通常情况下你可以用普通的 JavaScript 类定义一个组件:

class Greeting extends React.Component {

render() {

return <h1>Hello, {this.props.name}</h1>;

}

}

如果你不使用 ES6 ，你可以使用 create-react-class 方法代替：

var createReactClass = require('create-react-class');

var Greeting = createReactClass({

render: function() {

return <h1>Hello, {this.props.name}</h1>;

}

});

**声明默认 Props**

在函数和 ES6 classes(类) 中，defaultProps 被定义为组件本身的属性：

class Greeting extends React.Component {

// ...

}

Greeting.defaultProps = {

name: 'Mary'

};

在 createReactClass() 中，你需要在所传递的对象中定义 getDefaultProps() 方法:

var Greeting = createReactClass({

getDefaultProps: function() {

return {

name: 'Mary'

};

},

// ...

});

**设置初始化 State(状态)**

在 ES6 classes(类) 中，你可以在构造函数通过给 this.state 赋值来定义初始状态:

class Counter extends React.Component {

constructor(props) {

super(props);

this.state = {count: props.initialCount};

}

// ...

}

在 createReactClass() 函数中，你可以提供一个指定的 getInitialState 方法来返回初始状态:

var Counter = createReactClass({

getInitialState: function() {

return {count: this.props.initialCount};

},

// ...

});

**自动绑定**

在以 ES6 classes(类) 方式声明的 React 组件中，方法遵循与普通 ES6 的 class 中相同的语义。也就是说方法不会自动绑定 this 到实例中，你必须在构造函数中显式的使用 .bind(this) :

class SayHello extends React.Component {

constructor(props) {

super(props);

this.state = {message: 'Hello!'};

// 将handleClick方法绑定到当前类，这样handleClick 的this就指向当前类了

this.handleClick = this.handleClick.bind(this);

}

handleClick() {

alert(this.state.message);

}

...

}

在 createReactClass() 中，并不需要这么做，因为方法可以自动绑定。

var SayHello = createReactClass({

getInitialState: function() {

return {message: 'Hello!'};

},

handleClick: function() {

alert(this.state.message);

},

render: function() {

return (

<button onClick={this.handleClick}>

Say hello

</button>

);

}

});