极客学院
jikexueyuan.com

# 表单详解

# 表单详解 — **课程概要**

- 不可控组件和可控组件介绍
- 不同表单元素的使用
- 事件处理函数复用
- 表单组件自定义

# 不可控组件和可控组件介绍

# 不可控组件和可控组件介绍

- 什么是不可控组件
- 什么是可控组件
- 为什么组件要可控
- 实例演示

# 不可控组件和可控组件介绍 — 什么是不可控组件

```
<input type="text" defaultValue="Hello World!" />
```

```
var inputValue = ???
```

```
var inputValue = React.findDOMNode(this.refs.input).value
```

# 不可控组件和可控组件介绍 — 什么是可控组件

```
<input type="text" defaultValue={this.state.value} />
```

```
var inputValue = this.state.value
```

# 不可控组件和可控组件介绍 — **为什么组件要可控**

组件可控的好处：

- 符合React的数据流
- 数据存储在state中，便于使用
- 便于对数据进行处理

# 不可控组件和可控组件介绍 — **实例演示**

不可控组件实例

可控组件实例

# 不同表单元素的使用

# 不同表单元素的使用

- 表单元素介绍
- 实例演示

# 不同表单元素的使用 — **表单元素介绍**

<label>

<input>

<textarea>

<select>

```
<select value={this.state.helloTo}
    onChange={this.handleChange}>
    <option value="one">一</option>
    <option value="two">二</option>
</select>
```

# 不同表单元素的使用 — **实例演示**

表单元素实例

# 事件处理函数复用

# 事件处理函数复用

- 为什么要复用事件处理函数
- 事件处理函数的两种复用方式
- 实例演示

# 事件处理函数复用 — 为什么要复用事件处理函数

onChange = {this.handleChange1}

onChange = {this.handleChange2}

onChange = {this.handleChange3}

onChange = {this.handleChange4}

onChange = {this.handleChange5}

onChange = {this.handleChange}

# 事件处理函数复用 — **事件处理函数的两种复用方式**

```
handleChange: function(name, event) {
    ...
}
{this.handleChange.bind(this, 'input1')}
```

bind复用

```
handleChange: function(event) {
    var name = event.target.name
}
{this.handleChange}
```

name复用

# 事件处理函数复用 — **实例演示**

bind复用

name复用

# 表单组件自定义

# 表单组件自定义

- 为什么要自定义表单组件
- 表单组件的两种定义方式
- 实例演示

# 表单组件自定义 — **为什么要自定义表单组件**

自定义表单组建原因

- 内因：表单本身具备特殊性：样式统一、信息内聚、行为固定
- 外因：本质上是组件嵌套，组织和管理组件的一种方式

# 表单组件自定义 — 表单组件的两种定义方式

```
var MyForm = React.createClass({
    submitHandler: function (event) {
        event.preventDefault();
        alert(this.refs.radio.state.value);
    },
    render: function () {
        return <form onSubmit={this.submitHandler}>
            <Radio ref="radio" name="my_radio" defaultValue="B">
                <option value="A">First Option</option>
                <option value="B">Second Option</option>
                <option value="C">Third Option</option>
            </Radio>
            <button type="submit">Speak</button>
        </form>;
    }
});
```

不可控的自定义组件

# 表单组件自定义 — 表单组件的两种定义方式

```
var MyForm = React.createClass({
    getInitialState: function () {
        return {my_radio: "B"};
    },
    handleChange: function (event) {
        this.setState({
        my_radio: event.target.value
        });
    },
    submitHandler: function (event) {
        event.preventDefault();
        alert(this.state.my_radio);
    },
    render: function () {
        return <form onSubmit={this.submitHandler}>
        <Radio name="my_radio"
            value={this.state.my_radio}
            onChange={this.handleChange}>
            <option value="A">First Option</option>
            <option value="B">Second Option</option>
            <option value="C">Third Option</option>
        </Radio>
        <button type="submit">Speak</button>
        </form>;
    }
});
```

可控的自定义组件

# 表单组件自定义 — **实例演示**

不可控组的自定义组件

可控的自定义组件

# 表单详解

本课程中我们学习了React中表单的使用方式，你应当掌握了以下知识：

- 可控组件和不可控组件
- 表单元素的使用
- 复用事件处理函数
- 自定义表单组件

你可以使用这些技巧来编写更具交互性、更符合React特性的组件，如果想继续提高，你可以在极客学院继续学习React系列课程。

# 极客学院

jikexueyuan.com

中国最大的IT职业在线教育平台