## 02.js笔记—事件

**（一）基本概念**

1、何为事件

事件是JS捕捉到的发生在网页上的行为：点击鼠标，滚动鼠标，敲键盘。

2、重要么？——交互体验的核心功能

**（二）如何绑定事件处理函数**

1、ele.onxxx=function(event){}

a) 兼容性很好，但是一个元素的同一个时间上只能绑定一个处理程序

b) 句柄式写法，基本等同于写在HTML行间上

2、obj.addEventListener(type,fn,false)

a) IE9以下不兼容，可以为一个事件绑定多个处理程序（引用地址不同）

3、 obj.attachEvent('on'+type,fn);

a) IE专有( ie11例外)，一个事件同样可以绑定多个处理程序（引用地址不同）

b) <meta http-equiv="X-UA-Compatible" content="IE=10" />，解决 IE11向后兼容 IE10的问题

// 小练习：使用原生js的addEventListener，给每一个li 元素绑定一个click 事件，输出它们的顺序

<ul>

<li>a</li>

<li>a</li>

<li>a</li>

<li>a</li>

</ul>

// 当事件出现在 for循环里绑定并且事件处理函数用到了 i时，注意闭包问题！

**（三）事件处理程序的运行环境this**

1、ele.onxxx=function(){}

程序的 this指向的是 dom元素本身

2、obj.addEventListener(type,fn,false);

程序的 this指向的是 dom元素本身

3、obj.attachEvent('on'+type,fn);

程序的 this指向 window

// 封装兼容性的addEvent(elem,type,handle);方法

**（四）解除事件绑定**

1、ele.onclick=false/''/null;

2、ele.removeEventListener(type,fn,false);

3、ele.detachEvent("on"+type,fn);

// 注：若绑定的是匿名函数，则永远无法解除

**（五）事件处理程序模型**

1、事件冒泡：

结构上（非视觉上）嵌套关系的元素，会存在事件冒泡的功能，即同一事件，自子元素冒泡向父元素。（自底向上）

2、事件捕获

结构上（非视觉上）嵌套关系的元素，会存在事件捕获的功能，即同一事件，自父元素捕获至子元素（事件源元素）。（自顶向下）

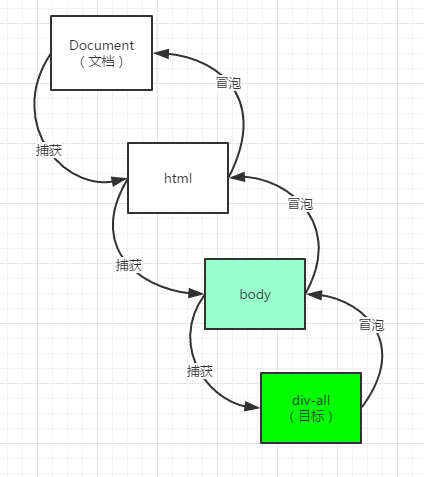
// 将addEventListener的第三个参数设置为true则可启用事件捕捉模型

// ie8以下由于不支持addEventListener，故没有事件捕获的事件处理模型

// focus,blur,change,submit,reset,select等事件不冒泡

3、事件的三个阶段

a) 先捕获，后目标，再冒泡



//对于目标元素而言，不存在事件处理模型，事件触发则执行，先绑定谁就执行谁

b) 默认在冒泡的时候执行事件（onclick/attach 和 addEventListener传false的时候）

c) addEventListener绑定事件，如果把第三个参数设置为true，则在捕捉的时候执行事件

d) 整个事件处理过程，会有个event事件对象在整个事件过程传播（W3C标准，ie8及其以下没有）

**（六）取消冒泡和阻止默认事件**

1、取消(停止)冒泡

a) W3C标准 event.stopPropagation();但不支持 ie9以下版本

b) event.cancelBubble=true; ie8及ie8以下可用

// 封装一个取消冒泡的函数stopBubble(event)，或者使用三目运算符

2、阻止默认事件

默认事件——表单提交（当表单中只有一个input\_text框时,回车即被提交），a标签跳转，右键菜单等

a) return false; 以对象属性的方式注册的事件才生效，用addEventListener/attachEvent这种是不生效的

b) event.preventDefault(); W3C标准，IE9以下不兼容

c) event.returnValue=false;兼容IE

// 封装阻止默认事件的函数 cancelDefault(event)，或者三目运算符

// 小练习：右键弹出自定义菜单(事件类型：contextmenu)

**（七）事件源对象**

1、事件对象：

a) 事件处理函数形参 ev(event)，W3C制定的标准，IE9以下不行

b) 全局对象 window.event用于，IE8及其以下

// 兼容性写法 var event= ev|| window.event

c) 事件对象上存储着事件发生时键盘敲入的编号信息(eg:event.which)

d) 事件对象上还存储着事件发生时的位置信息等。

// 小练习：实现案例，京东放大镜

2、事件源对象：

event.target 火狐只有这个

event.srcElement IE6/78只有这个

这两 chrome都有 var target= event.target|| event.srcElement

// 兼容性写法 var ele=event.target|| event.srcElement

// 小练习：使用原生js，add Event Listener，给每一个li 元素绑定一个click 事件，输出它们的内容

<ul>

<li>1</li>

<li>2</li>

<li>3</li>

<li>4</li>

</ul>

//若for循环之后增加几个li呢？ 若li增加到3000亿个呢？

**（八）事件委托**

1、利用事件冒泡，和事件源对象进行处理

2、优点：

a) 性能：不需要循环所有的元素一个个绑定事件

b) 灵活：当有新的子元素时不需要重新绑定事件

// 小练习（承上启下）：利用 onmousedown onmouseup onmousemove实现连续拖拽方块

// 将拖拽封装成一个函数drag(elem); 用addEventListener，且停止mousedown冒泡。

**（九）鼠标事件**

1、click、mousedown、mousemove、mouseup、contextmenu、mouseover、mouseout、mouseenter(html5标准)、mouseleave(html5标准)

a) mousedown-->mouseup-->click

b) mousedown是鼠标按下一次就只触发一次；

c) mousemove与 mousedown无关，相互独立；

d) 补充：移动端没有 mousedown事件，对应的是 touchstart--touchmove--touchend

2、DOM3标准规定：click事件只能监听左键，mousedown和 mouseup来判断鼠标键

3、用 event.button来区分鼠标的按键，0/1/2

// 判断 mousedown--mouseup中间是发生的拖拽还是发生的 click的判断问题。

a标签如果你是点击就页面跳转，如果你是长摁就弹出警告框。

**（十）键盘事件**

1、keydown、keyup、keypress

a) keydown-->keypress-->keyup

b) keydown在按键按下之后会一直不断被触发

2、keydown和 keypress的区别

a) keydown可以响应任意按键(除了Fn键)，keypress只可以响应字符类键盘按键（event.charCode）

b) keypress返回 ASCII码，可以转换成相应字符（String.fromCharCode(event.charCode)，区分大小写）

// keydown常用于绑定操作类事件处理，keypress常用于绑定字符类事件处理

**（十一）文本操作事件**

1、input、focus、blur、change

a) input监听 input框在聚焦状态下的文本变化 ，例：百度搜索

b) change监听 input框在失焦后的文本变化（与前一次失焦状态的文本对比变化）

c) focus和blur只在聚焦和失焦的一刻触发一次

// 使用 onfocus和 onblur模拟类似html5属性placeholder的功能，聚焦失焦状态下改变文本内容文本颜色；（参考新浪首页、58同城案例）

**（十二）窗体操作类(window上的事件)**

1、scroll、load

a) scroll：常用于绑定在window对象上，例网易严选 fixed定位

//window.onscroll=function(){

var srcollx=window.pageXoffset||document.body.scrollLeft+document.document Element.srollLeft;

}

b) load：常用于绑定一个操作script之后元素的事件处理函数，等待网页资源下载完毕再执行

补充：

网页加载顺序：url-->下载页面-->domTree，cssTree并行-->渲染树renderTree-->绘制页面-->继续下载图片资源，下载完毕再放到页面上去🡪onload

domTree：domTree的形成，是先把元素挂到 domTree上去，再把属性 img\_src放到渲染树上去

window.onload：等待页面所有资源下载完成才执行，所以它是最慢的，常规需求（主程序）都用不到它

**（十三）总结--ie兼容性问题：**

**①** 根据节点关系查找元素节点，即遍历元素节点树的方法

**②** 获取元素显示的样式集合 window.getComputedStyle(elem,null)-->elem.currentStyle

**③** 滚动条滚动距离

window.pageXoffset-->document.body.scrollLeft + document.documentElement.scrollLeft

**④** 事件绑定 addEventListener-->attachEvent

**⑤** 事件解除 removeEventListener(type,fn,flase)-->detachEvent("on"+type,fn)

**⑥** 事件运行环境，即this的指向问题（elem与window）

attachEvent(type,fn)-->addAttachEvent( type,function(){ fn.call(element)} )

**⑦** 事件处理函数中的事件对象 var event=ev || window.event

**⑧** 事件源对象 var target=event.target || event.srcElement

**⑨** 事件阻止冒泡 event.stopPropagation()-->event.cancelBubble=true

**⑩** 事件阻止默认事件 event.preventDefault()-->event.returnValue=false