# 一 DOM

### 3.3 获取前一个节点

节点.previousSibling

IE678中指前一个元素节点（标签）。

在标准浏览器指的是前一个节点（包括空文档和换行节点）

节点.previousElementSibling

在火狐谷歌IE9都指的是前一个元素节点。

总结：在IE678中用previousSibling，标准浏览器用previousElementSibling。

### 3.4 获取第一个子节点

父节点.firstChild

IE678中指第一个子元素节点（标签）。

在标准浏览器中指第一个节点（包括空文档和换行节点）。

父节点.firstElementChild

在火狐谷歌IE9都指的第一个元素节点。

同理用法还有获取最后一个子节点的方法：

lastChild lastElementChild

### 3.5 获取所有子节点

子节点数组 = 父节点.childNodes; 获取所有节点。

childNodes是标准属性，返回包括HTML节点，所 有属性，文本节点。

注意：火狐 谷歌等高本版会把换行也看做是子节点

nodeType == 1 表示元素节点(标签)

nodeType == 2 表示属性节点

nodeType == 3 表示文本节点

子节点数组 = 父节点.children; 用的最多。

children是非标准属性，只返回HTML节点，不返回文本节点，

注意：虽然不是标准的DOM属性，但它和innerHTML方法一样，得到了几乎所有浏览器的支持。children在IE6/7/8中包含注释节点。

### 3.6 通过索引获取兄弟节点

节点自己.parentNode.children[index];随意得到兄弟节点。

## 4 节点属性

获取：getAttribute(名称)

设置：setAttribute(名称, 值)

删除：removeAttribute(名称)

注意：IE6、7不支持。

## 5 节点使用

### 5.1 创建节点

新的标签（节点） = document.createElement(“标签名”);

### 5.2 插入节点

父节点.appendChild(新节点); 父节点的最后插入一个新节点

父节点.insertBefore(新节点,参考节点)在参考节点前插入;

如果参考节点为null，那么他将在节点最后插入一个节点。

### 5.3 删除节点

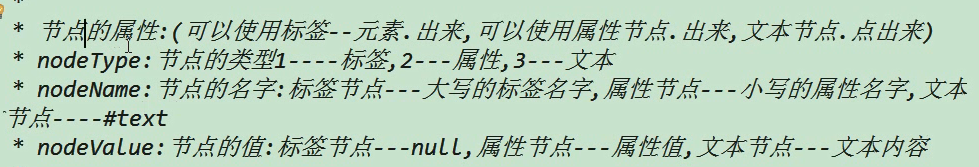
父节点.removeChild（子节点）；必须制定要删除的子节点

不知道父级的情况下，可以这么写：节点.parentNode.removeChild(node)

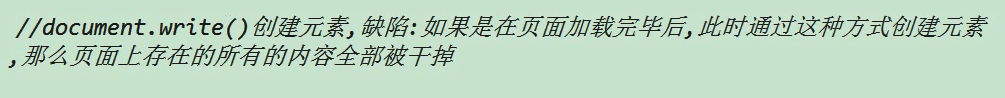
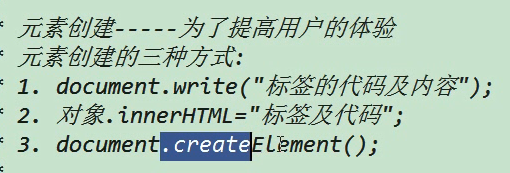
### 5.4 复制节点

新节点=要复制的节点.cloneNode(参数) ; 参数可选复制节点

## 6 节点类型



## 7 创建标签、节点的三种方式



# 二 事件

## 1 JS与常见事件

JS是以事件驱动为核心的一门语言。

事件的三要素是：事件源、事件、事件驱动程序。

三句话：获取事件源、绑定事件、书写事件驱动程序。

获取事件源：document.getElementById(“box”);

绑定事件： box.onclick = function(){ 程序 };

书写事件驱动程序：以后要学习的关于DOM的操作。

常见事件：



焦点可以使浏览器区分用户输入的对象。但是并不是所有的元素（div）都能接收焦点，能够响应用户操作的元素才有焦点(button)。

## 2 事件对象

当一个事件发生时，事件的详细信息都会保存到一个对象中，这个事件对象即event对象。

ie：内置了event对象，默认值是null

chrome：内置了全局event对象，默认值是undefined

火狐：没有内置，只能通过事件函数第一个参数定义

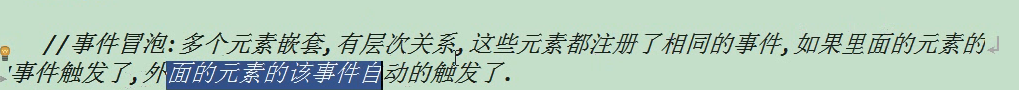
标准浏览器：通过事件函数第一个参数获取

非标准浏览器：event

兼容写法：

|  |
| --- |
| **function** *fn*(ev) {  **var** ev = ev || **event**; } |

## 3 事件冒泡



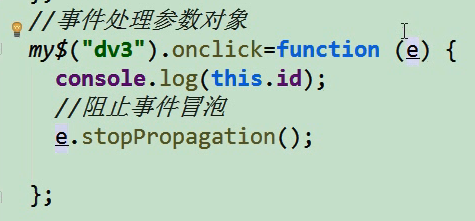
事件冒泡：当一个元素接收到事件的时候，会把接收到的所有事件传播给父级，一直到顶级window。事件冒泡与元素的显示界面无关，只与元素之间的结构关系有关。

冒泡机制：通常我们给一个元素添加了事件，只是添加了事件处理函数。并不是没有添加事件，就不能接收该事件。同理，冒泡事件中，子级触发了一个点击事件，而父级没有写点击事件的处理函数，那么父级仍然是接收了该事件，只是不知道要如何处理而已。

阻止冒泡：在当前要阻止冒泡的事件函数中调用event.cancelBubble = true;

event.cancelBubble = true; //IE不支持

火狐方法：



IE不支持原因：方法里没有事件对象，只有window.event

## 4 事件绑定

以前的绑定：

obj.onclick = fn;

obj.onclick = fn2; //fn2会覆盖fn1.

新的事件绑定：

IE：obj.attachEvent(‘onclick’,fn1);

没有捕获，事件名有on，绑定多事件执行顺序为正序，this为window。

标准：obj.addEventListener(‘onclick’,fn1,false);

有捕获，事件名没有on，执行顺序倒序，this为触发该事件的对象。

false：冒泡方式；true：捕获方式，这是一个可选项，默认冒泡。

由于IE下，事件函数内部的this指向的是window，这是不能容忍的，需要使用call()修改指向。

**兼容写法：**

|  |
| --- |
| **function *bind*(obj,eName,fn) {  if(obj.addEventListener){  obj.addEventListener(eName,fn,false);  }else {  obj.attachEvent('on' + eName,function () {  fn.call(obj);  });  } }** |

IE下是没有捕获的，在标准浏览器下，默认不捕获，使用false，意思就是以冒泡形式触发函数。

<div class =’father>

<div class=’son’>

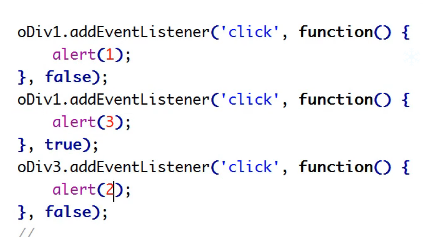
</div>

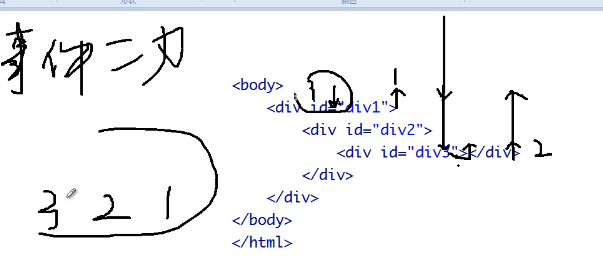
</div>

如果是false，冒泡形式：执行从son开始往外的事件函数

如果是true，捕获形式：执行从father开始往内的事件函数

案例：三个div，div1包裹着div2，div2包裹着div3





解析：

当我们点击div3的时候，事件是从最外层依次往里进入的，在进入时会按照层级依次循环div1，div2，div3是否捕获：

捕获（参数为true）则执行该事件，执行完传递给下一层，

不捕获（参数为false时），直接传递给下一层，

到达真正的点击对象div3时，开始冒泡：

依次查看div3，div2，div1是否执行冒泡。

## 5 事件取消

第一种绑定形式取消：

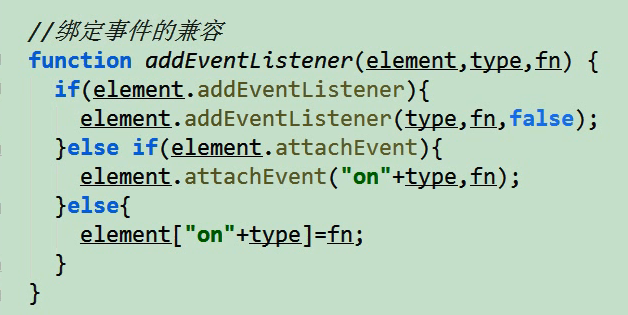
节点.onclick = null;

第二种绑定形式取消：

IE下：节点.detachEvent(事件名称,事件函数);

标准：节点.removeEventListener(事件名称,事件函数,是否捕获默认false);





## 6 事件默认行为

当一个事件发生时，浏览器会默认做一些事情，如何阻止？

该行为是哪个事件触发，就在这个事件处理函数中使用 return false

但是retrun false只能阻止传统的obj.on事件名=fn形式触发的默认行为，如果事件是以addEventListner绑定，那么取消该行为需要使用：

event.preventDefault();

## 7 mouseover与mouseenter

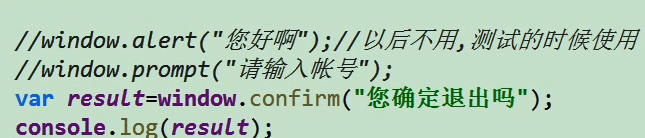
mouseover/mouseout事件，鼠标经过的时候会触发多次，每遇到一个子元素就会触发一次。

mouseenter/mouseleave事件，鼠标经过的时候只会触发一次。

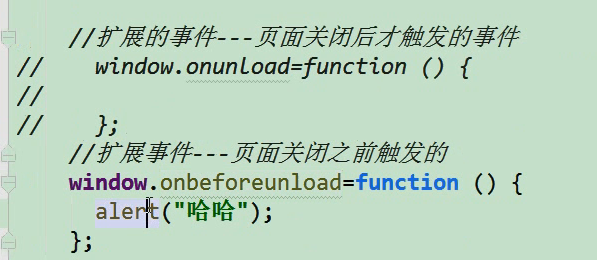
# 四 BOM

BOM(Browser Object Model)即浏览器对象模型，定义了JS操作浏览器的方法和属性。BOM中的方法属性大部分都属于window对象。

系统对话框：



谷歌不支下列这2个事件：



open(url,打开方式)

打开新的一个窗口，url为空；

打开一个空白页，打开方式为空，新窗口方式打开；

对应的方法是 close()