Portals 提供了一种很好的方法，将子节点渲染到父组件 DOM 层次结构之外的 DOM 节点。

ReactDOM.createPortal(child, container)

第一个参数（child）是任何[可渲染的 React 子元素](http://www.css88.com/react/docs/react-component.html" \l "render)。第二个参数（container）则是一个 DOM 元素。

**用法**

当你从组件的 render 方法返回一个元素时，它将被装载到父节点 DOM 中：

render() {

// React 装载一个新的 div，并将 children 渲染到这个 div 中

return (

<div>

{this.props.children}

</div>

);

}

然而，有时候将子元素插入到 DOM 节点的其他位置会有用的：

render() {

// React 不会创建一个新的 div。 它把 children 渲染到 `domNode` 中。

// `domNode` 可以是任何有效的 DOM 节点，不管它在 DOM 中的位置。

return ReactDOM.createPortal(

this.props.children,

domNode,

);

}

**通过 Portals 进行事件冒泡**

一个从 portal 内部会触发的事件会一直冒泡至包含 React tree 的祖先。

意思是：

B组件作为A组件的子组件，虽然B没有直接挂在A的Dom下面，但B的事件冒泡还是会冒泡到A

假设如下 HTML 结构：

// These two containers are siblings in the DOM

const appRoot = document.getElementById('app-root');

const modalRoot = document.getElementById('modal-root');

class Modal extends React.Component {

constructor(props) {

super(props);

this.el = document.createElement('div');

}

componentDidMount() {

modalRoot.appendChild(this.el);

}

componentWillUnmount() {

modalRoot.removeChild(this.el);

}

render() {

return ReactDOM.createPortal(

this.props.children,

this.el,

);

}

}

class Parent extends React.Component {

constructor(props) {

super(props);

this.state = {clicks: 0};

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

}

handleClick() {

// This will fire when the button in Child is clicked,

// updating Parent's state, even though button

// is not direct descendant in the DOM.

this.setState(prevState => ({

clicks: prevState.clicks + 1

}));

}

render() {

return (

<div onClick={this.handleClick}>

<p>Number of clicks: {this.state.clicks}</p>

<p>

Open up the browser DevTools

to observe that the button

is not a child of the div

with the onClick handler.

</p>

<Modal>

<Child />

</Modal>

</div>

);

}

}

function Child() {

// The click event on this button will bubble up to parent,

// because there is no 'onClick' attribute defined

return (

<div className="modal">

<button>Click</button>

</div>

);

}

ReactDOM.render(<Parent />, appRoot);