Redux 2.1_CW Exercises

ex01: create redux store

challenge

Create a file named `counterReducer.js`. In this file, define a reducer function called `counterReducer` that handles the state for a counter. The initial state and action types are provided. Make sure to import the necessary functions from Redux.

Action Types:

- `add`: Increment the counter by 1.
- `minus`: Decrement the counter by 1.

solution

```
const counterReducer = (state = defaultState, action) => {
  switch (action.type) {
    case 'add':
      return {
        ...state,
        counter: state.counter + 1,
      }
    case 'minus':
      return {
        ...state,
        counter: state.counter - 1,
      }
    default:
      return state
  }
}

export default counterReducer
```

COPY

ex02: attach counterReducer to store

challenge

Install react-redux dependency

In the `store.js` file, create a Redux store by importing the necessary functions from Redux and using the `counterReducer`

solution

```
import { createStore } from 'redux'

const store = createStore(counterReducer)

export default store
```

COPY

ex03: connect redux store to react app

challenge

In the `index.js` file, wrap the `App` component with the `Provider` component and connect it to the Redux store.

solution

```
import { StrictMode } from 'react'
import ReactDOM from 'react-dom'
import { Provider } from 'react-redux' // Import Provider
import store from './store'
import App from './App'

const rootElement = document.getElementById('root')
ReactDOM.render(
  <StrictMode>
    <Provider store={store}>
      <App />
    </Provider>
  </StrictMode>,
  rootElement,
)
```

COPY

ex04: access state from redux store

challenge

In the `App.js` file, use the `useSelector` hook to access the `counter` value from the Redux store and display it in the component.

solution

```
import { useSelector } from 'react-redux'

export default function App() {
  let counter = useSelector((state) => state.counter)

  return (
    <div className='App'>
      <div> Counter: {counter} </div>
      {/* ... */}
    </div>
  )
}
```

COPY

ex05: add `handleClick`

challenge

Add event listeners to the "add" and "minus" buttons in the `App.js` file, calling the `handleClick` function with the appropriate action type when each button is clicked.

solution

```
export default function App() {
  // ...

  return (
    <div className='App'>
      <div> Counter: {counter} </div>
      {/* Add event listeners to buttons */}
      <button onClick={(e) => handleClick('add')}> add </button>
      <button onClick={(e) => handleClick('minus')}> minus </button>
    </div>
  )
}
```

COPY

ex06: dispatch actions

challenge

In the `App.js` file, use the `useDispatch` hook to access the `dispatch` function from Redux. Implement the `handleClick` function to dispatch actions when the "add" and "minus" buttons are clicked.

solution

```
import { useDispatch } from 'react-redux'
```

```
export default function App() {
  // ...

  const dispatch = useDispatch()

  const handleClick = (type) => {
    dispatch({ type })
  }

  return (
    <div className='App'>
      <div> Counter: {counter} </div>
      {/* ... */}
    </div>
  )
}
```

COPY

entire solution

https://codesandbox.io/s/redux-without-toolkit-with-react-forked-8spdvl