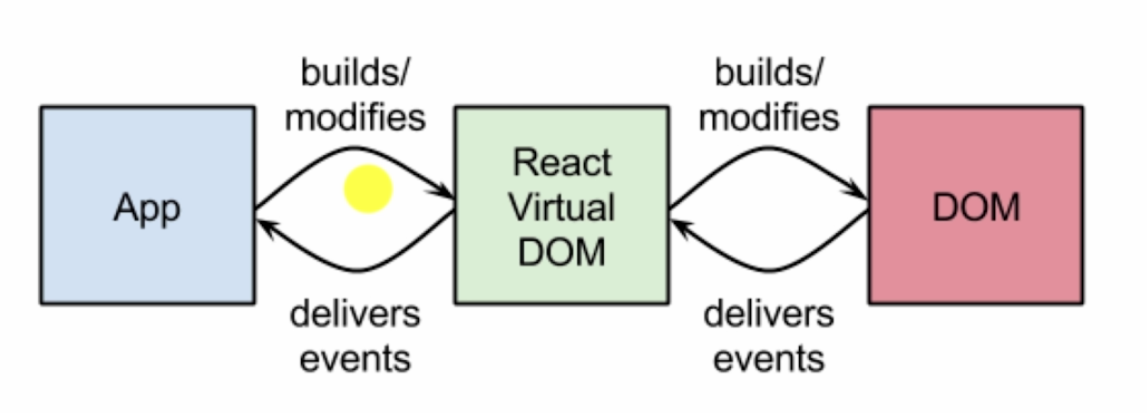
**1.React的虚拟DOM的结构**



优点：在数据结构层面降低时间复杂度，

2.React组件

组件定义：

export default class 组件名 extends React.Compont{

render(){

return()

}

}

注：export default 是将该组件导出外部，这样才能使别的文件引用并使用。

return函数中只能返回一个HTML节点。

命名规范，组件名类似于类名

入口定义：

ReactDOM.render(<组件名/>,需要渲染的页面的id);

多组件嵌套：

可以使用div将多个节点包裹起来返回

组件参数化：

声明：var 参数名 = <组件名/>

使用：{参数名}

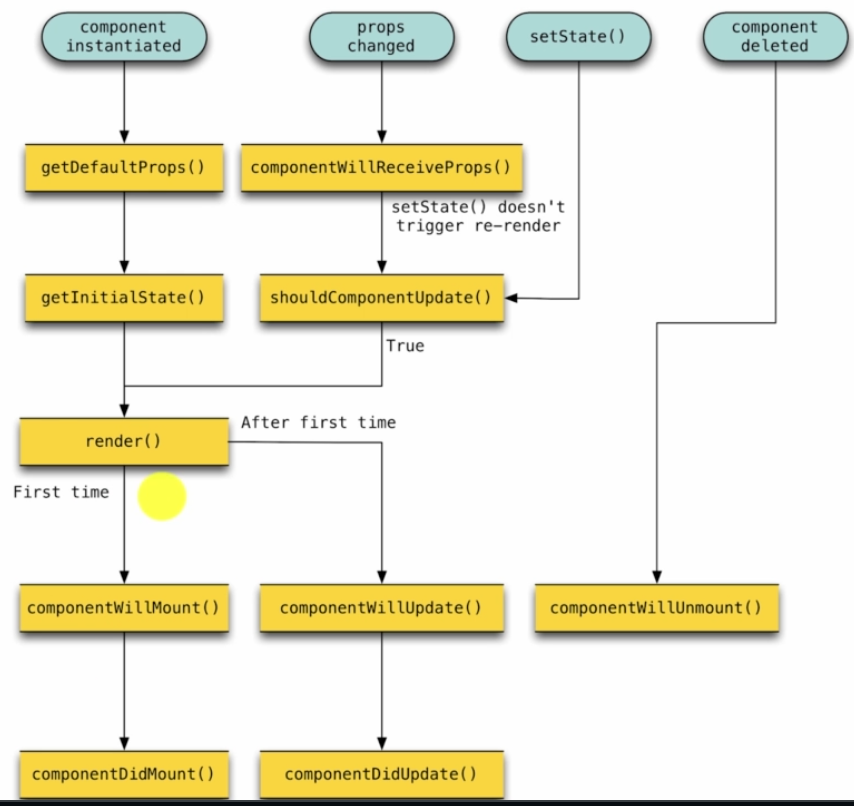
3.JSX内置表达式

表达式要写在{}里

注释：{/\*\*/}

HTML可以使用unicode转码

4.生命周期（见下页图）



5.React的state属性

this.state

state属性的作用域只属于当前类，不污染其他模块。

6.React的props属性

state和props同属于component。

state属于当前类自身属性，props属于当前类外来属性。

赋值时使用{}

7.事件与数据的双向绑定

当前页面参数绑定：

this.setState();

调用方法：this.方法名/this.方法名.bing(this)【不同的方式是因为解析的文件类型不同(es5/es6)】

父子页面参数绑定（子🡪父）：

在子页面调用父页面的事件传递【该事件必须从父页面传递给子页面，子页面才能调用。】

this.方法名.bing(this,参数)：

参数可以随意设置，可以为该方法提供入参。

**8.可复用组件**

**页面间传递参数是对参数进行校验(简单校验)：**

**组件.propTypes = {**

**参数名：React.PropTypes.类型/ React.PropTypes.类型.属性【前者只是限制该参数的类型，后者是限定该参数的性质(是否必输等)】**

**}；**

**组件.defaultProps = 默认常量；【该常量必须在该组件开头声明，表示其他页面传递参数时，给非必传的参数设置默认值】**

**父页面向孙页面及以下页面快速传参的方法：**

**子页面使用孙组件时，使用{...this.props}可以将子页面从父页面获取的参数全部传给孙页面，以此类推。**

**9.组件的Refs**

**第一种方式（原生js【不推荐】）：**

**let 节点名 = document.findElementById(‘’);**

**ReactDOM.findDOMNode(节点名).属性= ...;**

**eg：ReactDOM.findDOMNode(节点名).style.color= ‘red’;**

**第二种方式：**

**给组件设置ref属性，this.refs.组件的ref名。**

**操作与第一种方式相同。**

**注意：**

**Refs是访问到组件内部DOM节点唯一可靠的方法。**

**Refs会自动销毁对子组件的引用。**

**不要在render或render之前对Refs进行调用，要在事件内使用。**

**不要滥用Refs。**

**10.独立组件间共享Mixins**

**需要安装react-mixins插件。**

**在不同的组件间公用功能，共享代码。**

**和页面具有类似的声明周期。**

**11.React内联样式**

**第一种添加样式方式：**

**一般以常量形式声明：**

**const 样式名 = {**

**样式1：{**

**键：值，**

**eg：**

**backgroundColor： “red”，【第一种】**

**“padding-top”：“15px”【第二种】**

**}，**

**样式2：{**

**...**

**}，...**

**}；**

**使用：**

**<标签 style={样式名.样式1...}/>**

**第二种添加样式方式：**

**创建css公共样式文件，在html中引入。使用时，为标签设置className的值。**

**内联样式的局限：动画，伪类(hover)等不能使用。**

**内联样式可以使用表达式使用 样式：(this.state.xxx)？xx：xx【第一种内联样式】**

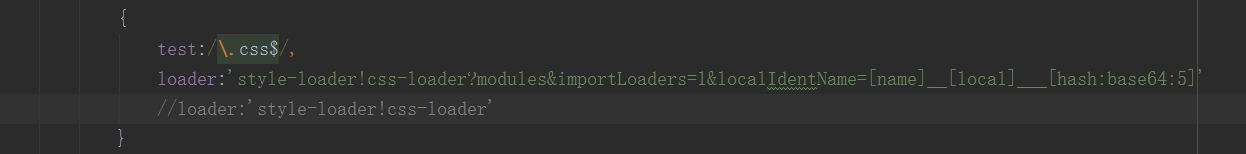
**12.css模块化**

**为什么要模块化？**

**全局污染、命名混乱、依赖管理不彻底、无法共享变量、代码压缩不彻底。**

**模块化**

**wenpack.config.js文件中对css文件处理**



**模块化声明：**

**单独组件可以单独创建该组件的css文件。**

**模块化使用：**

**var 样式名 = require(样式文件路径)；**

**<标签 className={样式名}/>**

**模块化的优点：**

**1.所有的样式都是文件本身的，解决了命名冲突和全局污染**

**2.class名生成规则配置灵活，可以此来压缩class名**

**3.只需引用组件的js就可以搞定所有组件的js和css**

**4.依然是css，不用额外学习**

**13.jsx样式与css互转**

**工具【CSS TO REACT】：**

**可以将css样式快速转换成react形式的样式。**

**14.ant-design框架**

**安装：**

**cnpm install antd –save**

**使用：**

**import【组件名】from ‘antd’；**

**15-0．react-router2升级react-router4**

**1.所有的组件都在react-router-dom里面**

**import{Router，Route，hashHistory}from‘react-router’；**

**改为：**

**import{BrowserRouter，Route，Switch}from‘react-router-dom’；**

**2.Router只能有一个子节点，所以可以使用<Switch/>标签包裹【<Switch/>也在dom里】**

**3.**

**15.react-router路由**

**<Router/>属性exact：使路由的匹配更为严格，默认为true表示严格匹配。**

**Router4的嵌套路由：**