在常规的 React 数据流中，[props](http://www.css88.com/react/docs/components-and-props.html) 是父组件与子组件交互的唯一方式。要修改子元素，你需要用新的 props 去重新渲染子元素。然而，在少数情况下，你需要在常规数据流外强制修改子元素。如下：

**Ref**

### ref可以让我们获取到原始的dom元素

### 何时使用 Refs

### 处理focus、文本选择或者媒体播放

### 触发强制动画

### 集成第三方DOM库

### 不要过度使用 Refs

### 在 DOM 元素上添加 Ref

### React 支持给任何组件添加特殊属性。ref 属性接受回调函数，并且在组件 装载(mounted) 或者 卸载(unmounted) 之后，回调函数会立即执行。

class CustomTextInput extends React.Component {

constructor(props) {

super(props);

this.focus = this.focus.bind(this);

}

focus() {

// 通过使用原生API，显式地聚焦text输入框

this.textInput.focus();

}

render() {

// 在实例中通过使用`ref`回调函数来存储text输入框的DOM元素引用(例如:this.textInput)

return (

<div>

<input

type="text"

ref={(input) => { this.textInput = input; }} />

<input

type="button"

value="Focus the text input"

onClick={this.focus}

/>

</div>

);

}

}

### 为 类(Class) 组件添加 Ref

### 当 ref 属性用于类(class)声明的自定义组件时，ref 回调函数收到的参数是装载(mounted)的组件实例。

class CustomTextInput extends React.Component {

// ...

}

class AutoFocusTextInput extends React.Component {

componentDidMount() {

this.textInput.focus();

}

render() {

return (

<CustomTextInput

// this.textInput保存的是CustomTextInput的实例

ref={(input) => { this.textInput = input; }} />

);

}

}

需要注意的是，这种方法仅对以类(class)声明的 CustomTextInput 有效：

### Refs 与 函数式组件

### 你不能在函数式组件上使用 ref 属性，因为它们没有实例:

### 对父组件暴露 DOM 节点

### 在极少数情况下，你可能希望从父组件访问子节点的 DOM 节点。通常不建议这样做，因为它会破坏组件的封装，但偶尔也可用于触发焦点或测量子 DOM 节点的大小或位置。

在这种情况下，我们建议在子节点上暴露一个特殊的属性。子节点将会获得一个函数属性，并将其作为 ref 属性附加到 DOM 节点。这允许父代通过中间件将 ref 回调给子代的 DOM 节点。

function CustomTextInput(props) {

return (

<div>

<input ref={props.inputRef} />

</div>

);

}

class Parent extends React.Component {

render() {

return (

<CustomTextInput

inputRef={el => this.inputElement = el}

/>

);

}

}