**JS基础**

1. 闭包 <https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Closures>

闭包是由函数以及创建该函数的词法环境组合而成。这个环境包含了这个闭包创建时所能访问的所有局部变量。

1. React组件
2. 组件名称必须以大写字母开头（小写字母开头会被认为是html的原生标签）；
3. 组件在使用时必须定义或引入；
4. 组件的返回值只能有一个根元素
5. Props属性只能读
6. React更新UI
7. 调用 ReactDOM.render()
8. 调用setState() 来更新组件局部状态
9. React中setState()
10. 不能直接更新状态，需要调用setState()更新状态

this.state.comment=”Hello”// 不会重新渲染组件

this.setState({comment: ‘Hello’})；// 会重新渲染组件

1. 状态更新可能是异步的

React 可以将多个setState() 调用合并成一个调用来提高性能

通过setState(Callback())可以解决同步问题

1. 状态更新合并，React 将你提供的对象合并到当前状态。
2. 数据自顶向下流动（即单向数据流），任何状态始终由某些特定组件所有，并且从该状态导出的任何数据或 UI 只能影响树中下方的组件。
3. React中key
4. 元素的key只有在它和它的兄弟节点对比时才有意义
5. key会作为给React的提示，但不会传递给你的组件。
6. Promise：ES6 原生提供的一个对象  
   1） 对象的状态不受外界影响  
   2）一旦状态改变，就不会再变，任何时候都可以得到这个结果。  
   3）缺点：无法取消 Promise；内部抛出的错误，不会反应到外部；Pending 状态时，无法得知目前进展到哪一个阶段。  
   new Promise(function(resolve, reject) {  
    if (/\* 异步操作成功 \*/){  
    resolve(value);  
    } else {  
    reject(error);  
    }  
   }).then(function(value) {  
    // success  
   }, function(value) {  
    // failure  
   }).catch(function(err){  
    console.log(err);  
    });
7. JavaScript代码必须 通过HTML/XHTML文档才能执行。有两种实现方式：  
   1）将JavaScript代码放到<head>标签中的<script>标签之间；  
   2）将JavaScript代码扩展为.js文件，在<head>中<script>标签的src属性指向该文件。  
   注：最好的做法是将<script>放到HTML文档的最后，</body>标签之前。
8. JavaScript是一种弱类型的语言，可以在任何阶段改变变量的数据类型。例如：  
   var age = “33”;  
   age = 33;
9. 字符串必须包含在引号里，可以使单引号或双引号。字符串里包含双引号，就把整个字符串放在单引号里；字符串里包含单引号，就把整个字符串放在双引号里。（也可以使用转译字符）
10. 数组声明：var names = Array();或var names = Array(6);
11. 一个数组中可以混合多种数据类型。
12. if(a = ‘0’)，a = ‘0’是赋值运算，赋值运算总是返回真，除了a = false和a = ‘’
13. ==操作符并不表示严格相等，例如空字符和false比较（ “” == false）返回true
14. 不严格相等==，严格相等===
15. DOM三种获取元素节点的方法：通过ID获取、通过标签名获取和通过类名获取
16. JavaScript语言区分字母大小写
17. 为标签指定多个类名：在标签类名字符串参数中有空格分隔各个类名。
18. 节点类型分为：元素节点、属性节点和文本节点等。
19. DOM的工作模式（“表里不一”）：先加载文档的静态内容，再动态刷新，且动态刷新不影响文档的静态内容。
20. childNodes属性获取该元素的全部子元素，返回该元素的全部子元素的数组。
21. function addLoadEvent (func) {  
     var oldOnLoad = window.onload;  
     if(typeof window.onload != ‘function’) {  
     window.onload = func;  
     } else {  
     window.onload = function() {  
     oldOnLoad();  
     func();  
     }  
     }  
    }
22. nodeName属性总是返回一个大写字母的值。
23. js设置垂直居中需同时设置line-height与height一致并且vertical-align:middle
24. DOM.innerHtml与document.write()的区别：
    1. write是DOM方法,向文档写入HTML表达式或JS代码，可列出多个参数，参数被顺序添加到文档中 ；innerHTML是DOM属性,设置或返回调用元素开始结束标签之间的HTML元素。
    2. document.write会重写整个页面，innerHTML返回或替换DOM树中调用元素原先的所有子节点。
    3. innerHTML能够实现html和js的分离，document.write则不能。document.write放在window.onload 中执行时，会将页面上的所有内容清除包括标题；innerHtml则只会更新调用元素原先的所有子节点。
25. ajax对页面的请求以异步方式发送到服务器，实现页面的部分更新。ajax的核心技术是XMLHttpRequest对象。
26. XMLHttpRequest的open(请求方法，请求问价，请求参数)方法
27. XMLHttpRequest的onreadystatechange是一个事件处理函数，在XMLHttpRequest对象获取到服务器响应的时候被触发。
28. 使用ajax时，注意同源策略。
29. getElementById：获取指定ID的元素

getElementsByTagName：获取指定的标签的全部元素

getAttribute：获取元素的相关属性

setAttribute：设置元素的相关属性

createElement：创建元素节点

createTextNode：创建文本节点

appendChild：在指定父节点元素上添加子元素

insertBefore：在指定元素之前插入一个元素节点

1. 网页由三层信息构成：结构层（(Html/XHtml）、表示层（css）)和行为层（js/DOM）。
2. DOM style属性只能获取内嵌样式，不能检索外部CSS文件
3. nodeType：1为元素节点，2为属性节点
4. 响应事件时修改元素的样式：
5. 在CSS文件中加一个新类的样式申明
6. 响应事件中，通过DOM改变元素的class属性或className属性。元素原来没有class属性时直接赋值，有class属性时在原来的class属性上追加新样式。
7. offsetParent：定位父级，与当前元素最近的经过定位(position不等于static)的父级元素
8. position属性：
   1. static：position的默认值，元素按顺序出现在浏览器窗口里。
   2. relative：与static相似，区别在于relative可以通过float属性将元素从文档正常顺序中摆脱出来。
   3. absolute：将元素摆放在“容器”的任何位置。这个“容器”可以是文档，也可以是有着fixed、absolute和relative属性的元素。
   4. fixed：相对浏览器窗口的绝对定位
9. overflow属性：
   1. visible：不裁剪溢出内容，溢出内容呈现在容器元素的显示区域以外。
   2. hidden：隐藏溢出内容，只显示容器元素显示区域内的内容。
   3. scroller：隐藏溢出的内容，但显示一个滚动条，可以滚动看到隐藏内容。
   4. auto：类似于scroller，区别auto时浏览器只在确实发生溢出时显示滚动条。
10. 网页分为：结构层（html）、样式层（css）和行为层（JavaScript）