# JavaScript-DOM+BOM

DOM一般属于内核，BOM一般属于内核外

DOM

概念：

Document Object Model文档对象模型

获取或设置文档中标签的属性

标签\标记vs元素

元素：浏览器预生成的对象

按照html文档中的标记嵌套关系和属性/值的对应关系，浏览器预生成一套对象，对象满足嵌套关系，可以由前端开发工程师直接使用，这些对象就是DOM

一个html网页文件就是一个document

每个标签对应一个element

html element是root element;任何document有且只有一个root element

从root element开始， 会形成逐级嵌套的元素，形成一个树，这个树叫做DOM tree

DOM能做的三件事

1. 操作元素
2. 控制元素样式
3. 事件

元素的交互响应

DOM中使用JS的正确姿势

1. JS直接写在HTML中（不可取）

<input type="button" id="btn" value=" 按钮”onclick="alert( 'Hello')" />

1. JS与HTML分离
2. 把JS以及所有的交互响应从HTML中剥离（更多）

[JS-DOM+BOM\书写方式.html](file:///C:\Users\彭彭彭\AppData\Roaming\Microsoft\Word\JS-DOM+BOM\书写方式.html)

javascript获取节点的方法

doucument.getElementById()

doucument.getElementsByTagName() (伪数组)

doucument.getElementsByName() (伪数组)

doucument.getElementByClassName() (伪数组)

伪数组不能使用for in 可以使用forEach()和for()

在JS中使用选择器

选择出符 合选择器要求的第一个元素

document.querySelector()

选择出所有符合选择器的元素(伪数组)

document.querySelectorAll()

在JS中不控制样式，样式全部在CSS中控制，JS中可以控制类名

常用技巧：

让text无效 disabled=true

让a标签跳转失效 return false

radio checkbox选中 checked=true/false

自定义属性

data-\* dataset

事件

onclick 点击

onfocus 在没获得焦点的时候点击获得焦点

onblur 在获得焦点的时候点击失去焦点(主要是文本框)

onmouseover 鼠标移入

onmouseout 鼠标移出

onmousedown 属性在鼠标按钮在元素上按下时触发

onmousemove 事件会在鼠标指针移动时发生

onmouseup 释放鼠标左键

innerText、innerHTML与textContent区别

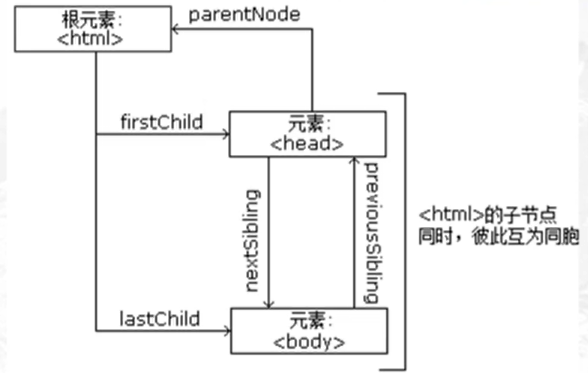
innerText:设置文本内容，兼容Chrome, FireFox, IE8

textContent:设置文本内容，Chrome,FF支持,IE8不支持

innerHTML:设置的是HTML原始字符串

a1.innerHTML=’<span>哈哈</span>’; 加入的是’哈哈’

a1.innerText=’<span>哈哈</span>’; 加入的是’<span>哈哈</span>’



HTML基础

节点(Node)

HTML元素( Element)

HTML属性(Attribute )

标题

节点

DOM元素节点树

* + - 父(parent) 子(child) 和同胞( sibling)等术语用于描述这些关系。父节点拥有子节点。同级的子节点被称为同胞( 兄弟或姐妹)
    - 在节点树中，顶端节点被称为根(root)
    - 每个节点都有父节点、除了根(它没有父节点)
    - 一个节点可拥有任意数量的子节点
    - 同胞是拥有相同父节点的节点

DOM-Node



定义：

nodeType

nodeName

nodeValue

childNodes（类型：NodeList，里面有所有的Node）

children（类型：HTMLCollection，只有元素）

parentNode

parentElement（原来IE用的，为了保证兼容性在Chrome也可以用）

previousSibling

Chrome和FireFox上前一个表示Node

IE8上表示前一个Element

previousElementSibling

Chrome和FireFox上表示前一个Element

IE8上不支持

nextSibling

Chrome和FireFox上后一个表示Node

IE8上表示后一个Element

nextElementSibling

Chrome和FireFox上表示后一个Element

IE8上不支持

firstChild

Chrome和FireFox上表示第一个Node

IE8上表示第一个Element

firstElementChild

Chrome和FireFox上表示第一个Element

IE8上不支持

lastChild

Chrome和FireFox上表示最后一个Node

IE8上表示最后一个Element

lastElementChild

Chrome和FireFox上表示最后一个Element

IE8上不支持

IE8和Chrome的兼容性问题

IE8不支持Node

对于Element

var firstChild = div.firstElementChild || div.firstChild;

var lastChild = div.lastElementChild || div.lastChild;

var previousSibling = ul.previousElementSibling || ul.previousSibling;

var nextSibling = ul.nextElementSibling || ul.nextSibling;

创建节点

1. document. write( )

2.对象.innerHTML

3. document.createElement()

document.createAttribute()

setAttributeNode()

setAttribute(‘type’, ‘button’)

getAttribute()

我们更多的时候使用第二种方法,因为第二种 方法更加简单。

第三种方法能够帮我们产生一个暂时不挂在DOM tree.上的元素.

Element.attributes 属性返回一个类似数组的动态对象，成员是该元素标签的所有属性节点对象

属性操作两种方式:

1. obj.xxx 俗称点操作，比如: obj.className = 'red';

2. obj.setAttribute(name,value) 比如: obj.setAttribute('class','red');

区别在于:

. 操作不能操作自定义属性,attribute可以

删除/改动节点

remove()

removeChild()

replaceChild(new,old)

事件

如果某件事情发生在了某个元素身上以后，允许JS引擎自动调用的回调函数(Callback)

事件绑定的三种方法

1. onType

两个特点:

1,兼容性非常好，相当于直接写DOM上的Attribute

2,如果调用两次，后面的回调会覆盖前面的回调

2.addEventListener(‘type’,function)

两个特点:

1.如果调用两次，都有效

2.IE8及以下不支持

3. attachEvent( 'onType', function)

特点:

1. Chrome不支持

2.在这个会调用中,this -> window

3.只有IE的attachEvent, this是window,其他的都是被触发的元素

事件的解绑定

1. onType = null
2. removeEventListener('type',function)

1. removeEventListener 只能删除addEvent Listener加上去的

2.删除时,必须有函数;如果你不知道函数,JS途径没有办法查询,更无法删除

3.作为程序员,怎么能够知道某个元素上挂了哪些事件?

Chrome上面的Event Listeners查看

1. detachEvent('ontype',function)

[改变attachEvent的this指向，导致无法detachEvent的解决方法。](http://itakeo.com/blog/2015/10/15/bindbug/)

事件的冒泡和捕获

事件的冒泡:

如果一个元素触发事件，那么其所有祖先元素都会依次触发该事件，这种机制称为冒泡

addEventListener有三个参数，第三个参数是表明是否是捕获，如果第三个参数不写，那么事件触发的传递方式就是冒泡，即从底向上

addEventListener(type,func,false)

事件的捕获（只有Chrome完全支持IE不支持）

顺序与事件的冒泡正好相反

addEventListener(type,func,true)

捕获和冒泡的执行顺序:

1.祖先节点开始， 按照从高到低,触发捕获

2.到节点本身,按照绑定顺序，触发冒泡+捕获

3. .再到祖先节点，按照从低到高，触发冒泡

冒泡的取消：事件的触发，到了某个元素，停止了

Chrome:e.stopPropagation

IE:e.cancelBubble=true

事件的起源：Chrome:e.target

IE:e.srcElement(Chrome和Firfox也支持)

阻止默认事件行为：

1. return false
2. （1）Chrome:e.preventDefault()

（2）IE: e.returnValue = false;

事件的委托：

触发事件的起源target;利用冒泡+事件的起源，避免在非常大数量的元素上绑定事件，而只在少量父元素上绑定事件。

在使用事件的委托时，下面任何一个层次的点击都有可能会触发它，需要对点击哪些层次需要做改变有一个清醒的认识

BOM

概念：Browser Object Model(全局变量Window)

BOM就是介绍window上的一些预制对象以及方法,主要处理浏览器窗口和框架

对话框：

1. alert()
2. prompt() 输入框
3. confirm() 判断框

加载事件

1. window.onload
2. DOMContentLoaded

两者相同点：存在时DOM已经建立了

不同点：

Src：CSS外部文件,img,他们的加载是异步的,不会阻塞DOM的parse过程

1. HTML文件下载完了，DOM parse完了

2.所有的CSS外部文件,img下载完了

通常来说，都是1先完,2后完

window.onload,在2完成后触发

DOMContentLoaded, 在1完成触发，速度更快

location对象:

location.reload() 强制刷新页面

通过JS进行页面跳转

location.href

改变url地址。要这样写：location.href="url"

location.replace(url)

将地址替换成新url，该方法通过指定URL替换当前缓存在历史里（客户端）的项目。与以上两者的区别在于：在replace之后，浏览历史就被清空了（href与assign方法会产生历史记录）。因此若使用replace页面跳转后是不能后退的。

location.assign(url)

加载 URL 指定的新的HTML文档，效果与location.href相当。

history对象

history.forward() 前进

history.back() 后退

navigator

navigator.userAgent 查看浏览器版本

动画

原生的简单的定时器动画

1.运动的对象

2.起始状态和终止状态

* + - * + 对象本身
        + 样式
        + 样式的值
  1. 时间跨度、时间间隔（至少20ms）
  2. 设定定时器
  3. 定时器停止

InnerWidth innerHeight

返回窗口的文档显示区的宽度、高度（不包括浏览器控制台、菜单栏、工具栏）

offsetLeft offsetRight 该对象相对于最近的有定位的父节点左上点距离

offsetWidth offsetHeight = padding + width/height + bordor-width

scrollLeft scrollTop 内容节点对于可视区域左上点距离

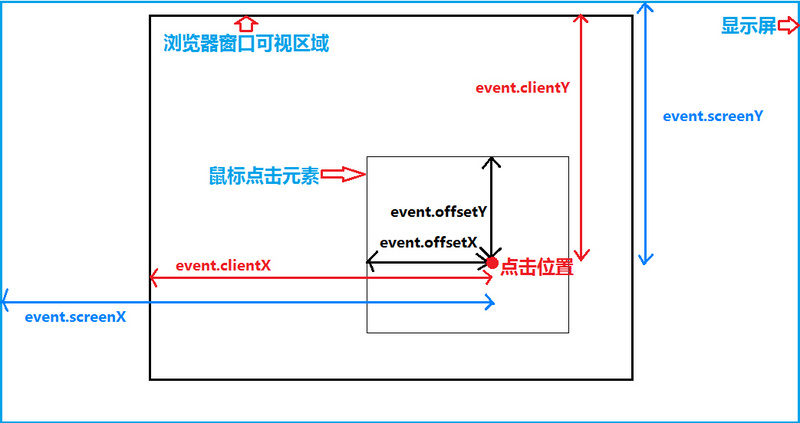
scrollWidth scrollHeight

div1.onscroll:在div1中有overflow:scroll;时，表示滚动条滚动事件

pageXOffset pageYOffset

clientLeft clientTop 可视区域的边框内角点对外角点距离

clientWidth clientHeight 可视区域的宽高



改变元素的类

Element.setAttribute(‘class’, ‘’)

Element.className = ‘’

自定义属性[data-\*](https://www.cnblogs.com/yaobolove/p/7367408.html)

变速动画1，分析

