**一、JSX中的事件绑定**

1. 普通函数（括号方式）

this：undefined 不能操作当前元素

render(){  
 return <button className='btn btn-success'  
 onClick={this.support}>  
 支持</button>;  
}  
  
support(ev){  
 //=>THIS:undefined（不是我们理解的当前操作的元素）  
 //=>ev.target：通过事件源可以获取当前操作的元素（一般很少操作，因为框架主要是数据驱动所有DOM的改变）  
}

2. 普通函数(bind)

让方法中的this变成当前类的实例，可以操作属性和状态：

onClick={this.support.bind(this)}

render(){  
 //=>THIS:实例  
 return <button className='btn btn-success'  
 onClick={this.support.bind(this)}>  
 支持</button>;  
}  
  
support(ev){  
 //=>THIS:实例  
}

3. 箭头函数（常用）

this:继承上下文中的this（实例），真实项目中，给JSX元素绑定的事件方法一般都是箭头函数，目的是为了保证函数中的this还是实例

render(){  
 //=>THIS:实例  
 return <button className='btn btn-success'  
 onClick={this.support}>  
 支持</button>;  
}  
  
support=ev=>{  
 //=>THIS:继承上下文中的THIS（实例）, 真实项目中，给JSX元素绑定的事件方法一般都是箭头函数，目的是为了保证函数中的THIS还是实例  
}

**二、声明周期函数（钩子 函数）**

描述一个组件或者程序从创建到销毁的过程，我们可以在过程中基于钩子函数完成一些自己的操作（例如：在第一次渲染完成做什么，或者在二次即将重新渲染之前做什么等）

【**基本流程**】

constructor 创建一个组件

componentWillMount 第一次渲染之前

render 第一次渲染

componentDidMount 第一次渲染之后

【**修改流程**：当组件的状态数据发生改变（set-state）或者传递给组件的属性发生改变（重新调用组件传递不同的属性）都会引发render重新执行（渲染也是差异渲染）】

shouldComponentUpdate是否允许组件重新渲染（允许则执行后面函数，不允许则直接结束就可以）

componentWillUpdata 重新渲染之前

render 第二次及 以后重新渲染

componentDidUpdate 重新渲染之后

componentWillReceiveProps：父组件把传递给子组件的属性发生改变后触发的钩子函数

【**卸载**：原有渲染的内容是不消失的，只不过以后不能基于数据改变视图了】

componentWillUnmount ：卸载组件之前（一般不用）