React AJAX React Refs

React 表单与事件

本章节我们将讨论如何在 React 中使用表单。

HTML 表单元素与 React 中的其他 DOM 元素有所不同,因为表单元素生来就保留一些内部状态。

在 HTML 当中，像 <input>, <textarea>, 和 <select> 这类表单元素会维持自身状态，并根据用户输入进行更新。但在React 中，可变的状态通常保存在组件的状态属性中，并且只能用 setState() 方法进行更新。

一个简单的实例

在实例中我们设置了输入框 input 值 **value = {this.state.data}**。在输入框值发生变化时我们可以更新 state。我们可以使用 **on Change** 事件来监听 input 的变化，并修改 state。

React 实例

```
class HelloMessage extends React.Component {
constructor(props) {
super(props);
this.state = {value: 'Hello Runoob!'};
this.handleChange = this.handleChange.bind(this);
}
handleChange(event) {
this.setState({value: event.target.value});
}
render() {
var value = this.state.value;
return <div>
<input type="text" value={value} onChange={this.handleChange} />
<h4>{value}</h4>
</div>;
}
}
ReactDOM.render(
<HelloMessage />,
document.getElementById('example')
);
```

尝试一下 »

上面的代码将渲染出一个值为 Hello Runoob! 的 input 元素，并通过 onChange 事件响应更新用户输入的值。

实例 2

在以下实例中我们将为大家演示如何在子组件上使用表单。 **onChange** 方法将触发 state 的更新并将更新的值传递到子组件的输入框的 **value** 上来重新渲染界面。

你需要在父组件通过创建事件句柄 (**handleChange**) ，并作为 prop (**updateStateProp**) 传递到你的子组件上。

React 实例

```
class Content extends React.Component {
render() {
return <div>
<input type="text" value={this.props.myDataProp} onChange={this.props.updateStateProp} />
<h4>{this.props.myDataProp}</h4>
</div>;
}
}
class HelloMessage extends React.Component {
constructor(props) {
super(props);
this.state = {value: 'Hello Runoob!'};
this.handleChange = this.handleChange.bind(this);
}
handleChange(event) {
this.setState({value: event.target.value});
}
render() {
var value = this.state.value;
return <div>
<Content myDataProp = {value}
updateStateProp = {this.handleChange}></Content>
</div>;
}
}
ReactDOM.render(
<HelloMessage />,
document.getElementById('example')
);
```

尝试一下 »

Select 下拉菜单

在 React 中，不使用 selected 属性，而在根 select 标签上用 value 属性来表示选中项。

React 实例

```
class FlavorForm extends React.Component {
constructor(props) {
super(props);
this.state = {value: 'coconut'};
this.handleChange = this.handleChange.bind(this);
this.handleSubmit = this.handleSubmit.bind(this);
}
handleChange(event) {
this.setState({value: event.target.value});
}
handleSubmit(event) {
alert('Your favorite flavor is: ' + this.state.value);
event.preventDefault();
}
render() {
return (
```

```
<form onSubmit={this.handleSubmit}>
<label>
选择您最喜欢的网站
<select value={this.state.value} onChange={this.handleChange}>
<option value="gg">Google</option>
<option value="rn">Runoob</option>
<option value="tb">Taobao</option>
<option value="fb">Facebook</option>
</select>
</label>
<input type="submit" value="提交" />
</form>
);
}
}
ReactDOM.render(
<FlavorForm />,
document.getElementById('example')
);
```

尝试一下 »

多个表单

当你有处理多个 input 元素时，你可以通过给每个元素添加一个 name 属性，来让处理函数根据 `event.target.name` 的值来选择做什么。

React 实例

```
class Reservation extends React.Component {
constructor(props) {
super(props);
this.state = {
isGoing: true,
numberOfGuests: 2
};
this.handleInputChange = this.handleInputChange.bind(this);
}
handleInputChange(event) {
const target = event.target;
const value = target.type === 'checkbox' ? target.checked : target.value;
const name = target.name;
this.setState({
[name]: value
});
}
render() {
return (
<form>
<label>
是否离开:
<input
name="isGoing"
type="checkbox"
```

```
checked={this.state.isGoing}
onChange={this.handleInputChange} />
</label>
<br />
<label>
访客数:
<input
name="numberOfGuests"
type="number"
value={this.state.numberOfGuests}
onChange={this.handleInputChange} />
</label>
</form>
);
}
}
```

尝试一下 »

React 事件

以下实例演示通过 onClick 事件来修改数据：

React 实例

```
class HelloMessage extends React.Component {
constructor(props) {
super(props);
this.state = {value: 'Hello Runoob!'};
this.handleChange = this.handleChange.bind(this);
}
handleChange(event) {
this.setState({value: '菜鸟教程'})
}
render() {
var value = this.state.value;
return <div>
<button onClick={this.handleChange}>点我</button>
<h4>{value}</h4>
</div>;
}
}
ReactDOM.render(
<HelloMessage />,
document.getElementById('example')
);
```

尝试一下 »

当你需要从子组件中更新父组件的 **state** 时，你需要在父组件通过创建事件句柄 (**handleChange**) ，并作为 prop (**updateStateProp**) 传递到你的子组件上。实例如下：

React 实例

```
class Content extends React.Component {
render() {
return <div>
<button onClick = {this.props.updateStateProp}>点我</button>
<h4>{this.props.myDataProp}</h4>
</div>
}
}
class HelloMessage extends React.Component {
constructor(props) {
super(props);
this.state = {value: 'Hello Runoob!'};
this.handleChange = this.handleChange.bind(this);
}
handleChange(event) {
this.setState({value: '菜鸟教程'})
}
render() {
var value = this.state.value;
return <div>
<Content myDataProp = {value}
updateStateProp = {this.handleChange}></Content>
</div>;
}
}
ReactDOM.render(
<HelloMessage />,
document.getElementById('example')
);
```

尝试一下 »

React AJAX React Refs

点我分享笔记