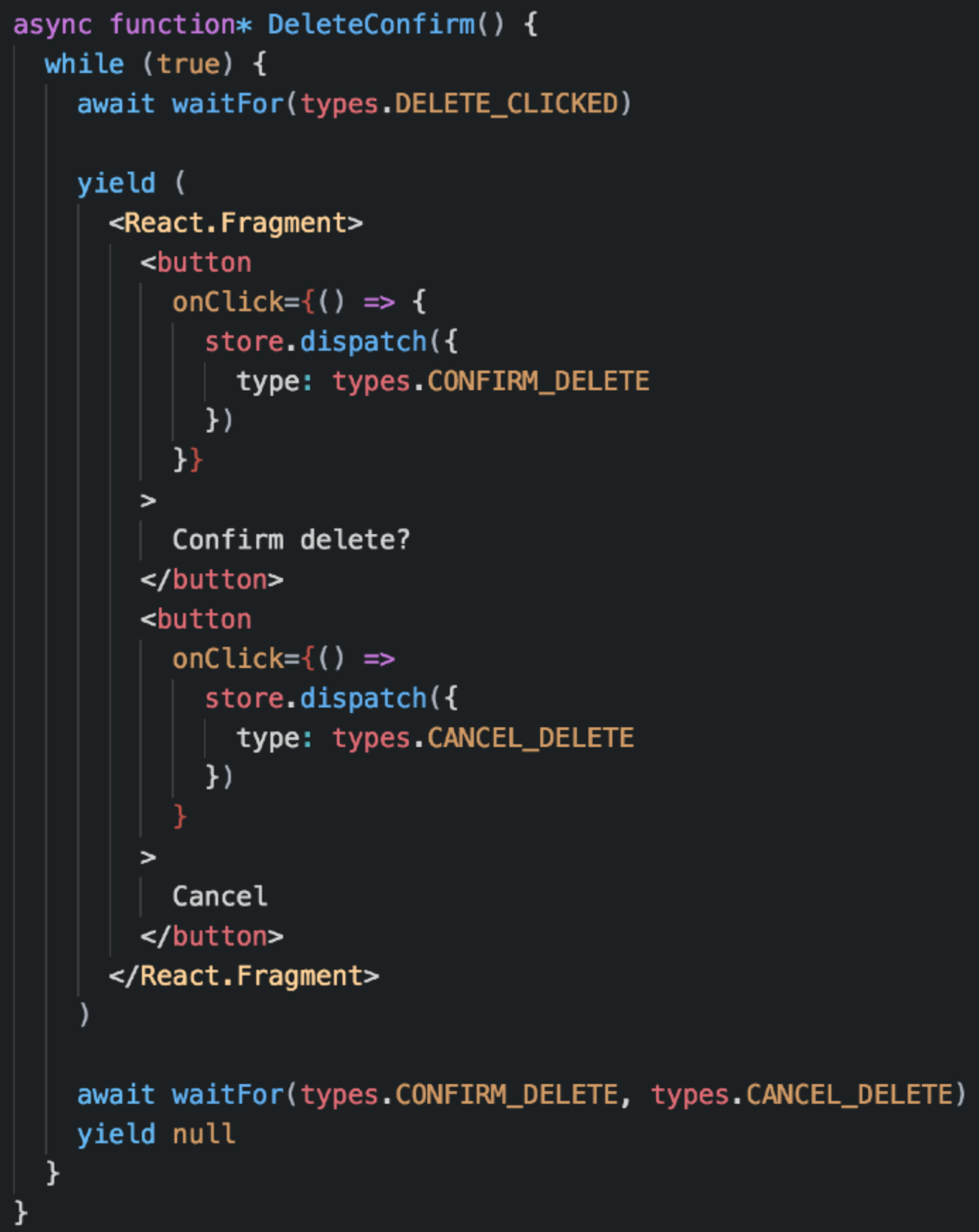
```
Console was cleared
dispatching ▸Object {type: "DELETE_CLICKED", payload: Object, bpThread: true}
dispatching ▸Object {type: "DELETE_TODO", payload: Object, bpThread: true}
```

“Classic” approach State-based



```
class DeleteConfirm extends React.Component {
  state = {
    showConfirm: false
  };
  render() {
    return this.state.showConfirm ? (
      <React.Fragment>
        <button
          onClick={() => {
            this.setState({
              showConfirm: false
            });
          }}
        >
          Confirm delete?
        </button>
        <button
          onClick={() => {
            this.setState({
              showConfirm: false
            });
          }}
        >
          Cancel
        </button>
      </React.Fragment>
    ) : null;
  }
}
```

Scenario-based



```
async function* DeleteConfirm() {
  while (true) {
    await waitFor(types.DELETE_CLICKED)

    yield (
      <React.Fragment>
        <button
          onClick={() => {
            store.dispatch({
              type: types.CONFIRM_DELETE
            })
          }}
        >
          Confirm delete?
        </button>
        <button
          onClick={() =>
            store.dispatch({
              type: types.CANCEL_DELETE
            })
          }
        >
          Cancel
        </button>
      </React.Fragment>
    )

    await waitFor(types.CONFIRM_DELETE, types.CANCEL_DELETE)
    yield null
  }
}
```