Returning the State from the Reducer

A reducer should never be allowed to alter the state of the app. We'll see the correct way to handle the state below.

When returning state from reducers, there's something that may put you off at first.

However, if you already write good ReactJS code , then you should be familiar with this already:

> You should not mutate the state received in your Reducer. Instead, you should always return a new copy of the state.

Technically, you should never do this:

```
export default (state, action) => {
  switch (action.type) {
    case "SET_TECHNOLOGY":
      state.tech = action.tech;
      return state;
    default:
      return state;
  }
};
```

This is exactly why the reducer written in the previous lesson returned this:

```
return {
  ...state,
  tech: action.tech
};
```

Instead of mutating (or changing) the state received from the reducer, I am returning a **new** object. This object has all the properties of the previous state object thanks to the ES6 spread operator, **...state**. However, the **tech** field is updated to what comes in from the action, **action.text**.

Also, every Reducer you write should be a pure function with no side-effects I.e No API calls or updating a value outside the scope of the function.

Got that? Hopefully, yes.

Now, the Cashier isn't ignoring our actions. They're in fact giving us cash now! After doing this, click the buttons. Does it work now? *Gosh it still doesn't work*.

The text doesn't update.

What in the world is wrong this time? We will tackle state updates in the next lesson.