

When delete is clicked, block delete until
confirmation

File Editor

> Open Tabs

Files

public

src

actions

componen.

containers

JS DeleteCo

JS Notificat

JS TodoInp

JS TodoIter

JS TodoList

reducers

sagas

selectors

JS TodoApp.js

JS bThreads.js

JS bp.js

JS bpMiddle...

JS createServ.

JS index.js

JS store.js

.gitignore

README....

package.j...

shrinkwra...

Dependencies

< JS bThreads.js src

1 import types from './actions/types'

2

3 export default [

4 function* whenDeleteIsClickedDeleteTodo() {

5 while (true) {

6 yield {

7 wait: ['DELETE_CLICKED']

8 }

9 yield {

10 request: ['DELETE_TODO'],

11 payload: this.lastPayload

12 }

13 }

14 }

15]

16

Console Problems Tests

Console was cleared

dispatching ▶Object {type: "DELETE_CLICKED", payload: Object, bpThread: true}

dispatching ▶Object {type: "DELETE_TODO", payload: Object, bpThread: true}

>

https://2oq97j758p.codesandbox.

Todo App ✨

Add

~~Learn React~~ ✖

All Active Completed

Confirm delete? Cancel



File Editor

> Open Tabs

Files

public

src

actions

components

containers

JS DeleteCo

JS Notificat

JS TodoInp

JS TodoIter

JS TodoList

reducers

sagas

selectors

JS TodoApp.js

JS bThreads.js

JS bp.js

JS bpMiddle...

JS createServ

JS index.js

JS store.js

.gitignore

README....

package.j...

shrinkwra...

Dependencies

< JS bThreads.js src

```
1 import types from './actions/types'
2
3 export default [
4   function* whenDeleteIsClickedDeleteTodo() {
5     while (true) {
6       yield {
7         wait: ['DELETE_CLICKED']
8       }
9       yield {
10        request: ['DELETE_TODO'],
11        payload: this.lastPayload
12      }
13    }
14  }
15 ]
16
```

Console

Problems

Tests

Console was cleared

dispatching ▶ Object {type: "DELETE_CLICKED", payload: Object, bpThread: true}

dispatching ▶ Object {type: "DELETE_TODO", payload: Object, bpThread: true}



https://2oq97j758p.codesandbox.



Todo App ✨

 Add

Learn React



All Active Completed

Confirm delete? Cancel

When delete is clicked, block delete until confirmation

The image shows a development environment with a code editor on the left and a web browser on the right. The code editor displays the `bThreads.js` file in the `src` directory. The code defines a `whenDeleteIsClickedDeleteTodo` function that uses a `while` loop to block execution until a confirmation is received.

```
1 import types from './actions/types'
2
3 export default [
4   function* whenDeleteIsClickedDeleteTodo() {
5     while (true) {
6       yield {
7         wait: ['DELETE_CLICKED']
8       }
9       yield {
10        request: ['DELETE_TODO'],
11        payload: this.lastPayload
12      }
13    }
14  }
15 ]
16
```

The web browser shows a `Todo App` interface. It includes an input field with an `Add` button, a `Learn React` button with a red close icon, and a filter section with `All`, `Active`, and `Completed` tabs. Below the filters, there are `Confirm delete?` and `Cancel` buttons.

The browser's address bar shows the URL `https://2oq97j758p.codesandbox.`

The bottom of the image shows the `Console` tab, which displays the following log messages:

```
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
  }
}
```