事件类型、事件目标、事件监听函数、事件对象、时间传播、事件捕获。

事件类型？

主要有3个来源：

1.3级DOM事件，如click、mouseover。

2.H5的新API定义了新事件，比如历史管理（History）、拖放（Drag）、跨域通信（postMessage）、视频以及音频的播放。

3.移动端事件，基于触摸和手势等事件。

传统事件类型？

1. 表单事件  
   控件一般都会触发onchange事件。表单元素在失去和获得焦点的时候会触发focus和blur事件，这两个事件不会冒泡，其他都会，这是仅在表单中focus和blur不冒泡。按钮如果不设置type和事件处理程序的话，是没有任何作用的。提交表单和重置表单都会触发<form>元素的submit和reset事件，这两个事件都有一些默认操作，当然可以取消默认操作。也可以取消事件冒泡。  
   注意：submit和reset事件只能在表单元素<form>上注册和触发，试想，只有整个表单才可以提交，而单个的按钮怎么可能有“提交”，它只是触发submit事件。  
   onchange oninput ontextinput 三者区别
2. <input type="file">的value值为选中的第一个文件的文件路径，为了保护隐私阻止恶意软件猜测文件路径，该值以C:\fakepath\为前缀。如果想获得其他文件的文件名，可以通过HTMLInputElement.files返回一个FileList对象。每一个File都包括name：文件名、lastModified：最后修改时间、size：文件大小、type：文件的MIME类型。

上传文件的时候<form enctype="multipart/form-data">,且<input type="file" multiple accept=".doc" required>。accept 属性并不会验证选中文件的类型. 他只是为开发这提供了一种引导用户做出期望行为的方式而已, 用户还是有办法绕过浏览器的限制.

var form = document.getElementById('form1');

var formdata = new FormData(form);

1. Window事件，document对象的DOMContentLoaded事件和readystatechange事件是替代方案。Window对象的unload和load事件是相反的，但你无法取消默认操作，但是beforeunload事件却可以取消默认操作。除此之外，window对象还有scroll事件（节流）和resize事件（回流）。
2. 鼠标事件、mousedown—mouseup---click。Click是单击鼠标或者键盘触发的，dblclick是双击鼠标左键触发。Mousemove是在元素内部移动时不断触发，消耗很大。Mouseover和mouseout是进入元素时触发，但这两个事件要冒泡。所以HTML5新定义了mouseenter和mouseleave事件，它们不冒泡。一般情况下，不会监听mousemove事件，因为消耗太大，mouseenter足够了。除此之外，单击鼠标右键，在上下文菜单出现之前会触发contextmenu事件，可以取消。  
   鼠标事件的事件对象：  
   clientX：鼠标的视口坐标  
   clinetY：鼠标的视口坐标  
   shiftKey、altKey、ctrlKey、metaKey表示事件发生时是否有键盘辅助键  
   button表示是哪个键被点击，每个浏览器可能都不同，一般0代表左键、1代表中键、2代表右键。
3. 滚轮事件  
   DOM3级是使用wheel事件作为标准版本。浏览器一般使用滚轮唉缩放文档，但是可以通过取消事件来阻止这些默认操作。
4. Focus和blur是不支持冒泡的，但是新定义的focusin和focusout是支持冒泡的。
5. 文本事件   
   textinput input change
6. 键盘事件  
   keydown keyup
7. 注册时间处理函数
8. HTML标签设置事件处理程序
9. <input type="text" onclick="Click()">问题是这种方法存在两个问题。一是时差问题，如果这种这个Click函数在后面声明，会报错。其次，这种方法内容和行为没有分离。
10. 设置JS对象属性为事件处理程序
11. document.getElementById('aaa').ontextinput=function(e){
12. if(e.data==='a') console.log(66);
13. alert("dada");

}

//这种方法也有缺点，只能设置为每种事件设置一个事件处理函数

//而且这种方式也没法控制事件流（冒泡或者捕获）

1. document.getElementById('aaa').ontextinput=null;//取消事件
2. DOM2级时间处理程序
3. document.getElementById('aaa').addEventListener("click",function(){},true)
4. //第三个参数控制事件流，true为捕获，false为冒泡，默认false。
5. //可以设置多个事件处理函数，按照设置的顺序执行。
6. document.getElementById('aaa').removeEventListener("click",,)
7. //传入的参数应该完全相同，也就是说如果给add传入的是匿名函数，是无法移除的
8. 老IE的DOM2级
9. document.getElementById('aaa').attachEvent("onclick",function(){})
10. //IE的是没有控制时间流的，因为IE只支持冒泡
11. document.getElementById('aaa').detachEvent('onclick',,)
12. 取消事件默认操作  
    事件处理程序的return返回值是false，仅当是DOM0级和HTML标签的时候才可以取消默认操作。
13. function canclePrevet(e){
14. var event=e||window.event;
15. if(event.preventDefault){
16. event.preventDefault();
17. }else if(event.returnValue){
18. event.returnValue=false;
19. }
20. return false;
21. }
22. 取消冒泡  
    从概念上讲，在调用完stopPropagation函数之后，就会立即停止对后续节点的访问，但是会执行完绑定到当前节点上的所有事件处理程序；而调用stopImmediatePropagation函数之后，除了所有后续节点，绑定到当前元素上的、当前事件处理程序之后的事件处理程序就不会再执行了。
23. function stopPro(e){
24. var event=e||window.event;
25. if(event.stopPropagation){
26. event.stopPropagation();
27. }else if(event.cancleBubble){
28. event.cancleBubble=true;
29. }
30. }
31. 事件代理  
    事件代理又称事件委托。之所以有事件委托是为了减少DOM操作。  
    假如有100个li，它们的事件处理程序都是一样的。难道你真的循环这100个li添加事件处理函数？这就相当于操作了100次DOM。如果使用事件委托，只需要一次DOM操作。  
    事件委托的原理是事件流的冒泡阶段。  
    利用event.target和event.srcElement(IE)来判断触发事件的源节点。
32. document.getElementsByTagName('ul')[0].onclick=function(event){
33. var event=event||window.event;
34. var target=event.target||event.srcElement;
35. if(target.nodeName.toLowerCase==='li'){
36. console.log(666)
37. }
38. }

除此之外，使用事件代理还不用担心新添加的节点不会有事件。  
 var oBtn = document.getElementById("btn");

var oUl = document.getElementById("ul1");

var aLi = oUl.getElementsByTagName('li');

var num = 4;

//鼠标移入变红，移出变白

for(var i=0; i<aLi.length;i++){

aLi[i].onmouseover = function(){

this.style.background = 'red';

};

aLi[i].onmouseout = function(){

this.style.background = '#fff';

}

}

//添加新节点

oBtn.onclick = function(){

num++;

var oLi = document.createElement('li');

oLi.innerHTML = 111\*num;

oUl.appendChild(oLi);

};

}

添加节点的程序由于在注册事件处理程序之后，所以新添加的节点不会有事件处理程序。var oBtn = document.getElementById("btn");

var oUl = document.getElementById("ul1");

var aLi = oUl.getElementsByTagName('li');

var num = 4;

//事件委托，添加的子元素也有事件

oUl.onmouseover = function(ev){

var ev = ev || window.event;

var target = ev.target || ev.srcElement;

if(target.nodeName.toLowerCase() == 'li'){

target.style.background = "red";

}

};

oUl.onmouseout = function(ev){

var ev = ev || window.event;

var target = ev.target || ev.srcElement;

if(target.nodeName.toLowerCase() == 'li'){

target.style.background = "#fff";

}

};

//添加新节点

oBtn.onclick = function(){

num++;

var oLi = document.createElement('li');

oLi.innerHTML = 111\*num;

oUl.appendChild(oLi);

};

1. 因为每个节点总有click事件可以触发吧，总是要冒泡的。
2. var Add = document.getElementById("add");
3. var Remove = document.getElementById("remove");
4. var Move = document.getElementById("move");
5. var Select = document.getElementById("select");
7. Add.onclick = function(){
8. alert('添加');
9. };
10. Remove.onclick = function(){
11. alert('删除');
12. };
13. Move.onclick = function(){
14. alert('移动');
15. };
16. Select.onclick = function(){
17. alert('选择');
18. }
20. 上述使用事件代理之后还可以减少内存开销。

拖放事件。

拖放，就是在拖放源和拖放目标之间传递数据。它有两个事件集。一个是在拖放源上触发，一个是在拖放目标上触发。