**JavaScript中的事件对象**

 　 JavaScript中的事件对象是非常重要的，恐怕是我们在项目中使用的最多的了。在触发DOM上的某个事件时，会产生一个事件对象event，这个对象中包含这所有与事件有关的信息。下面将会讲到DOM中的事件对象、IE中的事件对象以及跨浏览器的事件对象三个部分。

　　对于事件处理程序，大家可以看我的博文《[JavaScript中的五种事件处理程序](http://www.cnblogs.com/zhuzhenwei918/p/6139281.html)》。

**第一部分：DOM事件对象**

　　兼容DOM的浏览器会将一个event对象传入到事件处理程序中，**无论是HMTL特性、DOM0级还是DOM2级，都会传入这个event对象。**这个对象具有下面的属性和方法。

　　DOM事件对象的属性有：

* **bubbles**布尔值 只读 表明事件是否冒泡
* cancelable 布尔值 只读 表明是否可以取消事件的默认行为
* currentTarget  元素 只读  表明事件处理程序正在处理的元素
* defaultPrevented 布尔值 只读 表明是否调用了preventDefault()方法
* detail 数值 只读 表明与事件相关的细节信息
* eventPhase  数值 只读  表明事件处理程序的阶段：1为捕获阶段，2为处于目标阶段，3为冒泡阶段
* target  元素 只读 事件的目标
* trusted  布尔值  只读  true表示事件为浏览器生成，false表示开发人员通过js创建
* type  字符串  只读  表明被触发事件的类型

　　DOM事件对象的方法有：

* preventDefault()   只读 用来取消事件的默认行为
* stopPropagation()  取消事件的冒泡
* stopImmediatePropagation() 取消事件的捕获或者冒泡（DOM3级新增方法）

　　下面举例说明上述几个属性和方法的使用：

**例1**：

preventDefault()：此方法可以阻止事件的默认行为，比如，点击a标签，本来应该会跳转到相应的url，但是我们可以使用方法阻止，如下所示:

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <title>preventDefault</title>  </head>  <body>  <div id="wrap">  <div id="div">  <a id="link" href="http://www.cnblogs.com/zhuzhenwei918">918之初</a>  </div>  </div>  <script>  var link=document.getElementById("link");  link.addEventListener("click",function(){  alert(event.cancelable);//true 表示可以取消事件的默认行为  event.preventDefault();// 取消事件的默认行为  alert(event.type); //click 即事件对象类型即点击 click  alert(event.trusted);//undefined 可能是因为目前还不支持此属性  alert(event.defaultPrevented); //true,即默认被阻止，因为上面我们使用了preventDefault()方法  alert(event.eventPhase);//2 表示处于目标阶段  },false);  var div=document.getElementById("div");  div.addEventListener("click",function(){  alert(event.target); //http://www.cnblogs.com/zhuzhenwei918 target为事件的实际目标 即a标签被点击之后，实际上是应该跳转的  alert(event.currentTarget);//[object HTML DivElement]  alert(this);//[object HTML DivElement]，可以看出event.currentTarget 和this始终是相同的。  alert(event.bubbles);// true 表示事件冒泡  alert(event.eventPhase); //3 表示处于冒泡阶段  event.stopPropagation(); // 使用此方法来阻止继续向上冒泡  },false);  var wrap=document.getElementById("wrap");  wrap.onclick=function(){  alert("wrap");//不显示，因为在之前已经阻止事件冒泡了。  };  </script>  </body>  </html>  [复制代码](javascript:void(0);) |

**例2：**

实际上，我们也可以**不使用event而是在function中传入一个参数来当作event使用（DOM0级和DOM2级都可以）**，如下所示：

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <title>bubble</title>  <style>  button{  background: red;  color:white;  }  #third{  width: 50px;  height: 50px;  border:thin solid red;  }  #second{  width: 100px;  height: 100px;  border:thin solid red;  }  #first{  width:200px;  height: 200px;  border:thin solid red;  }  </style>  </head>  <body>  <div id="first">  <div id="second" >  <div id="third" >  <button id="button">事件冒泡</button>  </div>  </div>  </div>  <script>  document.getElementById("button").addEventListener("click",function(e){  alert(e.cancelable);//捕获被阻止，这里没有弹出窗口  },true);  document.getElementById("third").addEventListener("click",function(e){  alert(e.currentTarget);//捕获被阻止，这里没有弹出窗口  },true);  document.getElementById("second").addEventListener("click",function(e){  alert("second");//弹出窗口  e.stopImmediatePropagation(); //阻止事件捕获  },true);  document.getElementById("first").addEventListener("click",function(e){  alert("first"); //弹出窗口  alert(e.eventPhase); //1处于捕获阶段  },true);  </script>  </body>  </html>  [复制代码](javascript:void(0);) |

**例3：**

**实际上，利用event的type属性结合switch语句我们还可以做更多的事情！**

比如我们需要在点击按钮、划过按钮和划出按钮时做不同的事情，我们可以这样写（**这里使用DOM0级添加事件处理程序，方法一样，事件对象依然是event,也可以传入一个参数，以该参数作为对象**）：

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <title>bubble</title>  <style>  #div{  width: 200px;  height: 200px;  }  </style>  </head>  <body>  <div id="div"></div>  <button id="button">点击我</button>  <script>  var button=document.getElementById("button");  var div=document.getElementById("div");  button.onclick=function(){  div.style.backgroundColor="red";  };  button.onmouseover=function(){  div.style.border="thin red solid";  };  button.onmouseout=function(){  alert("mouseout");  };  </script>  </body>  </html> |

这样写当然可以得到想要的效果，但是我们还可以这样写：

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <title>bubble</title>  <style>  #div{  width: 200px;  height: 200px;  }  </style>  </head>  <body>  <div id="div"></div>  <button id="button">点击我</button>  <script>  var button=document.getElementById("button");  var div=document.getElementById("div");  var handler=function(event){  switch(event.type){  case "click":  div.style.backgroundColor="red";  break;  case "mouseover":  div.style.border="thin red solid";  break;  case "mouseout":  alert("mouseout");  break;  }  };  button.onclick=handler;  button.onmouseover=handler;  button.onmouseout=handler;  </script>  </body>  </html>  [复制代码](javascript:void(0);) |

这样写的封装性似乎会更好一些！

**例4**

html事件处理程序也可以使用event对象，只是这里不存在传入参数的问题了。。如下所示:

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <title>html对象</title>  </head>  <body>  <button onclick="alert(event.type)">点我</button>  <!-- 点击之后弹出窗口click -->  </body>  </html>  [复制代码](javascript:void(0);) |

到这就讲完了DOM中的事件对象，那么接下来就要谈谈特立独行的IE了。

## 第二部分：IE中的事件对象

　　DOM中的事件对象时不需要区分我们是通过何种方式来添加事件处理程序的，如DOM0级、DOM2级和HTML事件处理程序，对象都直接使用event即可。而IE中却不是这么简单。

　　我们把IE中的事件对象分为三种情况来讲

* 第一种：DOM0级方法添加事件处理程序时的事件对象
* 第二种：IE事件添加事件处理程序时的事件对象
* 第三种：HTML方法添加事件处理程序时的事件对象

### 第一种：DOM0级方法添加事件处理程序的事件对象

　　这时，**event对象作为了window对象的一个属性**。举例如下：

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <title>对象</title>  </head>  <body>  <button id="button">点我</button>  <script>  var button=document.getElementById("button");  button.onclick=function(){  var event=window.event;  alert(event.type); //click 这在chrome中也可使用  };  </script>  </body>  </html> |

### 第二种：IE事件添加事件处理程序时的事件对象

　　　这时需要有event对象传入事件处理程序函数中，如下所示：

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <title>对象</title>  </head>  <body>  <button id="button">点我</button>  <script>  var button=document.getElementById("button");  button.attachEvent("click",function(event){  alert(event.type); //"click"  });  </script>  </body>  </html> |

这里也可以通过window对象来访问event对象，不过最好还是使用传入参数的方法。

第三种：HTML事件处理程序中的对象

　　这个方法与DOM中的相同。举例如下：

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <title>对象</title>  </head>  <body>  <button id="button" onclick="alert(event.type)">点我</button>  <script>  </script>  </body>  </html> |

另外，IE中的事件对象也有和DOM对象相似的属性和方法，下面的属性和方法是其特有的：

* **cancelBubble  布尔值  可读可写  默认值为false，设置为true时可以取消事件的冒泡  与DOM中的stopPropagation()方法的作用类似**
* **returnValue  布尔值  可读可写  默认值为true，设置为false时我们可以取消事件的默认行为，与DOM中的preventDefault()方法类似**
* **srcElement 元素  只读  事件的目标 ,和DOM中的target属性相同**
* **type  字符串 只读 即为被触发事件的类型**

## 第三部分：跨浏览器的事件对象

　　实际上，这一部分就是兼容前面两者实现对象的跨浏览器。在《[探究JavaScript的五种事件处理程序](http://www.cnblogs.com/zhuzhenwei918/p/6139281.html#5)》中用到了EventUtil对象，这里我们可以对之加强。

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <title>对象</title>  </head>  <body>  <button id="button" onclick="alert(event.type)">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;  }  },  getEvent:function(event){  return event?event:window.event;  },  getTarget:function(event){  return event.target||event.srcElement;  },  preventDefault:function(event){  if(event.preventDefault){  event.preventDefault();  }else{  event.returnValue=false;  }  },  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;  }  },  stopPropagation:function(event){  if(event.stopPropagation){  event.stopPropagation();  }else{  event.cancelBubble=true;  }  }  };  </script>  </body>  </html> |

其中，getEvent返回了event对象的引用，event是一般的对象，window.event时考虑了IE的对象。getTarget方法返回事件的目标。 preventDefault（）方法阻止了事件的默认行为。stopPropagation方法阻止了事件的冒泡。

下面举一个简单的例子：

|  |
| --- |
| var button=document.getElementById("button");  button.onclick=function(){  event=EventUtil.getEvent(event);//获得对象  EventUtil.stopPropagation(event); //阻止事件冒泡  alert("button clicked"); //弹出窗口  };  document.body.onclick=function(){  alert("body clicked");//未弹出窗口，因为阻止了事件冒泡  }; |