事件流和事件委托

事件流

从页面中接收事件的顺序。

事件流的三个阶段：

1.事件捕获

从document到触发事件的那个节点，即自上而下的去触发事件

2.处于目标阶段

3.事件冒泡

自下而上的去触发事件

事件委托（事件代理）

事件委托就是利用事件冒泡，只对它的父级指定一个事件处理程序，就可以管理某一类型的所有事件。

使用事件委托的原因（优点）

1.减少dom操作，提高性能

为页面上很多个节点添加事件，需要不断的与dom节点进行交互，访问dom的次数越多，引起浏览器重绘与重排的次数也就越多，就会延长整个页面的交互就绪时间。

如果用事件委托，就会将所有的操作放到js程序里面，与dom的操作就只需要交互一次，这样就能大大的减少与dom的交互次数，提高性能。

2.减少函数，节省内存

每个函数都是一个对象，是对象就会占用内存，对象越多，内存占用率就越大，性能就越差。比如100个li，每个li都添加事件，就要占用100个内存空间。

如果用事件委托，那么我们就可以只对它的父级（如果只有一个父级）这一个对象进行操作，这样就只需要一个内存空间就够了，性能就会更好。

3.简化更新

简化dom节点更新时，相应事件的更新。比如添加或删除子标签时不用绑定和解绑事件。

缺点

1. 事件委托基于冒泡，对于不冒泡的事件不支持。
2. 层级过多，冒泡过程中，可能会被某层阻止掉。
3. 理论上委托会导致浏览器频繁调用处理函数，虽然很可能不需要处理。所以建议就近委托，比如在table上代理td，而不是在document上代理td。
4. 把所有事件都用代理就可能会出现事件误判。比如，在document中代理了所有button的click事件，另外的人在引用改js时，可能不知道，造成单击button触发了两个click事件。

实现

实现：点击li，弹出123

HTML页面：

```
<ul id="ul1">
    <li>111</li>
    <li>222</li>
    <li>333</li>
```

```
    <li>444</li>
</ul>
```

传统方式：

```
window.onload = function(){
    var oUl = document.getElementById("ul1");
    var aLi = oUl.getElementsByTagName('li');
    for(var i=0;i<aLi.length;i++){
        aLi[i].onclick = function(){
            alert(123);
        }
    }
}
```

事件委托方式：

```
window.onload = function(){
    var oUl = document.getElementById("ul1");
   oUl.onclick = function(){
        alert(123);
    }
}
```

用父级ul做事件处理，当li被点击时，由于冒泡原理，事件就会冒泡到ul上，因为ul上有点击事件，所以事件就会触发。当然，这当点击ul的时候，也是会触发的。

实现：点击ul不触发，只有点击li才触发

HTML页面：

```
<ul id="ul1">
    <li>111</li>
    <li>222</li>
    <li>333</li>
    <li>444</li>
</ul>
```

Event对象提供了一个target属性，可以返回事件的目标节点。也就是说，target就可以表示当前事件操作的dom节点。

```
    var oUl = document.getElementById("ul1");
    oUl.onclick = function(ev){
        var ev = ev || window.event;
        var target = ev.target || ev.srcElement;
//标准浏览器用ev.target，IE浏览器用event.srcElement
        if(target.nodeName.toLowerCase() == 'li'){          //返回的是大写的，转成小写再做比较
                alert(123);
                alert(target.innerHTML);
        }
    }
```

实现：点击每个li都有不同效果

HTML页面：

```
<div id="box">
        <input type="button" id="add" value="添加" />
```

```
        <input type="button" id="remove" value="删除" />
        <input type="button" id="move" value="移动" />
        <input type="button" id="select" value="选择" />
    </div>
```

传统方式：

```
window.onload = function(){
            var Add = document.getElementById("add");
            var Remove = document.getElementById("remove");
            var Move = document.getElementById("move");
            var Select = document.getElementById("select");

            Add.onclick = function(){
                alert('添加');
            };
            Remove.onclick = function(){
                alert('删除');
            };
            Move.onclick = function(){
                alert('移动');
            };
            Select.onclick = function(){
                alert('选择');
            }
        }
```

事件委托方式：

```
window.onload = function(){
            var oBox = document.getElementById("box");
            oBox.onclick = function (ev) {
                var ev = ev || window.event;
                var target = ev.target || ev.srcElement;
                if(target.nodeName.toLocaleLowerCase() == 'input'){
                    switch(target.id){
                        case 'add' :
                            alert('添加');
                            break;
                        case 'remove' :
                            alert('删除');
                            break;
                        case 'move' :
                            alert('移动');
                            break;
                        case 'select' :
                            alert('选择');
                            break;
                    }
                }
```

```
            }
        }
```

只用一次dom操作就能完成所有的效果。

实现：添加节点

HTML页面：

```
<input type="button" name="" id="btn" value="添加" />
    <ul id="ul1">
        <li>111</li>
        <li>222</li>
        <li>333</li>
        <li>444</li>
    </ul>
```

传统方式：

```
window.onload = function(){
            var oBtn = document.getElementById("btn");
            var oUl = document.getElementById("ul1");
            var aLi = oUl.getElementsByTagName('li');
            var num = 4;

            function mHover () {        //为了让新添加的节点也有事件，将for循环用一个mHover函数包起来
                //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';
                    }
                }
            }
            mHover ();

            //添加新节点
            oBtn.onclick = function(){
                num++;
                var oLi = document.createElement('li');
                oLi.innerHTML = 111*num;
                oUl.appendChild(oLi);
                mHover ();              //给新增节点添加事件
            };
        }
```

事件委托方式：

```
window.onload = function(){
            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);
        };
    }
```

新添加的子元素是带有事件效果的。