jQuery 使用手册

翻译整理：Young.J
官方网站：http://jquery.com

jQuery 是一款同 prototype 一样优秀 js 开发库类，特别是对 css 和 XPath 的支持，使我们写 js 变得更加方便！如果你不是个 js 高手又想写出优 秀的 js 效果，jQuery 可以帮你达到目的！

下载地址：Starterkit （http://jquery.bassistance.de/jquery-starterkit.zip）
jQuery Downloads （http://jquery.com/src/）

下载完成后先加载到文档中，然后我们来看个简单的例子！

```
<script language="javascript" type="text/javascript">
    $(document).ready(function(){
        $("a").click(function() {
          alert("Hello world!");
        });
    });
<script>
```

上边的效果是点击文档中所有 a 标签时将弹出对话框，$("a") 是一个 jQuery 选择器，$本身表示一个 jQuery 类，所有$()是构造一个 jQuery 对象，click()是这个对象的方法，同理$(document)也是一个 jQuery 对象，ready(fn)是$(document)的方法，表示当 document 全部下载完毕时执行函数。

在进行下面内容之前我还要说明一点$("p")和$("#p")的区别,$("p")表示取所有 p 标签(<p></p>)的元素,$("#p")表示取 id 为"p"()的元素.

我将从以下几个内容来讲解 jQuery 的使用:

1:核心部分
2:DOM 操作
3:css 操作
4:javascript 处理
5:动态效果
6:event 事件
7:ajax 支持
8:插件程序

一、 核心部分

1. $(expr)

说明：该函数可以通过 css 选择器，Xpath 或 html 代码来匹配目标元素，所有的 jQuery 操作都以此为基础

参数：expr：字符串，一个查询表达式或一段 html 字符串

例子：

未执行 jQuery 前：

```
  <p>one</p>
  <div>
      <p>two</p>
  </div>
   <p>three</p>
   <a href="#" id="test" onClick="jq()" >jQuery</a>
```

jQuery 代码及功能：

```
function jq(){
   alert($("div > p").html());
}
```

运行：当点击 id 为 test 的元素时，弹出对话框文字为 two，即 div 标签下 p 元素的内容

```
function jq(){
   $("<div><p>Hello</p></div>").appendTo("body");
}
```

运行：当点击 id 为 test 的元素时，向 body 中添加“ Hello ”

2. $(elem)

说明：限制 jQuery 作用于一个特定的 dom 元素，这个函数也接受 xml 文档和 windows 对象

参数： elem：通过 jQuery 对象压缩的 DOM 元素

例子：

未执行 jQuery 前：

```
<p>one</p>
 <div>
   <p>two</p>
 </div>
<p>three</p>
<a href="#" id="test" onClick="jq()">jQuery</a>
```

jQuery 代码及功能：

```
function jq(){
   alert($(document).find("div > p").html());
}
```

运行：当点击 id 为 test 的元素时，弹出对话框文字为 two，即 div 标签下 p 元素的内容

```
function jq(){
  $(document.body).background("black");
}
```

运行：当点击 id 为 test 的元素时，背景色变成黑色

3. $(elems)

说明：限制 jQuery 作用于一组特定的 DOM 元素

参数： elem：一组通过 jQuery 对象压缩的 DOM 元素

例子：

未执行 jQuery 前：

```
<form id="form1">
      <input type="text" name="textfield">
      <input type="submit" name="Submit" value="提交">
  </form>
  <a href="#" id="test" onClick="jq()">jQuery</a>
```

jQuery 代码及功能：

```
function jq(){
  $(form1.elements ).hide();
}
```

运行：当点击 id 为 test 的元素时，隐藏 form1 表单中的所有元素。

4. $(fn)

说明：$(document).ready()的一个速记方式，当文档全部载入时执行函数。可以有多个$(fn)当文档载入时，同时执行所有函数！

参数：fn (Function):当文档载入时执行的函数！

例子：

```
$( function(){
   $(document.body).background("black");
})
```

运行：当文档载入时背景变成黑色，相当于 onLoad。

5. $(obj)

说明：复制一个 jQuery 对象，

参数：obj (jQuery): 要复制的 jQuery 对象

例子：

未执行 jQuery 前：

```
<p>one</p>
<div>
  <p>two</p>
</div>
<p>three</p>
<a href="#" id="test" onClick="jq()">jQuery</a>
```

jQuery 代码及功能：

```
function jq(){
   var f = $("div");
   alert($(f).find("p").html())
}
```

运行：当点击 id 为 test 的元素时，弹出对话框文字为 two，即 div 标签下 p 元素的内容。

6. each(fn)

说明：将函数作用于所有匹配的对象上

参数：fn (Function): 需要执行的函数

例子：

未执行 jQuery 前：

```
<img src="1.jpg"/>
<img src="1.jpg"/>
<a href="#" id="test" onClick="jq()">jQuery</a>
```

jQuery 代码及功能：

```
function jq(){
  $("img").each(function(){
     this.src = "2.jpg"; });
}
```

运行：当点击 id 为 test 的元素时，img 标签的 src 都变成了 2.jpg。

7. eq(pos)

说明：减少匹配对象到一个单独得 dom 元素

参数：pos (Number): 期望限制的索引，从 0 开始

例子：

未执行 jQuery 前：

```
<p>This is just a test.</p>
<p>So is this</p>
<a href="#" id="test" onClick="jq()">jQuery</a>
```

jQuery 代码及功能：

```
function jq(){
   alert($("p").eq(1).html())
}
```

运行：当点击 id 为 test 的元素时，alert 对话框显示：So is this，即第二个<p>标签的内容

8. get() get(num)

说明：获取匹配元素，get(num)返回匹配元素中的某一个元素

参数：get (Number): 期望限制的索引，从 0 开始

例子：

未执行 jQuery 前：

```
<p>This is just a test.</p>
<p>So is this</p>
<a href="#" id="test" onClick="jq()">jQuery</a>
```

jQuery 代码及功能：

```
function jq(){
   alert($("p").get(1).innerHTML);
}
```

运行：当点击 id 为 test 的元素时，alert 对话框显示：So is this，即第二个<p>标签的内容

注意 get 和 eq 的区别，eq 返回的是 jQuery 对象，get 返回的是所匹配的 dom 对象，所有取$("p").eq(1)对象的内容用 jQuery 方法 html()，而取$("p").get(1)的内容用 innerHTML

9. index(obj)

说明：返回对象索引

参数：obj (Object): 要查找的对象

例子：

未执行 jQuery 前：

```
<div id="test1"></div>
<div id="test2"></div>
<a href="#" id="test" onClick="jq()">jQuery</a>
```

jQuery 代码及功能：

```
function jq(){
   alert($("div").index(document.getElementById('test1')));
```

```
    alert($("div").index(document.getElementById('test2')));
}
```

运行：当点击 id 为 test 的元素时，两次弹出 alert 对话框分别显示 0，1

10. size()　Length

说明：当前匹配对象的数量，两者等价

例子：

未执行 jQuery 前：

```
<img src="test1.jpg"/>
<img src="test2.jpg"/>
<a href="#" id="test" onClick="jq()">jQuery</a>
```

jQuery 代码及功能：

```
function jq(){
    alert($("img").length);
}
```

运行：当点击 id 为 test 的元素时，弹出 alert 对话框显示 2，表示找到两个匹配对象

二、 DOM 操作

1. 属性

我们以为例，在原始的 javascript 里面可以用 var o=document.getElementById('a')取的 id 为 a 的节点对象，在用 o.src 来取得或修改该节点的 scr 属性，在 jQuery 里$("#a")将得到 jQuery 对象[ ]，然后可以用 jQuery 提供的很多方法来进行操作，如$("#a").scr()将得到 5.jpg,$("#a").scr("1.jpg")将该对象 src 属性改为 1,jpg。下面我们来讲 jQuery 提供的众多 jQuery 方法，方便大家快速对 DOM 对象进行操作

herf()　herf(val)

说明：对 jQuery 对象属性 herf 的操作。

例子：

未执行 jQuery 前

```
<a href="1.htm" id="test" onClick="jq()">jQuery</a>
```

jQuery 代码及功能：

```
function jq(){
  alert($("#test").href());
  $("#test").href("2.html");
}
```

运行：先弹出对话框显示 id 为 test 的连接 url，在将其 url 改为 2.html，当弹出对话框后会看到转向到 2.html

同理，jQuery 还提供类似的其他方法，大家可以分别试验一下：

herf()　herf(val)　html()　html(val)　id()　id (val)　name()　name (val)　rel()　rel (val)

src()　src (val)　title()　title (val)　val()　val(val)

2. 操作

after(html)　在匹配元素后插入一段 html

```
<a href="#" id="test" onClick="jq()">jQuery</a>
```

jQuery 代码及功能：

```
function jq(){
    $("#test").after("<b>Hello</b>");
}
```

执行后相当于：

```
<a href="#" id="test" onClick="jq()">jQuery</a><b>Hello</b>
```

after(elem) after(elems) 将指定对象 elem 或对象组 elems 插入到在匹配元素后

```
<p id="test">after</p><a href="#" onClick="jq()">jQuery</a>
```

jQuery 代码及功能

```
function jq(){
   $("a").after($("#test"));
}
```

执行后相当于

```
<a href="#" onClick="jq()">jQuery</a><p id="test">after</p>
```

append(html)在匹配元素内部，且末尾插入指定 html

```
<a href="#" id="test" onClick="jq()">jQuery</a>
```

jQuery 代码及功能：

```
function jq(){
   $("#test").append("<b>Hello</b>");
}
```

执行后相当于

```
<a href="#" onClick="jq()">jQuery<b>Hello</b></a>
```

同理还有 append(elem) append(elems) before(html) before(elem) before(elems)请执行参照 append 和 after 的方来测试、理解！

appendTo(expr) 与 append(elem)相反

```
<p id="test">after</p><a href="#" onClick="jq()">jQuery</a>
```

jQuery 代码及功能

```
function jq(){
    $("a"). appendTo ($("#test"));
}
```

执行后相当于

```
<p id="test">after<a href="#" onClick="jq()">jQuery</a> </p>
```

clone() 复制一个 jQuery 对象

```
<p id="test">after</p><a href="#" onClick="jq()">jQuery</a>
```

jQuery 代码及功能：

```
function jq(){
    $("#test").clone().appendTo($("a"));
}
```

复制$("#test")然后插入到<a>后,执行后相当于

```
<p id="test">after</p><a href="#" onClick="jq()">jQuery</a><p id="test">after</p>
```

empty() 删除匹配对象的所有子节点

```
<div id="test">
    <span>span</span>
    <p>after</p>
  </div>
  <a href="#" onClick="jq()">jQuery</a>
```

jQuery 代码及功能：

```
function jq(){
   $("#test").empty();
}
```

执行后相当于

```
<div id="test"></div><a href="#" onClick="jq()">jQuery</a>
```

insertAfter(expr) insertBefore(expr)

按照官方的解释和我的几个简单测试 insertAfter(expr) 相当于 before(elem),insertBefore(expr)相当于 after (elem)

prepend (html) prepend (elem) prepend (elems) 在匹配元素的内部且开始出插入

通过下面例子区分 append(elem) appendTo(expr) prepend (elem)

```
<p id="a">p</p>
<div>div</div>
```

执行**$("#a").append($("div"))** 后相当于

```
<p id="a">
 P
  <div>div</div>
</p>
```

执行**$("#a").appendTo($("div"))** 后 相当于

```
<div>
  div
  <p id="a">p</p>
</div>
```

执行**$("#a").prepend ($("div"))** 后 相当于

```
<p id="a">
  <div>div</div>
```

```
  P
</p>
```

remove() 删除匹配对象

注意区分 empty()，empty()移出匹配对象的子节点，remove()，移出匹配对象

wrap(htm) 将匹配对象包含在给出的 html 代码内

```
<p>Test Paragraph.</p> <a href="#" onClick="jq()">jQuery</a>
```

jQuery 代码及功能：

```
function jq(){
    $("p").wrap("<div class='wrap'></div>");
}
```

执行后相当于

```
<div class='wrap'><p>Test Paragraph.</p></div>
```

wrap(elem) 将匹配对象包含在给出的对象内

```
<p>Test Paragraph.</p><div id="content"></div>
<a href="#" onClick="jq()">jQuery</a>
```

jQuery 代码及功能：

```
function jq(){
    $("p").wrap( document.getElementById('content') );
}
```

执行后相当于

```
<div id="content"><p>Test Paragraph.</p></div>
```

3. 遍历、组合

add(expr) 在原对象的基础上在附加符合指定表达式的 jquery 对象

```
<p>Hello</p><p><span>Hello Again</span></p>
<a href="#" onClick="jq()">jQuery</a>
```

jQuery 代码及功能：

```
function jq(){
   var f=$("p").add("span");
   for(var i=0;i < $(f).size();i++){
   alert($(f).eq(i).html());}
}
```

执行$("p")得到匹配<p>的对象，有两个，add("span")是在("p")的基础上加上匹配的对

象，所有一共有 3 个，从上面的函数运行结果可以看到$("p").add("span")是 3 个对象的集合，分别是[Hello]，[Hello Again]，[Hello Again]。

add(el) 在匹配对象的基础上在附加指定的 dom 元素。

$("p").add(document.getElementById("a"));

add(els) 在匹配对象的基础上在附加指定的一组对象，els 是一个数组。

```
<p>Hello</p><p><span>Hello Again</span></p>
```

jQuery 代码及功能：

```
function jq(){
   var f=$("p").add([document.getElementById("a"), document.getElementById("b")])
   for(var i=0;i < $(f).size();i++){
        alert($(f).eq(i).html());}
}
```

注意 els 是一个数组，这里的[ ]不能漏掉。

ancestors () 一依次以匹配结点的父节点的内容为对象,根节点除外（有点不好理解，看看下面例子就明白了）

```
<div>
  <p>one</p>
  <span>
  <u>two</u>
  </span>
</div>
```

jQuery 代码及功能：

```
function jq(){
   var f= $("u").ancestors();
   for(var i=0;i < $(f).size();i++){
    alert($(f).eq(i).html());}
}
```

第一个对象是以<u>的父节点的内容为对象，[<u>two</u>]

第一个对象是以<u>的父节点的父节点（div）的内容为对象，[one <u>two</u>]

一般一个文档还有<body>和<html>，依次类推下去。

ancestors (expr) 在 ancestors（）的基础上之取符合表达式的对象

如上各例子讲 var f 改为 var f= $("u").ancestors(“div”),则只返回一个对象：

[one <u>two</u>]

children() 返回匹配对象的子介点

```
<p>one</p>
<div id="ch">
   <span>two</span>
```

```
</div>
```

jQuery 代码及功能：

```
function jq(){
   alert($("#ch").children().html());
}
```

$("#ch").children()得到对象[two].所以.html()的结果是”two”

children(expr) 返回匹配对象的子介点中符合表达式的节点

```
<div id="ch">
    <span>two</span>
    <span id="sp">three</span>
</div>
```

jQuery 代码及功能

```
function jq(){
   alert($("#ch").children(“#sp”).html());
}
```

$("#ch").children()得到对象[twothree].

$("#ch").children(“#sp”)过滤得到[three]

parent () parent (expr)取匹配对象父节点的。参照 children 帮助理解

contains(str) 返回匹配对象中包含字符串 str 的对象

```
  <p>This is just a test.</p><p>So is this</p>
```

jQuery 代码及功能：

```
function jq(){
   alert($("p").contains("test").html());
}
```

$("p")得到两个对象，而包含字符串”test”只有一个。所有$("p").contains("test")返回 [This is just a test.]

end() 结束操作,返回到匹配元素清单上操作前的状态.

filter(expr) filter(exprs) 过滤现实匹配符合表达式的对象 exprs 为数组,注意添加“[ ]”

```
<p>Hello</p><p>Hello Again</p><p class="selected">And Again</p>
```

jQuery 代码及功能：

```
function jq(){
   alert($("p").filter(".selected").html())
}
```

$("p")得到三个对象，$("p").contains("test")只返回 class 为 selected 的对象。

find(expr) 在匹配的对象中继续查找符合表达式的对象

```
<p>Hello</p><p id="a">Hello Again</p><p class="selected">And Again</p>
```

jQuery 代码及功能

```
function jq(){
   alert($("p").find("#a").html())
}
```

在$("p")对象中查找 id 为 a 的对象。

is(expr) 判断对象是否符合表达式,返回 boolen 值

```
<p>Hello</p><p id="a">Hello Again</p><p class="selected">And Again</p>
```

jQuery 代码及功能：

```
function jq(){
   alert($("#a").is("p"));
}
```

在$("#a ")是否符合 jquery 表达式。

大家可以用$("#a").is("div"); ("#a").is("#a")多来测试一下

next() next(expr) 返回匹配对象剩余的兄弟节点

```
<p>Hello</p><p id="a">Hello Again</p><p class="selected">And Again</p>
```

jQuery 代码及功能

```
function jq(){
      alert($("p").next().html());
     alert($("p").next(".selected").html());
}
```

$("p").next()返回 [Hello Again , And Again]两个对象

$("p").next(".selected)只返回 [And Again]一个对象

prev () prev (expr) 参照 next 理解

not(el) not(expr) 从 jQuery 对象中移出匹配的对象，el 为 dom 元素，expr 为 jQuery 表达式。

```
<p>one</p><p id="a">two</p>
<a href="#" onclick="js()">jQuery</a>
```

jQuery 代码及功能：

```
function js(){
   alert($("p").not(document.getElementById("a")).html());
   alert($("p").not(“#a”).html());
}
```

$("p")由两个对象，排除后的对象为[one]

siblings () siblings (expr) jquery 匹配对象中其它兄弟级别的对象

```
<p>one</p>
<div>
 <p id="a">two</p>
</div>
<a href="#" onclick="js()">jQuery</a>
```

jQuery 代码及功能：

```
function js(){
    alert($("div").siblings().eq(1).html());
}
```

$("div").siblings() 的结果实返回两个对象 [one , jQuery]

alert($("div").siblings("a")返回一个对象[jQuery]

4. 其他

addClass(class) 为匹配对象添加一个 class 样式

removeClass (class) 将第一个匹配对象的某个 class 样式移出

attr (name) 获取第一个匹配对象的属性

```
<img src="test.jpg"/><a href="#" onclick="js()">jQuery</a>
```

jQuery 代码及功能：

```
function js(){
   alert($("img").attr("src"));
}
```

返回 test.jpg

attr (prop) 为第一个匹配对象的设置属性，**prop** 为 **hash** 对象，用于为某对象批量添加众多属性

```
<img/><a href="#" onclick="js()">jQuery</a>
```

jQuery 代码及功能：

```
function js(){
   $("img").attr({ src: "test.jpg", alt: "Test Image" });
}
```

运行结果相当于

attr (key,value) 为第一个匹配对象的设置属性，**key** 为属性名，**value** 为属性值

```
<img/><a href="#" onclick="js()">jQuery</a>
```

jQuery 代码及功能

```
function js(){
   $("img").attr("src","test.jpg");
}
```

运行结果相当于

removeAttr (name) 将第一个匹配对象的某个属性移出

```
<img alt="test"/><a href="#" onclick="js()">jQuery</a>
```

jQuery 代码及功能：

```
function js(){
```

```
    $("img"). removeAttr("alt");
}
```

运行结果相当于<img />

toggleClass (class)　将当前对象添加一个样式，不是当前对象则移出此样式，返回的是处理后的对象

```
<p>Hello</p><p class="selected">Hello Again</p><a href="#" onclick="js()">jQuery</a>
```

$("p")的结果是返回对象 [Hello , Hello Again]

$("p").toggleClass("selected") 的结果是实返回对象 [Hello , Hello Again]

三、 CSS 操作

传统 javascript 对 css 的操作相当繁琐，比如 css 取它的 background 语法是 document.getElementById("a").style.background，而 jQuery 对 css 更方便的操作，$("#a").background()，$("#a").background(“red”)

$("#a")得到 jQuery 对象[<div id="a" ... /div>]

$("#a").background()将取出该对象的 background 样式。

$("#a").background(“red”)将该对象的 background 样式设为 redjQuery 提供了以下方法，来操作 css

background () background (val) color() color(val) css(name) css(prop) css(key, value) float() float(val) height() height(val) width() width(val) left() left(val) overflow() overflow(val) position() position(val) top() top(val)

这里需要讲解一下 css(name) css(prop) css(key, value)，其他的看名字都知道什么作用了！

```
<div id="a" style="background:blue; color:red">css</div><P id="b">test</P>
```

css(name) 获取样式名为 name 的样式

$("#a").css("color") 将得到样式中 color 值 red，("#a").css("background ")将得到 blue

css(prop) prop 是一个 hash 对象，用于设置大量的 css 样式

$("#b").css({ color: "red", background: "blue" });

最终效果是 test ,{ color: "red", background: "blue" }，hash 对象，color 为 key，"red"为 value，

css(key, value) 用于设置一个单独得 css 样式

$("#b").css("color","red");最终效果是 test

四、 JavaScript 处理

$.browser() 判断浏览器类型，返回 boolen 值

```
$(function(){
  if($.browser.msie) {
     alert("这是一个 IE 浏览器");}
  else if($.browser.opera) {
     alert("这是一个 opera 浏览器");}
})
```

当页面载入式判断浏览器类型，可判断的类型有 msie、mozilla、opera、safari

$.each(obj, fn) obj 为对象或数组，fn 为在 obj 上依次执行的函数，注意区分$().each()

```
$.each( [0,1,2], function(i){ alert( "Item #" + i + ": " + this ); });
```

分别将 0，1，2 为参数，传入到 function(i)中

```
$.each({ name: "John", lang: "JS" }, function(i){ alert( "Name: " + i + ", Value: " + this );
```

{ name: "John", lang: "JS" }为一个 hash 对象，依次将 hash 中每组对象传入到函数中

$.extend(obj, prop) 用第二个对象扩展第一个对象

```
var settings = { validate: false, limit: 5, name: "foo" };
var options = { validate: true, name: "bar" };
$.extend(settings, options);
```

执行后 settings 对象为{ validate: true, limit: 5, name: "bar" }

可以用下面函数来测试

```
$(function(){
    var settings = { validate: false, limit: 5, name: "foo" };
     var options = { validate: true, name: "bar" };
```

```
    $.extend(settings, options);
     $.each(settings, function(i){ alert( i + "=" + this ); });
})
```

$.grep(array,fn) 通过函数 fn 来过滤 array，将 array 中的元素依次传给 fn，fn 必须返回一个 boolen，如 fn 返回 true，将被过滤

```
$(function(){
    var arr= $.grep( [0,1,2,3,4], function(i){ return i > 2; });
    $.each(arr, function(i){ alert(i); });
})
```

我们可以看待执行$.grep 后数组[0,1,2,3,4]变成[0，1]

$.merge(first, second) 两个参数都是数组，排出第二个数组中与第一个相同的，再将两个数组合并

```
$(function(){
    var arr = $.merge( [0,1,2], [2,3,4] )
    $.each(arr, function(i){ alert(i); });
})
```

可以看出 arr 的结果为[0,1,2,3,4]

$.trim(str) 移出字符串两端的空格

$.trim(" hello, how are you? ")的结果是"hello, how are you?"

五、 动态效果

在将这部分之前我们先看个例子，相信做网页的朋友都遇到 n 级菜单的情景，但点击某菜单按钮时，如果它的子菜单是显示的，则隐藏子菜单，如果子菜单隐藏，则显示出来，传统的 javascript 做法是先用 getElementById 取出子菜单所在容器的 id，在判断该容器的 style.display 是否等于 none，如果等于则设为 block,如果不等于这设为 none，如果在将效果设置复杂一点，当点击按钮时，不是忽然隐藏和显示子菜单，而是高度平滑的转变，这时就要通过 setTimeout 来设置子菜单的 height 了，再复杂一点透明度也平滑的消失和显现，这时显现下来需要编写很多代码，如果 js 基础不好的朋友可能只能从别人写好的代码拿过来修改了！jQuery 实现上面效果只需要 1 句话就行，$("#a").toggle("slow"), ,学完 jQuery 后还需要抄袭修改别人的代码吗？下面我们逐个介绍 jQuery 用于效果处理的方法。

hide() 隐藏匹配对象

```
<p id="a">Hello Again</p><a href="#" onClick=' ("#a").hide()'>jQuery</a>
```

当点击连接时,id 为 a 的对象的 display 变为 none。

show() 显示匹配对象

hide(speed) 以一定的速度隐藏匹配对象，其大小（长宽）和透明度都逐渐变化到 0，speed 有 3 级("slow", "normal", "fast")，也可以是自定义的速度。

show(speed) 以一定的速度显示匹配对象，其大小（长宽）和透明度都由 0 逐渐变化到正常

hide(speed, callback) show(speed, callback) 当显示和隐藏变化结束后执行函数 callback

toggle() toggle(speed) 如果当前匹配对象隐藏，则显示他们，如果当前是显示的，就隐藏，toggle(speed),其大小（长宽）和透明度都随之逐渐变化。

```
<img src="1.jpg" style="width:150px"/>
<a href="#" onClick='$("img").toggle("slow")'>jQuery</a>
```

fadeIn(speeds) fadeOut(speeds) 根据速度调整透明度来显示或隐藏匹配对象，注意有别于 hide(speed)和 show(speed)，fadeIn 和 fadeOut 都只调整透明度，不调整大小

```
<img src="1.jpg" style="display:none"/><a href="#" onClick='$("img ").fadeIn("slow")'> jQu
ery </a>
```

点击连接后可以看到图片逐渐显示。

fadeIn(speed, callback) fadeOut(speed, callback) callback 为函数，先通过调整透明度来显示或隐藏匹配对象，当调整结束后执行 callback 函数

```
<img src="1.jpg"/>
<a href="#" onClick='$("img ").fadeIn("slow",function(){ alert("Animation Done."); })'> jQue
ry </a>
```

点击连接后可以看到图片逐渐显示,显示完全后弹出对话框

fadeTo(speed, opacity, callback) 将匹配对象以 speed 速度调整倒透明度 opacity，然后执行函数 callback。Opacity 为最终显示的透明度(0-1).

```
<img src="1.jpg"/><br>
<a href="#" onClick='$("img ").fadeTo("slow",0.55,function(){ alert("Animation Done."); })'>
 jQuery </a>
```

大家可以看一下自己看看效果，如果不用 jQuery，编写原始 javascript 脚本可能很多代码！

slideDown(speeds) 将匹配对象的高度由 0 以指定速率平滑的变化到正常！

```
<img src="1.jpg" style="display:none"/>
<a href="#" onClick='$("img ").slideDown("slow")'>jQuery</a>
```

slideDown(speeds,callback) 将匹配对象的高度由 0 变化到正常！变化结束后执行函数 callback

slideUp("slow") **slideUp(speed, callback)** 匹配对象的高度由正常变化到 **0**

slideToggle("slow") 如果匹配对象的高度正常则逐渐变化到 **0**，若为 **0**，则逐渐变化到正常

六、 事件处理

hover(Function, Function) 当鼠标 **move over** 时触发第一个 **function**，当鼠标 **move out** 时触发第二个 **function**

样式：<style>.red{color:#FF0000}</style>

Html 代码： sdf

jQuery 代码及效果

```
$(function(){
 $("#a").hover(function(){$(this).addClass("red");},
                function(){ $(this).removeClass("red");
               });
})
```

最终效果是当鼠标移到 id 为 a 的层上时图层增加一个 red 样式，离开层时移出 red 样式

toggle(Function, Function) 当匹配元素第一次被点击时触发第一个函数，当第二次被点击时触发第二个函数

样式：<style>.red{color:#FF0000}</style>

Html 代码： sdf

jQuery 代码及效果

```
$(function(){
 $("#a"). toggle (function(){$(this).addClass("red");},
                function(){ $(this).removeClass("red");
               });
})
```

最终效果是当鼠标点击 id 为 a 的层上时图层增加一个 red 样式，离开层时移出 red 样式

bind(type, fn) 用户将一个事件和触发事件的方式绑定到匹配对象上。

trigger(type) 用户触发 **type** 形式的事件。**$("p").trigger("click")**

还有：**unbind()** **unbind(type)** **unbind(type, fn)**

Dynamic event(Function)　绑定和取消绑定提供函数的简捷方式

例：

```
$("#a").bind("click",function() {
                        $(this).addClass("red");
})
```

也可以这样写：

```
$("#a").click(function() {
                $(this).addClass("red");
});
```

最终效果是当鼠标点击 id 为 a 的层上时图层增加一个 red 样式，

jQuery 提供的函数

用于 browers 事件

error(fn)　load(fn)　unload(fn)　resize(fn)　scroll(fn)

用于 form 事件

change(fn)　select(fn)　submit(fn)

用于 keyboard 事件

keydown(fn)　keypress(fn)　keyup(fn)

用于 mouse 事件

click(fn)　dblclick(fn)　mousedown(fn)　mousemove(fn)
mouseout(fn)　mouseover(fn)　mouseup(fn)

用于 UI 事件

blur(fn)　focus(fn)

以上事件的扩展再扩展为 5 类

举例，click(fn) 扩展 click() unclick() oneclick(fn) unclick(fn)

click(fn)：增加一个点击时触发某函数的事件

click()：可以在其他事件中执行匹配对象的 click 事件。

unclick ()：不执行匹配对象的 click 事件。

oneclick(fn)：只增加可以执行一次的 click 事件。

unclick (fn)：增加一个点击时不触发某函数的事件。

上面列举的用于 browers、form、keyboard、mouse、UI 的事件都可以按以上方法扩展。

七、　Ajax 支持

通用方式：

$.ajax(prop)　通过一个 ajax 请求，回去远程数据，prop 是一个 hash 表，它可以传递的 key/value 有以下几种。

(String)type：数据传递方式(get 或 post)。

((String)url：数据请求页面的 url

((String)data：传递数据的参数字符串，只适合 post 方式

((String)dataType：期待数据返回的数据格式(例如 "xml", "html", "script",或 "json")

((Boolean)ifModified：　当最后一次请求的相应有变化是才成功返回，默认值是 false

((Number)timeout:设置时间延迟请求的时间。可以参考$.ajaxTimeout

((Boolean)global：是否为当前请求触发 ajax 全局事件，默认为 true

((Function)error：当请求失败时触发的函数。

((Function)success：当请求成功时触发函数

((Function)complete：当请求完成后出发函数

jQuery 代码及说明

```
$.ajax({url: "ajax.htm",
      success:function(msg){
                $(div"#a").html(msg);
          }
  });
```

将 ajax.htm 返回的内容作为 id 为 a 的 div 内容

```
$.ajax({ url: "ajax.aspx",
          type:"get",
         dataType:"html",
        data: "name=John&location=Boston",
        success:function(msg){
                    $("#a").html(msg);
                  }
      });
```

用 get 方式向 ajax.aspx 页面传参数，并将返回内容负给 id 为 a 的对象。

$.ajaxTimeout(time) 设置请求结束时间

$.ajaxTimeout(5000)

其它简化方式：

$.get(url, params, callback)　用 get 方式向远程页面传递参数，请求完成后处理函数，除了 url 外，其它参数任意选择！

```
$.get( "ajax.htm" , function(data){ $("#a").html(data)  })

$.get(    "ajax.asp",

        { name: "young", age: "25" },

        function(data){ alert("Data Loaded: " + data); }
```

```
)
```

$.getIfModified(url, params, callback) 用 get 方式向远程页面传递参数，从最后一次请求后如果数据有变化才作出响应，执行函数 callback

$.getJSON(url, params, callback) 用 get 方式向远程 json 对象传递参数，请求完成后处理函数 callback。

$.getScript(url, callback) 用 get 方式载入并运行一个远程 javascript 文件。请求完成后处理函数 callback。

$.post(url, params, callback) 用 post 方式向远程页面传递参数，请求完成后处理函数 callback

load(url, params, callback) 载入一个远程文件并载入页面 DOM 中，并执行函数 callback

```
$("#a").load("ajax.htm", function() { alert("load is done"); } );
```

向 ajax.htm 页面发出请求，将返回结果装入 id 为 a 的内容中，然后再执行函数 callback。

loadIfModified(url, params, callback) 用 get 方式向远程页面传递参数，从最后一次请求后如果数据有变化才作出响应，将返回结果载入页面 DOM 中，并执行函数 callback

ajaxStart(callback) 当 ajax 请求发生错误是时执行函数 callback

ajaxComplete(callback) 当 ajax 请求完成时执行函数 callback

ajaxError(callback) 当 ajax 请求发生错误时执行函数 callback

ajaxStop(callback) 当 ajax 请求停止时执行函数 callback

ajaxSuccess(callback) 当 ajax 请求成功时执行函数 callback

八、 jQuery 插件

随着 jQuery 的广泛使用，已经出现了大量 jQuery 插件，如 thickbox，iFX，jQuery-googleMap 等，简单的引用这些源文件就可以方便的使用这些插件。这里我简单的介绍一些网址供大家参考，这些网站头提供了大量的 demo，并且使用及其简单，即使 E 文不好，也能快速掌握！

http://jquery.com/plugins 官方推荐

http://interface.eyecon.ro/demos 效果超级棒，使用更简单，一定有你喜欢的！

http://www.dyve.net/jquery/

http://bassistance.de/jquery-plugins

还有其它很多插件，大家可以 google 以下，如果大家发现好的了，可以留言共享以下！