**1模块化-基于commonjs规范**

顶部声明依赖，底部导出内容，把组件逻辑放在中间。

**导入模块方式： var React=require(react’)**

**声明组件方式：var Logo=React.createClass({\*…..\*})**

**导出模块方式： module.exports=Logo**

**2基于ECMAscript模块**

**导入模块方式： import React from ‘react’**

**声明组件方式：var Logo=React.createClass({\*…..\*})**

**导出模块方式： export default Logo**

**.基于ECMAscript的类组件编写**

**修改前：var Logo=React.createClass({\*…..\*})**

**修改后：class Logo extends React.Component{/\*……..\*/}**

**几个区别，方法之间不再需要逗号隔开，方法通过render（）{}方式定义，不需要像之前一样render:function(){return}，类中无属性只有方法，需要属性可以由this.nam之类来定义。**

**另外关于commonjs规范的模块化和ECMAscript模块化的区别**

es6 {

　　export   :      '可以输出多个，输出方式为 {}' ，

　　export  default : ' 只能输出一个 ，可以与export 同时输出，但是不建议这么做'，

　　解析阶段确定对外输出的接口，解析阶段生成接口，

　　模块不是对象，加载的不是对象，

　　可以单独加载其中的某个接口（方法），

　　静态分析，动态引用，输出的是值的引用，值改变，引用也改变，即原来模块中的值改变则该加载的值也改变，

　　this 指向undefined

}

commonJS {

　　module.exports =  ...   :      '只能输出一个，且后面的会覆盖上面的' ，

　　exports. ...  : ' 可以输出多个'，

　　运行阶段确定接口，运行时才会加载模块，

　　模块是对象，加载的是该对象，

　　加载的是整个模块，即将所有的接口全部加载进来，

　　输出是值的拷贝，即原来模块中的值改变不会影响已经加载的该值，

　　this 指向当前模块

}