# JavaScript进阶

第1天课堂笔记

目录

[JavaScript进阶 1](#_Toc17246)

[目录 2](#_Toc26383)

[一、我们学过的页面特效模型总结 3](#_Toc6145)

[二、事件深入 4](#_Toc5323)

[2.1 事件流 4](#_Toc1147)

[2.2 DOM0级事件绑定 5](#_Toc1136)

[2.3 DOM2级事件绑定 6](#_Toc21492)

[2.4 低版本IE的事件绑定 8](#_Toc494)

[2.5 经典轮子 9](#_Toc16988)

[三、事件对象 10](#_Toc562)

[3.1 通用事件对象属性和方法 10](#_Toc24252)

[3.2 clientX、clientY、screenX、screenY 11](#_Toc22185)

[3.3 IE中的event 11](#_Toc24883)

# 一、我们学过的页面特效模型总结

页面特效，必须熟记各种模型，模型掌握了，就无敌于天下了。

“设计师、视觉设计师、产品经理，尽管提需求！我将嗷嗷做出来！”

到今天为止，我们学过的页面效果模型：

□ 显示/隐藏模型 ：　10天内免登陆

□ 更改html属性模型 ： className的交换、current类的交换、图片的src、选项卡

□ 普通运动模型

□ 无缝连续滚动模型

□ 传统轮播模型

□ 间歇模型

□ 呼吸轮播模型

□ 异形滚动模型

□ 百叶窗模型

□ 手风琴模型

这两天我们还要学习的模型：

□ 拼图轮播模型

□ 拖拽模型

□ 鼠标滚轮效果模型

□ 浏览器窗口尺寸模型

□ 动画出场出场模型

□ 滚滚屏模型

□ 元素offsetLeft相关模型

今后CSS3还将学习：

□ 3D

□ 炫酷效果

# 二、事件深入

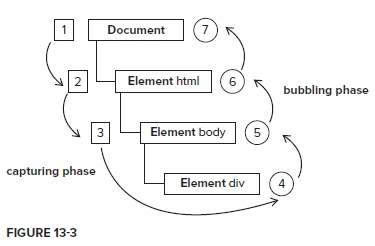
## 2.1 事件流

当你单击了某个元素，单击事件不仅仅发生在这个元素上，你也单击了它的父元素、父元素的父元素、……它的祖先元素，甚至单击了整个页面。



“事件流”描述的是页面上各个元素接收事件的顺序。

我们为了描述事件的传播顺序，特意人为规定了两个阶段：捕获阶段capture pahse、冒泡阶段bubbling phase。点击页面上的一个元素，事件在哪个阶段触发，这取决于添加事件监听的方法。



## 2.2 DOM0级事件绑定

DOM分为级别，DOM0级、1级、2级、3级，是不同的标准，标准一直在升级。

我们之前学习的

|  |
| --- |
| 1. oDiv.onclick = function(){ 2. } |

这种注册监听的写法，就是DOM0级的事件绑定方法。就是把onclick当做属性添加给了oDiv元素。

通过实验，我们发现，**这种事件添加方法，只能监听冒泡过程**。事件的捕获阶段，没有监听成功。

**这里需要注意，在IE9、Chrome里面，事件会冒泡到window对象，而IE6、7、8仅仅冒泡到document对象。**

**另外，用这种方法绑定的监听，this指的是触发这个事件的元素，没有任何的浏览器兼容问题。**

**DOM0级还有一种写法，直接将监听写在标签里面，工作中不许用：**

|  |
| --- |
| 1. <div id="box1" onclick="alert('我是box1');"> 2. <div id="box2" onclick="alert('我是box2');"> 3. <div id="box3" onclick="alert('我是box3');"> 5. </div> 6. </div> 7. </div> |

写在HTML标签里面代码耦合性很强，三层没有完全分离，工作中不用的。

这种写法，也是监听冒泡阶段。

**用DOM0级添加事件监听，同一个元素不能有两个同样事件监听，比如：**

|  |
| --- |
| 1. box1.onclick = function(){ 2. alert("哈哈"); 3. } 4. box1.onclick = function(){ 5. alert("嘻嘻"); 6. } |

以JS代码后出现的为准，它会覆盖先写的函数。所以点击box1弹出嘻嘻。

## 2.3 DOM2级事件绑定

DOM1级规范中，没有对事件进行改动。

DOM2级做了新的规范，不用on\*\*\*来绑定监听了，而是使用一个方法

|  |
| --- |
| 1. addEventListener(); |

add添加，Event事件，Listener监听

它接受三个参数：什么事件、函数、是否监听捕获阶段。

|  |
| --- |
| 1. oBox.addEventListener("click",function(){ 3. },false); |

第1个参数：事件名不用写on， click、mouseover 、mouseout

第2个参数：函数可以是匿名函数，也可以是有名函数

第3个参数：布尔值，true表示监听捕获、false表示监听冒泡阶段

比如：

|  |
| --- |
| 1. box1.addEventListener("click", function(){ 2. alert("box1被点击，捕获阶段"); 3. }, true); |

第三个参数是true，表示监听box1的捕获阶段的单击事件。

比如小题目：

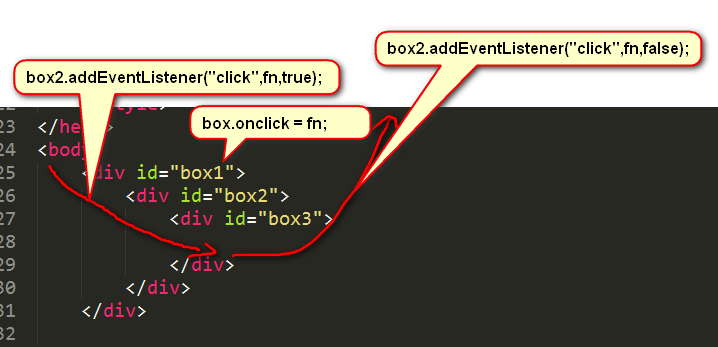
|  |
| --- |
| 1. <div id="box1"> 2. <div id="box2"> 3. <div id="box3"> 5. </div> 6. </div> 7. </div> |

对应的JS：

|  |
| --- |
| 1. box2.addEventListener("click", function(){ 2. alert(1); 3. }, true); //捕获阶段 4. box1.onclick = function(){ 5. alert(2); //冒泡阶段 6. } 7. box2.addEventListener("click",function(){ 8. alert(3); 9. },false); //冒泡阶段 |

弹出顺序1、3、2.

解释图：



我们来看看this是什么？就是触发函数的这个元素。

addEventListener可以重复添加相同事件名的事件：

|  |
| --- |
| 1. box1.addEventListener("click", function(){ 2. alert("嘻嘻"); 3. }, false); 4. box1.addEventListener("click", function(){ 5. alert("哈哈"); 6. }, false); |

我们给box1的绑定了两个事件冒泡阶段的监听，不会覆盖，两个监听的函数都会执行，按照代码执行顺序。

弹出嘻嘻，然后哈哈。

## 2.4 低版本IE的事件绑定

IE永远是个奇葩，但是现在20%的用户在使用IE8，我们祝愿他们健康幸福。

IE6、7、8不支持addEventListener()方法，支持

|  |
| --- |
| 1. oDiv.attachEvent(“onclick”,函数); |

没有第三个参数，也就是说，不能选择监听捕获、冒泡。只能监听冒泡。

|  |
| --- |
| 1. box1.**attachEvent**("onclick", function(){ 2. alert("box1"); 3. }); |

第一个参数，必须写on，和addEventListener()不一样；

第二个参数，就是事件处理函数

没有第三个参数，只能监听冒泡。所以和on\*\*\*写法一样。

低版本IE的事件监听attachEvent：**事件处理函数里面的this，不是触发事件的这个元素，而是window对象！**

|  |
| --- |
| 1. box1.attachEvent("onclick", function(){ 2. alert(this === window); //true 3. }); |

同一个事件名的多个监听，会反着执行：

|  |
| --- |
| 1. box1.attachEvent("onclick", function(){ 2. alert(1); 3. }); 4. box1.attachEvent("onclick", function(){ 5. alert(2); 6. }); 7. box1.attachEvent("onclick", function(){ 8. alert(3); 9. }); |

弹出3、2、1。

## 2.5 经典轮子

经典轮子：得到计算后样式fetchComputedStyle、运动框架animate、addEvent绑定监听、getElementsByClassName得到元素。

|  |
| --- |
| 1. function addEvent(obj,eventtype,fn){ 2. if(obj.addEventListener){ 3. obj.addEventListener(eventtype,fn,false); 4. }else if(obj.attachEvent){ 5. obj.attachEvent("on" + eventtype,function(){ 6. fn.call(obj); 7. }); 8. }else{ 9. obj["on" + eventtype] = fn; 10. } 11. } |

上面我们学习的就是一些理论知识，总结一下：

事件流永远是先下后上，先捕获后冒泡。但是不同的添加监听的方式，决定了能够监听那一部分。

● DOM0级的方式，只能监听冒泡阶段。不能有同名的事件，会覆盖。this是触发事件的这个元素。高版本浏览器会冒泡到window，低版本浏览器冒泡到document。

● DOM2级的方法，addEventListener()，可以自由设置冒泡、捕获。第三个参数是use capture ，true就是捕获，false就是冒泡。事件名不加on，可以有同名事件，会顺序执行，不覆盖。this是触发事件的这个元素。会冒泡到window。

● IE6、7、8是自己的方法，attachEvent()，只能监听冒泡阶段。没有第三个参数。事件名写on。可以有同名事件，会反着执行（事件栈）。this很蛋疼，是window而不是触发事件的这个元素。事件仅仅冒泡到document。

面试意义大于实际意义。

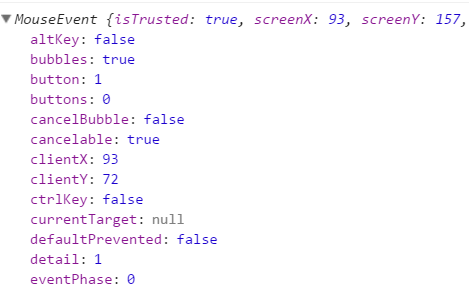
jQuery中用的就是我们的DOM2级，也有轮子。jQuery事件同名不会被覆盖。

# 三、事件对象

任何的事件处理函数，我们的浏览器、JS引擎会默认往里面传一个实参，就是事件对象。

通常用形参event来接收：

|  |
| --- |
| 1. oDiv.onclick = function(event){ 2. console.log(event); 3. } |



所有这次事件的细节，都被封装到了这个event对象里面。

比如clientX、clientY就是这次点击的位置。

## 3.1 通用事件对象属性和方法

● event.type 返回事件的类型，没有on， 比如”click”

● event.target 返回你点击的最小的那个元素，即使这个元素身上没有监听，也是返回它

● event.currentTarget 返回自己，this一定和event.currentTarget是一个元素，都是自己

● event.bubbles 返回一个布尔值，表示这个事件是否冒泡

|  |
| --- |
| 1. oDiv.onmouseover = function(event){ 2. console.log(event.bubbles); 3. } |

比如onmouesover的event.bubbles就是true;

但是onmouseenter的 event.bubbles就是false;

简称onmouseover冒泡，onmouseenter不冒泡。

onmouseover、onmouseenter IE6、7、8、9、10全面兼容！反而是chrome30之前不兼容。所以现在可以看做全线兼容，就嗷嗷用就完事儿了。

● stopPropagation() 停止传播事件流

|  |
| --- |
| 1. event.stopPropagation(); |

● preventDefault() 阻止默认事件

|  |
| --- |
| 1. event.preventDefault(); |

## 3.2 clientX、clientY、screenX、screenY

event.clientX

event.clientY

event.screenX

event.screenY

全线兼容，表示事件触发这一瞬间的鼠标位置。

clientX表示鼠标的位置，距离浏览器窗口左边边的距离

clientY表示鼠标的位置，距离浏览器窗口上边边的距离

screenX表示鼠标的位置，距离屏幕左边边的距离

screenY表示鼠标的位置，距离屏幕上边边的距离

## 3.3 IE中的event

IE浏览器的event对象是window对象的属性，而不是事件的实参。

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |