sourcemind

REDUX - TOOLKIT

Redux Toolkit

sourcemind

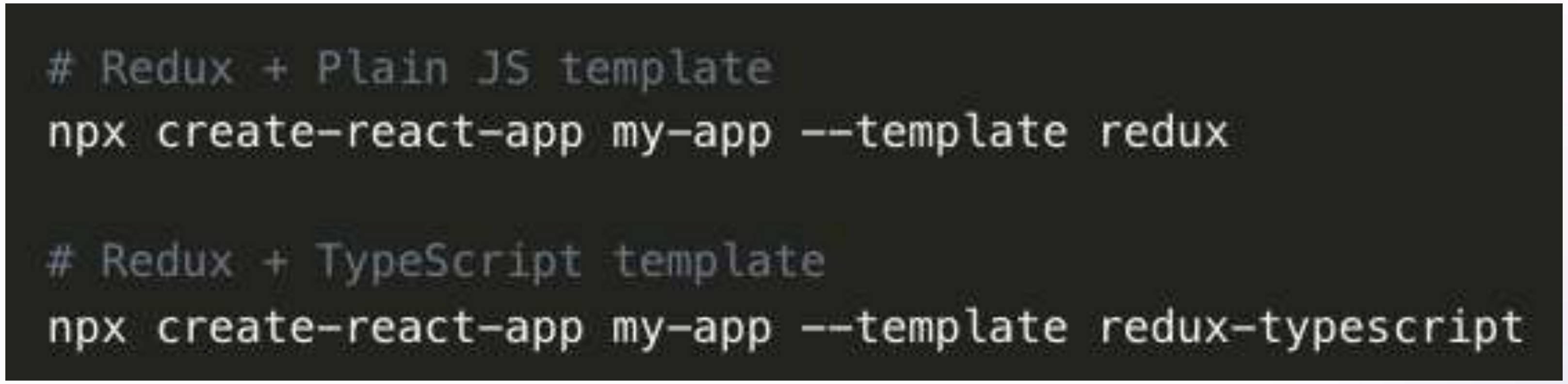
Redux Toolkit

The official, opinionated, batteries-included toolset for efficient Redux development

sourcemind

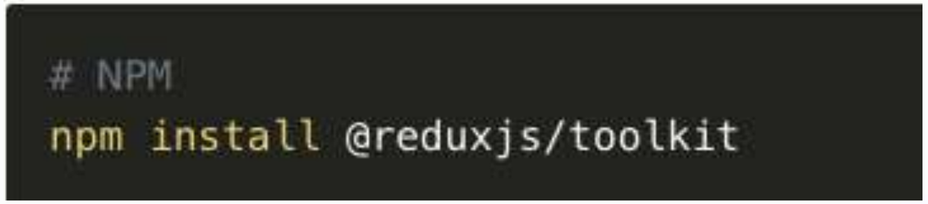
Installation

Using Create React App



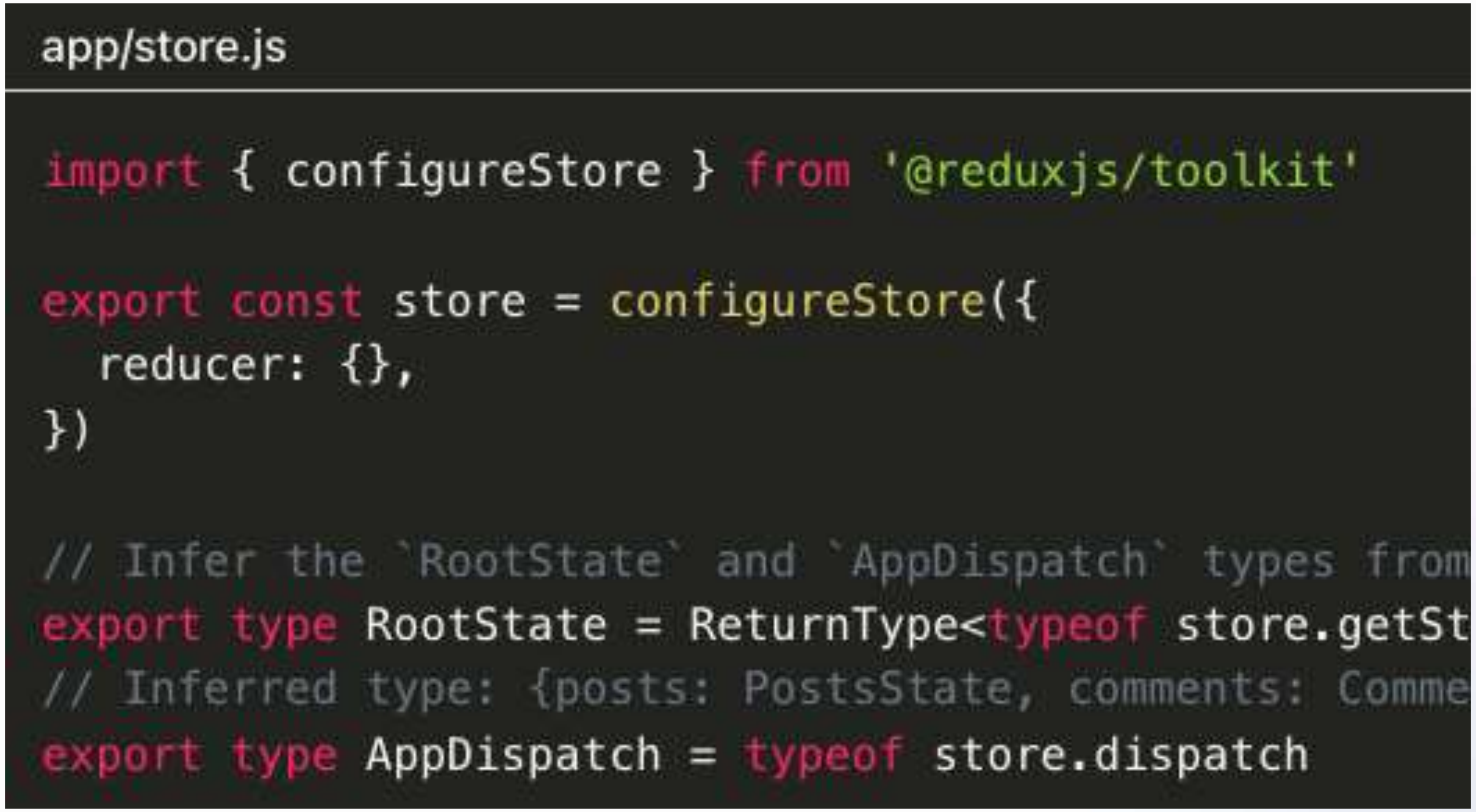
```
# Redux + Plain JS template
npx create-react-app my-app --template redux

# Redux + TypeScript template
npx create-react-app my-app --template redux-typescript
```

An existing App



```
# NPM
npm install @reduxjs/toolkit
```

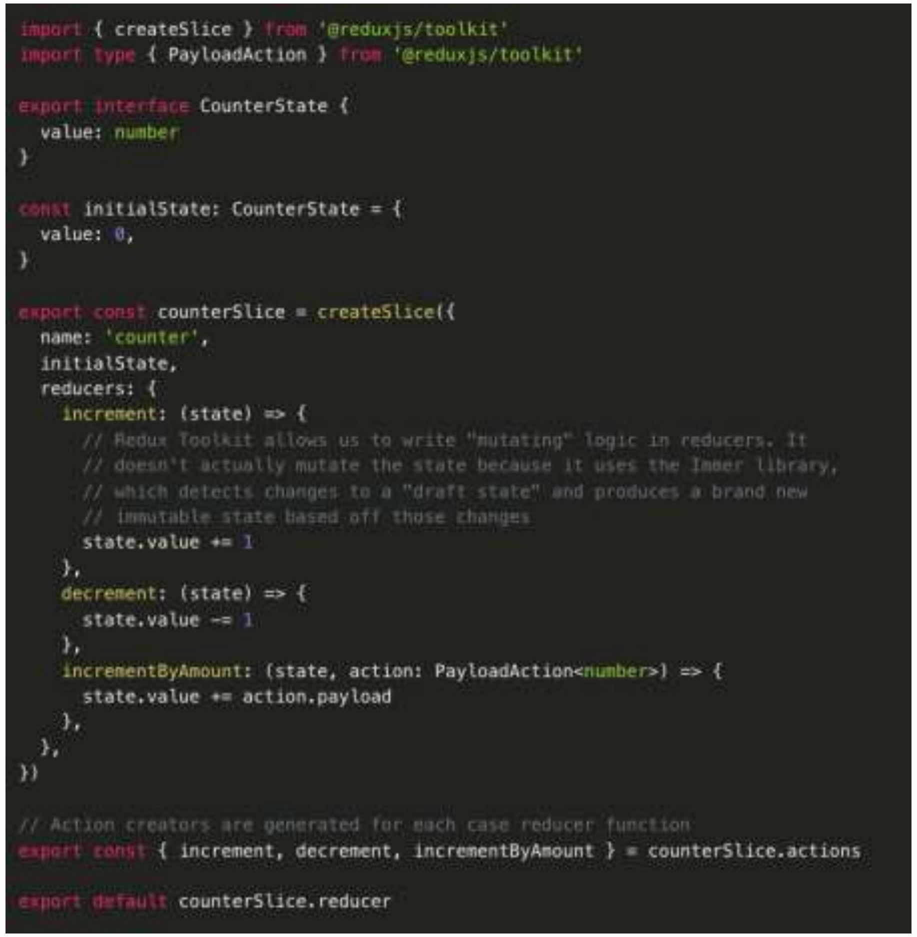
Create a Redux Store



app/store.js

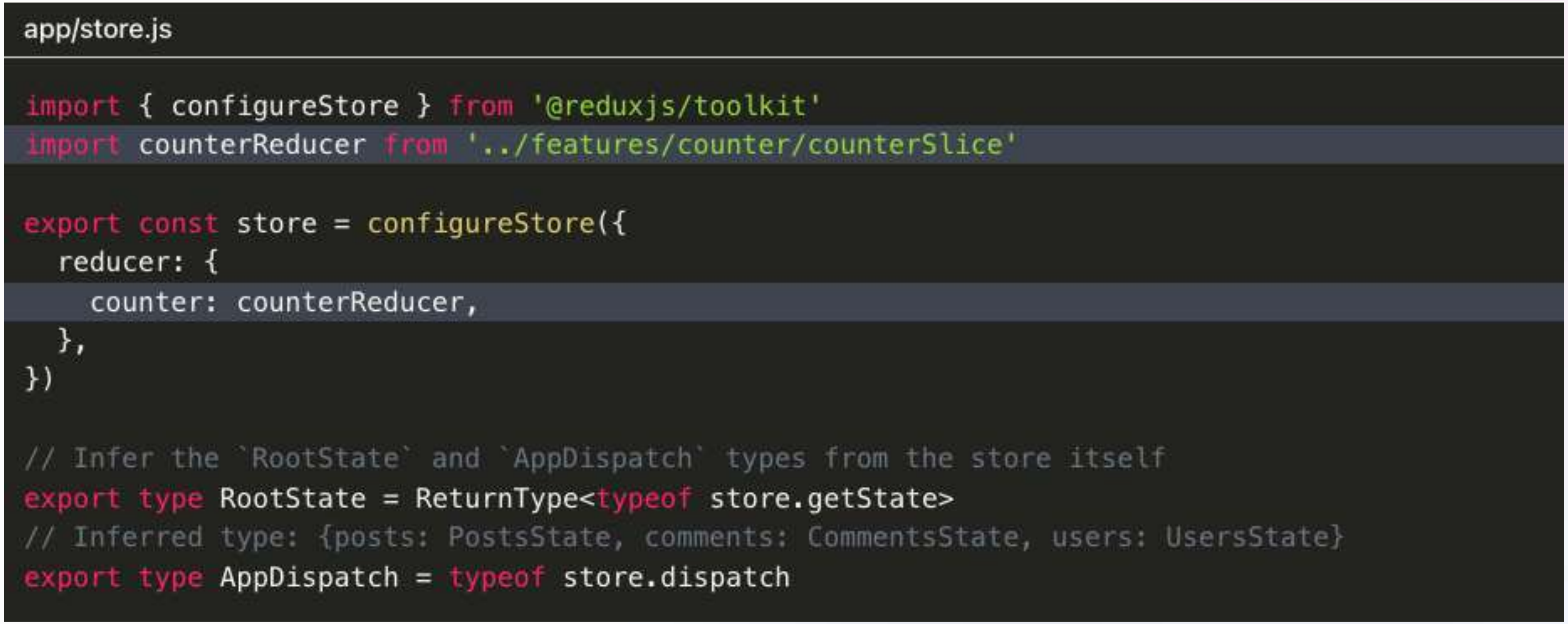
```
import { configureStore } from '@reduxjs/toolkit'

export const store = configureStore({
  reducer: {},
})

// Infer the `RootState` and `AppDispatch` types from
export type RootState = ReturnType<typeof store.getSt
// Inferred type: {posts: PostsState, comments: Comme
export type AppDispatch = typeof store.dispatch
```

Slice



```typescript
import { createSlice } from '@reduxjs/toolkit'
import type { PayloadAction } from '@reduxjs/toolkit'

export interface CounterState {
  value: number
}

const initialState: CounterState = {
  value: 0,
}

export const counterSlice = createSlice({
  name: 'counter',
  initialState,
  reducers: {
    increment: (state) => {
      // Redux Toolkit allows us to write "mutating" logic in reducers. It
      // doesn't actually mutate the state because it uses the Immer library,
      // which detects changes to a "draft state" and produces a brand new
      // immutable state based off those changes
      state.value += 1
    },
    decrement: (state) => {
      state.value -= 1
    },
    incrementByAmount: (state, action: PayloadAction<number>) => {
      state.value += action.payload
    },
  },
})

// Action creators are generated for each case reducer function
export const { increment, decrement, incrementByAmount } = counterSlice.actions

export default counterSlice.reducer
```

Add Slice Reducers to the Store

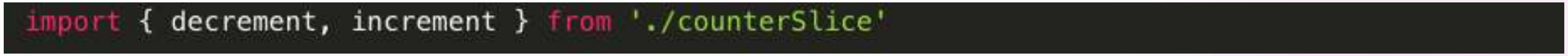


app/store.js

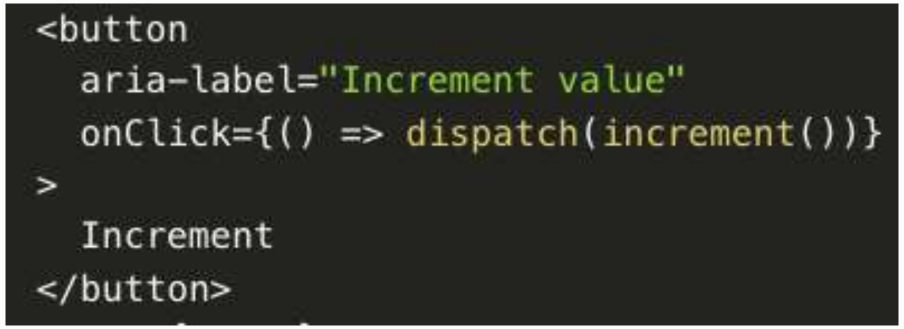
```
import { configureStore } from '@reduxjs/toolkit'
import counterReducer from '../features/counter/counterSlice'

export const store = configureStore({
  reducer: {
    counter: counterReducer,
  },
})

// Infer the `RootState` and `AppDispatch` types from the store itself
export type RootState = ReturnType<typeof store.getState>
// Inferred type: {posts: PostsState, comments: CommentsState, users: UsersState}
export type AppDispatch = typeof store.dispatch
```

sourcemind

Using In React Component



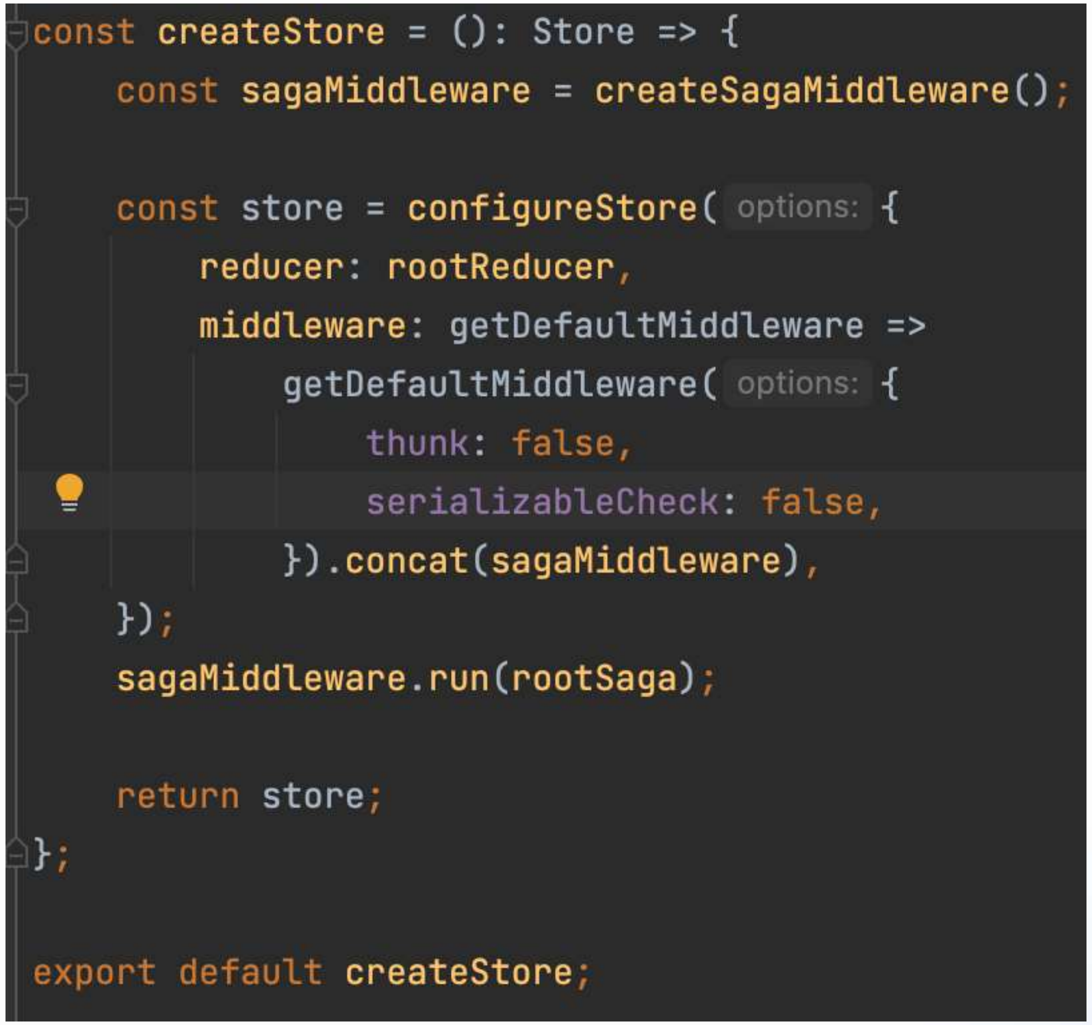
```
import { decrement, increment } from './counterSlice'
```



```
<button
  aria-label="Increment value"
  onClick={() => dispatch(increment())}
>
  Increment
</button>
```

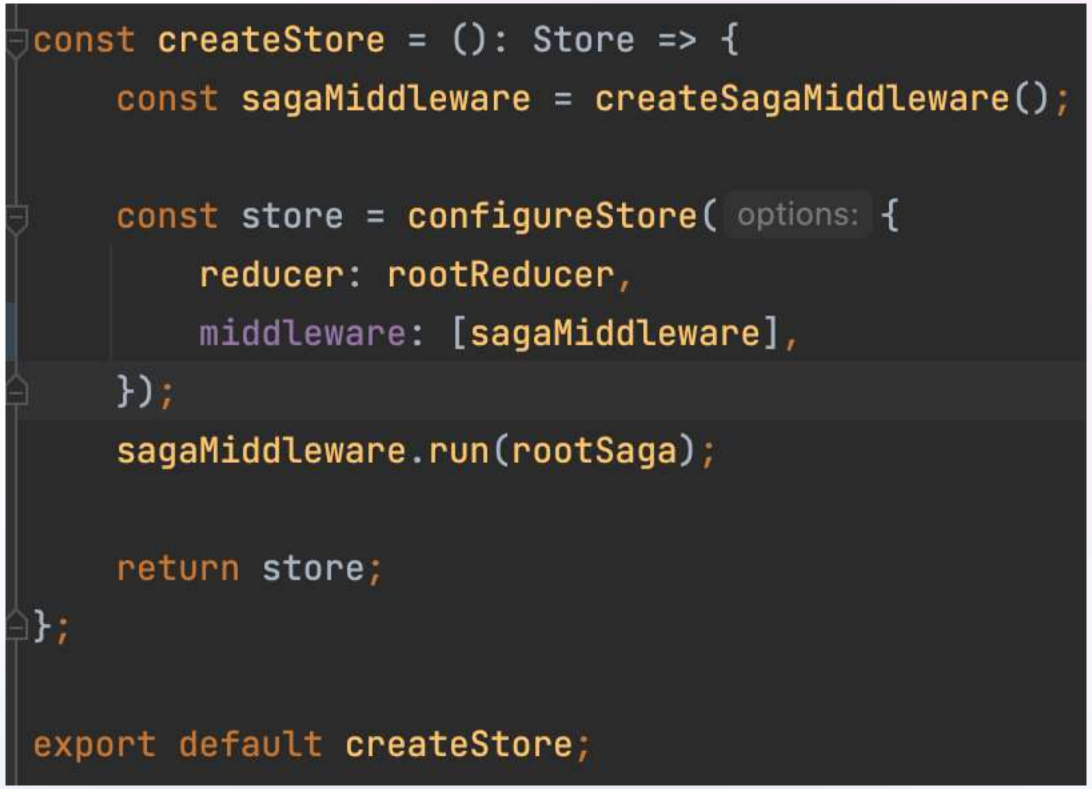
sourcemind

Usage with Redux-Saga



```
const createStore = (): Store => {
    const sagaMiddleware = createSagaMiddleware();

    const store = configureStore( options: {
        reducer: rootReducer,
        middleware: getDefaultMiddleware =>
            getDefaultMiddleware( options: {
                thunk: false,
                serializableCheck: false,
            }).concat(sagaMiddleware),
    });
    sagaMiddleware.run(rootSaga);

    return store;
};

export default createStore;
```

or

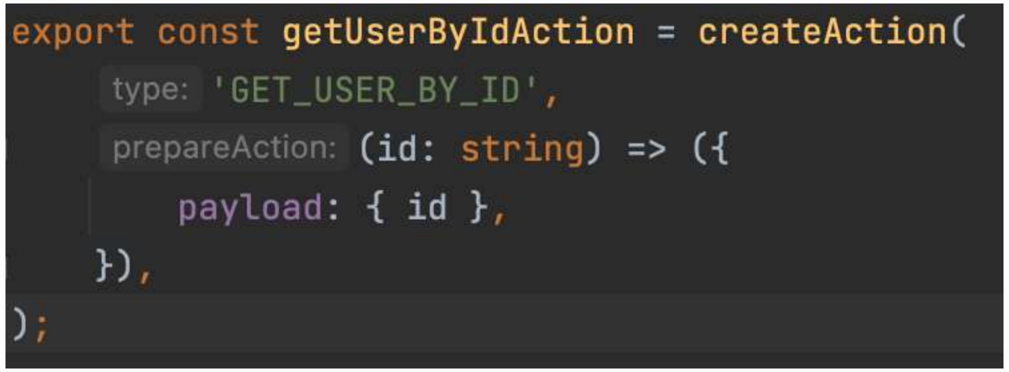


```
const createStore = (): Store => {
    const sagaMiddleware = createSagaMiddleware();

    const store = configureStore( options: {
        reducer: rootReducer,
        middleware: [sagaMiddleware],
    });
    sagaMiddleware.run(rootSaga);

    return store;
};

export default createStore;
```

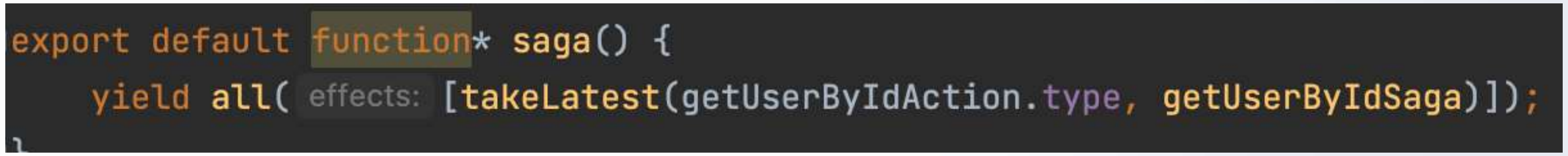
sourcemind

Usage with Redux-Saga

Actions



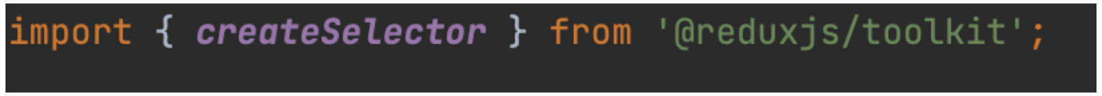
```
export const getUserByIdAction = createAction(
    type: 'GET_USER_BY_ID',
    prepareAction: (id: string) => ({
        payload: { id },
    }),
);
```

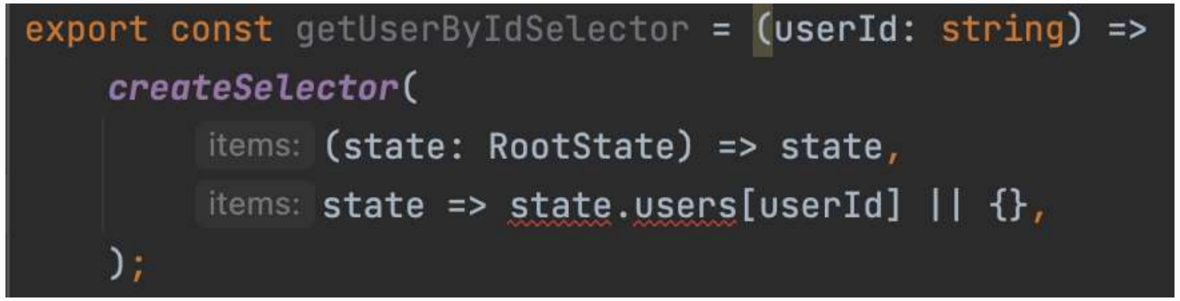


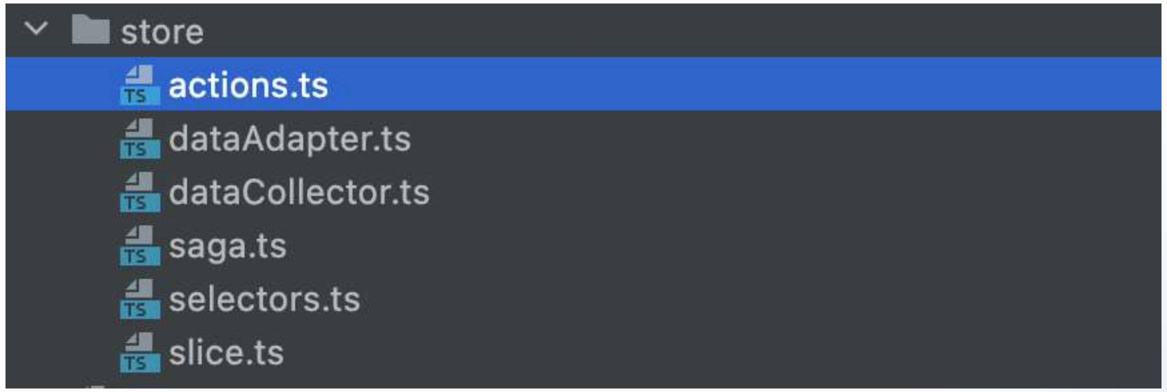
```
export default function* saga() {
    yield all( effects: [takeLatest(getUserByIdAction.type, getUserByIdSaga)]);
```

sourcemind

Create Selector


```
import { createSelector } from '@reduxjs/toolkit';
```


```
export const getUserByIdSelector = (userId: string) =>
    createSelector(
        items: (state: RootState) => state,
        items: state => state.users[userId] || {},
    );
```


```
store
    actions.ts
    dataAdapter.ts
    dataCollector.ts
    saga.ts
    selectors.ts
    slice.ts
```

sourcemind