事 件

# Dom0 dom2 定义区别

1. 事件流，冒泡与捕获
2. 事件目标target currentTarget

# dom2和dom0的定义和区别

**DOM0**

分为2个：一是在标签内写onclick事件

　　　　  二是在JS写onlicke=function（）{}函数

**1）**

<input id="myButton" type="button" value="Press Me" onclick="alert('thanks');" >

**2）**

document.getElementById("myButton").onclick = function () {

alert('thanks');

}

**DOM2**

只有一个：监听方法，原生有两个方法用来添加和移除事件处理程序：addEventListener()和removeEventListener()。

它们都有三个参数：第一个参数是事件名（如click）；

　　　　　　　　　第二个参数是事件处理程序函数；

　　　　　　　　   第三个参数如果是true则表示在捕获阶段调用，为false表示在冒泡阶段调用。

* addEventListener():可以为元素添加多个事件处理程序，触发时会按照添加顺序依次调用。
* removeEventListener():不能移除匿名添加的函数。

由于div是层层嵌套的，所以就有了内层和外层的概念，当事件发生时，内外是如何联系的呢？即事件传递的顺序，事件流，像一个单位，学校一样。发生事情级级上报。

事件冒泡和事件捕获分别由微软和网景公司提出，这两个概念都是为了解决页面中事件流（事件发生顺序）的问题。

document.getElementById("myTest").attachEvent("onclick", function(){alert(1)});  
//等价于  
document.getElementById("myTest").addEventListener("click", function(){alert(1)}, false);

只有2级DOM包含3个事件：事件捕获阶段、处于目标阶段和事件冒泡阶段

<span>

<a></a>

</span>

点击a后capturing(捕捉)阶段事件传播会从document-> span->a，然后发生在a，最后bubbling(冒泡)阶段事件传播会从a->span->document 。

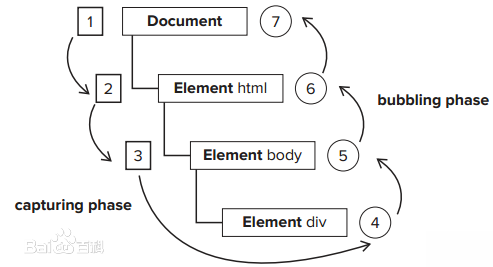
区别：如果定义了两个dom0级事件，dom0级事件会覆盖

dom2不会覆盖，会依次执行

dom0和dom2可以共存，不互相覆盖，但是dom0之间依然会覆盖

事件的冒泡与捕获：

如图所示：



事件流，事件传播：

传播(从里到外或者从外向里)。为什么要传播呢？

例如加给ul事件时，点击li，li会向上传播到ul.找到绑定的点击事件

因为事件源本身（可能）并没有处理事件的能力，即处理事件的函数（方法）并未绑定在该事件源上。例如我们点击一个按钮时，就会产生一个click事件，但这个按钮本身可能不能处理这个事件，事件必须从这个按钮传播出去，从而到达能够处理这个事件的代码中（例如我们给按钮的onclick属性赋一个函数的名字，就是让这个函数去处理该按钮的click事件），或者按钮的父级绑定有事件函数，当该点击事件发生在按钮上，按钮本身并无处理事件函数，则传播到父级去处理。

传播分为冒泡与捕获

* 事件冒泡 是从触发事件的那个节点一直到document，是自下而上的去触发事件。
* 事件捕获指的是从document到触发事件的那个节点，即自上而下的去触发事件。
* 事件委托：利用了事件冒泡的原理，
* 阻止事件委托和冒泡：在事件函数中执行event.stopPropagation()
* 事件的兼容性：

var evt = event || widow.event;

范例：

var \_ul=document.querySelector("ul");

var lis=document.querySelectorAll("ul li");

\_ul.onclick=function (event){

var evt = event || widow.event;

console.log("ul 收到事件",evt);

}

lis[0].onclick=function (event){

console.log("li 收到事件");

event.stopPropagation();//阻止后，上级无法收到事件

}

lis[1].onclick=function (event){

console.log("li 收到事件")

}

“DOM2级事件”规定的事件流包括三个阶段：事件捕获阶段、处于目标阶段和事件冒泡阶段。首先发生的是事件捕获，然后处于目标阶段，最后才事件冒泡。

事件目标：target currentTarget

用最简单的语言总结：  
target: 代表当前目标对象（事件作用的对象）  
currentTarget: 代表注册监听器的对象  
this: 和currentTarget一样，即它们的作用相等。

1、target发生在事件流的目标阶段，而currentTarget发生在事件流的整个阶段（捕获、目标和冒泡阶段）  
2、只有当目标流处于目标阶段的时候才相同

3、而当事件流处于捕获和冒泡阶段时，target指向被点击的对象，而currentTarget指向当前事件活动的对象，通常是事件的祖元素。

实例讲解

事件冒泡和事件捕捉：

<div id="parent">

　　<div id="child" class="child"></div>

</div>

<script type="text/javascript">

document.getElementById("parent").addEventListener("click",function(e){

console.log("parent事件被触发，"+this.id);

},false);

document.getElementById("child").addEventListener("click",function(e){

console.log("child事件被触发，"+this.id);

},false);

</script>

上述代码给child和parent元素都设置了点击事件，并且都是采用的事件冒泡机制。   
1.当点击child元素时，先触发child的点击事件，再触发parent的点击事件，打印相应的内容；   
2.当点击parent元素时，只触发parent的点击事件，因为parent元素冒泡上去没有dom元素设置了click事件的函数；

<div id="parent">

　　<div id="child" class="child"></div>

</div>

<script type="text/javascript">

document.getElementById("parent").addEventListener("click",function(e){

console.log("parent事件被触发，"+this.id);

},true);

document.getElementById("child").addEventListener("click",function(e){

console.log("child事件被触发，"+this.id);

},true);

</script>

上述代码采用事件捕捉   
1.当点击child元素时，先触发parent的点击事件，再触发child的点击事件，打印相应的内容；   
2.当点击parent元素时，只触发parent的点击事件，因为parent元素从上面捕捉下来一直到parent元素没有dom元素设置了click事件的函数；

总结：所以当点击child元素时，不论是设置的事件捕捉还是事件冒泡，都会触发parent的点击事件，除非不给parent元素设置点击事件；

事件委托：

当有多个类似的元素需要绑定事件时，一个一个去绑定即浪费时间，又不利于性能，这时候就可以用到事件委托，给他们的一个共同父级元素添加一个事件函数去处理他们所有的事件情况，根据下面代码分析;

<ul id="1ist">

<li>111</li>

<li>222</li>

<li>333</li>

<li>444</li>

</ul>

<script type="text/javascript">

document.getElementById('1ist').addEventListener('click',function(e){

e.target.innerHTML = "被点击";

});

</script>

可能大家会疑惑，我们并没有给li分别设置点击函数，为什么点击相应的li却会产生效果，我们可以这么理解，虽然我们没有给li设置点击事件，但是默认的点击事件是采用的事件冒泡，冒泡到父级元素的点击事件是，用事件的target属性判断进行点击的元素