## 面试题总结

### 事件注册(浅析)

1. **最原始的注册**

<p id="para" title="cssrain demo!" onclick="test()" >test</p>

<script>

function test(){

alert("test");

}

</script>

1. **JavaScript要跟HTML结构实现分离**

**Js性能：**

**1.尽量减少DOM的访问和标记，减少document.getElementByTagName（）之类的方法的使用，把这类方法的存储变量中。**

**2.合并脚本，样式文件，图片文件等，以便减少加载时的请求数量。**

**3.压缩脚本，减少脚本的空格和注释。**

<p id="para" title="cssrain demo!">test</p>

<script>

function test(){

alert("test");

}

function pig(){

alert("pig");

}

window.onload = function(){

document.getElementById("para").onclick = test;

document.getElementById("para").onclick = pig;

}

</script>

只能输出第二个函数。

**3.这时候我们需要用到attachEvent方法**

<p id="para" title="cssrain demo!">test</p>

<script>

function test(){

alert("test");

}

function pig(){

alert("pig");

}

window.onload = function(){

document.getElementById("para").attachEvent("onclick",test);

document.getElementById("para").attachEvent("onclick",pig);

}

</script>

只支持ie,放到firefox中执行后,发现并不能正常运行。

**4.firefox中注册事件使用,同时兼容ie**

<p id="para" title="cssrain demo!">test</p>

<script>

function test(){

alert("test");

}

function pig(){

alert("pig");

}

window.onload = function(){

var element = document.getElementById("para");

if(element.addEventListener){ // firefox , w3c

element.addEventListener("click",test,false);

element.addEventListener("click",pig,false);

} else { // ie

element.attachEvent("onclick",test);

element.attachEvent("onclick",pig);

}

}

</script>

二、attachEvent和addEventListener的区别

**Mozilla中：**addEventListener的使用方式：   
  
target.addEventListener(type, listener, useCapture);   
  
target： 文档节点、document、window 或 XMLHttpRequest。   
type： 字符串，事件名称，不含“on”，比如“click”、“mouseover”、“keydown”等。   
listener ：实现了 EventListener 接口或者是 JavaScript 中的函数。   
useCapture ：是否使用捕捉，一般用 false 。例如：document.getElementById("testText").addEventListener("keydown", function (event) { alert(event.keyCode); }, false);

移除事件：removeEventListener(event,function,capture/bubble);

**IE中：**target.attachEvent(type, listener);   
target： 文档节点、document、window 或 XMLHttpRequest。   
type： 字符串，事件名称，含“on”，比如“onclick”、“onmouseover”、“onkeydown”等。   
listener ：实现了 EventListener 接口或者是 JavaScript 中的函数。 例如：document.getElementById("txt").attachEvent("onclick",function(event) {alert(event.keyCode);});

移除事件：detachEvent(event,function);

优先级：

attachEvent方法，为某一事件附加其它的处理事件。（不支持Mozilla系列）   
addEventListener方法 用于 Mozilla系列  
举例: document.getElementById("btn").onclick = method1;   
document.getElementById("btn").onclick = method2;   
document.getElementById("btn").onclick = method3;如果这样写,那么将会只有medhot3被执行

写成这样：   
var btn1Obj=document.getElementById("btn1"); //object.attachEvent(event,function);   
btn1Obj.attachEvent("onclick",method1);   
btn1Obj.attachEvent("onclick",method2);   
btn1Obj.attachEvent("onclick",method3);

执行顺序为method3->method2->method1   
如果是Mozilla系列，并不支持该方法，需要用到addEventListener

var btn1Obj = document.getElementById("btn1");   
//element.addEventListener(type,listener,useCapture);   
btn1Obj.addEventListener("click",method1,false);   
btn1Obj.addEventListener("click",method2,false);   
btn1Obj.addEventListener("click",method3,false);执行顺序为method3->method2->method1

网上：method1->method2->method3

三、事件流

1.定义

从页面中接收事件的顺序。也就是说当一个事件产生时，这个事件的传播过程，就是事件流。

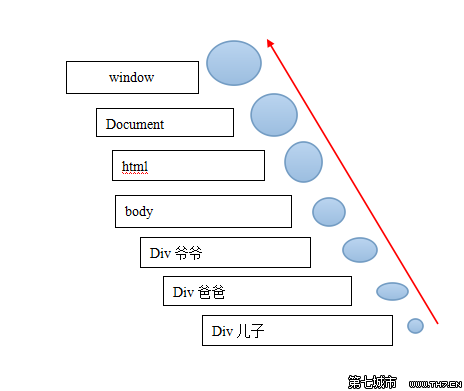
1. **IE的事件流**

IE中的事件流叫事件冒泡；事件冒泡：事件开始时由最具体的元素接收，然后逐级向上传播到较为不具体的节点（文档）。对于html来说，就是当一个元素产 生了一个事件，它会把这个事件传递给它的父元素，父元素接收到了之后，还要继续传递给它的上一级元素，就这样一直传播到document对象

1. 实际结果（见eventStream.html）



这就是事件冒泡，它会把你这个click事件，一级一级的向上传递，如果相应的元素也绑定click事件处理程序（这里强调是click事件，如果你是给绑定了其它事件，那没用），那么它的这个事件处理程序也会执行，也就产生了上面的结果了；

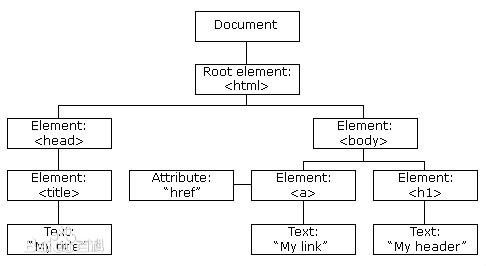


形象的就是跟水里的鱼吐泡泡似的，慢慢的向上传递。

1. **事件捕获(见eventCapture.Html)**

事件捕获是网景（Netscape）提出来的，事件捕获是不太具体的元素应该更早接受到事件，而最具体的节点应该最后接收到事件。他们的用意是在事件到达 目标之前就捕获它；也就是跟冒泡的过程正好相反，以html的click事件为例，document对象（DOM级规范要求从document开始传播， 但是现在的[浏览器](http://cpro.baidu.com/cpro/ui/uijs.php?adclass=0&app_id=0&c=news&cf=1001&ch=0&di=128&fv=20&is_app=0&jk=d4e7be4bbca373e2&k=%E4%AF%C0%C0%C6%F7&k0=%E4%AF%C0%C0%C6%F7&kdi0=0&luki=4&mcpm=0&n=10&p=baidu&q=smileking_cpr&rb=0&rs=1&seller_id=1&sid=e273a3bc4bbee7d4&ssp2=1&stid=9&t=tpclicked3_hc&td=1682280&tu=u1682280&u=http://www.th7.cn/web/js/201504/98405.shtml&urlid=0" \t "http://www.th7.cn/web/js/201504/_blank)是从window对象开始的）最先接收到click事件的然后事件沿着DOM树依次向下传播，一直传播到事件的实际目标。

**DOM（文档对象模型(Document Object Model)）**



**DOM0事件**

<input onclick="alert(event)" />

<form><input name="hehe" value="hehe" /><input onclick="alert(hehe.value)">  
所有DOM0的事件作用域被扩展了哇;  
</form>

　DOM0的事件具有极好的跨浏览器优势, 会以最快的速度绑定, 如果你通过DOM2绑定要等到JS运行, DOM0不用, 因为DOM0是写在元素上面的哇;

**DOM2事件**

DOM2事件的冒泡和捕获

### 四、自定义事件

1. **事件**

很多DOM对象都有原生的事件支持，向div就有click、mouseover等事件，事件机制可以为类的设计带来很大的灵活性，相信.net程序员深 有体会。随着web技术发展，使用JavaScript自定义对象愈发频繁，让自己创建的对象也有事件机制，通过事件对外通信，能够极大提高开发效率。

1. **示例（见customize1.HTML）**

1.没有共同性。如果再定义一个控件，还得写一套类似的结构处理

2.事件绑定有排斥性。只能绑定了一个close事件处理程序，绑定新的会覆盖之前绑定

3.封装不够完善。如果用户不知道有个 close\_handler的句柄，就没有办法绑定该事件，只能去查源代码

**3.最后结果（见customize2.HTML）**

学习原型**<http://www.cnblogs.com/wangfupeng1988/p/3979290.html>**

1. angular.js的一些主要指令及优缺点

<http://www.runoob.com/angularjs/angularjs-directives.html>