Making the Reducer Count

With the reducer receiving the action, we must specify that we want it to update the app's state based on the particular action.

Up until now, the reducer we've worked with hasn't done anything particularly smart. It's like a Cashier who is new to the job and does nothing with our WITHDRAW_MONEY intent.

What exactly do we expect the reducer to do?

For now, here's the initialState we passed into createStore when the STORE was created.

```
const initialState = { tech: "React" };
export const store = createStore(reducer, initialState);
```

When a user clicks any of the buttons, thus passing an action to the reducer, the new state we expect the reducer to return should have the action text in there!

Here's what I mean.

Current state is

```
{tech: "React"}
```

Given a new action of type SET_TECHNOLOGY, and text, React-Redux :

```
{
    type: "SET_TECHNOLOGY",
    text: "React-Redux"
}
```

What do you expect the new state to be? Yeah,

```
{tech: "React-Redux"}
```

```
{tech: 'React-Redux' }
```

The only reason we dispatched an action is because we want a new application state!

Like I mentioned earlier, the common way to handle different action types within a reducer is to use the Javascript switch statement as shown below:

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

Now we switch over the action type. But why?

Well, if you went to see a Cashier, you could have many different actions in mind.

You could want to WITHDRAW_MONEY, or DEPOSIT_MONEY or maybe just SAY_HELLO.

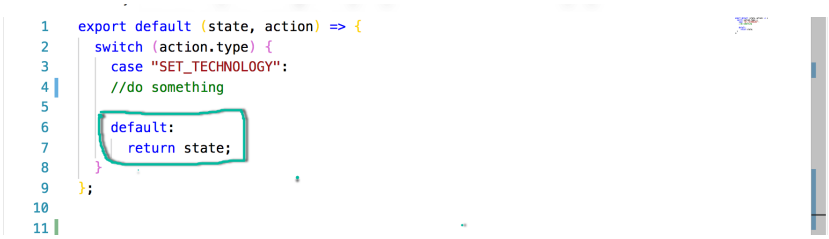
The Cashier is smart, so it will take in your action and respond based on your intent.

This is exactly what we're doing with the Reducer. The switch statement checks the type of the action.

Hey, What do you want to do? Withdraw, deposit, whatever...

After that, we then handle the known **cases** we expect. For now, there's just one case which is SET_TECHNOLOGY.

And by default, be sure to just return the **state** of the app.



So far so good.

The Cashier (Reducer) now understands our action. However, they aren't giving us any money (state) yet. Let's do something within the **case**.

Here's the updated version of the reducer. One that actually gives us *money* :)

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

Aw, yeah!

You see what I'm doing there?

I'll explain what's going on in the next section.