# Child Components Update Sibling Components

Patterns within patterns within patterns!

In lesson 1, you learned your first React programming pattern: a stateful, parent component passes down a prop to a stateless, child component.

In lesson 2, you learned that lesson 1's pattern is actually part of a larger pattern: a stateful, parent component passes down an event handler to a stateless, child component. The child component then uses that event handler to update its parent's state.

In this lesson, we will expand the pattern one last time. A child component updates its parent's state, and the parent passes that state to a sibling component.

An understanding of this final pattern will be very helpful in the wild, not to mention in the next React course. Click Next and we'll build an example!

# One Sibling to Display, Another to Change

One of the very first things that you learned about components is that they should only have one job.

In the last lesson, Child had two jobs:

1 - Child displayed a name.

2 - Child offered a way to change that name.

You should make like Solomon and divide Childin two: one component for displaying the name, and a different component for allowing a user to change the name.

In the code editor, select **Child.js**. Notice that these lines have vanished:

<h1> Hey, my name is {this.props.name}! </h1>

The new version of Child renders a dropdown menu for changing the name, and that's it.

Select **Sibling.js** in the code editor.

Read through the render function's returnstatement.

Here, the name is displayed! Or at least it will be displayed, once you've done a little editing.

That brings us to the essential new concept for this lesson: you will have one stateless component display information, and a different stateless component offer the ability to changethat information.

**Pass the Right props to the Right Siblings**

Look at **Parent.js** in the code editor.

Three things have changed in this file since the last Lesson:

1. Sibling has been required on line 4.
2. A <Sibling /> instance has been added to the render function on line 27.
3. <Sibling /> and <Child /> have been wrapped in a <div></div>, since JSX expressions must have only one outer element.

# Display Information in a Sibling Component

You're on the last step!

You've passed the name down to <Sibling />as a prop. Now <Sibling /> has to displaythat prop.

**Stateless Components Inherit From Stateful Components Recap**

You just executed your first complete React programming pattern!

Let's review. Follow each step in the code editor:

* A *stateful* component class defines a function that calls this.setState. (**Parent.js**, lines 15-19)
* The stateful component passes that function down to a stateless component. (**Parent.js**, line 24)
* That *stateless* component class defines a function that calls the passed-down function, and that can take an *event object*as an argument. (**Child.js**, lines 10-13)
* The stateless component class uses this new function as an event handler. (**Child.js**, line 20)
* When an event is detected, the parent's state updates. (A user selects a new dropdown menu item)
* The stateful component class passes down its state, distinct from the ability to *change*its state, to a different stateless component. (**Parent.js**, line 25)
* That stateless component class receives the state and displays it. (**Sibling.js**, lines 5-10)
* An instance of the stateful component class is rendered. One stateless child component displays the state, and a different stateless child component displays a way to *change*the (**Parent.js**, lines 23-26)

This pattern occurs in React all the time! The more that you see it, the more that its elegance will become clear.

Being introduced to this pattern is your first step towards understanding how React apps fit together! You'll get more practice using it throughout this course, as well as in the course after this one.