事件监听

什么是“事件监听”

- DOM允许我们书写JavaScript代码以让HTML元素对事件作出反应
- 什么是“事件”：用户与网页的交互动作
  - 当用户点击元素时
  - 当鼠标移动到元素上时
  - 当文本框的内容被改变时
  - 当键盘在文本框中被按下时
  - 当网页已加载完毕时
  - ……

什么是“事件监听”

- “监听”，顾名思义，就是让计算机随时能够发现这个事件发生了，从而执行程序员预先编写的一些程序

- 设置事件监听的方法主要有onxxx和addEventListener()两种，二者的区别将在“事件传播”一课中介绍

最简单的设置事件监听的方法

- 最简单的给元素设置事件监听的方法就是设置它们的onxxx属性，像这样：

```
oBox.onclick = function () {
    // 点击盒子时，将执行这里的语句
}
```

常见的鼠标事件监听

事件名	事件描述
onclick	当鼠标单击某个对象
ondblclick	当鼠标双击某个对象
onmousedown	当某个鼠标按键在某个对象上被按下
onmouseup	当某个鼠标按键在某个对象上被松开
onmousemove	当某个鼠标按键在某个对象上被移动
onmouseenter	当鼠标进入某个对象（相似事件onmouseover）
onmouseleave	当鼠标离开某个对象（相似事件onmouseout）

常见的键盘事件监听

事件名	事件描述
onkeypress	当某个键盘的键被按下（系统按钮如箭头键和功能键无法得到识别）
onkeydown	当某个键盘的键被按下（系统按钮可以识别，并且会先于onkeypress发生）
onkeyup	当某个键盘的键被松开

常见的表单事件监听

事件名	事件描述
onchange	当用户改变域的内容
onfocus	当某元素获得焦点（比如tab键或鼠标点击）
onblur	当某元素失去焦点
onsubmit	当表单被提交
onreset	当表单被重置

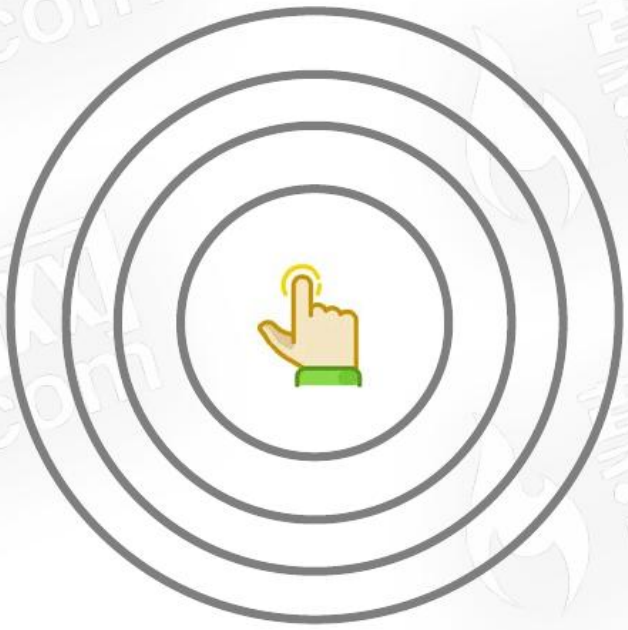
常见的页面事件监听

事件名	事件描述
onload	当页面或图像被完成加载
onunload	当用户退出页面

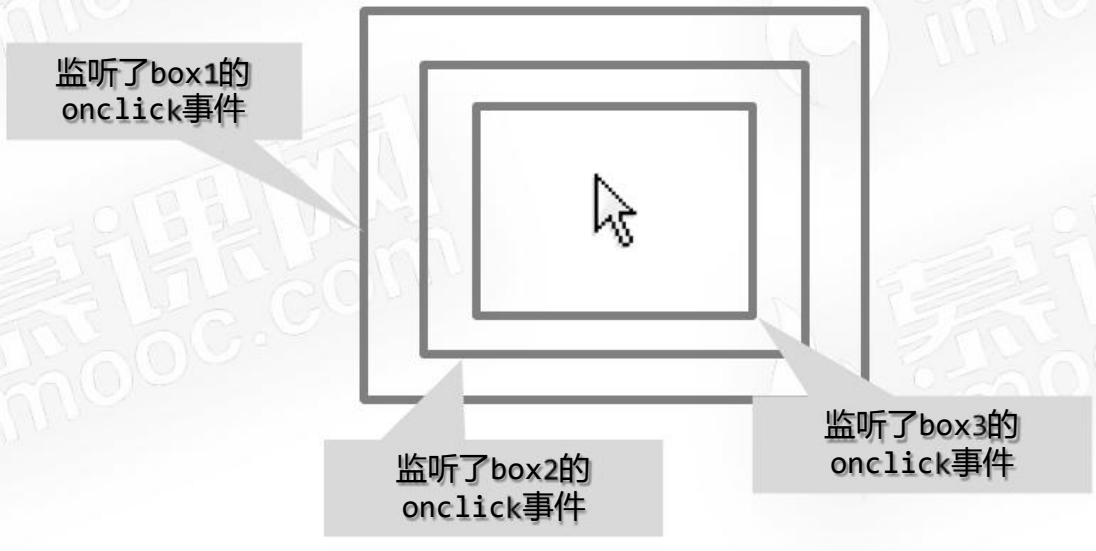
* 更多有关窗口或页面的事件在BOM课程中介绍

事件传播

思考一个问题

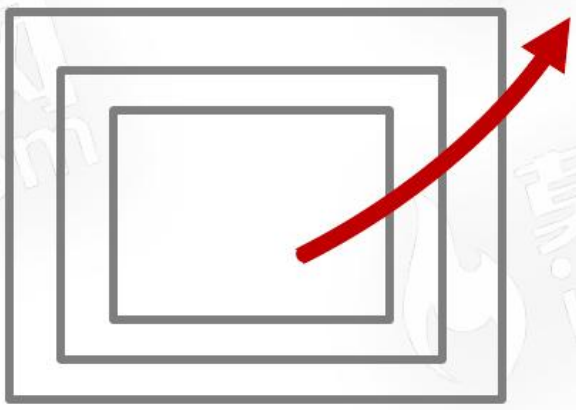


研究：当盒子嵌套时事件监听的执行顺序



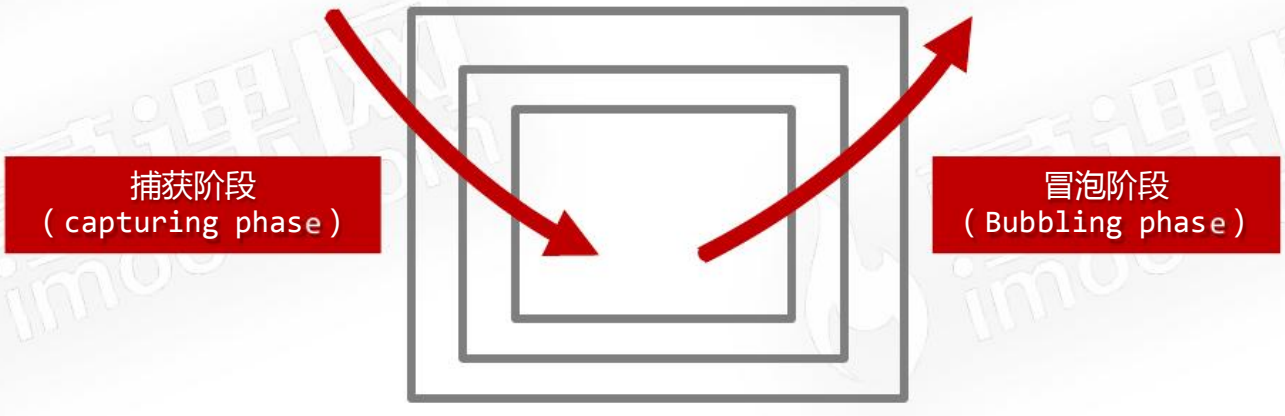
测试小结论（不成熟）

- 通过刚才的测试，我们感觉事件的传播是从内到外的



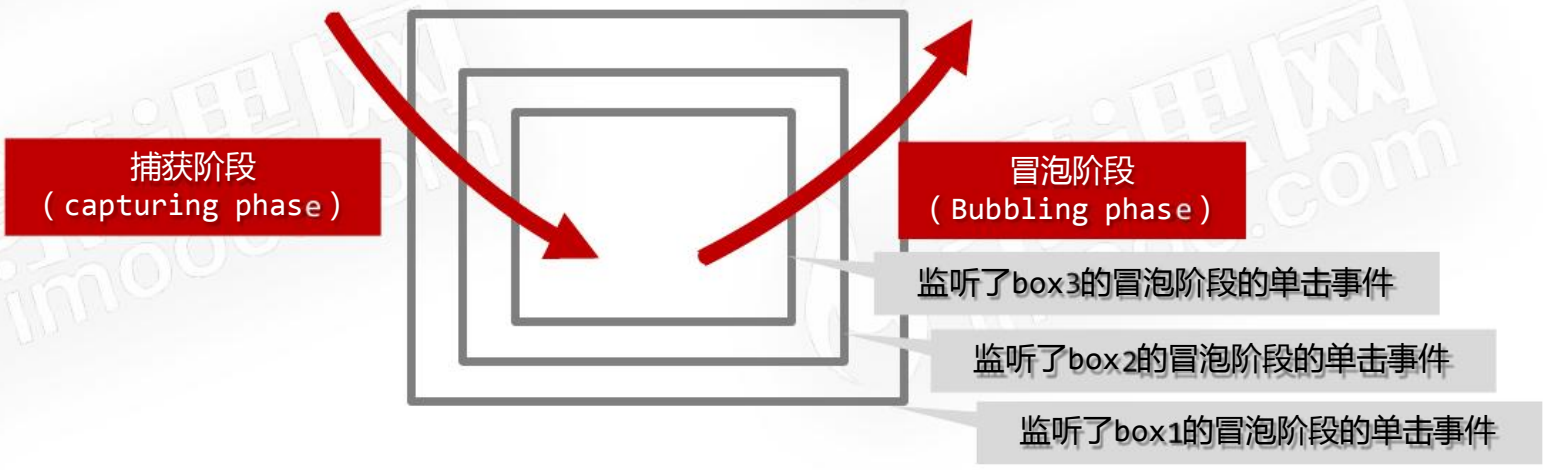
事件的传播

- 实际上，事件的传播是：先从外到内，然后再从内到外



onxxx写法只能监听冒泡阶段

- onxxx这样的写法只能监听冒泡阶段



addEventListener()方法

- DOM0级事件监听：只能监听冒泡阶段

```
oBox.onclick = function () {

};
```

- DOM2级事件监听：

```
oBox.addEventListener('click', function () {
    // 这是事件处理函数
}, true);
```

事件名不加on

true监听捕获阶段
false监听冒泡阶段

注意事项

- 最内部元素不再区分捕获和冒泡阶段，会先执行写在前面的监听，然后执行后写的监听

- 如果给元素设置相同的两个或多个同名事件，则DOM0级写法后面写的会覆盖先写的；而DOM2级会按顺序执行

事件对象

什么是事件对象

- 事件处理函数提供一个形式参数，它是一个对象，封装了本次事件的细节

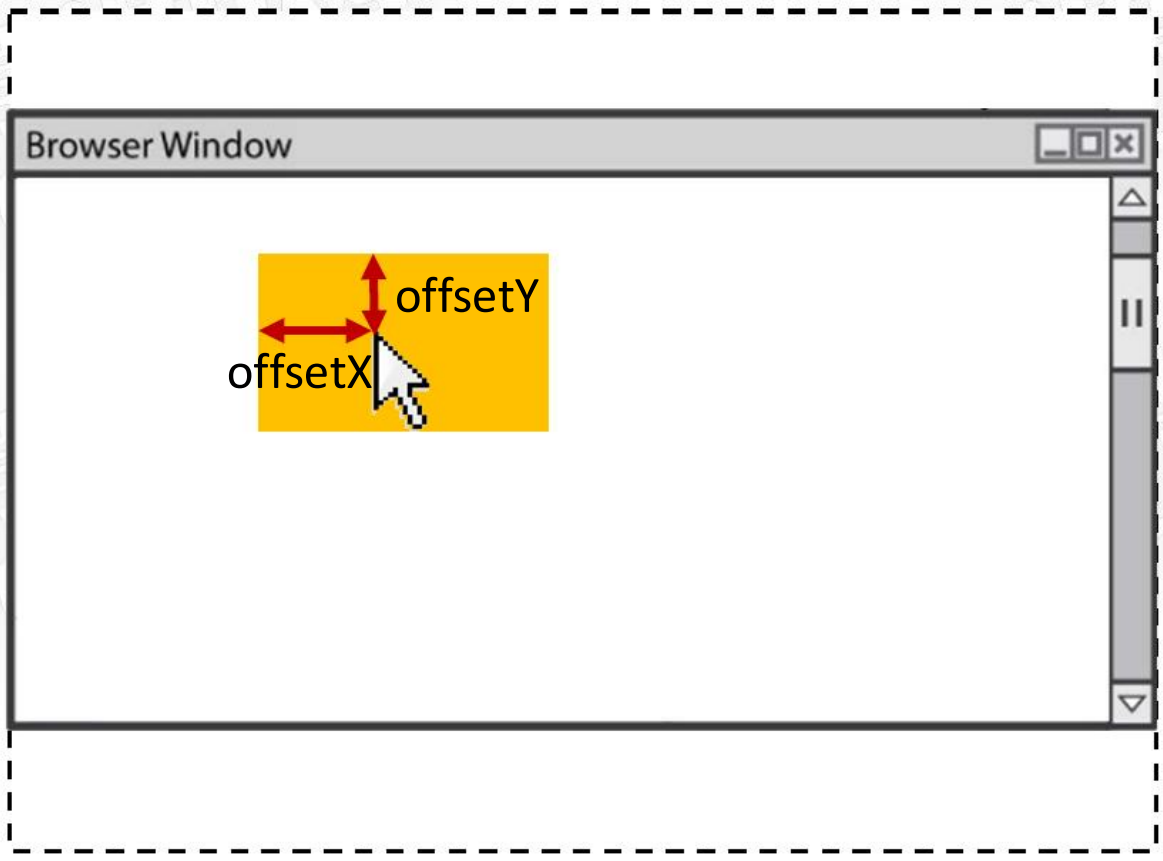
- 这个参数通常用单词event或字母e来表示

```
oBox.onmousemove = function (e) {
    // 对象e就是这次事件的“事件对象”
};
```

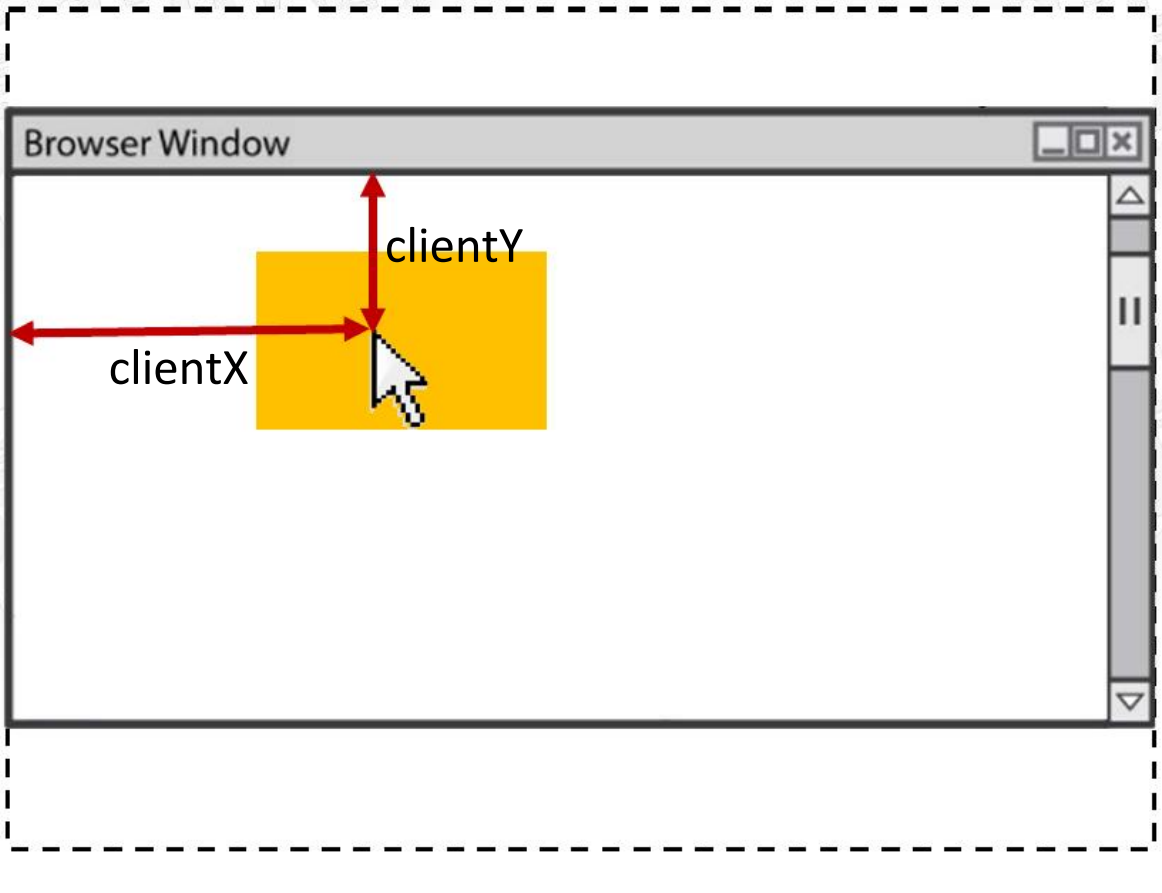
鼠标位置

属性	属性描述
clientX	鼠标指针相对于浏览器的水平坐标
clientY	鼠标指针相对于浏览器的垂直坐标
pageX	鼠标指针相对于整张网页的水平坐标
pageY	鼠标指针相对于整张网页的垂直坐标
offsetX	鼠标指针相对于事件源元素的水平坐标
offsetY	鼠标指针相对于事件源元素的垂直坐标

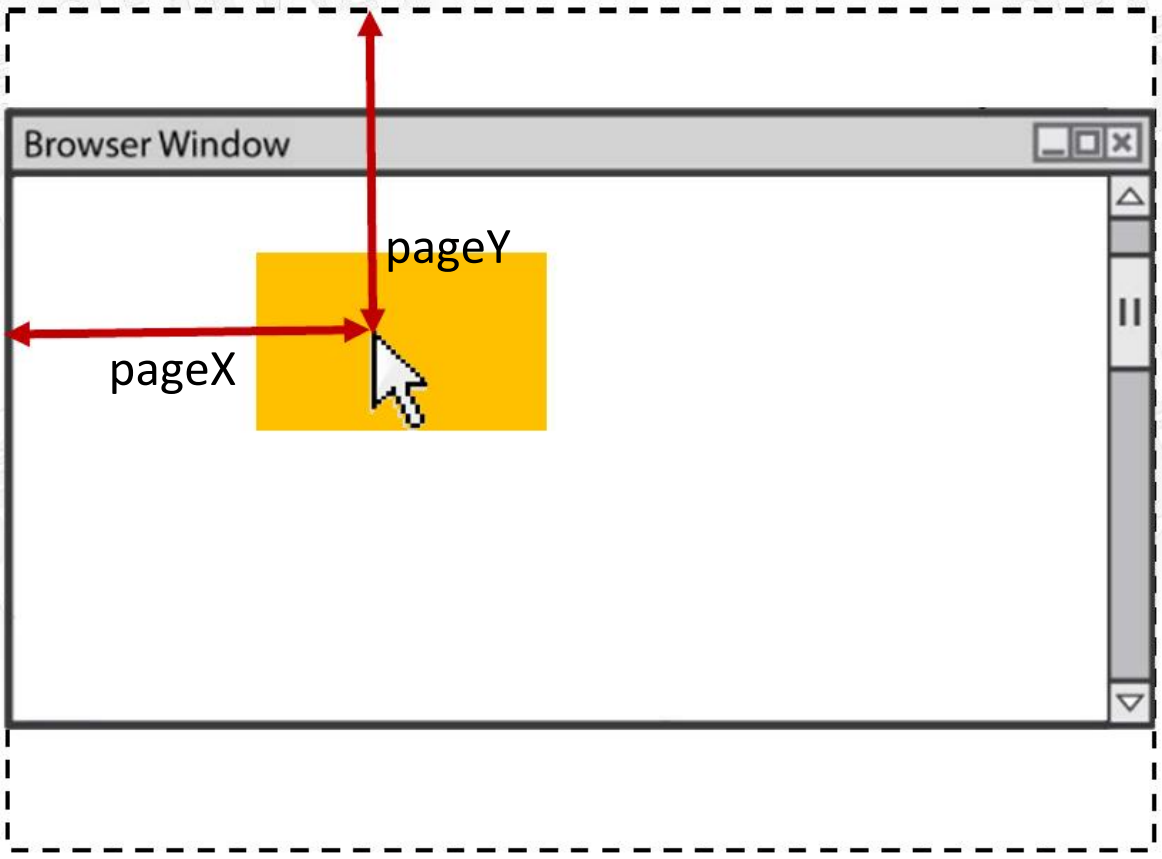
鼠标位置



鼠标位置


Browser Window
clientY
clientX

鼠标位置


Browser Window
pageY
pageX

e.charCode和e.keyCode属性

- e.charCode属性通常用于onkeypress事件中，表示用户输入的字符的“字符码”
- e.keyCode属性通常用于onkeydown事件和onkeyup中，表示用户按下的按键的“键码”

charCode字符码

字符	字符码
数字0 ~ 数字9	48 ~ 57
大写字母A ~ Z	65 ~ 90
小写字母a ~ z	97 ~ 122

keyCode键码

按键	键码
数字0 ~ 数字9	48 ~ 57（同charCode键码完全相同）
字母不分大小a ~ z	65 ~ 90（同charCode键码的大写字母A ~ Z，而keyCode不分大小写，一律为65 ~ 90）
四个方向键← ↑ → ↓	37、38、39、40
回车键	13
空格键	32

小案例

- 制作一个特效：按方向键可以控制页面上的盒子移动

e.preventDefault()方法

- e.preventDefault()方法用来阻止事件产生的“默认动作”
- 一些特殊的业务需求，需要阻止事件的“默认动作”

小案例1

- 制作一个文本框，只能让用户在其中输入小写字母和数字，其他字符输入没有效果

小案例2

- 制作鼠标滚轮事件：当鼠标在盒子中向下滚动时，数字加1；反之，数字减1

- 鼠标滚轮事件是onmousewheel，它的事件对象e提供deltaY属性表示鼠标滚动方向，向下滚动时返回正值，向上滚动时返回负值

e. stopPropagation()方法

- e.stopPropagation()方法用来阻止事件继续传播

- 在一些场合，非常有必要切断事件继续传播，否则会造成页面特效显示出bug

小案例

- 制作一个弹出层：点击按钮显示弹出层，点击网页任意地方，弹出层关闭

事件委托

批量添加事件监听

- 题目：页面上有一个无序列表<ul>，它内部共有20个<li>元素，请批量给它们添加点击事件监听，实现效果：点击哪个<li>元素，哪个<li>元素就变红

批量添加事件监听的性能问题

- 每一个事件监听注册都会消耗一定的系统内存，而批量添加事件会导致监听数量太多，内存消耗会非常大

- 实际上，每个<li>的事件处理函数都是不同的函数，这些函数本身也会占用内存

新增元素动态绑定事件

- 题目：页面上有一个无序列表<ul>，它内部没有<li>元素，请制作一个按钮，点击这个按钮就能增加一个<li>元素。并且要求每个增加的<li>元素也要有点击事件监听，实现效果点击哪个<li>元素，哪个<li>元素就变红

动态绑定事件的问题

- 新增元素必须分别添加事件监听，不能自动获得事件监听
- 大量事件监听、大量事件处理函数都会产生大量消耗内存

事件委托

- 利用事件冒泡机制，将后代元素事件委托给祖先元素

```
<ul id="list">
    <li>列表项</li>
    <li>列表项</li>
    <li>列表项</li>
    <li>列表项</li>
    <li>列表项</li>
</ul>
```

监听onclick事件

不管点击任何一个<li>元素，事件都会通过事件冒泡传给祖先元素

e.target和e.currentTarget属性

- 事件委托通常需要结合使用e.target属性

属性	属性描述
target	触发此事件的最早元素，即“事件源元素”
currentTarget	事件处理程序附加到的元素

事件委托的使用场景

- 当有大量类似元素需要批量添加事件监听时，使用事件委托可以减少内存开销

- 当有动态元素节点上树时，使用事件委托可以让新上树的元素具有事件监听

使用事件委托时需要注意的事项

- onmouseenter和onmouseover都表示“鼠标进入”，它们有什么区别呢？

  答：onmouseenter不冒泡，onmouseover冒泡。

- 使用事件委托时要注意：不能委托不冒泡的事件给祖先元素

使用事件委托时需要注意的事项

- 最内层元素不能再有额外的内层元素了，比如：

```
<ul id="list">
    <li><span>我是span</span>列表项</li>
    <li><span>我是span</span>列表项</li>
    <li><span>我是span</span>列表项</li>
</ul>
```