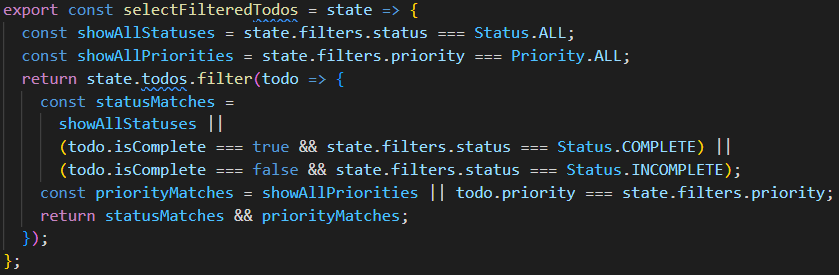
**Todo App:**

* Open Demos>React>redux in preferred editor
* Run npx create-react-app . in the CLI, once finished clean up files as you see fit
* Install React Redux and Redux Toolkit by running npm i react-redux @reduxjs/toolkit in the CLI
* Inside of the src directory, create the app and features directories
* In the src directory, create status and priority enum files
* Inside of the features directory, create the todos and filters directories
* In the todos directory, create the todosSlice.js file
  + Create an initial state array with 2 todo objects with id, text, priority, and isComplete values
  + Use the create slice method to create a todosSlice with a name of todos, initialState, and reducers for adding a todo object, toggling a todo’s isComplete status, and updating the todo’s priority
  + Export the todoSlice.actions and todoSlice.reducer as well as a selector that selects all todos
* In the app directory, create the store.js file
  + Create the store using the configureStore reducer and pass the todos reducer into the reducer object
  + Export the store as the default
* In index.js, import Provider from react-redux and wrap the app component in the Provider component
* Pass the store into the Provider
* In the todos directory, create a TodoList component and fill it with boilerplate React code
* Render TodoList in the App component and run the app to show that it is rendering properly
* In TodoList, use the useSelector hook to select all of the todos from todosSlice
  + Map through the todos array to return a div element containing the todo text, a dropdown for the priority, and a checkbox for isComplete
  + Render the new array of divs and switch over to the browser to show that both initial todos are rendering, but no changes to the todos register in the redux devtools
  + Add any CSS you want to make the todos look better
* Create a new file in the todos directory called TodoItem
  + Move the rendered element from the todos map into TodoItem and render a TodoItem instead, passing in the todo as a prop
  + Ensure that the select element has a value attribute set to todo.priority and the checkbox element has a checked attribute set to todo.isComplete
* In todosSlice, change one of the initial todos’ priotiry to high and isComplete to true and show that these input now render differently in the browser
* In TodoItem, create a handleIsCompleteToggle function that dispatches the todoToggled action and pass it into the checkbox element as an onChange event
  + Create a handlePriorityChange function that dispatches the todoPriorityChanged action and pass it into the select element as an onChange event
  + Switch to the browser and ensure that Redux DevTools is open
  + Change each todo item’s priority and check the checkbox, show that the state is changing in Redux and the actions are being tracked
* In the filters directory, create a new file called filtersSlice.js
  + Create an initial state object with a status set to Status.ALL and a priority set to Priority.ALL
  + Create a filtersSlice with the createSlice method and pass in an object with the name ‘filters’, the initial state, and reducers for changing the priority and status filters
  + Export the reducer actions, selector methods for selecting the current priority filter and status filter, and export the filtersSlice.reducer as default
* Add the filtersReducer to the store
* In the filters directory, create a new component called Filters.jsx and fill it with boilerplate component code
  + Use the useSelector hook to select the priority and status filters from the filtersSlice
  + Render a panel that allows users to select the current status and priority filters from dropdown elements
  + Ensure that each select element has a value attribute set to the priority filter or the status filter
  + Add any CSS you want to make this look nice and render the Filters component under the TodoList component in the App component
* Switch to the browser and ensure that the filter panel is rendering correctly
* In Filters, create functions for handling the status and priority filter changes that dispatch their related actions from filtersSlice
  + Pass the handler functions into their related select elements as onChange events
* Switch to the browser and ensure that the Redux DevTools panel is open
  + Change each filter a few times and show that the actions are being recorded and the Redux state updated
* In todosSlice, export a new selector that selects the todos that match our filters



* In TodoList, change the select function passed into useSelector to be selectFilteredTodos
* Switch to the browser and ensure that the Redux DevTools panel is open
  + Change the todos’ priorities and status and then change the filter options to show that our todo list begins filtering by status and priority

**Redux DevTools:**

* Open the same Redux demo from earlier
* In the todos directory, create a new file called AddTodo.jsx and fill it with boilerplate component code
  + Use the useState hook to create a state variable called newTodo and a setter for the variable
  + Render a form with a controlled text input for the newTodo text and a button (of type button)
  + Create a handleSubmit function for the button click that dispatches the todoAdded action
  + Create another function that calls handleSubmit if you press enter while in the text input
  + Add any CSS you want
* Render the AddTodo component in the App component above the TodoList component
* Ensure the app is running and switch to the browser
* Open the DevTools panel and show the students that the action is tracked and state updated when you add a new todo
  + Open the action panel and show that it list the action type as well as the payload passed
  + Open the state panel and show that it lists all of our initial state as well as our new todo
  + Open the diff panel and show that it list the difference between the state before and after the action
  + Open the trace panel and show that it traces the action through the code
  + Open the test panel and show that we can write tests right in the browser to ensure that redux will react as expected
  + Open the state panel back up and switch between the Tree, Chart, and Raw views to show the different ways in which data can be displayed