State and Subscriptions

Initializing state

1. Give your initial state a shape sort of like this:

```
const initialState = {
  name: {}, location: {}, login: {}
};
```

2. Run and test your page. You should see no errors, but also no data.
3. Populate initial state with personal values of your own:
   - The name object has first and last properties
   - The location object has a street, city, state, and postcode properties.
   - You should also add email and cell properties directly in the state object.
4. Run and test again. You should see your own data in the summary at the top and in the input fields below.
5. Change the data in the email and cell fields at the bottom. Do they change? (They should not). But are you seeing something in the console? (You should).

Responding to data changes

6. Note that whenever any value in any field is changed, a method is called. Look for it in Register.js.
7. In that method, we have a big switch statement capable of handling any of our defined fields. Your mission will be to notify the store that the email field and cell fields have changed. (You can ignore the other fields for now).
8. Find the case for the email field. Add a dispatch like so:

```
store.dispatch({type: "SET_EMAIL", email:e.target.value});
```

9. As you know, this will tell redux we want to run the reducer.
10. Edit store.js. Add a new condition for the action type "SET_EMAIL". If the action is "SET_EMAIL", return back a new object that looks just like the old one, but is using the action's email property.
11. Run and test. Once again, you should have no errors but if you stop it in the debugger or console.log the state that is coming out of that reducer function, you should see a complete object but it will have the new email address. This will prove that the data is changing.
12. Do the same for the cell field.

Subscribing

Now that the data is changing, let's refresh the page based upon it.

13. Edit App.js. In the constructor, subscribe to a method called this.refresh.
14. Add a method called refresh:

```
refresh = () => {
  // You have to fill in this line here.
  this.setState(newState);
}
```

15. Run and test. You will know it is working when you can change the email or cell number values and watch the page change live.

16. Bonus! If you have time, implement a reducer case for one property of name or location.