Events

- In the real world, you are probably going to change your component state based on some user interaction.
- This is where events come in. You can add methods to your components to handle events.
- Event Types

Events - Example

```
class EventExample extends Component {
  save() {
    alert('clicked save!');
  },
  render() {
    return <div>
        <button onClick={this.save}>Save</button>
    </div>;
  }
};
```

Function binding

- Be careful when your event functions are using `this`:
- React in ES6 does not automatically bind your functions for event handlers. Any event handlers need to be bound with one of two ways;
- Using the `constructor` function or using an arrow function

Forms

- The different kinds of form components are 'input', 'option' and 'textarea'
- React form components are special in that their property values can be changed by the user as well as the code.
- We call these special props "Interactive Props".

Forms - Interactive Props

- value: supported by `<input>` and `<textarea>` components.
- checked: supported by `<input>` components of type checkbox or radio.
- selected: supported by `<option>` components.

Controlled Components

- form components that have the property "value" defined.
- if you're going to define the "value" property you must define the "onChange" handler so that you can be notified everytime the user modifies that form element.

Example

```
class FormTest extends Component {
  constructor(props) {
    super(props);
    this.state = {
      name: 'David'
    };

    this.handleNameChange = this.handleNameChange.bind(this);
  }
  render() {
    return <div>
      <input type='text' value={this.state.name} onChange={this.handleNameChange}/>
    </div>
  }
  handleNameChange(event) {
    this.setState({
      name: event.target.value
    });
  }
}
```

Uncontrolled Componenets

- Simply, form components that do not have a "value" property defined.
- You can define the "defaultValue" if you want the component to be initialized with some value.

Example - Select Form Elements

```
<select value="CA" onChange={this.handleSelectChange}>
  <option value="CA">California</option>
  <option value="FL">Florida</option>
  <option value="NY">New York</option>
</select>
```

Exercise - Form Edit

- Define a Component that displays a user's first and last name in two div tags on the screen along with an 'edit' button.
- When the user clicks on the 'edit' button, the user's first and last name are displayed in two input fields along with a 'save' and 'cancel' button. The 'edit' button, and two div tags should not be visible after the user presses 'edit'!
- The user is then allowed to make any changes they want and either press 'save' or 'cancel'
- If they press 'save' any changes they made are reflected into the two divs
- If they press 'cancel' any changes they made are discarded and the divs go back to what they were before pressing 'edit'

Exercise - Folder Contents Toggle

- Write a Folder component that displays the name of a folder called 'Home'.
- Under the folder there should be three files in an unordered list: 'File1', 'File2', 'File3'.
- Add a button that toggles the visiblity of the files on the screen everytime it is clicked.
- eg: Initially the folder and files are all on the screen, but when the 'Toggle' button is clicked the files disappear, then when it is clicked again, the files reappear.