Chapter 22: React Forms

Section 22.1: Controlled Components

A controlled component is bound to a value and its changes get handled in code using event based callbacks.

```
class CustomForm extends React.Component {
constructor() {
    super();
    this.state = {
        person: {
            firstName: '',
            lastName: ''
        }
    }
}

handleChange(event) {
    let person = this.state.person;
    person[event.target.name] = event.target.value;
    this.setState({person});
}

render() {
    return (
        <form>
            <input
            type="text"
            name="firstName"
            value={this.state.firstName}
            onChange={this.handleChange.bind(this)} />

            <input
            type="text"
            name="lastName"
            value={this.state.lastName}
            onChange={this.handleChange.bind(this)} />
        </form>
    )
}
```

}

In this example we initialize state with an empty person object. We then bind the values of the 2 inputs to the individual keys of the person object. Then as the user types, we capture each value in the `handleChange` function. Since the values of the components are bound to state we can rerender as the user types by calling `setState()`.

NOTE: Not calling `setState()` when dealing with controlled components, will cause the user to type, but not see the input because React only renders changes when it is told to do so.

It's also important to note that the names of the inputs are same as the names of the keys in the person object. This allows us to capture the value in dictionary form as seen here.

```
handleChange(event) {
    let person = this.state.person;
    person[event.target.name] = event.target.value;
    this.setState({person});
```

```
}
```

`person[event.target.name]` is the same is a `person.firstName || person.lastName`. Of course this would depend on which input is currently being typed in. Since we don't know where the user will be typing, using a dictionary and matching the input names to the names of the keys, allows us to capture the user input no matter where the onChange is being called from.