**事件流**

* **事件冒泡：**如果点击了页面中的元素，则click事件会按照节点树的顺序往上传递的过程。
* **事件捕捉：**事件的处理将从DOM层次的根开始，而不是从触发事件的目标元素开始，事件被从目标元素的所有祖先元素依次往下传递。在这个过程中，事件会被从文档根到事件目标元素之间各个继承派生的元素所捕获，如果事件监听器在被注册时设置了useCapture属性为true,那么它们可以被分派给这期间的任何元素以对事件做出处理；否则，事件会被接着传递给派生元素路径上的下一元素，直至目标元素。
* DOM规定事件流包括3阶段，依次是事件捕获、处于目标阶段和事件冒泡阶段。[Dome1](dome1.html)

**事件处理**

1. **HTML事件处理程序：**直接在标签里添加对应的事件。缺点1：可能在浏览器还没有解析函数之前就点击了元素导致错误，可以用 onclick=”try(fun();)catch()”这样就把错误捕获了，浏览器不处理错误**。**缺点2：js代码与HTML不分离。
2. **DOM0级事件处理程序：**用这种方法绑定的事件需先对元素进行引用，具有良好跨浏览器性能。Div.onclick=fun。用这种方法事件处理的时候，是在元素的作用域中进行，换句话说，this引用当前元素。而div.onclick=null可以删除事件（DOM0级和HTML都可以删除）
3. **DOM2级事件处理程序：**使用addEventListener()传入3个参数，处理的事件，事件的函数，布尔值。布尔值为true则捕获的时候处理事件，为false则冒泡的时候处理事件与DOM0一样。[Dome1](dome1.html)。当添加了2次同一事件时，会按照顺序运行事件函数。只能通过removeEventListener()解除绑定，传入3个参数，处理的事件，事件函数的引用值！！！，布尔值。布尔值决定删除捕获过程的事件还是冒泡过程的事件。Ie不兼容。
4. **IE事件处理：**attachEvent()添加事件，detachEvent()移除事件，传入的事件名为’onclick’之类，而DOM2传入’click’。都只在冒泡过程绑定事件，这个时候在全局作用域中运行，this是对window的引用。当添加了2次同一事件时，会按照**相反的**顺序运行事件函数。**所以这个方法添加的匿名函数和DOM2一样无法被覆盖。**
5. **跨浏览器添加移除事件：**

var EventUtil = {

addHandler : function(elem, type, handler){

if(elem.addEventListener){

elem.addEventListener(type, handler, false);

}

else if(elem.attachEvent){

elem.attachEvent("on" + type, handler);//添加多个同一类型的handler时，IE方式的规则是最后添加的最先触发

}

else{

if(typeof elem["on" + type] === 'function'){

var oldHandler = elem["on" + type];

elem["on" + type] = function(){

oldHandler();

handler();

}

}

else{

elem["on" + type] = handler;//支持添加多个事件处理器

}

}

}

**事件对象**

在标签内加onclick=”alert(event.type)”等效函数方法调用。[Dome2](dome2.html).

属性：

|  |  |  |  |
| --- | --- | --- | --- |
| 属性/方法 | 类型 | 读/写 | 说明 |
| bubbles | Bool | **只读** | 表明事件是否冒泡 |
| cancelable | B | 表明是否可以取消事件的默认行为 |
| currentTarget | Element | 其事件处理程序当前正在处理事件的那个元素 |
| detail | Int | 与事件相关的细节信息 |
| eventPhase | Int | 调用事件处理程序的阶段：1代表捕获，2代表“处于目标”，3代表冒泡 |
| preventDefaut() | Fun | 取消事件的默认行为。如果cancelable是true的话可以调用这个方法 |
| stopPropagation() | Fun | 取消事件的进一步获取或者冒泡。如果bubbles是true可以调用这个方法。 |
| target | Element | 事件的目标 |
| type | String | 被触发事件的类型 |
| view | AbstractView | 与事件关联的抽象视图，等同于发生事件的window对象。 |

* 在事件处理函数内部，对象this始终等于currentTarget的值，而target则只包含实际的目标。在事件委托中，currentTarget指向父元素(this)，而target指向触发函数的子元素。
* 使用event.type和switch函数可以实现一个函数处理多个事件。[Dome3](dome3.html)
* 要阻止特定事件的默认行为可以用preventDefault()方法，比如a的链接。Cancelable属性为true时，才可以使用这个方法。stopPropagation()方法在bubbles属性为true时可以调用。eventPhase三种状态[dome4](dome4.html)

**IE中的事件对象**

|  |  |  |  |
| --- | --- | --- | --- |
| 属性/方法 | 类型 | 读/写 | 说明 |
| cancelBubble | Bool | 读/写 | 默认值为false，设为true就可以取消事件冒泡(IE无捕获) |
| returnValue | Bool | 读/写 | 默认值为true,设为false就可以取消事件默认行为。 |
| srcElement | Element | 读 | 事件目标（同target） |
| Type | String | 读 | 杯触发事件的类型 |

**事件类型**

1. **UI事件**

* 元素获得焦点：focus,元素失去焦点：blur。与HTML事件一样。

1. **鼠标事件**

* Click, dblclick, mousedown, mouseup, mouseout, mouseover, mousemove。
* 所有元素支持鼠标事件，所有鼠标事件都会冒泡也可以取消。如果取消mousedown或者up，click就不会被触发。
* 鼠标事件中event有clientX，clientY，screenX，screenY。
* 鼠标事件event中有4个修改键，shift、alt、ctrl、meta，按下这些键时，event对应的属性会变成true。[Dome5](dome5.html)
* Mouseover和mouseout事件event提供相关元素属性，relatedTarget。在IE中mouseover触发时，是fromElement，mouseout是toElement。
* 鼠标按钮：DOM中的button属性有3个值，0表示主鼠标按钮，1表示中间的鼠标按钮（滚轮），2表示次鼠标按钮。IE中大有不同，0表示没有按下按钮，1表示按下了主鼠标按钮，2表示次鼠标按钮，3表示同时按下主次鼠标按钮，4表示阿霞了中间（滚轮），5表示同事按下主鼠标按钮和中间按钮，6表示次鼠标按钮和中间，7表示同时按下三个鼠标按钮。

**键盘事件**

* 有3个鼠标事件，keydown，keypress，keyup。虽然所有的元素都支持这三个事件，但只有在用户输入文本时才最常用。首先触发keydown，然后紧跟着keypress，最后触发keyup。

1. 在keydown和keyup事件中，event对象有一个keyCode属性，于特定键对应，数字键和字母对应ASCII码中的数字和小写字母。
2. 字符编码发生keypress事件意味着按下的键会影响到屏幕中的文本的显示。Firefox、chrome、safari的event对象都支持一个charCode属性，属性值为按下键代表那个字符的ASCII码。无charCode则使用keyCode。区分大小写。如果按住shirt则keyCode与keydown和keyup时的keyCode不同，与此时的charCode一样。
3. DOM3中引入了一个新事件，叫textInput。在可编辑区域中输入字符时，会触发这个事件，与keypress不同，区别1：能获得焦点的元素都可以触发keypress事件，单只有可编辑区才能触发textInput事件。区别2：是textInput事件只会在用户按下能够输入实际的字符的键时才会被触发，而keypress在按下能够影响文本显示的键时才触发。[Dome6](dome6.html)

**HTML事件**

Load，unload，abort，error，select，change，submit，reset，resize，scroll，focus，blur

1. **加载事件**

一般使用window.onload，也可以给body加onload事件，图像上也可以触发load事件。

* 可以使用createElement（‘img’）创建，img.src属性设定了之后图片就开始加载。也可以使用image对象实现，image也可以设置src属性，但是不同的是，image对象无法加入到DOM树中，只用来加载图片。Var image = image();IE不提供srcElement，其他浏览器targte，值为ducoment。
* Script标签只能也会触发onload，IE不兼容，与图像不同，script元素的src属性并将该元素添加到文档后才会开始下载js文件。
* Link标签也有onload事件，IE和Opera支持。以便开发人员确定样式表是否加载完毕。与href属性相同，在指定href属性并将link元素添加到文档之后才开始加载。

1. **卸载(unload)事件**

* 用户从一个页面切换到另一个页面，会发生unload事件。利用这个事件最多的情况是清除引用，以避免内存泄漏。这个事件所有浏览器提供target属性，值为ducoment。
* 与load一样，可以给window加也可以给body加。

1. **调整(resize)大小**

* 当浏览器窗口被调整到一个新的高度或宽度时，就会触发resize。IE不提供srcElement，其他浏览器targte，值为ducoment。
* 在IE、safari、chrome、opera会在浏览器窗口变化了1px时就触发resize事件，然后不断重新触发。而firefox只会在用户停止调整窗口大小时才会触发resize事件。

1. **滚动(scroll)事件**

* Scroll事件表示页面中元素的变化。在混杂模式下通过body的scrollTop和scrollLeft来监控这一变化，而标准模式下除safari之外的所以浏览器都通过html来反应。

1. **事件变动**

变动时间在DOM中一部分发生变化时给出提示。不是所有浏览器都支持所有事件。即使不支持所有变动事件，Opera9.5及以前的版本在执行代码是返回true。Safari3和chrome支持除DOMAttrModified之外所有事件，执行代码会错误得返回true。Firefox只支持一部分变动事件，返回false，IE不支持任何变动事件。

1. 节点移除：

* 在使用removeChild和replaceChild时，触发DOMNodeRemoved事件，event.target是被移除的节点，event.relatedNode包含对目标父节点的引用。事件触发时，被移除节点的parentNode仍存在。这个事件会冒泡。
* 如果移除的节点包含子节点，那么其所有的子节点以及这个被移除的节点会相继触发DOMNodeRemovedFromDocument事件，这个事件不会冒泡。Target是子节点。
* 紧跟着父元素触发DOMSubtreeModified事件。Target是父节点。

1. 节点插入

* 使用appendChild、replaceChild、insertBefore方法首先触发DOMNodeInserted事件，target是被插入的节点，event.relatedNode包含对父节点的引用。
* 紧接着在新插入到节点上触发DOMNodeInsertedIntoDocument事件。这个事件不冒泡，必须在插入之前为它添加这个事件处理。Target是插入的节点。
* 最后一个父元素触发DOMSubtreeModified。

1. 特性变化

* 特性值被修改时触发DOMAttrModified事件，target是被修改的那个元素，relateNode是被修改的Attr节点。此外，event还有4个属性，attrName，attrChange：表示变化类型：1表示修改，2表示添加，3表示移除，prevValue修改前的特性（AttrChange为2则返回空），newValue（AttrChange为3则返回空）。
* 之后会触发DOMSubtreeModified事件，target是这个元素。
* 由于2个事件都冒泡，可以在document是添加事件。

1. 文本变化

* 文本节点变化时，先触发DOMCharacterDataModified事件，target是文本节点，还有2个属性，prevValue和newValue。
* 然后在文本节点 触发DOMSubtreeModified。

1. **专有事件**
2. 上下文菜单事件：contextmenu，window右键触发，mac ctrl+单击触发。可以通过event.preventDefault和IE中的event.retrurnValue的值设为false取消。[Dome7](dome7.html)
3. 卸载前事件：beforeunload,之所以有这个事件是为了让开发人员有可能在页面卸载前阻止这一操作。必须给event.returnvalue设置要显示的字符串。[Dome8](dome8.html)
4. 鼠标滚轮：mousewheel和DOMMouseScroll。

* IE6首先实现mousewheel事件，而后除firefox外都实现这个事件。无论向上滑还是向下滑都会触发这个事件，可以在任何元素上触发，最后冒泡到Document（Ie）或window（其他）。
* 还包含一个特殊属性wheelDelta,当用户向前滚动鼠标滚轮时，wheelDelta值为120的倍数，向后滚动时是-120的倍数。要注意opera9,5之前的版本正负值颠倒。需要通过（client.engine.opera && client.engine.opera < 9.5）? –event.wheelDelta:event.wheelDelta;兼容。
* 在火狐中，支持一个名为DOMMouseScroll的类似事件。有关的信息保存在detail中，这个值是3的倍数，夸浏览器兼容时需要\*40倍。不能用DOM方法直接添加。
* iPhone和iPod中，两只手指同时放在屏幕上而且页面因手指移动而滚动时，触发事件。[Dome9](dome9.html)

1. DOMContentLoaded事件，window的load事件要在加载完一切时触发，这个过程可能加载外部资源过多而破费周折，而DOMContentLoaded事件则在形成DOM树之后触发，不理会图片、js文件和css文件是否加载完毕。意味着用户能尽早地与页面进行交互。Target是document。对于不支持该事件的浏览器，建议设置一个0毫秒的超时调用，但无法保证在load事件前运行。
2. 移动设备事件

* 方向变化事件：orientationchange事件。用户改变了设备的查看模式，就触发这个事件，此时的event不包含任何信息，唯一相关信息可以通过window.orientation访问。
* 触摸事件。Touchstart、touchmove、touchend、touchcancel。这些事件都可以冒泡。每个事件event对象都提供常见属性：bubbles、cancelable、view、clientX、clientY、screenX、screenY、detail、altKey、shiftKey、ctrlKey、metaKey。
* 除了以上这些还包括3个追踪触摸的属性，touches,targetTouchs,changeTouches。每个Touch对象包含下列属性，clientX，clientY（视口中的），identifier，pageX，pageY（触摸目标在页面中的X，Y），screenX，screenY（屏幕中的），target。[Dome10](dome10.html)

**内存和性能**

1. 事件委托：利用事件冒泡或者捕获，减少事件绑定的元素。好处：

* Document对象很快就能访问，无需等待load事件就可以为他添加事件处理。
* 页面中设置事件处理程序所需事件少了。所需的DOM引用少了。
* 整个页面占用的内存空间少了。

1. 移除事件处理程序：代码之间的事件链接越多打开网页越慢。

移除事件时，先移除事件处理程序（div.onclick = null; removeEventListener, detachEvent），再清除元素。

**模拟事件**