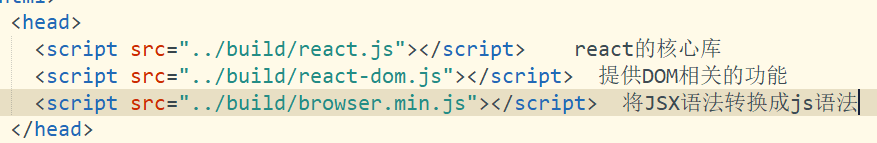
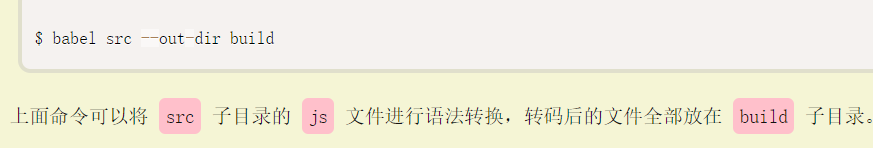
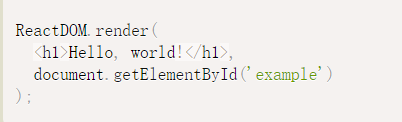
**一**：HTML 模板





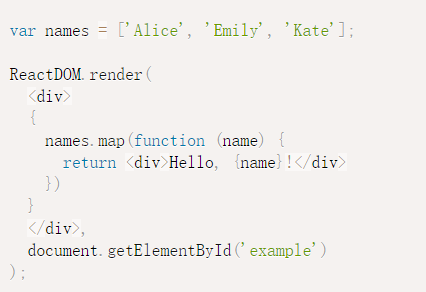
**二： ReactDOM.render() 此方法是将模板转换成HTML语言，并插入指定的DOM节点**



**此是把h1标题，插入到example节点。 Example是一个标签节点**

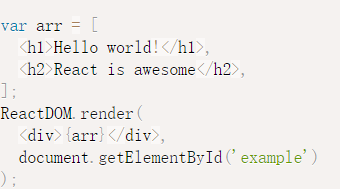
**三：JSX 语法**

**HTML语言直接写在Javascript语言中，不加任何引号，就是JSX语法，如上面的h1 标签**



**JSX的基本语法规则是：遇到HTML标签，（以<开头），就用HTML解析，遇到代码块（以{ 开头 ，就用javascript 解析**

**2 JSX也允许直接在模板插入js变量 ，如果是个数组，则会展开这个数组的所有成员**

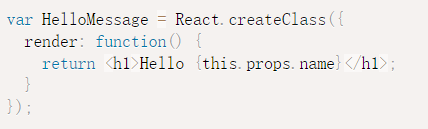


**结果就是展示了arr里的所有值**

**四：组件**

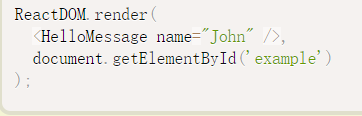
**React.createClass 方法就是用于生成一个组件类**

**例如：**



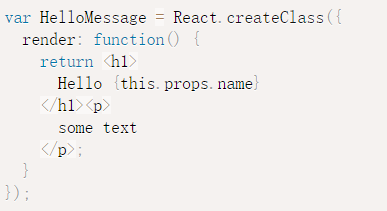
**注意点：HelloMessage 是一个变量，首字母一定要大写。第二个是，每个组件都要有一个render的方法，用于输出组件。**

**2**



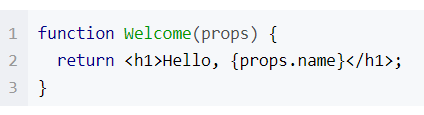
**模板插入<HelloMessage/>时，会自动生成HelloMessage的一个实例**

**3 组件类只能包含一个顶层标签，否则报错，错误示例如下**



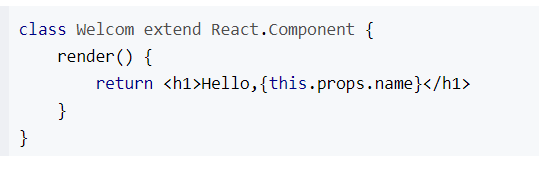
**定义组件的两个方式：**

**1 最简单的方式是创建一个javascript函数**



**2 Es6的class类**

**可以通过setState直接赋值**



**4 组件用法与原生的HTML标签一致，可以任意加入属性。**

**例如：**<HelloMessage name="John"> ，就是 HelloMessage 组件加入一个 name 属性，值为 John。

**组件的属性，可以在组件类的对象上获取，用this.props。如上面的Name 属性，就可以通过this.props.name读取，**

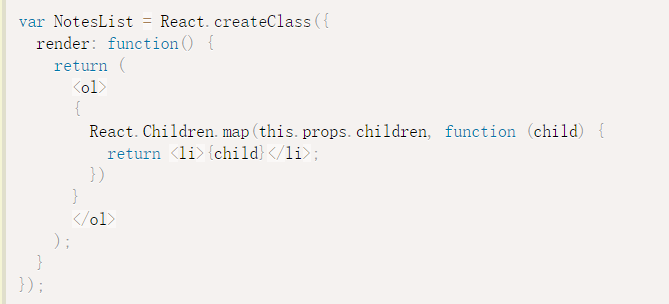
**特别注意： 如果是属性class，要写成className, for属性需要写成htmlFor ,因为class和for是javascript的保留字**

**五： this.props.children 组件的所有的子节点**

**React提供的工具方法 react.Children 来处理this.props.children。**

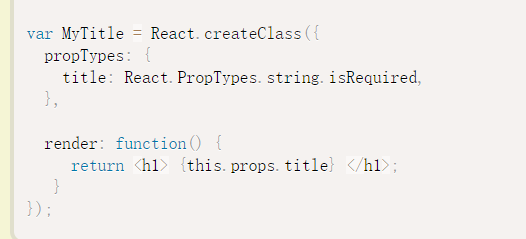
**用React.Children.map 来遍历子节点**

**例：**



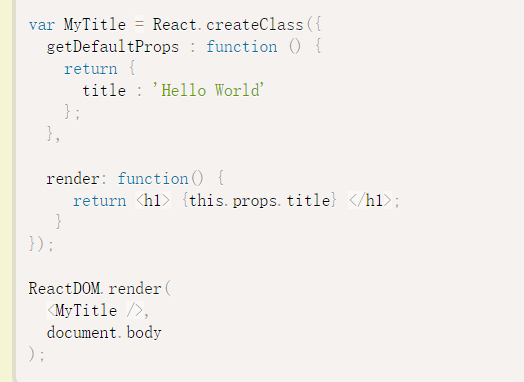
**六： PropTypes 组件属性**

**用来验证组件实例的属性是否符合要求**



**此是**title的属性是字符串必填

**GetDefaultProps 方法用来设置组件属性的默认值**



**此的默认值是：Hello World;**

**七：获取真实的DOM节点**

**获取组件的真实DOM节点，用ref属性**

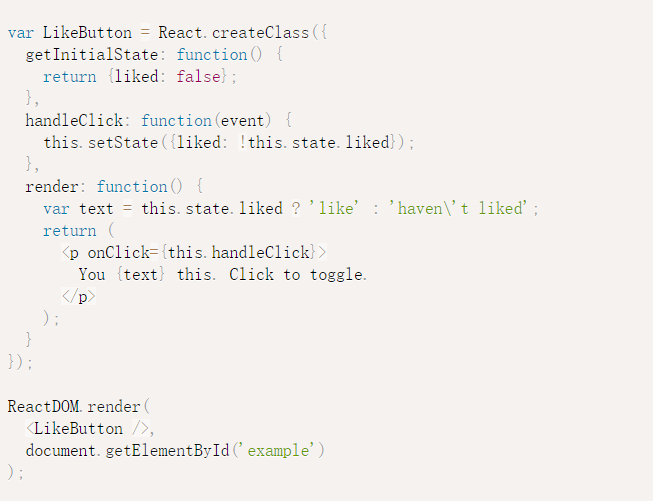


**此属性和vue的ref属性使用一样**

**用法： this.refs.[fefName];**

**因为this.refs.[fefName]属性获取的是真实的DOM ,所以必须等到虚拟DOM插入文档后才能使用此属性，否则报错。**

**八： this.state 状态**



**this.state 对属性进行读取**

**getInitialState 方法，用于定义初始状态**

**this.setState 方法修改状态值。每次修改后自动调用this.render方法，再次渲染组件**

**注意： this.props和this.state都用来描述组件特点，this,props表示那些一旦定义就不再改变的特性，this.state是随用户互动而改变。**

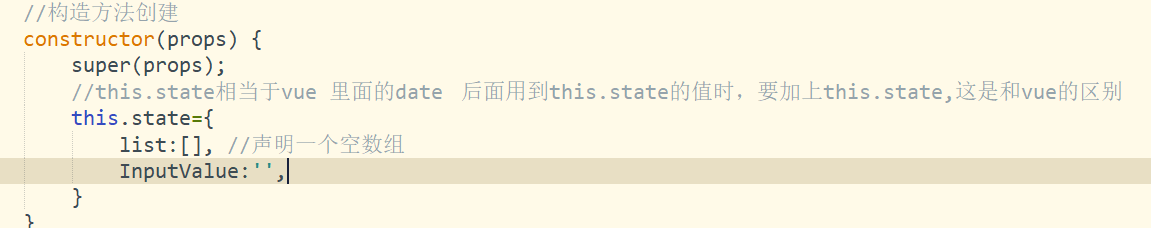
**九： 表单**

**注意：** new Date().toLocaleTimeString() 是把时间转换成字符串形式，HH:MM:SS

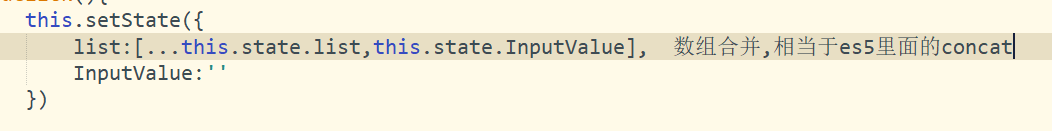
**做TodoList的一些问题**

1. **在事件上用 onClick={this.事件名.bind(this)}**

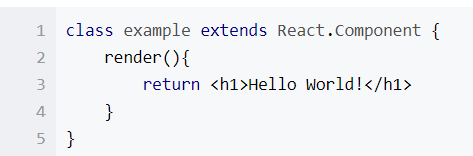
**2 每个事件都需要在this.setState({ }) 里面去做**



**3**

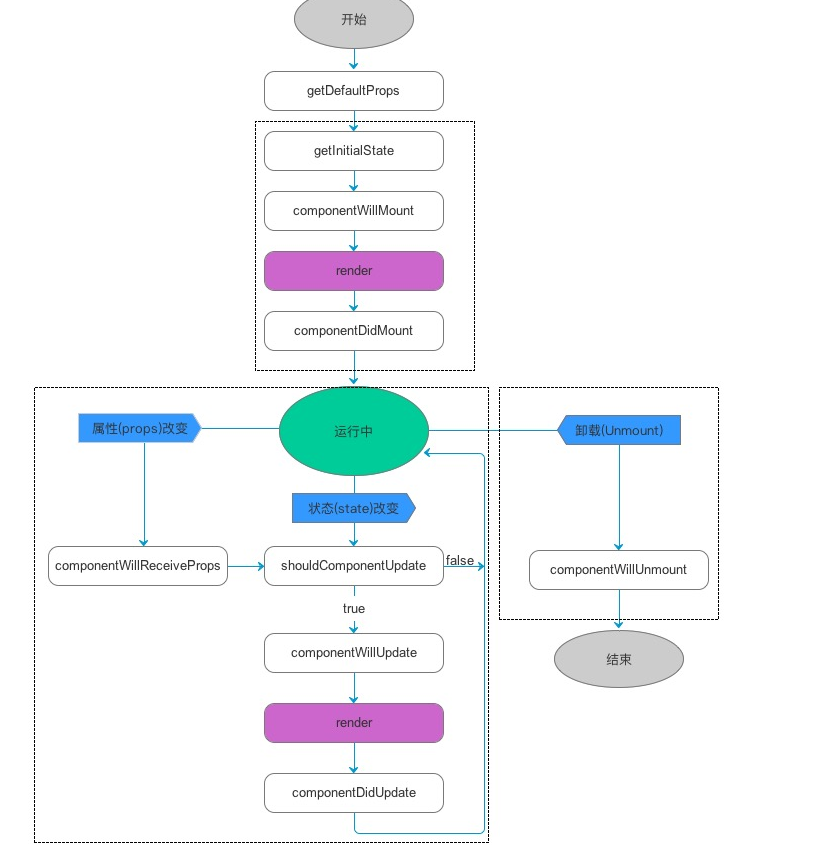


**React.component**



**生命周期方法**





**class example extends React.Component{**

**一：这是装配 1 2 3**

**1 constructor(props){ //这是生命周期的第一个方法**

**super(props) //在其他一切之前调用**

**this.state={ } 要声明的东西在此声明，相当于vue 里的data**

**}**

**2 componentWillMount(){**

**此方法是在装配前被调用，在render()之前，在这设置的状态，同步的不会重染。（在服务端渲染调起的生命周期钩子函数，一般用constructor代替）**

**}**

1. **componentDidMount(){**



**在组件被装配后立即调用，若需要远端加载数据，适合实现网络请求，在该方法里设置状态会重新渲染。**

**}**

**二：更新**

**1 componentWillReceiveProps(){ 在装配了的组件接收到新的属性前调用。**

**this.props //当前的props**

**} Renser(){ } }**

**注意：它只会在组件props更新时才调动componentWillReceiveProps,**

**调用this.setState通常不会触发。**

1. **shouldComponentUpdate()**



**使用shouldComponentUpdate()让React知道当前状态或属性的改变是否不影响组件的输出。**

当前，若shouldComponentUpdate()返回false，而后componentWillUpdate()，render()， 和 componentDidUpdate()将不会被调用。

1. **componentWillUpdate()**



**当接受到新属性和状态时，componentWillUpdate()在渲染前被立即调用。该方法不会在初始化渲染时调用。**

**注意：不能在这调用this.setState() ,若要更新状态响应属性的调整，使用componentWillReceiveProps()代替**

1. **componentDidUpdate()**

**会在更新后立即被调用，不会在初始化渲染式时调用。**

**也是适合发请求的地方。**

**三：卸载**

**componentWillUnmount()**

**componentWillUnmount() 在组件被卸载和销毁之前调用，例如解绑定时器，取消网络请求。清理任何在componentDidMount环节创建的DOM元素。**