**探究JavaScript中的五种事件处理程序**

　　我们知道JavaScript与HTML之间的交互是通过事件实现的。**事件最早是在IE3和Netscape Navigator 2中出现的，当时是作为分担服务器运算负载的一种手段。**通俗地理解，事件就是用户或浏览器自身执行的某种操作。而事件处理程序即为响应某个事件的函数。抽出主干，即**事件处理程序为函数。**  我们又把事件处理程序称为事件**侦听器。**  事件处理程序是以"on"开头的，因此对于事件on的时间处理程序即为onclick。时间处理程序在JavaScript中大致有五种，下面会根据这五种不同的时间处理程序分为5部分来介绍。

1. [HTML事件处理程序](https://www.cnblogs.com/zhuzhenwei918/p/6139281.html#1)
2. [DOM0级事件处理程序](https://www.cnblogs.com/zhuzhenwei918/p/6139281.html#2)
3. [DOM2级事件处理程序](https://www.cnblogs.com/zhuzhenwei918/p/6139281.html#3)
4. [IE事件处理程序](https://www.cnblogs.com/zhuzhenwei918/p/6139281.html#4)
5. [跨浏览器的事件处理程序](https://www.cnblogs.com/zhuzhenwei918/p/6139281.html#5)

**第一部分：HTML事件处理程序**

　　什么使HTML事件处理程序呢？显然，通过名字就可以猜到，它是卸载HTML中的函数（事件处理程序）。初学者大多用到的事件处理程序即为HTML事件处理程序。下面举例：

例1：

<button onclick="alert('success')">点我</button>

这条代码即为事件处理程序，点击button后，会弹出弹框，显示success。

**特点：HTML事件处理程序中Javascript代码作为了onclick特性的值，因此，我们不能在JavaScript代码中使用未经转义的HTML语法字符，如&(和号)、""（双引号）、<(小于号)、>（大于号）等等。**所以这个例子中字符串我使用了单引号而没有使用双引号。看下面在JavaScript代码中使用了未经转义的HTML语法字符。

例2：

<button onclick="alert("success")">点我</button>

这时，**我在success外使用了HTML语法字符""(双引号)，这时不会弹出窗口，而是报错语法错误。**但是我如果还是希望使用双引号呢？ 这时就要**用&quot;实体来代替HTML中的语法字符。**如下例所示：

例3：

<button onclick="alert(&quot;success&quot;)">点我</button>

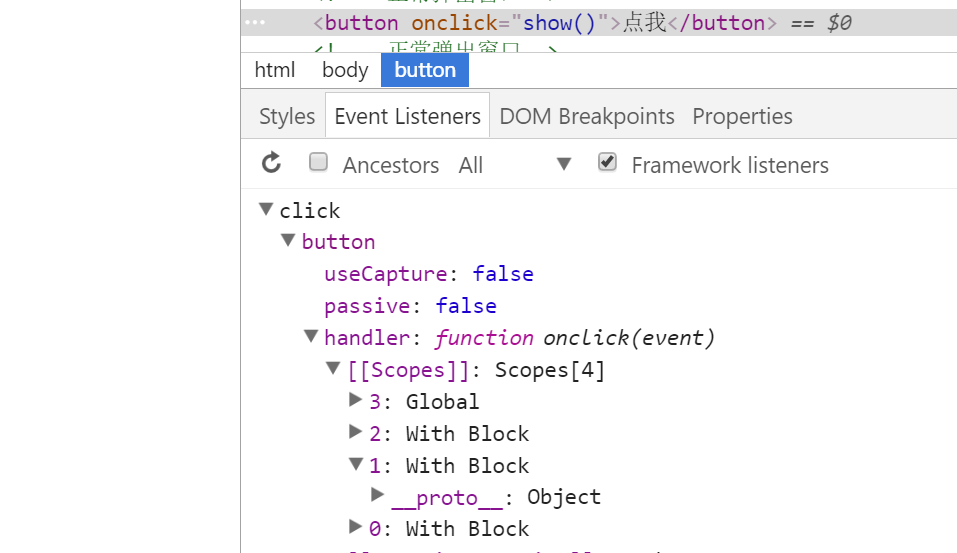
<!-- 正常弹出窗口-->

这个例子中**我们在JavaScript代码中使用了HTML实体而没有使用HTML语法字符，**这时就不会报错了。

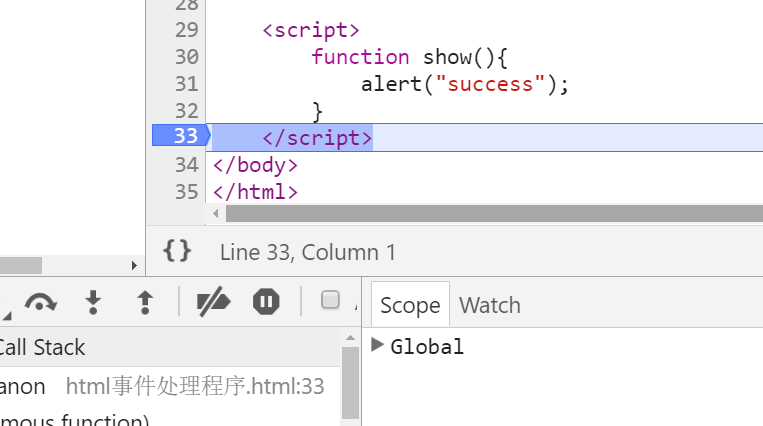
例4：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | <button onclick="show()">点我</button>  <!--  正常弹出窗口-->    <script>      function show(){          alert("success");      }  </script> |

**这个例子中我们调用函数，而把函数定义放在了script中，这样也是可以的。因为：事件处理程序中的代码在执行时，有权访问到全局作用域中的任何代码。这句话怎么理解呢？  实际上，我们可以在chrome中观察button标签的作用域链。如下所示：**

****

　　接下来我们再看看script所在的作用域，如下图所示：



可以看到script标签就在全局作用域。

　　也就是说目前button中的HTML事件处理函数在作用域链的最前端，而Script在全局作用域，所以“**事件处理程序中的代码在执行时，有权访问到全局作用域中的任何代码。”**这句话就不难理解了。

例5：

<button onclick="alert(event.type)">点我</button>

这时浏览器弹出窗口显示：click。这个例子是什么意思呢？注意到我并没有在event.type外加单引号，说明这并不是字符串。实际上，**event是局部对象--在触发DOM上的某个事件时，会产生一个事件对象event，这个对象包含着所有与事件有关的信息。**而这里是弹出了对象了类型，即为click。

**HTML事件处理程序的三个缺点（重点）：**

**1. 时差问题**。 因为用户可能在HTML元素一出现就开始触发相应事件，但是有可能该事件的脚本（如例4中show()函数的函数定义在script中）还没有加载完成，**此时不具备执行条件，故报错。**

**解决方法：将HTML事件处理程序封装在一个try-catch块中，以便错误不会浮出水面。**

<input type="button" value="click me" onclick="try{show();}catch(ex){}">

**2.这样扩展事件实例程序的作用域链在不同的浏览器中会导致不同的结果（例4中我是在chrome中查看的作用域链，其他浏览器不一定是这样的，请注意）。不同JavaScript引擎遵循的标识符解析规则略有差异，很有可能会在访问非限定对象成员时出错。**

**3.HTML和JavaScript代码紧密耦合。 结果是：如果要更换事件处理程序，就必须改动两个地方--HTML代码和JavaScript代码。**

那么怎么解决上面的问题呢？ DOM0级事件处理程序是一个不错的选择!

**第二部分：DOM0级事件处理程序**

DOM0级事件处理程序用的也非常普遍。之所以成为DOM0级，我认为是当时还没有出DOM标准，而IE和Netscape Navigator两者使用的时间处理程序（不知是否合理，望批评指正）。 总之，我们先看看下面的例子吧。

**例6：**

[复制代码](javascript:void(0);)

<button id="button">点我</button>

<script>

var button=document.getElementById("button");

button.onclick=function(){

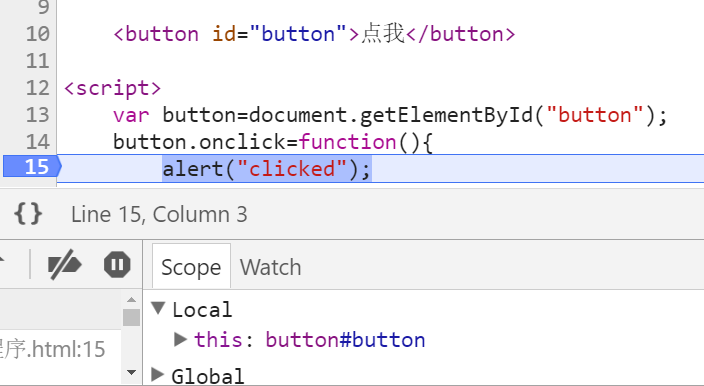
alert("clicked");

}

</script>

[复制代码](javascript:void(0);)

**即我们先在script中取得元素的引用，然后再将一个函数赋值给onclick事件处理程序。**之前介绍过，事件处理程序即为函数，而**button.onclick这种形式即函数作为了对象的方法。那么对象的方法即事件处理程序是在元素（对象）的作用域中运行而非在全局作用域中运行的，因为方法是属于对象的。(**注意：例4中事件处理程序是在全局作用域中运行的）。 **如果这个函数中存在this关键字，那么this就会指向这个对象。**下面我们在浏览器中证明事件处理程序是在元素的作用域中运行。



我们看到alert("clicked");确实是在button中运行的。

我们还可以通过下面的方式删除通过DOM0级方法指定的事件处理程序。

button.onclick=null;

通过上面的分析我们可以知道**DOM0级事件处理程序是非常不错的，它解决了HTML事件处理程序的三个缺点：**时差问题、作用域链导致的不同浏览器表现不一致问题和HTML和JavaScript紧密耦合问题。

　　但是，**DOM0级事件处理程序并不是完美**的，它同样有两个缺点：

1. 我们不能给一个元素同时添加两个事件。
2. 我们不能控制元素的事件流（捕获or冒泡）。

　　对于第二个问题后面会讲到，第一个问题举例如下：

[复制代码](javascript:void(0);)

<button id="button">点我</button>

<script>

var button=document.getElementById("button");

button.onclick=function(){

alert("clicked");

}

button.onclick=function(){

alert("again");

}

[复制代码](javascript:void(0);)

**虽然我对同一个元素设置了两个事件处理程序，但是最终的结果是：只有第二个事件有效（覆盖了第一个事件）**。当然，人类是聪明的动物，DOM2级事件很好的解决了这个问题!

**第三部分：DOM2级事件处理程序**

　　DOM2级事件处理程序定义了两个方法：

* **addEventListener()   ---添加事件侦听器**
* **removeEventListener()   ---删除事件侦听器**

**在博文的开头我就提到了事件处理程序即事件侦听器**。这两个方法都接收三个参数：

1. 要处理的事件名（注意：是时间名，所以没有on！），如click、mouseover等。
2. 作为事件处理程序的函数，如function(){alert("clicked");}
3. 表示事件流方式的布尔值。**false为冒泡阶段调用事件处理程序；true为捕获阶段调用事件处理程序。**对于冒泡和捕获这两种时间流可以看《[JavaScript中的两种事件流](http://www.cnblogs.com/zhuzhenwei918/p/6139880.html)》

下面通过两个例子加深理解：

**例7：**

[复制代码](javascript:void(0);)

<button id="button">点我</button>

<script>

var button=document.getElementById("button");

button.addEventListener("click",function(){

alert(this.id);

},false);

button.addEventListener("click",function(){

alert("another event");

},false);

</script>

[复制代码](javascript:void(0);)

**结果**：第一次弹出窗口：button。

　　　第二次弹出窗口：another event。

**结论**：**通过DOM2级事件处理程序，我们可以为同一个元素添加两个或更多的事件。事件根据顺序依次触发。且this同样指向当前元素，故函数在元素的作用域中执行。**

**this分析：**和前面的DOM0级事件处理程序一样，这里的addEventListener同样也可以看作对象的方法，不同之初在于，DOM0级的方法需要另外一个函数来赋值，而这里的方法是DOM2级规范预定义的。

**removeEventListener()这个删除事件处理程序的方法值得注意的是：使用addEventListener()来添加的事件处理程序只能通过它来移除，且需要传入相同的参数。**

**例8：**

[复制代码](javascript:void(0);)

<button id="button">点我</button>

<script>

var button=document.getElementById("button");

button.addEventListener("click",function(){

alert(this.id);

},false);

button.removeEventListener("click",function(){

alert("another event");

},false);

[复制代码](javascript:void(0);)

上述代码貌似可以移除click的事件处理程序，但是通过实验证明是不可以的，原因是：**事件处理程序为匿名函数时无法移除**。看下面的成功移除的例子：

**例9：**

[复制代码](javascript:void(0);)

<button id="button">点我</button>

<script>

var button=document.getElementById("button");

function handler(){

alert(this.id);

}

button.addEventListener("click",handler,false);

button.removeEventListener("click",handler,false);

</script>

[复制代码](javascript:void(0);)

**成功移除！**

**注意：1.传入方法的handler没有（），是因为这里都只是定义函数，而不是调用，需要注意。**

**2.这两个方法的第三个参数都是false，即事件处理程序添加到冒泡阶段。一般不使用true,因为低版本的IE不支持捕获阶段。**

**DOM2级事件处理程序成功地解决了前面所有事件处理程序的问题，堪称perfect!!!!**  然而总是特立独行的IE浏览器又有新花样，它也有自己的一套事件处理程序，下面我们就来看看吧。

**第四部分：IE事件处理程序**

　　IE事件处理程序中有类似与DOM2级事件处理程序的两个方法：

1. attachEvent()
2. detachEvent()

　　它们都接收两个参数：

1. 事件处理程序名称。如onclick、onmouseover，注意：这里不是事件，而是事件处理程序的名称，所以有on。
2. 事件处理程序函数。如function(){alert("clicked");}

**之所以没有和DOM2级事件处理程序中类似的第三个参数，是因为IE8及更早版本只支持冒泡事件流。**

**注意：**

**1.IE事件处理程序中attachEvent()的事件处理程序的作用域和DOM0与DOM2不同，她的作用域是在全局作用域中。因此，不同于DOM0和DOM2中this指向元素，IE中的this指向window。**

　　2.**同样，我们可以使用attachEvent()来给同一个元素添加多个事件处理程序。但是与DOM2不同，事件触发的顺序不是添加的顺序而是添加顺序的相反顺序。**

　　3.**同样地，通过attachEvent()添加的事件处理程序必须通过detachEvent()方法移除，同样的，不能使用匿名函数。**

　　4.**支持IE事件处理程序的浏览器不只有IE浏览器，还有Opera浏览器。**

**第五部分：跨浏览器的事件处理程序**

　　实际上，这一部分视为了跨浏览器使用，将前面的几部分结合起来就可以了。

　　这一部分需要创建两个方法：

* addHandler()  --这个方法职责是视情况来使用DOM0级、DOM2级、IE事件处理程序来添加事件。
* removeHandler()--这个方法就是移除使用addHandler添加的事件。

　　这两个方法接收相同的三个参数：

1. 要操作的元素--通过dom方法获取
2. 事件名称--注意：没有on，如click、mouseover
3. 事件处理程序函数--即handler函数

　　这两个方法的构造情况如下：

[复制代码](javascript:void(0);)

var EventUtil={

addHandler:function(element,type,handler){

if(element.addEventListener){

element.addEventListener(type,handler,false);//注意：这里默认使用了false（冒泡）

}else if(element.attachEvent){

element.attachEvent("on"+type,handler);

}else{

element["on"+type]=handler;

}

},

removeHandler:function(element,type,handler){

if(element.removeEventListener){

element.removeEventListener(type,handler,false);//注意：这里默认使用了false（冒泡）

}else if(element.detachEvent){

element.detachEvent("on"+type,handler);

}else{

element["on"+type]=null;

}

}

};

[复制代码](javascript:void(0);)

**即先判断DOM2级事件处理程序，再判断IE事件处理程序，最后使用DOM0级事件处理程序。**

**例10：**通过这个例子来使用上面构造的方法。

[复制代码](javascript:void(0);)

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>跨浏览器事件处理程序</title>

</head>

<body>

<button id="button">点我</button>

<script>

var EventUtil={

addHandler:function(element,type,handler){

if(element.addEventListener){

element.addEventListener(type,handler,false);//注意：这里默认使用了false（冒泡）

}else if(element.attachEvent){

element.attachEvent("on"+type,handler);

}else{

element["on"+type]=handler;

}

},

removeHandler:function(element,type,handler){

if(element.removeEventListener){

element.removeEventListener(type,handler,false);//注意：这里默认使用了false（冒泡）

}else if(element.detachEvent){

element.detachEvent("on"+type,handler);

}else{

element["on"+type]=null;

}

}

};

function handler(){

alert("clicked");

}

var button=document.getElementById("button");

EventUtil.addHandler(button,"click",handler);

</script>

</body>

</html>

[复制代码](javascript:void(0);)

最后浏览器成功弹出“clicked”。