Passing Data Between React Components

Ruth M. Pardee Follow

Jun 8, 2016 · 3 min read

Because of React's one-way data flow, it often can be tricky to see how data can flow from one component to another. Data sometimes needs to be able to move from children to parent, parent to children, or between siblings.

Here is example structure of the React components which comprise my app:

```
App
  |
```

```
    |__ InputBar
    |
    |__ ToDoList
          |
          |__ToDoItem
```

Parent to Child—Use Prop

This is the easiest direction in React to transfer data. If I have access to data in my parent component that I need my child component to have access to, I can pass it as a prop to the child when I instantiate it within the parent.

In my example, if I need to pass something from the App to the ToDoList:

```
  class App extends React.Component {
```

```
      render() {
```

```
    [... somewhere in here I define a
variable listName which I think will be
useful as data in my ToDoList component...]

        return (
            <div>
                <InputBar/>
                <ToDoList
listNameFromParent={listName}/>
            </div>
        );


    }
}
```

Now in the ToDoList component, if I use this.props.listNameFromParent I will have access to that data.

Child to Parent—Use a callback and states

This one is a bit trickier. If I have data in my child that my parent needs access to, I can do the following:

1. Define a callback in my parent which takes the data I need in as a parameter.
2. Pass that callback as a prop to the child (see above).
3. Call the callback using this.props.[callback] in the child (insert your own name where it says [callback] of course), and pass in the data as the argument.

Here's what that might look like if I had data in ToDoItem that I need to access in ToDoList:

```
class ToDoList extends React.Component {
```

```
    myCallback = (dataFromChild) => {
        [...we will use the dataFromChild
here...]
    },

    render() {
        return (
            <div>
                 <ToDoItem
callbackFromParent={this.myCallback}/>
            </div>
        );

    }
}
```

Now from within ToDoItem we can pass something to callbackFromParent:

```
class ToDoItem extends React.Component{
```

```
    someFn = () => {
        [...somewhere in here I define a
variable listInfo which    I think will be
useful as data in my ToDoList component...]


this.props.callbackFromParent(listInfo);
    },


    render() {
        [...]
    }
};
```

ToDoList will now be able to use listInfo within it's myCallback function!

But what if I want to use listInfo in a different function within ToDoList, not just in myCallback? With this implementation, I would only have access as a parameter passed into that one specific method.

Easy: set this parameter as a state within ToDoList. You can almost think of it as creating a variable within the scope of ToDoList that all the methods within that component can access. In that case my code defining ToDoList might look something like:

```
class ToDoList extends React.Component {

    constructor(props) {
        super(props);
        this.state = {
            listDataFromChild: null

        };
    },

    myCallback = (dataFromChild) => {
        this.setState({ listDataFromChild:
dataFromChild });
    },

    otherFn = () => {
```

```
        [...within this other function now I
still have access to
this.state.listDataFromChild...]


    }


    render() {
        return (
            <div>


                <ToDoItem
callbackFromParent={this.myCallback}/>


                [...now here I can pass
this.state.listDataFromChild as a prop to
any other child component...]

            </div>
        );


    }
});
```

Note the use of the arrow function in the definition of myCallback. This allows us to avoid using .bind when invoking, as it will retain the context of where it is called.

Between Siblings—Combine the above

Not surprisingly, to pass data between siblings, you have to use the parent as an intermediary. First pass the data from the child to the parent, as an argument into a callback from the parent. Set this incoming parameter as a state on the parent component, then pass it as a prop to the other child (see above example). The sibling can then use the data as a prop.

Passing data between React components can be a little tricky at first (without using Redux that is), but once you practice these three techniques you'll be able to pass data between whichever components you'd like.