# [系统学习DOM事件机制](http://www.cnblogs.com/chengzp/p/domevent.html)

本文将从以下几个方面介绍DOM事件：

1. 基本概念：DOM事件的级别
2. DOM事件模型，事件流
3. 描述DOM事件捕获的具体流程
4. Event对象的常见应用
5. 自定义事件

## **DOM事件的级别**



[IMG_257](http://www.cnblogs.com/chengzp/p/javascript:void(0);)

//DOM0

element.onclick = function(){}

//DOM2

element.addEventListener('click',function(){},false)

//DOM3

element.addEventListener('onmouseover',function(){},false)

[IMG_258](http://www.cnblogs.com/chengzp/p/javascript:void(0);)

**先解答一个小疑问：为啥没有DOM1级事件呢？**

　因为在1级DOM标准中没有定义与事件有关的内容，所以事件的级别是没有DOM1的。

下面简单介绍dom0，2，3的区别

　0级是最早的，而且目前所有的浏览器仍支持0级DOM模型。缺点是一个事件的处理程序只能对应一个函数

　删除DOM0事件处理程序，只要将对应事件属性置为null即可。

　在2级模型中可以为特定对象的事件注册多个事件监听。

　2级事件的删除

removeEventListener("click",function(){},false);

　3级事件的操作方式和2级事件是一样的。DOM3级事件模块在DOM2级事件的基础上重新定义了这些事件，也添加了一些新事件。包括IE9在内的主流浏览器都支持DOM2级事件，IE9也支持DOM3级事件。dom对以下事件做了改动

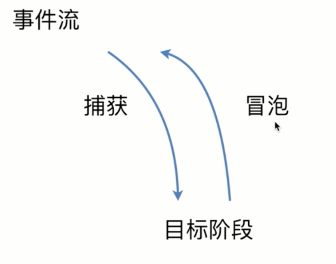
* UI事件
* 焦点事件
* 鼠标事件
* 滚轮事件
* 文本事件
* 键盘事件
* 合成事件
* 变动事件

关于dom事件级别更多相关内容，推荐大家一篇文章，写的挺详细的。[传送门](http://www.qdfuns.com/notes/11861/e21736a0b15bceca0dc7f76d77c2fb5a.html" \t "http://www.cnblogs.com/chengzp/p/_blank)

另：dom3中还定义了自定义事件，在文章下面中会讲到

## **事件模型  事件流**

**定义**：  
1.事件流描述的是从页面中接收**事件**的顺序,也可理解为**事件**在页面中传播的顺序。  
2.事件就是用户或浏览器自身执行的某种动作。诸如click(点击)、load(加载)、mouseover(鼠标悬停)。  
3.事件处理程序响应某个事件的**函数**就叫事件处理程序(或事件侦听器)

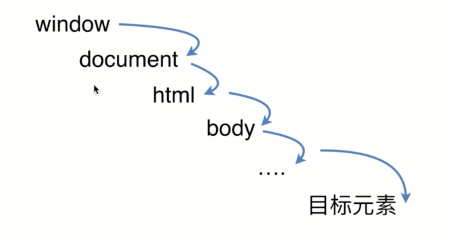


这两个事件流分别的是IE公司和netspace公司提出来的，冒泡事件流支持的浏览器更多。

 事件捕获阶段：事件从最上一级标签开始往下查找，直到捕获到事件目标(target)。  
 事件冒泡阶段：事件从事件目标(target)开始，往上冒泡直到页面的最上一级标签。

## **描述DOM事件捕获的具体流程**

事件捕获示意图



和事件冒泡相反，事件捕获是自上而下执行，我们只需要将addEventListener的第三个参数改为true就行。

下面通过代码示例来感受一下

[IMG_261](http://www.cnblogs.com/chengzp/p/javascript:void(0);)

<body>

<div id="ev">

<style media="screen">

#ev{

width: 300px;

height: 100px;

background: red;

color: #fff;

text-align: center;

line-height: 100px;

}

</style> 目标元素

</div>

<script type="text/javascript">

var ev = document.getElementById('ev');

ev.addEventListener('click', function (e) {

console.log('ev captrue');

}, true);

window.addEventListener('click', function (e) {

console.log('window captrue');

}, true);

document.addEventListener('click', function (e) {

console.log('document captrue');

}, true);

document.documentElement.addEventListener('click', function (e) {

console.log('html captrue');

}, true);

document.body.addEventListener('click', function (e) {

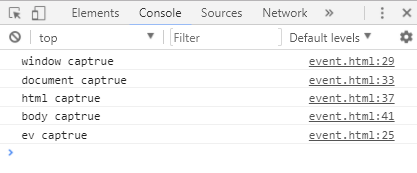
console.log('body captrue');

}, true);</script>

</body>

[IMG_262](http://www.cnblogs.com/chengzp/p/javascript:void(0);)

输出结果



 这个例子很好的验证了捕获是由上到下的。

注意：最上层的是window而不是html。

## **Event对象的常见应用**

event对象的方法是有很多的，像鼠标位置键盘key值的这样的就不多说了，下面介绍几个我认为比较重要的。

1.event.preventDefault()

　　此方法用来阻止默认的事件，比如阻止a标签的跳转。用法：

dom.addEventListener('click',function(event){

event.preventDefault()

},false)

2.event.stopPropagation()

　　阻止冒泡；让事件停留在当前dom而不会向上传递，

3.event.stopImmediatePropagation()

　　stopImmediatePropagation()函数用于阻止剩余的事件处理函数的执行，并防止当前事件在DOM树上冒泡。

4.event.target

　　target 事件属性可返回事件的目标节点（触发该事件的节点），如生成事件的元素、文档或窗口。

来一段代码

[IMG_264](http://www.cnblogs.com/chengzp/p/javascript:void(0);)

<html>

<head>

<script type="text/javascript">function getEventTrigger(event)

{

x=event.target;

alert("The id of the triggered element: "

+ x.id);

}</script>

</head>

<body >

<p id="p1" onmousedown="getEventTrigger(event)">

Click on this paragraph. An alert box will

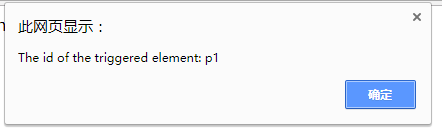
show which element triggered the event.</p>

</body>

</html>

[IMG_265](http://www.cnblogs.com/chengzp/p/javascript:void(0);)

输出结果



5.event.currentTarget

　　currentTarget 事件属性返回其监听器触发事件的节点，即当前处理该事件的元素、文档或窗口。

　　在捕获和起泡阶段，该属性是非常有用的，因为在这两个节点，它不同于 target 属性。

　　与event.target的区别

　　event.currentTarget指向事件所绑定的元素，而event.target始终指向事件发生时的元素

代码：

[IMG_267](http://www.cnblogs.com/chengzp/p/javascript:void(0);)

<html>

<head>

<script type="text/javascript">function getEventTrigger(event)

{

x=event.currentTarget;

alert("The id of the triggered element: "

+ x.id);

}</script>

</head>

<body >

<p id="p1" onmousedown="getEventTrigger(event)">

Click on this paragraph. An alert box will

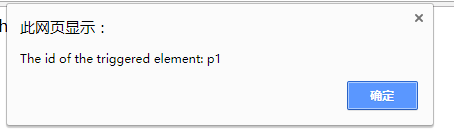
show which element triggered the event.</p>

</body>

</html>

[IMG_268](http://www.cnblogs.com/chengzp/p/javascript:void(0);)

结果



## **自定义事件**

这部分直接通过代码来感受一下吧

[IMG_270](http://www.cnblogs.com/chengzp/p/javascript:void(0);)

var eve = new Event('test');  
ev.addEventListener('test', function () {  
　　console.log('test dispatch');  
});  
setTimeout(function () {  
　　ev.dispatchEvent(eve);  
}, 1000)

[IMG_271](http://www.cnblogs.com/chengzp/p/javascript:void(0);)

一秒钟后，控制台打印

