Chapter 7: Forms and User Input

Section 7.1: Controlled Components

Controlled form components are defined with a `value` property. The value of controlled inputs is managed by React, user inputs will not have any direct influence on the rendered input. Instead, a change to the `value` property needs to reflect this change.

```
class Form extends React.Component {
  constructor(props) {
    super(props);

    this.onChange = this.onChange.bind(this);

    this.state = {
      name: ''
    };
  }

  onChange(e) {
    this.setState({
      name: e.target.value
    });
  }

  render() {
    return (
      <div>
        <label for='name-input'>Name: </label>
        <input
          id='name-input'
          onChange={this.onChange}
          value={this.state.name} />
      </div>
    )
  }
}
```

The above example demonstrates how the `value` property defines the current value of the input and the `onChange` event handler updates the component's state with the user's input.

Form inputs should be defined as controlled components where possible. This ensures that the component state and the input value is in sync at all times, even if the value is changed by a trigger other than a user input.

Section 7.2: Uncontrolled Components

Uncontrolled components are inputs that do not have a `value` property. In opposite to controlled components, it is the application's responsibility to keep the component state and the input value in sync.

```
class Form extends React.Component {
  constructor(props) {
    super(props);

    this.onChange = this.onChange.bind(this);

    this.state = {
      name: 'John'
```

```
    };
  }

  onChange(e) {
    this.setState({
      name: e.target.value
    });
  }

  render() {
    return (
      <div>
        <label for='name-input'>Name: </label>
        <input
          id='name-input'
          onChange={this.onChange}
          defaultValue={this.state.name} />
      </div>
    )
  }
}
```

Here, the component's state is updated via the `onChange` event handler, just as for controlled components. However, instead of a `value` property, a `defaultValue` property is supplied. This determines the initial value of the input during the first render. Any subsequent changes to the component's state are not automatically reflected by the input value; If this is required, a controlled component should be used instead.