React – Shared State

In this lab, you will change a react component that uses the parent to maintain state into an app that uses a singleton.

# Objectives

In this lab, you will

* Examine an app that uses a Parent to manage the state
* Modify the app to use a singleton to manage the state
* Configure a Listener to indicate changes to the state

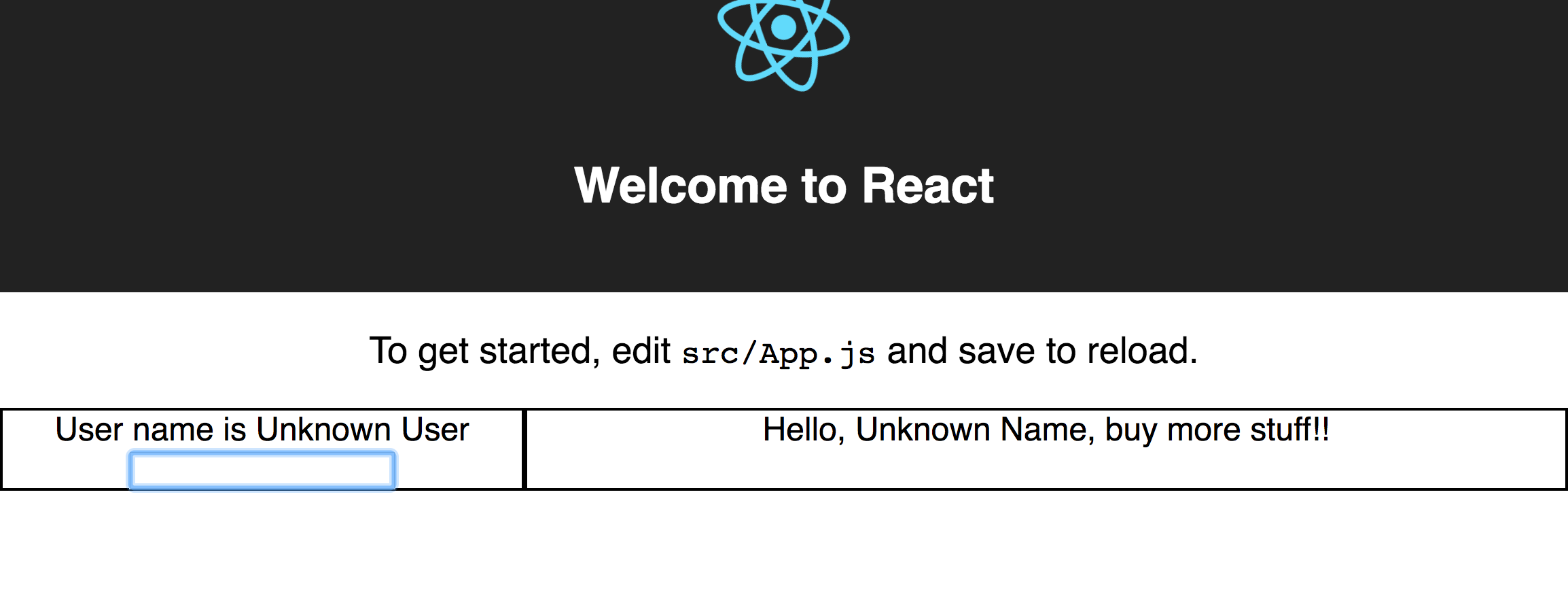
# Examine the Project

1. Open the solution/shared folder and start the application with:

yarn install

yarn start

1. Open the browser to <http://localhost:3000/> and see the below:



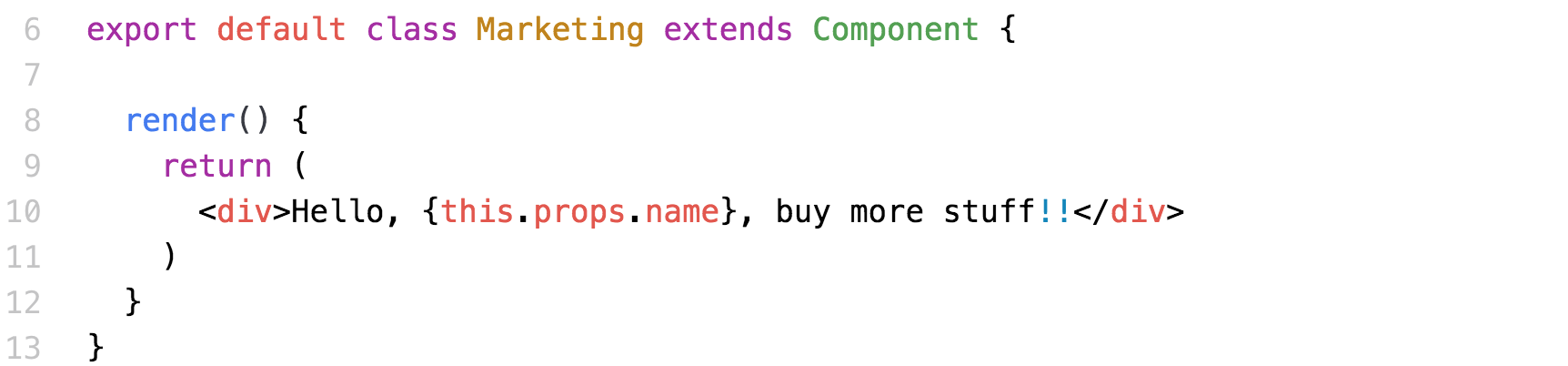
1. Enter a name and watch it change in BOTH components. These components do not necessarily have the same parent.
2. Stop the server.

# Start the Lab

1. Start the lab/shared server. Notice it works exactly the same, BUT it uses a Parent component to manage the clients. These clients MUST be direct children of the Parent or the intervening components must pass through the properties.
2. Examine the src/components/Login.js file shown below



1. In the above, there is a callback in line 16 that sends the user input to the parent. It also uses the name property in line 22 to show the current user.
2. On line 10 in Marketing.js below, it uses the property, name, to display the current user.



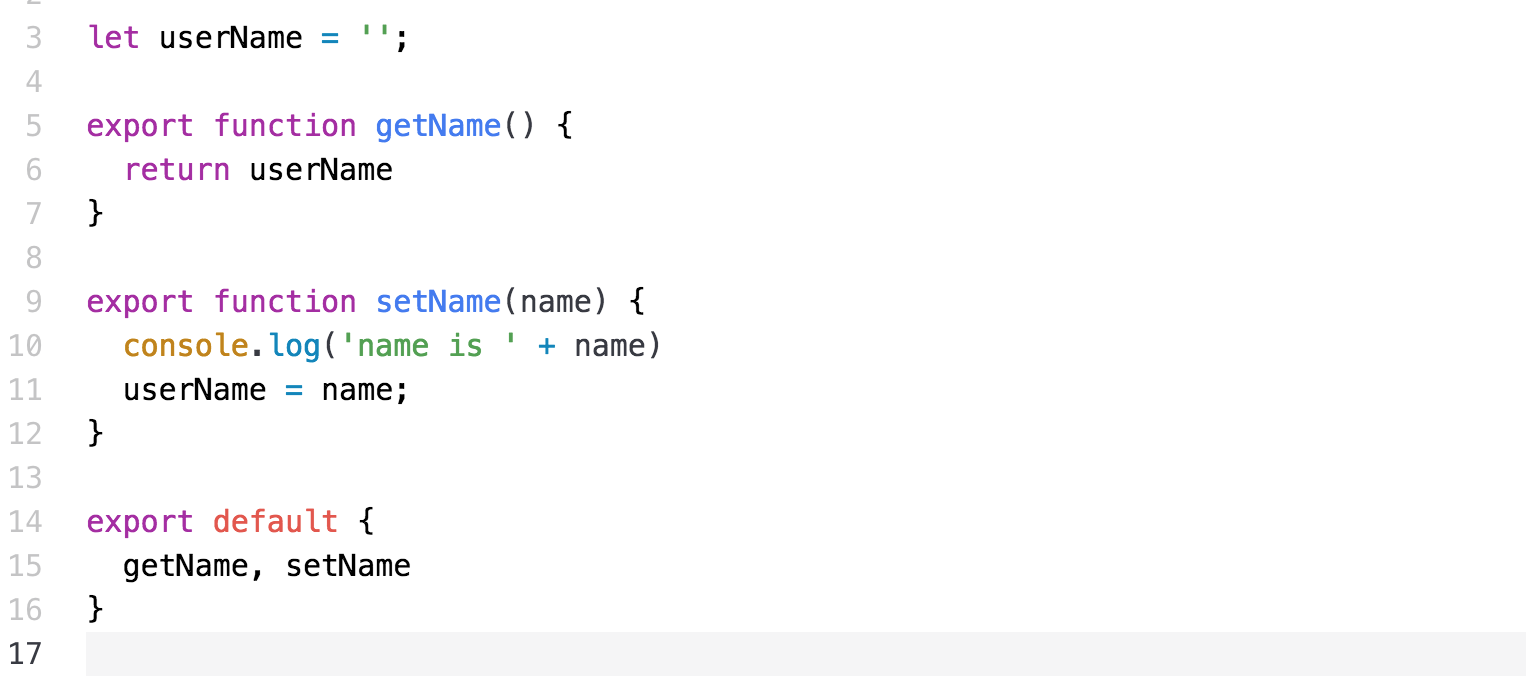
1. The parent is Layout.js shown below.



1. We use the state variable, name, to contain the current user.
2. This is very simple, but has a problem in that the <Login> and <Marketing> components MUST be children of the same parent.

# Convert to a Singleton

1. The singleton is UserInfo.js shown below.



1. Notice it exports two functions, getName() and setName().
2. Change the Login.js code to get / set the name in the singleton with:



1. Notice in line 18, we send the name to the UserInfo instead of the callback.
2. Notice in line 26, we get the user name from the singleton.
3. Edit the Marketing.js file and register a listener to the singleton so changes from UserInfo.setName() passes the name to this component as shown below.



1. Notice in lines 10-12, we define the state variable, name. In componentDidMount() on line 14, we register the listener and configure the callback to our method, changeName() on line 18.
2. In changeName() we set the state variable which triggers a re-render().

# Create a Listener

1. Edit the UserInfo.js file again and add the where we implement the Observer (Listener) pattern shown below.



1. Line 22 defines the array of listener callbacks. When the client code registers a new listener, it adds the listener to the array.
2. In emitEvent() on line 34, it calls all the registered callbacks.
3. Run the application. Now the two components are separated.
4. Have fun!

Congratulations. You have completed this lab.