**一.介绍**   
插件编写的目的是给已经有的一系列方法或函数做一个封装，以便在其他地方重复使用，方便后期维护。   
JQuery除了提供一个简单、有效的方式进行管理元素以及脚本，它还还提供了例外一种机制：即给核心模块增加自己的方法和额外的功能。通过这种机制，Jquery允许我们自己创建属于我们自己的插件，提高我们在开发过程中的效率。   
  
1.1JQuery的插件分为3中类型：   
（1）封装对象方法的插件（也就是对象级别的开发）   
这种类型的插件是我们今天需要讲的插件。   
（2）封装全局函数的插件（类级别开发）   
指可以把独立的函数添加到JQuery命名空间之下。   
添加一个全局函数，我们只需如下定义：   
jQuery.foo = function() {   
alert('This is a test. This is only a test.');   
};   
当然 你也可以添加多个全局函数：

.代码如下:

jQuery.foo = function() {   
alert('This is a test. This is only a test.');   
};   
jQuery.bar = function(param) {   
alert('This function takes a parameter, which is "' + param + '".');   
};

调用时和一个函数的一样的:jQuery.foo();jQuery.bar();或者$.foo();$.bar('bar');   
（3）选择器插件   
1.2编写JQuery插件需要注意的地方：   
（1）插件的推荐命名方法为：jquery.[插件名].js   
（2）所有的对象方法都应当附加到JQuery.fn对象上面，而所有的全局函数都应当附加到JQuery对象本身上。   
（3）在插件内部，this指向的是当前通过选择器获取的JQuery对象，而不像一般方法那样，内部的this指向的是DOM元素。   
（4）可以通过this.each 来遍历所有的元素   
（5）所有方法或函数插件，都应当以分号结尾，否则压缩的时候可能会出现问题。为了更加保险写，可以在插件头部添加一个分号（；），以免他们的不规范代码给插件带来 影响。   
（6）插件应该返回一个JQuery对象，以便保证插件的可链式操作。   
（7）避免在插件内部使用$作为JQuery对象的别名，而应当使用完整的JQuery来表示。这样可以避免冲突。   
1.3JQuery插件的机制   
JQuery提供了2个用于扩展JQuery功能的方法。即：   
①jQuery.fn.extend()   
②jQuery.extend()   
第一个就是我们前面说插件类型的第一个中情况，第二个就是指后面的2种情况。   
jQuery.extend()在插件中友一个很重要的功能是扩展已经有的object的对象。   
比如：   
var newSrc=$.extend(dest,src1,src2,src3...)   
它的含义是将src1,src2,src3...合并到dest中,返回值为合并后的dest,由此可以看出该方法合并。   
示例：   
var result=$.extend({},{name:"Tom",age:21},{name:"Jerry",sex:"Boy"})   
得到的结果是：   
result={name:"Jerry",age:21,sex:"Boy"}   
详细的 可以查看：[jQuery.extend 函数详解](http://www.jb51.net/article/29591.htm) 里面对这个方法有很好的讲解。  
  
官网：[JQuery.extend()与JQuery.fn.extend()](http://asimilia.wordpress.com/2008/12/17/jquery-extend-confusion/)  
**使用命名空间**虽然在jQuery命名空间中，我们禁止使用了大量的javaScript函数名和变量名。但是仍然不可避免某些函数或变量名将于其他jQuery插件冲突，因此我们习惯将一些方法封装到另一个自定义的命名空间。

.代码如下:

jQuery.myPlugin = {   
foo:function() {   
alert('This is a test. This is only a test.');   
},   
bar:function(param) {   
alert('This function takes a parameter, which is "' + param + '".');   
}   
};

采用命名空间的函数仍然是全局函数，调用时采用的方法：

.代码如下:

$.myPlugin.foo();   
$.myPlugin.bar('baz');

**二.第一个Jquery插件**   
如果需要写一个JQuery插件，你需要在$.fn对象后面增加一个属性名，这个属性名 其实 就是你的插件名。它的一般框架如下：

.代码如下:

(function( $ ) {   
$.fn.myPlugin = function() {   
// 这里开始写功能需求   
};   
})( jQuery );

现在我们需要写的插件的功能很简单，就是把一个对象给慢慢的隐藏起来。就是使用fadeOut()这个方法。   
OK，我们打开 VS 2012.新建一个jscript文件，并且命名为：MyPlugin.js，并在里面添加如下代码：

.代码如下:

(function ($) {   
$.fn.myPlugin = function () {   
this.fadeOut('normal');   
};   
})(jQuery);

**怎么用呢？很简单。**   
新建一个html页面，把jquery文件以及刚刚我们MyPlugin.js文件导入到本次页面中。如下：

.代码如下:

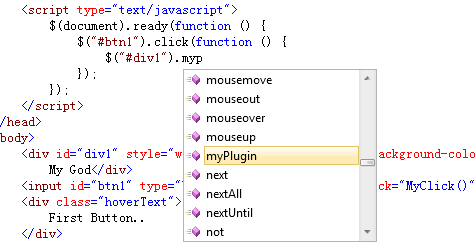
<script src="Scripts/jquery-1.4.1.js" type="text/javascript"></script>   
<script src="MyPlugin.js" type="text/javascript"></script>   
js代码：   
<script type="text/javascript">   
$(document).ready(function () {   
$("#btn1").click(function () {   
$("#div1").myPlugin();   
});   
});   
</script>

HTML代码：

.代码如下:

<div id="div1" style="width: 400px; height: 30px; background-color: Gray;">   
My God</div>   
<input id="btn1" type="button" value="button" onclick="MyClick()" />

好的，你现在点击网页上面的按钮的时候，div就会缓缓的隐藏。。。因为我们设定的是normal嘛，里面也可以设定一些值之类的。   
很兴奋的是，既然这个有智能提示，如下图： 



吧！   
**三.插件运用在多个元素控件中。**   
3.1 运用在多个元素控件中   
在上面编写JQuery插件需要注意的地方的第四点，写道 如果要遍历 则可以用this.each方法。$("ID").each 可以遍历jquery对象、数组以及集合。   
OK。知道了这样子，那么我们的新的代码如下：

.代码如下:

(function ($) {   
$.fn.hoverElement = function () {   
this.each(function () {   
$(this).hover(   
function () {   
$(this).addClass("Add");   
},   
function () {   
$(this).removeClass("Remove");   
}   
);   
})   
}   
})(jQuery);

上面主要用到.each()方法进行遍历。代码很简单，就是把当前的对象的背景色css样式在“Add”和“Remove”直接切换。   
  
HTML的代码是：

.代码如下:

<div class="hoverText">   
First Button..   
</div>   
<div class="hoverText">   
Second Button..   
</div>   
<div class="hoverText">   
Third Button..   
</div>

JS 代码：

.代码如下:

<script type="text/javascript">   
$(document).ready(function () {   
$(".hoverText").hoverElement();   
});   
</script>

很简单，不解释。   
3.2 链式操作   
链式操作？都听过吧。。。比如下面这一句话：   
$("#div1").css("color", "red").addClass("Add").animate({"width":"100px"},1000);   
就是能在当前的元素后面通过“.” 来实施更多的操作。这个动作特别的潇洒。   
那我们要怎么才能实现这种效果呢？很简单，我能只需要把对象回去就可以了。请注意上面的第六点：插件应该返回一个JQuery对象，以便保证插件的可链式操作。   
我们依旧看刚刚的例子：

.代码如下:

(function ($) {   
$.fn.hoverElement = function () {   
return this.each(function () {   
$(this).hover(   
function () {   
$(this).addClass("Add");   
},   
function () {   
$(this).removeClass("Remove");   
}   
);   
})   
}   
})(jQuery);

代码都一样，唯一区别的是：this.each(function () { 这个前面加了一个 return。这样就实现了我们的链式操作。   
然后 你这样：

.代码如下:

$(document).ready(function () {   
$(".hoverText").hoverElement().css("color","yellow");   
});

能看到 文字已经变成了yellow色的了。   
  
**四 自定义自己的插件**对于一个商业插件来说，自定义插件的样式是必不可少的。我们可以通过我们自己输入不同的样式，来改变开发者的默认样式。比如说最常见的 width、height、url、color等等。要是没有这些自定义的东西，那开发者开发的插件的利用价值就大大的减小了。   
OK，下面的这个实例的意思是 当我们hover一个对象的时候，它能改变它的text、background、foreground三个属性，也就是文本、背景色、前景色。用户可以设定他自己想设定的值，而不是固定死的。当然，如果用户没有设置，我们会给他一个默认值。   
定义这类插件的开发框架是：   
$.fn.YouPlugin = function (options) {...}   
为了防止一些偷懒的人，我们需要设置一些默认值，当它没有设置的时候，我们就用这些默认值。   
var defaultVal = {   
Text: 'Your mouse is over',   
ForeColor: 'red',   
BackColor: 'gray'   
};   
那默认值和用户传进来的值是怎么联合在一起的呢？这个就需要我们在开篇的时候讲到的$.extend()知识了。   
var obj = $.extend(defaultVal, options);   
这样子的话，就把用户自定义的值覆盖了默认用户的值。如果用户没定义值，就用系统自定义的。   
代码如下：

.代码如下:

(function ($) {   
$.fn.textHover = function (options) {//options 经常用这个表示有许多个参数。   
var defaultVal = {   
Text: 'Your mouse is over',   
ForeColor: 'red',   
BackColor: 'gray'   
};   
　　　　　　//默认值   
var obj = $.extend(defaultVal, options);   
return this.each(function () {   
var selObject = $(this);//获取当前对象   
var oldText = selObject.text();//获取当前对象的text值   
var oldBgColor = selObject.css("background-color");//获取当前对象的背景色   
var oldColor = selObject.css("color");//获取当前对象的字体的颜色   
selObject.hover(function () {//定义一个hover方法。   
selObject.text(obj.Text);//进行赋值   
selObject.css("background-color", obj.BackColor);//进行赋值   
selObject.css("color", obj.ForeColor);//进行赋值   
},   
function () {   
selObject.text(oldText);   
selObject.css("background-color", oldBgColor);   
selObject.css("color", oldColor);   
}   
);   
});   
}   
})(jQuery);

代码也很简单，上面都有些了解释，此刻不在罗嗦。   
怎么用呢？很简单。   
HTML code：

.代码如下:

<div id="div1" class="textBar">   
Mouse over here.....   
</div>   
<div id="div2" class="textBar">   
Mouse over here.....   
</div>

JS Code：

.代码如下:

$(document).ready(function () {   
$('#div1').textHover({   
Text: 'I am going to over..',   
ForeColor: 'yellow',   
BackColor: 'Red'   
});   
$('#div2').textHover({ Text: 'I am second div...' });   
});

就能看到效果了。