React组件的创建方式有两种，第一是可以通过函数式申明（多用于创建无状态组件），第二是通过类的继承（多用于创建有状态组件）。

🡪

**React组件无状态函数式组件申明**：

React组件的申明可以采用 class 组件名 extends React.Component的类的申明写法，除此以外我们还可以使用无状态函数式组件方式。它本质上就是一个常规的函数，接收一个 props 参数并返回一个元素。返回元素的方式便是JS函数的return方式。

# 例如：

function Item(props) {//接收props参数

return (

<div className='item'> //返回一个组件

{props.title}

<span

className='deleteItem'

onClick={props.remove(props.id)}

> x </span>

</div>

)

}

同时使用ES6 的箭头函数和解构，我们也可以这样编写：

const Item = ({id, title, remove}) =>

<div className='item'>

{title}

<span

className='deleteItem'

onClick={remove(id)}

> x </span>

</div>

函数式组件，有时也被称为无状态组件，没有任何生命周期方法，意味着每次上层组件状态发生变更时它们都会重新渲染，这就是因为缺少 shouldComponentUpdate 方法导致的。这也同样意味着您不能定义某些基于组件挂载和卸载的行为。

在函数式组件中既不能使用 this 关键字，也没有this.props或this.states或访问到this.refs。

无状态函数式组件主要的设计概念是使得容器型组件 (container component) 和 展示型组件 (presentational component) 的概念就会变得非常清晰。通过将逻辑和数据处理与 UI 展示剥离开来，我们就可以避免在展示型组件中处理任何的状态。无状态函数式组件强制实施了这样的风格，因为您无论如何都没有办法处理本地状态了。它强制您将任何的状态处理移交至上层的组件树，而让下层的组件只做它们所做的——关注 UI 的展示。

P.s 无状态组件(stateless component)由于无自身状态，所以在给定props的前提下，生成的html元素都肯定是一样的。而相比之下，有状态组件(stateful component)在给定自身的props前提下，可以使用自己独立的state生成不同的html元素，所以生成的html元素可以不一样。

P.s函数式组件和有状态组件语法上的区别

函数式组件，使用return返回要生成的内容。有状态组件则是以render()方法中的return方式。

函数式组件由于没有this,使用props.xxx来使用props。有状态组件则是以this.props.xxx来使用props.

🡪

**React组件有状态类继承组件申明**：

class InputControlES6 extends React.Component {

constructor(props) {

super(props); // 设置 initial state

this.state = { text: props.initialValue || 'placeholder' };

}

render() { return ….}

}