Form & Validation

CS568 – Web Application Development I

Assistant Professor Umur Inan
Computer Science Department
Maharishi International University

Maharishi International University - Fairfield, Iowa

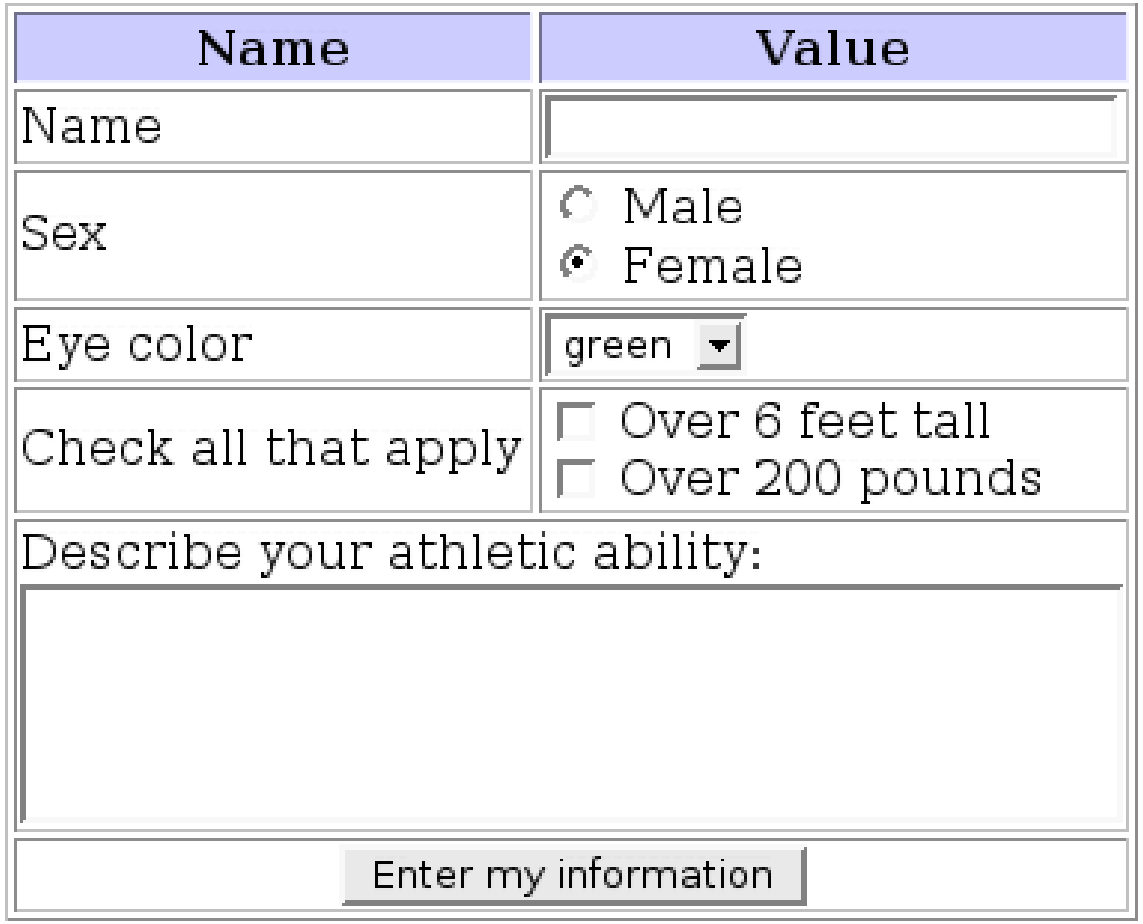
Content

- Controlled & Uncontrolled Components
- Custom Dynamic Input Components
- Form validation
- Refs
  - Imperative vs declarative animation
  - Integrating the React with the third-party libraries

Web form

A webform or HTML form on a web page allows a user to enter data that is sent to a server for processing. Forms can resemble paper or database forms because web users fill out the forms using checkboxes, radio buttons, or text fields.

React forms are slightly different than the HTML form.



Controlled Components

In HTML, form elements such as <input>, <textarea>, and <select> typically **maintain their own state** and update it based on user input.

In React, mutable state is typically kept in the state property of components, and **only updated with setState**().

An input form element whose value is controlled by React in this way is called a "**controlled component**".

```
class NameForm extends React.Component {
  constructor(props) {
    super(props);
    this.state = {value: ''};
    ...
  }

  handleChange(event) {
    this.setState({value: event.target.value});
  }

  handleSubmit(event) {
    event.preventDefault();
    alert('A name was submitted: ' + this.state.value);
  }

  render() {
    return (
      <form onSubmit={this.handleSubmit}>
        <input type="text" value={this.state.value} onChange={this.han
dleChange} />
        <input type="submit" value="Submit" />
      </form>
    ...
```

The textarea Tag

In HTML, a <textarea> element defines its text by its children:

```
<textarea>
  Hello there, this is some text in a text area
</textarea>
```

In React, a <textarea> uses a value attribute instead

```
<textarea value={this.state.value} onChange={this.handleChange} />
```

The select Tag

In HTML, the default option is selected with selected attribute.

```
<select>
  <option selected value="coconut">Coconut</option>
  <option value="mango">Mango</option>
</select>
```

In React, it uses a value attribute on the root select tag

```
<select value={this.state.value} onChange={this.handleChange}>
```

Form validation

There are three main reasons why we insist on validating forms:

1. We want to get the right data, in the right format
2. We want to protect our users
3. We want to protect ourselves' data

We can validate form data on server side and client side.

Before submitting data to the server, it is important to ensure all required form controls are filled out, in the correct format. This is called client-side form validation, and helps ensure data submitted matches the requirements set forth in the various form controls.

Form validation

There are three ways to validate the user input:

1. Writing your own validation library using built-in form validation uses HTML 5 features – You have full control on it. But it will take effort and time to get it done.
2. Using third-party libraries – Saves effort and time to develop validation. But you may not validate some complex fields.
3. Shared library in the company – Reusing the code but you need to ping to the other team to request changes.

You can validate the form on the following events:

- OnChange
- OnClick (of the submit button)

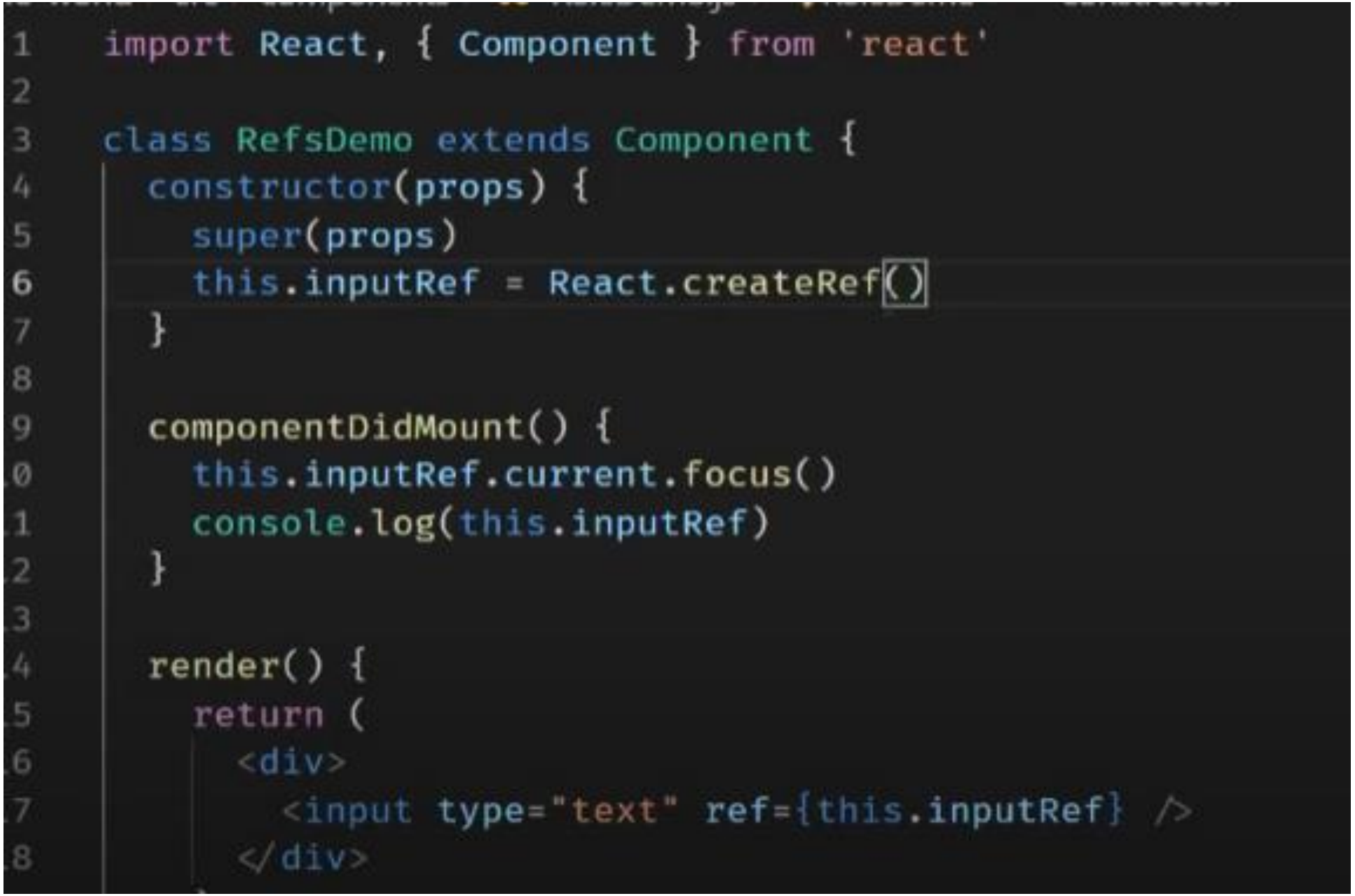
Refs

Refs provide a way to access DOM nodes or React elements created in the render method.

It gets the value from the DOM, not from the React state. It is uncotrolled by React.

When to Use Refs

- Managing focus, text selection, or media playback.
- Triggering imperative animations.
- Integrating with third-party DOM libraries.



```
import React, { Component } from 'react'

class RefsDemo extends Component {
  constructor(props) {
    super(props)
    this.inputRef = React.createRef()
  }

  componentDidMount() {
    this.inputRef.current.focus()
    console.log(this.inputRef)
  }

  render() {
    return (
      <div>
        <input type="text" ref={this.inputRef} />
      </div>
```