Redux Toolkit

INTERVIEW QUESTIONS-84

Follow on in

@DUVVURU KISHORE

The Redux Toolkit package is intended to be the standard way to write Redux logic.

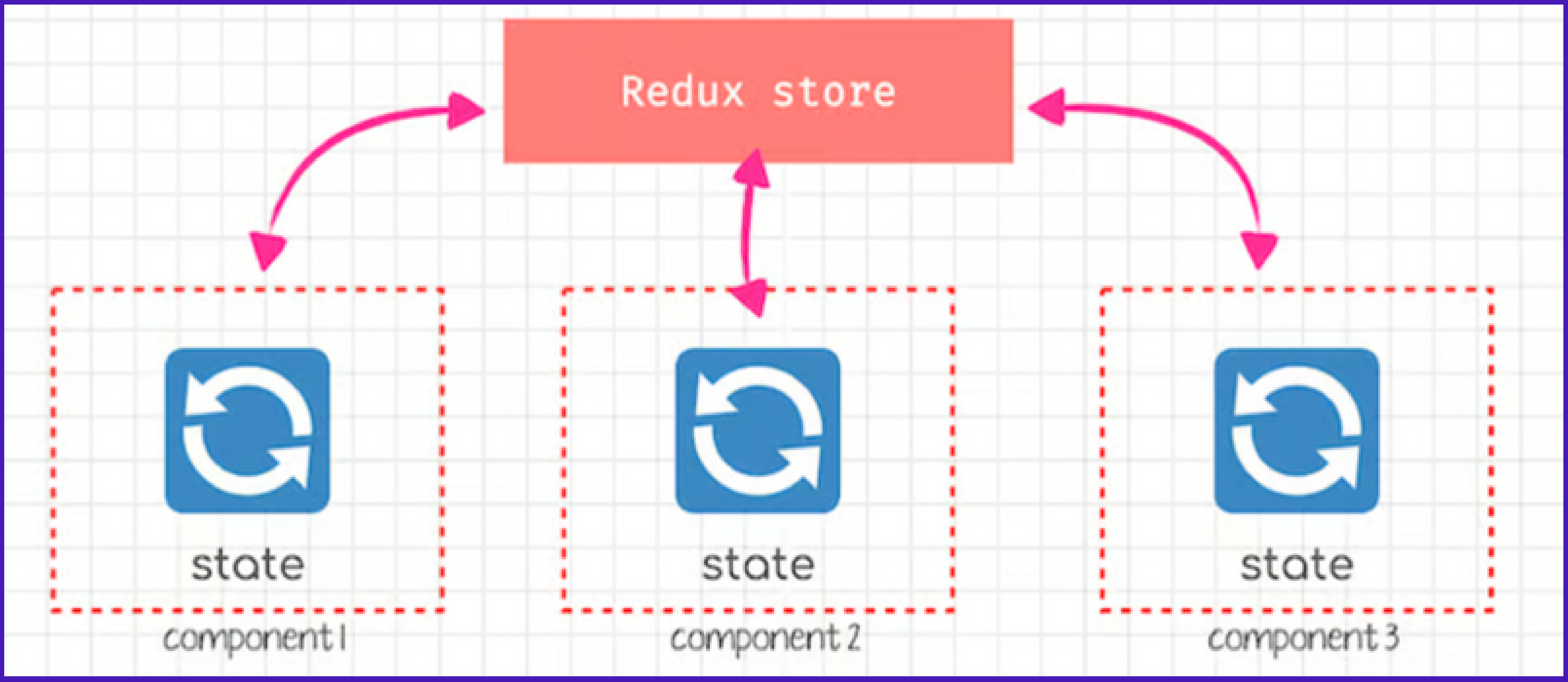
Changes from previous way of writing redux will be known just by scrolling through further slides

before going into the topic we will know why is Redux used for?

- Redux provides a centralized state management solution where components can access state from any level of the component tree without passing props explicitly.

- With Redux, components subscribe to specific parts of the state and are notified whenever that state changes, allowing for more flexible and decoupled component relationships.

- **Redux helps avoid prop drilling by maintaining global state in a store that any component can access, reducing the need to pass props through intermediate components.**



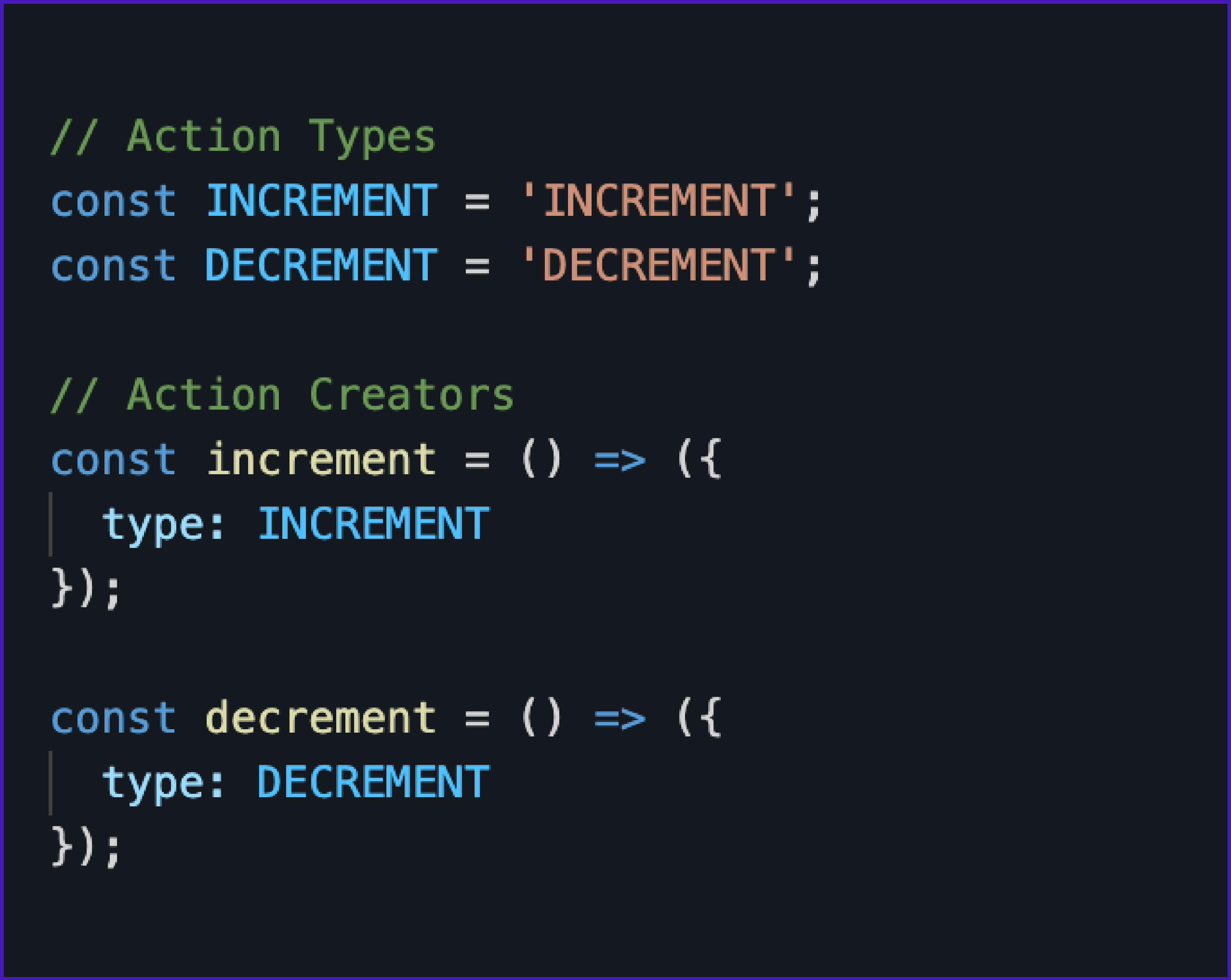
coming back to redux toolkit...In the previous approach of writing Redux, the process typically involved three stages:

- actions,
- creating reducers,
- and configuring the store.

When dealing with multiple reducers, developers utilized combineReducers to consolidate them into a single reducer, which was then integrated into the store setup.

Old way

Actions

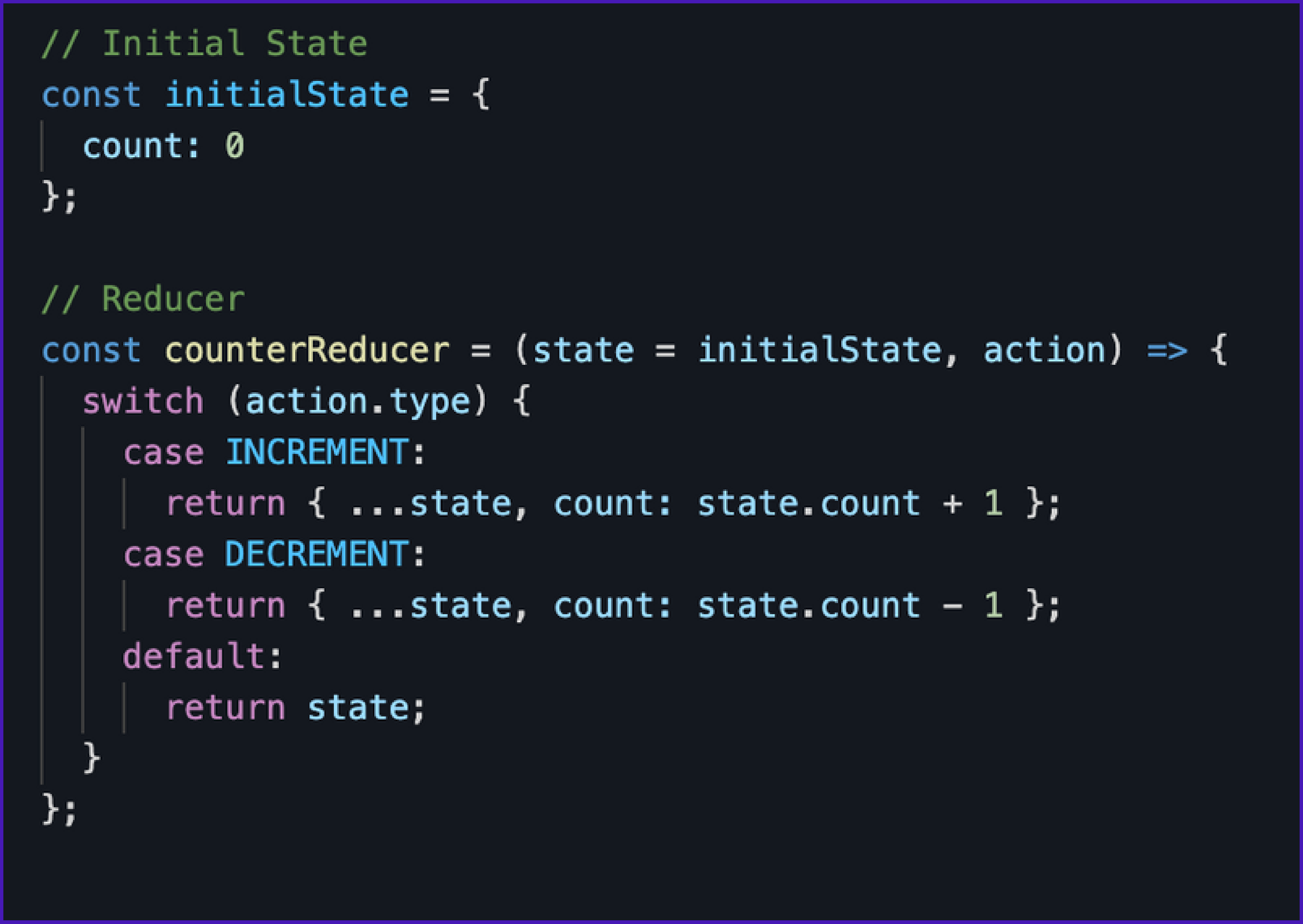


```
// Action Types
const INCREMENT = 'INCREMENT';
const DECREMENT = 'DECREMENT';

// Action Creators
const increment = () => ({
  type: INCREMENT
});

const decrement = () => ({
  type: DECREMENT
});
```

if your actions require additional data to be passed along (such as the amount to increment/decrement), you can include a payload property in your action creators to hold this data:

Reducers

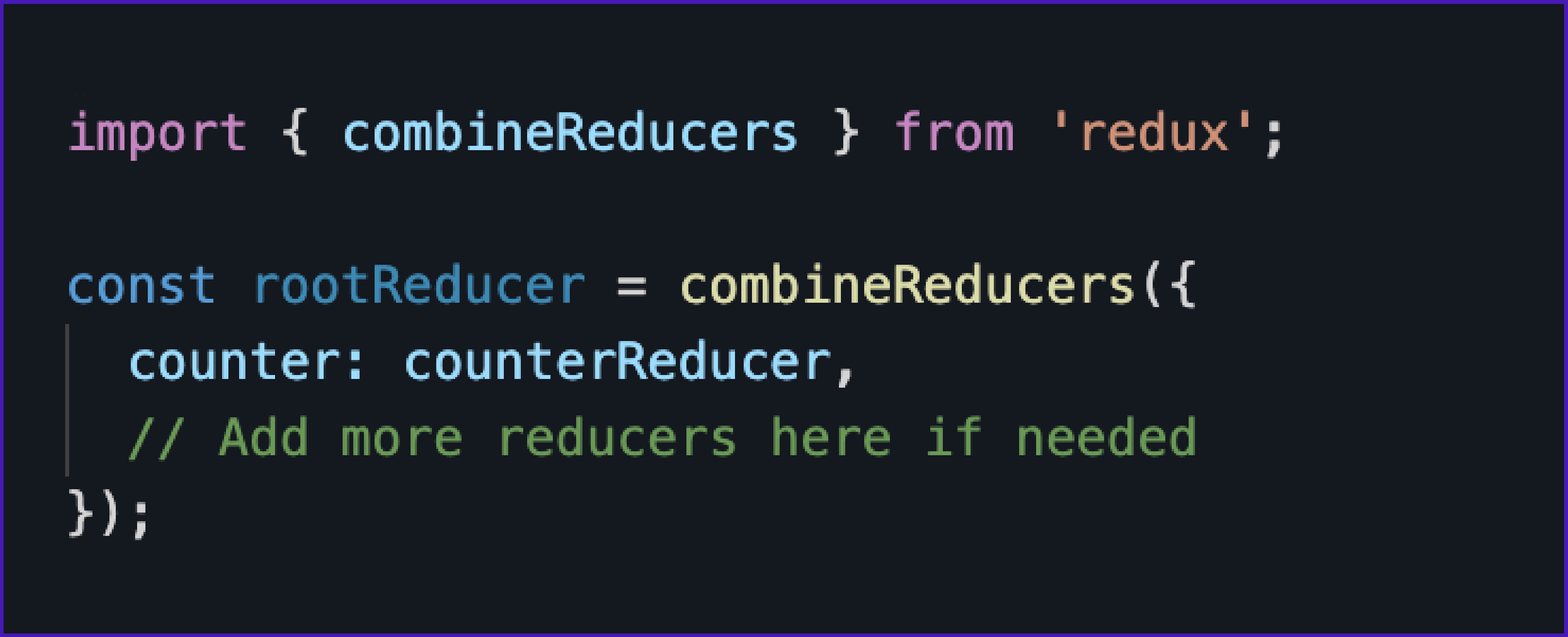
- **Reducers are pure functions responsible for specifying how the application's state should change in response to dispatched actions.**
- **They take the current state and an action as arguments and return the next state.**



```javascript
// Initial State
const initialState = {
  count: 0
};

// Reducer
const counterReducer = (state = initialState, action) => {
  switch (action.type) {
    case INCREMENT:
      return { ...state, count: state.count + 1 };
    case DECREMENT:
      return { ...state, count: state.count - 1 };
    default:
      return state;
  }
};
```

combine reducers

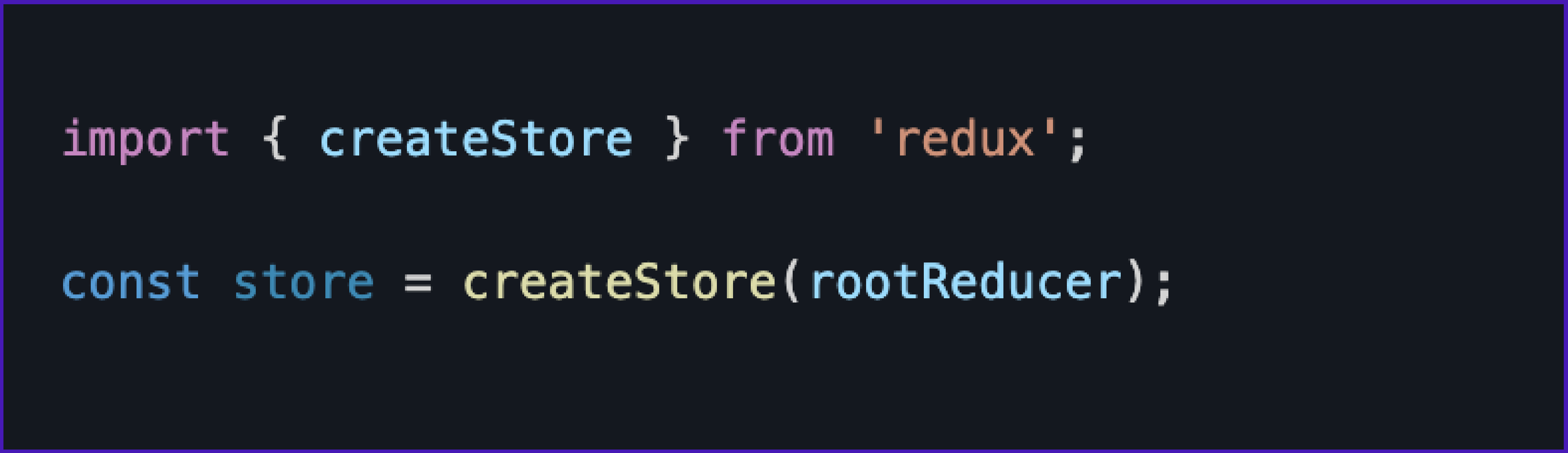
combineReducers is a utility function provided by Redux to combine multiple reducers into a single reducer function.



```javascript
import { combineReducers } from 'redux';

const rootReducer = combineReducers({
  counter: counterReducer,
  // Add more reducers here if needed
});
```

Store

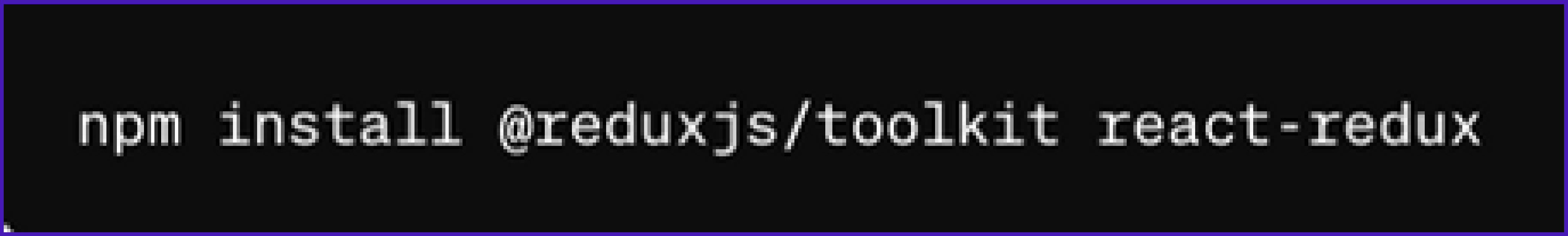
- The store is the central piece of the Redux architecture that holds the application state.
- It is created by passing the root reducer (or combined reducer) to the createStore function provided by Redux.



```
import { createStore } from 'redux';

const store = createStore(rootReducer);
```

Redux toolkit

installation:



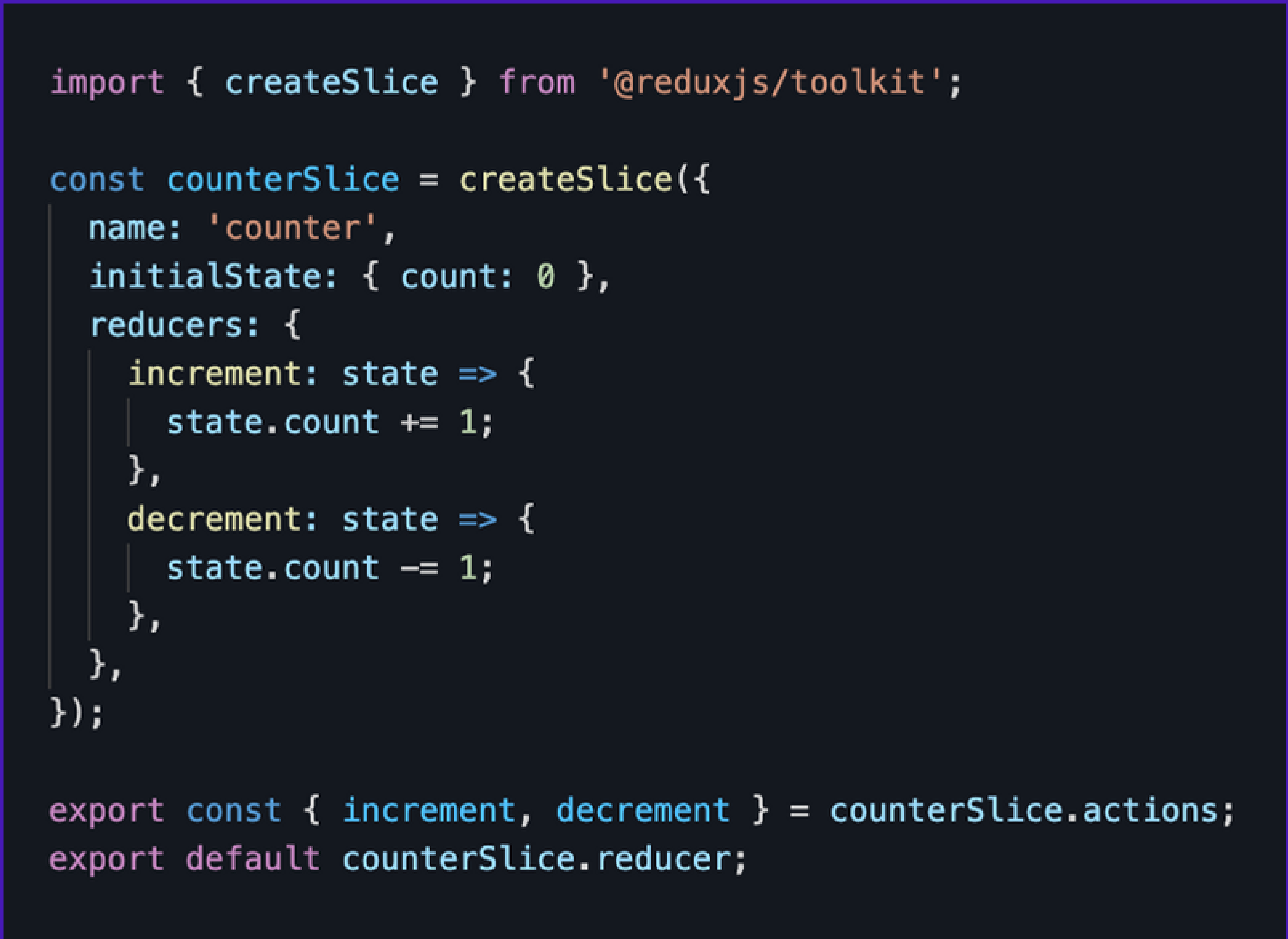
```
npm install @reduxjs/toolkit react-redux
```

Changes compared to old redux way:

- Createstore to ConfigureStore
- No combinereducers are nedded
- No need to use switch case and actions, this are replaced by Createslice

reducers and actions

createSlice that helps streamline the process of defining reducers and actions in Redux applications.

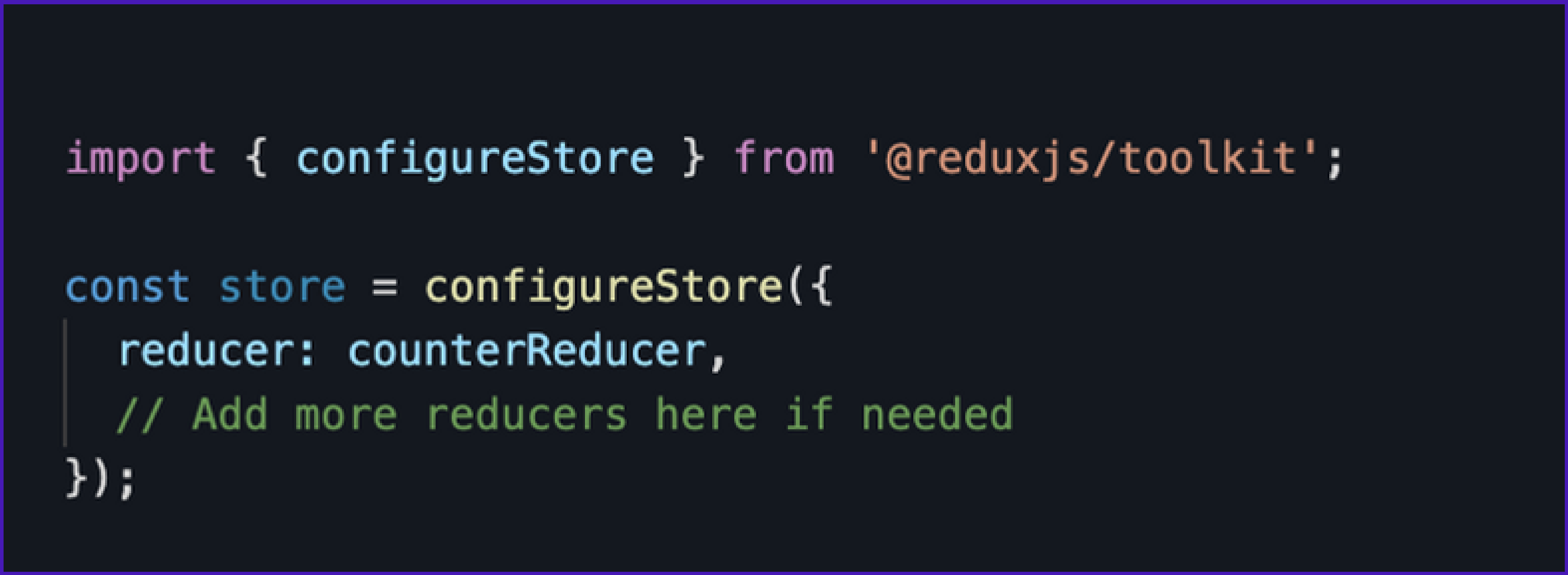


```
import { createSlice } from '@reduxjs/toolkit';

const counterSlice = createSlice({
  name: 'counter',
  initialState: { count: 0 },
  reducers: {
    increment: state => {
      state.count += 1;
    },
    decrement: state => {
      state.count -= 1;
    },
  },
});

export const { increment, decrement } = counterSlice.actions;
export default counterSlice.reducer;
```

Store

- **Redux Toolkit provides the configureStore function as a replacement for createStore to configure the Redux store with additional functionalities and middleware.**

- **configureStore automatically sets up middleware, including Redux DevTools Extension and thunk middleware for handling asynchronous logic.**



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

const store = configureStore({
  reducer: counterReducer,
  // Add more reducers here if needed
});
```

Practising more will make you good at the concept. do some work ...see the results

Follow on in

@Duvvuru Kishore