**Javascript面试题**

**JavaScript中如何检测一个变量是一个String类型？请写出函数实现**

typeof(obj) === "string"

typeof obj === "string"

obj.constructor === String

**请用js去除字符串空格？**

**方法一：使用replace正则匹配的方法**

去除所有空格: str = str.replace(/\s\*/g,"");

去除两头空格: str = str.replace(/^\s\*|\s\*$/g,"");

去除左空格： str = str.replace( /^\s\*/, “”);

去除右空格： str = str.replace(/(\s\*$)/g, "");

str为要去除空格的字符串，实例如下：

var str = " 23 23 ";

var str2 = str.replace(/\s\*/g,"");

console.log(str2); // 2323

**方法二：使用str.trim()方法**

str.trim()局限性：无法去除中间的空格，实例如下：

var str = " xiao ming ";

var str2 = str.trim();

console.log(str2); //xiao ming

同理，str.trimLeft()，str.trimRight()分别用于去除字符串左右空格。

**方法三：使用jquery,$.trim(str)方法**

$.trim(str)局限性：无法去除中间的空格，实例如下：

var str = " xiao ming ";

var str2 = $.trim(str)

console.log(str2); // xiao ming

**你如何获取浏览器URL中查询字符串中的参数？**

测试地址为：<http://www.runoob.com/jquery/misc-trim.html?channelid=12333&name=xiaoming&age=23>

实例如下：

function showWindowHref(){

var sHref = window.location.href;

var args = sHref.split('?');

if(args[0] == sHref){

return "";

}

var arr = args[1].split('&');

var obj = {};

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

var arg = arr[i].split('=');

obj[arg[0]] = arg[1];

}

return obj;

}

var href = showWindowHref(); // obj

console.log(href['name']); // xiaoming

***js*字符串操作函数**

　　我这里只是列举了常用的字符串函数，具体使用方法，请[参考网址](http://www.w3school.com.cn/jsref/jsref_obj_string.asp)。

* concat() – 将两个或多个字符的文本组合起来，返回一个新的字符串。
* indexOf() – 返回字符串中一个子串第一处出现的索引。如果没有匹配项，返回 -1 。
* charAt() – 返回指定位置的字符。
* lastIndexOf() – 返回字符串中一个子串最后一处出现的索引，如果没有匹配项，返回 -1 。
* match() – 检查一个字符串是否匹配一个正则表达式。
* substr() 函数 -- 返回从string的startPos位置，长度为length的字符串
* substring() – 返回字符串的一个子串。传入参数是起始位置和结束位置。
* slice() – 提取字符串的一部分，并返回一个新字符串。
* replace() – 用来查找匹配一个正则表达式的字符串，然后使用新字符串代替匹配的字符串。
* search() – 执行一个正则表达式匹配查找。如果查找成功，返回字符串中匹配的索引值。否则返回 -1 。
* split() – 通过将字符串划分成子串，将一个字符串做成一个字符串数组。
* length – 返回字符串的长度，所谓字符串的长度是指其包含的字符的个数。
* toLowerCase() – 将整个字符串转成小写字母。
* toUpperCase() – 将整个字符串转成大写字母。

**怎样添加、移除、移动、复制、创建和查找节点？**

 1）创建新节点

　　createDocumentFragment() //创建一个DOM片段  
　　createElement() //创建一个具体的元素  
　　createTextNode() //创建一个文本节点

2）添加、移除、替换、插入  
　　appendChild() //添加  
　　removeChild() //移除  
　　replaceChild() //替换  
　　insertBefore() //插入

3）查找  
　　getElementsByTagName() //通过标签名称  
　　getElementsByName() //通过元素的Name属性的值  
　　getElementById() //通过元素Id，唯一性

**写出3个使用this的典型应用**

（1）、在html元素事件属性中使用，如：

<input type=”button” onclick=”showInfo(this);” value=”点击一下”/>

（2）、构造函数

function Animal(name, color) {

　　this.name = name;

　　this.color = color;

}

（3）、input点击，获取值

<input type="button" id="text" value="点击一下" />

<script type="text/javascript">

var btn = document.getElementById("text");

btn.onclick = function() {

alert(this.value); //此处的this是按钮元素

}

</script>

(4)、apply()/call()求数组最值

var numbers = [5, 458 , 120 , -215 ];

var maxInNumbers = Math.max.apply(this, numbers);

console.log(maxInNumbers); // 458

var maxInNumbers = Math.max.call(this,5, 458 , 120 , -215);

console.log(maxInNumbers); // 458

**比较typeof与instanceof？**

相同点：JavaScript 中 typeof 和 instanceof 常用来判断一个变量是否为空，或者是什么类型的。

typeof的定义和用法：返回值是一个字符串，用来说明变量的数据类型。

细节：

(1)、typeof 一般只能返回如下几个结果：number,boolean,string,function,object,undefined。

(2)、typeof 来获取一个变量是否存在，如 if(typeof a!="undefined"){alert("ok")}，而不要去使用 if(a) 因为如果 a 不存在（未声明）则会出错。

(3)、对于 Array,Null 等特殊对象使用 typeof 一律返回 object，这正是 typeof 的局限性。

Instanceof定义和用法：instanceof 用于判断一个变量是否属于某个对象的实例。

实例演示：

a instanceof b?alert("true"):alert("false"); //a是b的实例？真:假

var a = new Array();

alert(a instanceof Array); // true

alert(a instanceof Object)  // true

如上，会返回 true，同时 alert(a instanceof Object) 也会返回 true;这是因为 Array 是 object 的子类。

function test(){};

var a = new test();

alert(a instanceof test) // true

细节：

(1)、如下，得到的结果为‘N’,这里的 instanceof 测试的 object 是指 js 语法中的 object，不是指 dom 模型对象。

if (window instanceof Object){ alert('Y')} else { alert('N');} // 'N'

**如何理解闭包？**

1、定义和用法：当一个函数的返回值是另外一个函数，而返回的那个函数如果调用了其父函数内部的其它变量，如果返回的这个函数在外部被执行，就产生了闭包。

2、表现形式：使函数外部能够调用函数内部定义的变量。

3、实例如下：

(1)、根据作用域链的规则，底层作用域没有声明的变量，会向上一级找，找到就返回，没找到就一直找，直到window的变量，没有就返回undefined。这里明显count 是函数内部的flag2 的那个count 。

var count=10; //全局作用域 标记为flag1

function add(){

var count=0; //函数全局作用域 标记为flag2

return function(){

count+=1; //函数的内部作用域

alert(count);

}

}

var s = add()

s();//输出1

s();//输出2

4、变量的作用域

要理解闭包，首先必须理解Javascript特殊的变量作用域。

变量的作用域分类：全局变量和局部变量。

特点：

1、函数内部可以读取函数外部的全局变量；在函数外部无法读取函数内的局部变量。

2、函数内部声明变量的时候，一定要使用var命令。如果不用的话，你实际上声明了一个全局变量！

 5、使用闭包的注意点

1）滥用闭包，会造成内存泄漏：由于闭包会使得函数中的变量都被保存在内存中，内存消耗很大，所以不能滥用闭包，否则会造成网页的性能问题，在IE中可能导致内存泄露。解决方法是，在退出函数之前，将不使用的局部变量全部删除。

2）会改变父函数内部变量的值。所以，如果你把父函数当作对象（object）使用，把闭包当作它的公用方法（Public Method），把内部变量当作它的私有属性（private value），这时一定要小心，不要随便改变父函数内部变量的值。

**什么是跨域？跨域请求资源的方法有哪些？**

**1、什么是跨域？**

由于浏览器同源策略，凡是发送请求url的协议、域名、端口三者之间任意一与当前页面地址不同即为跨域。存在跨域的情况：

* 网络协议不同，如http协议访问https协议。
* 端口不同，如80端口访问8080端口。
* 域名不同，如qianduanblog.com访问baidu.com。
* 子域名不同，如abc.qianduanblog.com访问def.qianduanblog.com。
* 域名和域名对应ip,如www.a.com访问20.205.28.90.

**2、跨域请求资源的方法：**

**(1)、porxy代理**

定义和用法：proxy代理用于将请求发送给后台服务器，通过服务器来发送请求，然后将请求的结果传递给前端。

实现方法：通过nginx代理；

注意点：1、如果你代理的是https协议的请求，那么你的proxy首先需要信任该证书（尤其是自定义证书）或者忽略证书检查，否则你的请求无法成功。

**(2)、CORS 【Cross-Origin Resource Sharing】**

定义和用法：是现代浏览器支持跨域资源请求的一种最常用的方式。

使用方法：一般需要后端人员在处理请求数据的时候，添加允许跨域的相关操作。如下：

res.writeHead(200, {

"Content-Type": "text/html; charset=UTF-8",

"Access-Control-Allow-Origin":'http://localhost',

'Access-Control-Allow-Methods': 'GET, POST, OPTIONS',

'Access-Control-Allow-Headers': 'X-Requested-With, Content-Type'

});

**(3)、jsonp**

定义和用法：通过动态插入一个script标签。浏览器对script的资源引用没有同源限制，同时资源加载到页面后会立即执行（没有阻塞的情况下）。

特点：通过情况下，通过动态创建script来读取他域的动态资源，获取的数据一般为json格式。

实例如下：

<script>

function testjsonp(data) {

console.log(data.name); // 获取返回的结果

}

</script>

<script>

var \_script = document.createElement('script');

\_script.type = "text/javascript";

\_script.src = "http://localhost:8888/jsonp?callback=testjsonp";

document.head.appendChild(\_script);

</script>

缺点：

　　1、这种方式无法发送post请求（这里）

　　2、另外要确定jsonp的请求是否失败并不容易，大多数框架的实现都是结合超时时间来判定。

**谈谈垃圾回收机制方式及内存管理**

**回收机制方式**

1、定义和用法：垃圾回收机制(GC:Garbage Collection),执行环境负责管理代码执行过程中使用的内存。

2、原理：垃圾收集器会定期（周期性）找出那些不在继续使用的变量，然后释放其内存。但是这个过程不是实时的，因为其开销比较大，所以垃圾回收器会按照固定的时间间隔周期性的执行。

3、实例如下：

function fn1() {

var obj = {name: 'hanzichi', age: 10};

}

function fn2() {

var obj = {name:'hanzichi', age: 10};

return obj;

}

var a = fn1();

var b = fn2();

fn1中定义的obj为局部变量，而当调用结束后，出了fn1的环境，那么该块内存会被js引擎中的垃圾回收器自动释放；在fn2被调用的过程中，返回的对象被全局变量b所指向，所以该块内存并不会被释放。

 4、垃圾回收策略：标记清除(较为常用)和引用计数。

**标记清除：**

　　定义和用法：当变量进入环境时，将变量标记"进入环境"，当变量离开环境时，标记为："离开环境"。某一个时刻，垃圾回收器会过滤掉环境中的变量，以及被环境变量引用的变量，剩下的就是被视为准备回收的变量。

　　到目前为止，IE、Firefox、Opera、Chrome、Safari的js实现使用的都是标记清除的垃圾回收策略或类似的策略，只不过垃圾收集的时间间隔互不相同。

**引用计数：**

　　定义和用法：引用计数是跟踪记录每个值被引用的次数。

　　基本原理：就是变量的引用次数，被引用一次则加1，当这个引用计数为0时，被视为准备回收的对象。

**内存管理**

1、什么时候触发垃圾回收？

垃圾回收器周期性运行，如果分配的内存非常多，那么回收工作也会很艰巨，确定垃圾回收时间间隔就变成了一个值得思考的问题。

IE6的垃圾回收是根据内存分配量运行的，当环境中的变量，对象，字符串达到一定数量时触发垃圾回收。垃圾回收器一直处于工作状态，严重影响浏览器性能。

IE7中，垃圾回收器会根据内存分配量与程序占用内存的比例进行动态调整，开始回收工作。

2、合理的GC方案：(1)、遍历所有可访问的对象; (2)、回收已不可访问的对象。

3、GC缺陷：(1)、停止响应其他操作；

4、GC优化策略：(1)、分代回收（Generation GC）;(2)、增量GC

**开发过程中遇到的内存泄露情况，如何解决的？**

1、定义和用法：

内存泄露是指一块被分配的内存既不能使用，又不能回收，直到浏览器进程结束。C#和Java等语言采用了自动垃圾回收方法管理内存，几乎不会发生内存泄露。我们知道，浏览器中也是采用自动垃圾回收方法管理内存，但由于浏览器垃圾回收方法有bug，会产生内存泄露。

2、内存泄露的几种情况:

(1)、当页面中元素被移除或替换时，若元素绑定的事件仍没被移除，在IE中不会作出恰当处理，此时要先手工移除事件，不然会存在内存泄露。

实例如下:

<div id="myDiv">

<input type="button" value="Click me" id="myBtn">

</div>

<script type="text/javascript">

var btn = document.getElementById("myBtn");

btn.onclick = function(){

document.getElementById("myDiv").innerHTML = "Processing...";

}

</script>

解决方法如下：

<div id="myDiv">

<input type="button" value="Click me" id="myBtn">

</div>

<script type="text/javascript">

var btn = document.getElementById("myBtn");

btn.onclick = function(){

btn.onclick = null;

document.getElementById("myDiv").innerHTML = "Processing...";

}

</script>

(2)、由于是函数内定义函数，并且内部函数--事件回调的引用外暴了，形成了闭包。闭包可以维持函数内局部变量，使其得不到释放。

实例如下：

function bindEvent(){

var obj=document.createElement("XXX");

obj.onclick=function(){

//Even if it's a empty function

}

}

解决方法如下：

function bindEvent(){

var obj=document.createElement("XXX");

obj.onclick=function(){

//Even if it's a empty function

}

obj=null;

}

**javascript面向对象中继承实现？**

面向对象的基本特征有：封闭、继承、多态。

在JavaScript中实现继承的方法：

1. 原型链（prototype chaining）

2. call()/apply()

3. 混合方式(prototype和call()/apply()结合)

4. 对象冒充

继承的方法如下：

1、prototype原型链方式：

1. function teacher(name){
2. this.name = name;
3. }
4. teacher.prototype.sayName = function(){
5. console.log("name is "+this.name);
6. }
7. var teacher1 = new teacher("xiaoming");
8. teacher1.sayName();
10. function student(name){
11. this.name = name;
12. }
13. student.prototype = new teacher()
14. var student1 = new student("xiaolan");
15. student1.sayName();
16. // name is xiaoming
17. // name is xiaolan

2、call()/apply()方法

function teacher(name,age){

this.name = name;

this.age = age;

this.sayhi = function(){

alert('name:'+name+", age:"+age);

}

}

function student(){

var args = arguments;

teacher.call(this,args[0],args[1]);

// teacher.apply(this,arguments);

}

var teacher1 = new teacher('xiaoming',23);

teacher1.sayhi();

var student1 = new student('xiaolan',12);

student1.sayhi();

// alert: name:xiaoming, age:23

// alert: name:xiaolan, age:12

3、混合方法【prototype,call/apply】

function teacher(name,age){

this.name = name;

this.age = age;

}

teacher.prototype.sayName = function(){

console.log('name:'+this.name);

}

teacher.prototype.sayAge = function(){

console.log('age:'+this.age);

}

function student(){

var args = arguments;

teacher.call(this,args[0],args[1]);

}

student.prototype = new teacher();

var student1 = new student('xiaolin',23);

student1.sayName();

student1.sayAge();

// name:xiaolin

// age:23

4、对象冒充

function Person(name,age){

this.name = name;

this.age = age;

this.show = function(){

console.log(this.name+", "+this.age);

}

}

function Student(name,age){

this.student = Person; //将Person类的构造函数赋值给this.student

this.student(name,age); //js中实际上是通过对象冒充来实现继承的

delete this.student; //移除对Person的引用

}

var s = new Student("小明",17);

s.show();

var p = new Person("小花",18);

p.show();

// 小明, 17

// 小花, 18

**javascript相关程序计算题**

**1、判断一个字符串中出现次数最多的字符，统计这个次数**

var str = 'asdfssaaasasasasaa';

var json = {};

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

if(!json[str.charAt(i)]){

json[str.charAt(i)] = 1;

}else{

json[str.charAt(i)]++;

}

};

var iMax = 0;

var iIndex = '';

for(var i in json){

if(json[i]>iMax){

iMax = json[i];

iIndex = i;

}

}

console.log('出现次数最多的是:'+iIndex+'出现'+iMax+'次');

结果如下：出现次数最多的是:a出现9次

**JavaScript 数组(Array)对象**

**1、Array相关的属性和方法**

这里只是做了相关的列举，具体的使用方法，[请参考网址](http://www.w3school.com.cn/jsref/jsref_obj_array.asp)。

**Array 对象属性**

constructor 返回对创建此对象的数组函数的引用。

length 设置或返回数组中元素的数目。

prototype 使您有能力向对象添加属性和方法。

**Array 对象方法**

concat() 连接两个或更多的数组，并返回结果。

join() 把数组的所有元素放入一个字符串。元素通过指定的分隔符进行分隔。

pop() 删除并返回数组的最后一个元素。

shift() 删除并返回数组的第一个元素

push() 向数组的末尾添加一个或更多元素，并返回新的长度。

unshift() 向数组的开头添加一个或更多元素，并返回新的长度。

reverse() 颠倒数组中元素的顺序。

slice() 从某个已有的数组返回选定的元素

sort() 对数组的元素进行排序

splice() 删除元素，并向数组添加新元素。

toSource() 返回该对象的源代码。

toString() 把数组转换为字符串，并返回结果。

toLocaleString() 把数组转换为本地数组，并返回结果。

valueOf() 返回数组对象的原始值

**2、编写一个方法 去掉一个数组的重复元素**

方法一：

var arr = [0,2,3,4,4,0,2];

var obj = {};

var tmp = [];

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

if( !obj[arr[i]] ){

obj[arr[i]] = 1;

tmp.push(arr[i]);

}

}

console.log(tmp);

结果如下： [0, 2, 3, 4]

 方法二：

var arr = [2,3,4,4,5,2,3,6],

arr2 = [];

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

if(arr2.indexOf(arr[i]) < 0){

arr2.push(arr[i]);

}

}

console.log(arr2);

结果为：[2, 3, 4, 5, 6]

 方法三：

var arr = [2,3,4,4,5,2,3,6];  
var arr2 = arr.filter(function(element,index,self){  
return self.indexOf(element) === index;  
});  
console.log(arr2);

结果为：[2, 3, 4, 5, 6]

[回到顶部](http://www.cnblogs.com/#_labelTop)

**jquery相关**

**1、 jQuery 库中的 $() 是什么？**

　　$() 函数是 jQuery() 函数的别称。$() 函数用于将任何对象包裹成 jQuery 对象，接着你就被允许调用定义在 jQuery 对象上的多个不同方法。你可以将一个选择器字符串传入 $() 函数，它会返回一个包含所有匹配的 DOM 元素数组的 jQuery 对象。

**2、如何找到所有 HTML select 标签的选中项？**

$('[name=selectname] :selected')

**3、$(this) 和 this 关键字在 jQuery 中有何不同？**

$(this) 返回一个 jQuery 对象，你可以对它调用多个 jQuery 方法，比如用 text() 获取文本，用val() 获取值等等。

而 this 代表当前元素，它是 JavaScript 关键词中的一个，表示上下文中的当前 DOM 元素。你不能对它调用 jQuery 方法，直到它被 $() 函数包裹，例如 $(this)。

**4、jquery怎么移除标签onclick属性？**

获得a标签的onclick属性: $("a").attr("onclick")

删除onclick属性：$("a").removeAttr("onclick");

设置onclick属性：$("a").attr("onclick","test();");

**5、jquery中addClass,removeClass,toggleClass的使用。**

$(selector).addClass(class)：为每个匹配的元素添加指定的类名

$(selector).removeClass(class)：从所有匹配的元素中删除全部或者指定的类，删除class中某个值；

$(selector).toggleClass(class)：如果存在（不存在）就删除（添加）一个类

$(selector).removeAttr(class);删除class这个属性；

**6、JQuery有几种选择器?**

(1)、基本选择器：#id，class,element,\*;

(2)、层次选择器：parent > child，prev + next ，prev ~ siblings

(3)、基本[过滤器](http://zhidao.baidu.com/search?word=%E8%BF%87%E6%BB%A4%E5%99%A8&fr=qb_search_exp&ie=utf8" \t "_blank)选择器：:first，:last ，:not ，:even ，:odd ，:eq ，:gt ，:lt

(4)、内容[过滤器](http://zhidao.baidu.com/search?word=%E8%BF%87%E6%BB%A4%E5%99%A8&fr=qb_search_exp&ie=utf8" \t "_blank)选择器： :contains ，:empty ，:has ，:parent

(5)、可见性[过滤器](http://zhidao.baidu.com/search?word=%E8%BF%87%E6%BB%A4%E5%99%A8&fr=qb_search_exp&ie=utf8" \t "_blank)选择器：:hidden ，:visible

(6)、属性过滤器选择器：[attribute] ，[attribute=value] ，[attribute!=value] ，[attribute^=value] ，[attribute$=value] ，[attribute\*=value]

(7)、子元素过滤器选择器：:nth-child ，:first-child ，:last-child ，:only-child

(8)、表单选择器： :input ，:text ，:password ，:radio ，:checkbox ，:submit 等；

(9)、表单过滤器选择器：:enabled ，:disabled ，:checked ，:selected

**7、jQuery中的Delegate()函数有什么作用？**

   delegate()会在以下两个情况下使用到：

 1、如果你有一个父元素，需要给其下的子元素添加事件，这时你可以使用delegate()了，代码如下：

$("ul").delegate("li", "click", function(){ $(this).hide(); });

 2、当元素在当前页面中不可用时，可以使用delegate()

**8、$(document).ready()方法和window.onload有什么区别？**

 (1)、window.onload方法是在网页中所有的元素(包括元素的所有关联文件)完全加载到浏览器后才执行的。

 (2)、$(document).ready() 方法可以在DOM载入就绪时就对其进行操纵，并调用执行绑定的函数。

**9、如何用jQuery禁用浏览器的前进后退按钮？**

实现代码如下：

<script type="text/javascript" language="javascript">

　　$(document).ready(function() {

　　　　window.history.forward(1);

　　　　//OR window.history.forward(-1);

　　});

</script>

**10、 jquery中$.get()提交和$.post()提交有区别吗？**

相同点：都是异步请求的方式来获取服务端的数据；

异同点：

1、请求方式不同：$.get() 方法使用GET方法来进行异步请求的。$.post() 方法使用POST方法来进行异步请求的。

2、参数传递方式不同：get请求会将参数跟在URL后进行传递，而POST请求则是作为HTTP消息的实体内容发送给Web服务器的，这种传递是对用户不可见的。

3、数据传输大小不同：get方式传输的数据大小不能超过2KB 而POST要大的多

4、安全问题： GET 方式请求的数据会被浏览器缓存起来，因此有安全问题。

**11、写出一个简单的$.ajax()的请求方式？**

$.ajax({

url:'http://www.baidu.com',

type:'POST',

data:data,

cache:true,

headers:{},

beforeSend：function(){},

success:function(){},

error:function(){},

complete:function(){}

});

**12、jQuery的事件委托方法bind 、live、delegate、on之间有什么区别？**

**(1)、bind 【jQuery 1.3之前】**

定义和用法：主要用于给选择到的元素上绑定特定事件类型的监听函数；

语法：bind(type,[data],function(eventObject))；

特点：

　　(1)、适用于页面元素静态绑定。只能给调用它的时候已经存在的元素绑定事件，不能给未来新增的元素绑定事件。

　　(2)、当页面加载完的时候，你才可以进行bind()，所以可能产生效率问题。

实例如下：$( "#members li a" ).bind( "click", function( e ) {} );

**(2)、live 【jQuery 1.3之后】**

定义和用法：主要用于给选择到的元素上绑定特定事件类型的监听函数；

语法：live(type, [data], fn);

特点：

　　(1)、live方法并没有将监听器绑定到自己(this)身上，而是绑定到了this.context上了。

　　(2)、live正是利用了事件委托机制来完成事件的监听处理，把节点的处理委托给了document，新添加的元素不必再绑定一次监听器。

　　(3)、使用live（）方法但却只能放在直接选择的元素后面，不能在层级比较深，连缀的DOM遍历方法后面使用，即$(“ul”").live...可以，但$("body").find("ul").live...不行；

实例如下：$( document ).on( "click", "#members li a", function( e ) {} );

**(3)、delegate 【jQuery 1.4.2中引入】**

定义和用法：将监听事件绑定在就近的父级元素上

语法：delegate(selector,type,[data],fn)

特点：

　　(1)、选择就近的父级元素，因为事件可以更快的冒泡上去，能够在第一时间进行处理。

　　(2)、更精确的小范围使用事件代理，性能优于.live()。可以用在动态添加的元素上。

实例如下：

$("#info\_table").delegate("td","click",function(){/\*显示更多信息\*/});

$("table").find("#info").delegate("td","click",function(){/\*显示更多信息\*/});

**(4)、on 【1.7版本整合了之前的三种方式的新事件绑定机制】**

定义和用法：将监听事件绑定到指定元素上。

语法：on(type,[selector],[data],fn)

实例如下：$("#info\_table").on("click","td",function(){/\*显示更多信息\*/});参数的位置写法与delegate不一样。

说明：on方法是当前JQuery推荐使用的事件绑定方法，附加只运行一次就删除函数的方法是one()。

 总结：.bind(), .live(), .delegate(),.on()分别对应的相反事件为：.unbind(),.die(), .undelegate(),.off()