React-Redux Part 2

Download Demo Code

Goals

- Avoid unnecessary re-rendering with ***useSelector***
- Include action types to avoid duplication
- Include action creators to reduce code
- Include React-Router with Redux

useSelector Revisited

Avoiding Renders

- If your callback to ***useSelector*** returns an object, then your component will re-render after every dispatch.
- This is true even if the values in the store haven't changed!

Example - Too Much Rendering

demo/selector-example/src/Math.js

```
function Math() {
  const { num1, num2 } = useSelector(st => ({
    num1: st.num1,
    num2: st.num2
  }));

  console.log("Math rendered");

  return (
    <div>
      <div>
        <h3>Math Facts</h3>
        <ul>
          <li>Sum: {num1 + num2}</li>
          <li>Difference: {num1 - num2}</li>
          <li>Product: {num1 * num2}</li>
          <li>Quotient: {num1 / num2}</li>
        </ul>
      </div>
    </div>
  );
}
```

This renders after every dispatch, even if ***num1*** and ***num2*** stay the same!

Reducing Renders

- There are two ways to minimize this kind of unnecessary re-rendering
- One approach us to use ***useSelector*** multiple times, so that we return multiple primitives rather than a single object.
- Another approach is to provide an *equality function* to ***useSelector***, which will tell it how to determine whether the values have actually changed or not.

Reducing Reducers - One Fix

demo/selector-example/src/MathLessRendering.js

```
import React from "react";
import { useSelector } from "react-redux";

/* Generate arithmetic facts about numbers in the redux store. */
function MathLessRendering() {
  const num1 = useSelector(st => st.num1);
  const num2 = useSelector(st => st.num2);

  console.log("MATH_LESS_RENDERING RENDERING");

  return (
    <div>
      <div>
        <h3>Math Facts (with less rendering)</h3>
        <ul>
          <li>Sum: {num1 + num2}</li>
          <li>Difference: {num1 - num2}</li>
          <li>Product: {num1 * num2}</li>
          <li>Quotient: {num1 / num2}</li>
        </ul>
      </div>
    </div>
  );
}
```

Reducing Reducers - Another Fix

demo/selector-example/src/MathLessRenderingAlt.js

```
import React from "react";
import { useSelector, shallowEqual } from "react-redux";

/* Generate arithmetic facts about numbers in the redux store. */
function MathLessRenderingAlt() {
  const { num1, num2 } = useSelector(
    st => ({ num1: st.num1, num2: st.num2 }),
    shallowEqual // does a shallow equality check on the values
  );

  console.log("MATH_LESS_RENDERING_ALT RENDERING");

  return (
    <div>
      <div>
        <h3>Math facts (also with less rendering)</h3>
        <ul>
          <li>Sum: {num1 + num2}</li>
          <li>Difference: {num1 - num2}</li>
          <li>Product: {num1 * num2}</li>
          <li>Quotient: {num1 / num2}</li>
        </ul>
      </div>
    </div>
  );
}
```

Common Redux Patterns

Action Types

- Move the value of the type property to a constant
- Define it once, reuse it multiple times!
- Better for tab completion in your editor

demo/counter/src/actionTypes.js

```
export const INCREMENT = "INCREMENT";
export const DECREMENT = "DECREMENT";
```

Our reducer using action types

demo/counter/src/actionTypes.js

```
export const INCREMENT = "INCREMENT";
export const DECREMENT = "DECREMENT";
```

demo/counter/src/rootReducer.js

```
import { INCREMENT, DECREMENT } from "./actionTypes";

const INITIAL_STATE = { count: 0 };

function rootReducer(state = INITIAL_STATE, action) {
  console.log("reducer ran; state & action:", state, action);

  switch (action.type) {
    case INCREMENT:
      return { ...state, count: state.count + 1 };

    case DECREMENT:
      return { ...state, count: state.count - 1 };

    default:
      return state;
  }
}
```

Action Creators

- Abstract the functions that create actions to their own file
- A function that creates an action is called an *action creator*

demo/counter/src/actions.js

```
import { INCREMENT, DECREMENT } from "./actionTypes";

export function increment() {
  return {
    type: INCREMENT
  };
}

export function decrement() {
  return {
    type: DECREMENT
  };
}
```

Our Component

demo/counter/src/Counter.js

```
import React from "react";
import { useSelector, useDispatch } from "react-redux";
import { increment, decrement } from "./actions";

function Counter() {
  const count = useSelector(st => st.count);
  const dispatch = useDispatch();

  function up(evt) { dispatch(increment()); }
  function down(evt) { dispatch(decrement()); }

  return (
    <div>
      <h2>Count is: {count}</h2>
      <button onClick={up}> + </button>
      <button onClick={down}> - </button>
    </div>
  );
}
```

React Router and Redux

Using React Router with Redux

```
<Provider store={store}>
  <BrowserRouter>
    <App/>
  </BrowserRouter>
</Provider>
```

Make sure you wrap ***<BrowserRouter>*** with ***<Provider>***. That's it!

Looking Ahead

Coming Up

- Async with Redux