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

$(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 "

$(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

$(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。

$(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。

$(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。

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。

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

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

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

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

jQuery 使用手册

二：DOM 操作

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)

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">aft
er</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>
```

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$("p")得到匹配 的对象，有两个，add("span")是在("p")的基础上加上匹配的对象，所有一共有 3 个，从上面的函数运行结果可以看到$("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>
```

Query 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>
```

Query 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]

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]

jQuery 使用手册

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 red

jQuery 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 + ", Valu
e: " + 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")'> jQuery </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."); })'> jQuery </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

2006-12-12

jQuery 中文入门教程全文已经译完

整理后的教程完整版在这里:

http://keel.sike.googlepages.com/jQuery_getting_started.html

已经给原文作者回复.如果有错误的地方恳请提出!

posted by Keel @ 9:57 PM **0 comments**

jQuery 中文入门指南，翻译加实例，jQuery 的起点教程(四)

五：Animate me(让我生动起来):使用 FX

一些生动的效果可以使用 show() 和 hide()来表现:

```
$(document).ready(function() {
 $("a").toggle(function() {
  $(".stuff").hide('slow');
 }, function() {
  $(".stuff").show('fast');
 });
});
```

你可以与 animate()联合起来创建一些效果,如一个带渐显的滑动效果:

```
$(document).ready(function() {
 $("a").toggle(function() {
  $(".stuff").animate({
   height: 'hide',
   opacity: 'hide'
  }, 'slow');
 }, function() {
  $(".stuff").animate({
   height: 'show',
```

```
    opacity: 'show'
  }, 'slow');
 });
});
```

很多不错的效果可以访问 **interface plugin collection**. 这个站点提供了很多 demos 和文档

这些效果插件是位于 jQuery 插件列表的前面的，当然也有很多其他的插件，比如我们下一章讲到的表格排序插件。

本章的相关链接:

- **jQuery FX Module**
- **Interface plugin**

Sort me(将我有序化):使用 tablesorter 插件(表格排序)

这个表格排序插件能让我们在客户端按某一列进行排序，引入 jQuery 和这个插件的 js 文件，然后告诉插件你想要哪个表格拥有排序功能。

要测试这个例子，先在 starterkit.html 中加上像下面这一行的代码:

```
<script                                    src="lib/jquery.tablesorter.js"
type="text/javascript"></script>
```

然后可以这样调用不着:

```
$(document).ready(function() {
 $("#large").tableSorter();
});
```

现在点击表格的第一行 head 区域，你可以看到排序的效果，再次点击会按倒过来的顺序进行排列。

这个表格还可以加一些突出显示的效果，我们可以做这样一个隔行背景色效果:

```
$(document).ready(function() {
 $("#large").tableSorter({
  stripingRowClass: ['odd','even'], // Class names for striping supplyed
as a array.
  stripRowsOnStartUp: true  // Strip rows on tableSorter init.
 });
});
```

关于这个插件的更多例子和文档可以在 **tablesorter 首页**找到.

几乎所有的特件都是这样用的:先 include 插件的 js 文件,然后在某些元素上使用插件定义的方法,当然也有一些参数选项是可以配置的

经常更新的插件列表可以从 jQuery 官方站 **on the jQuery site** 找到.

当你更经常地使用 jQury 时,你会发现将你自己的代码打包成插件是很有用处的,它能方便地让你的公司或者其他人进行重用.下一章我们将谈到如何构建一个自己的插件.

本章的相关链接:

- **Plugins for jQuery**
- **Tablesorter Plugin**

Plug me:制作自己的插件

写一个自己的 jQuery 插件是非常容易的,如果你按照下面的原则来做,可以让其他人也容易地结合使用你的插件.

1. 为你的插件取一个名字,在这个例子里面我们叫它"foobar".
2. 创建一个像这样的文件:jquery.[yourpluginname].js,比如我们创建一个 jquery.foobar.js
3. 创建一个或更多的插件方法,使用继承 jQuery 对象的方式,如:

```
jQuery.fn.foobar = function() {
 // do something
};
```

4. 可选的:创建一个用于帮助说明的函数,如:

```
jQuery.fooBar = {
 height: 5,
 calculateBar = function() { ... },
 checkDependencies = function() { ... }
};
```

你现在可以在你的插件中使用这些帮助函数了:

```
jQuery.fn.foobar = function() {
 // do something
 jQuery.foobar.checkDependencies(value);
 // do something else
};
```

5. 可选的 I:创建一个默认的初始参数配置,这些配置也可以由用户自行设定,如:

```
jQuery.fn.foobar = function(options) {
 var settings = {
  value: 5,
  name: "pete",
```

```
    bar: 655
  };
  if(options) {
    jQuery.extend(settings, options);
  }
};
```

现在可以无需做任何配置地使用插件了,默认的参数在此时生效:

```
$("...").foobar();
```

Or with some options:

```
$("...").foobar({
  value: 123,
  bar: 9
});
```

如果你 release 你的插件，你还应该提供一些例子和文档,大部分的插件都具备这些良好的参考文档.

现在你应该有了写一个插件的基础,让我们试着用这些知识写一个自己的插件.

很多人试着控制所有的 radio 或者 checkbox 是否被选中,比如:

```
$("input[@type='checkbox']").each(function() {
  this.checked = true;
  // or, to uncheck
  this.checked = false;
  // or, to toggle
```

```
 this.checked = !this.checked;
});
```

你可能想要把这个重写为一个插件,很直接地:

```
$.fn.check = function() {
 return this.each(function() {
  this.checked = true;
 });
};
```

这个插件现在可以这样用:

```
$("input[@type='checkbox']").check();
```

现在你应该还可以写出 uncheck()和 toggleCheck()了.但是先停一下,让我们的插件接收一些参数.

```
$.fn.check = function(mode) {
 var mode = mode || 'on'; // if mode is undefined, use 'on' as default
 return this.each(function() {
  switch(mode) {
  case 'on':
   this.checked = true;
   break;
  case 'off':
   this.checked = false;
   break;
  case 'toggle':
   this.checked = !this.checked;
   break;
```

```
    }
  });
};
```

这里我们设置了默认的参数,所以将"on"参数省略也是可以的,当然也可以加上"on","off", 或 "toggle",如:

```
$("input[@type='checkbox']").check();
$("input[@type='checkbox']").check('on');
$("input[@type='checkbox']").check('off');
$("input[@type='checkbox']").check('toggle');
```

如果有多于一个的参数设置会稍稍有点复杂,在使用时如果只想设置第二个参数,则要在第一个参数位置写入 null.

从上一章的 tablesorter 插件用法我们可以看到,既可以省略所有参数来使用或者通过一个 key/value 对来重新设置每个参数.

作为一个练习,你可以试着将 **第四部分** 的功能重写为一个插件.这个插件的骨架应该是像这样的:

```
$.fn.rateMe = function(options) {

 // instead of selecting a static container with $("#rating"), we now use the jQuery context
 var container = this;

 var settings = {
  url: "rate.php"
  // put more defaults here
  // remember to put a comma (",") after each pair, but not after the last one!
 };
```

```
 if(options) { // check if options are present before extending the
settings
  $.extend(settings, options);
 }

 // ...
 // rest of the code
 // ...

 return this; // if possible, return "this" to not break the chain
});
```

Next steps(下一步)

如果你想做更好的 javascript 开发,建议你使用一个叫 **FireBug** 的 firefox 插件.
它提供了断点调试(比 alert 强多了)、观察 DOM 变化等很多漂亮的功能

如果你还有未解决的问题，或者新的想法与建议，你可以使用 jQuery 的邮件列表 **jQuery mailing list**.

关于这个指南的任何事情，你可以写 **mail** 给作者或者发表评论在他的日志：**blog**.

关于这个指南的翻译任何事情，你可以写 **mail** 给我或者发表评论在我的日志：**blog**.

全文完.

jQuery 中文入门指南，翻译加实例，jQuery 的起点教程(三)

四.Rate me:使用 AJAX

在这一部分我们写了一个小小的 AJAX 应用，它能够 rate 一些东西（译者 Keel 注：就是对某些东西投票），就像在 youtube.com 上面看到的一样。

首先我们需要一些服务器端代码，这个例子中用到了一个 PHP 文件，读取 rating 参数然后返回 rating 总数和平均数。看一下 **rate.php** 代码。

虽然这些例子也可以不使用 AJAX 来实现，但显然我们不会那么做，我们用 jQuery 生成一个 DIV 容器，ID 是"rating"。

```
$(document).ready(function() {
// generate markup
var ratingMarkup = ["Please rate: "];
for(var i=1; i <= 5; i++) { ratingMarkup[ratingMarkup.length] = "" +
i + " ";
}
// add markup to container and
applier click handlers to anchors
$("#rating").append( ratingMarkup.join('')
).find("a").click(function(e) {
e.preventDefault();
// send requests
$.post("rate.php", {rating: $(this).html()}, function(xml) {
// format
result
var result = [
"Thanks for rating, current average: ",
$("average", xml).text(),
", number of votes: ",
$("count",
xml).text()
];
```

```
// output result
$("#rating").html(result.join(''));
} );
});
});
```

这段代码生成了 5 个链接，并将它们追加到 id 为"rating"容器中，当其中一个链接被点击时，该链接标明的分数就会以 rating 参数形式发送到 rate.php，然后，结果将以 XML 形式会从服务器端传回来，添加到容器中，替代这些链接。
如果你没有一个安装过 PHP 的 webserver，你可以看看这个在线的例子。
这是一个比较粗糙的例子，更完美的例子可以访问 **softonic.de** 点击 "Kurz bewerten!"。
更多的 AJAX 方法可以从这里找到，或者看看 **API** 文档下面的 AJAX。

（译者 Keel 注：这个在线实例从国内访问还是比较慢的，点击后要等一会儿才能看到结果，可以考虑对它进行修改，比如加上 loading，投票后加上再投票的返回链接等。此外，这个例子中还是有很多需要进一步消化的地方，看不懂的地方请参考 API 文档。）

一个在载入 AJAX 时经常发生的问题是：当载入一个事件句柄到一个需要载入内容的 HTML 文档时，你只能在内容载入后再加载这些句柄，为了防止代码重复执行，你可能用到如下一个 function:

```
// lets use the shortcut
$(function() {
var
addClickHandlers = function() {
$("a.clickMeToLoadContent").click(function()
{
$("#target").load(this.href, addClickHandlers);
});
};
```

```
addClickHandlers();
});
```

现在，addClickHandlers 只在 DOM 载入完成后执行一次，并且是在用户点击具有 clickMeToLoadContent 这个样式的链接后。

请注意 addClickHandlers 函数是作为一个本地变量定义的，而不是全局变量(如：function addClickHandlers() {...})，这样做是为了防止与其他的全局变量或者函数相冲突。

另一个常见的问题是关于回传(callback)的参数。你可以通过一个额外的参数指定回叫的方法，简单的办法是将这个回叫方法包含在一个其它的 function 中：

```
// get some data
var foobar = ...;
// specify
handler, it needs data as a paramter
var handler = function(data) {
...
};
// add click handler and pass foobar!
$('a').click(
function(event) { handler(foobar); } );

// if you need the context of
the original handler, use apply:
$('a').click( function(event) {
handler.apply(this, [foobar]); } );
```

用到简单的 AJAX 后，我们可以认为已经非常之"web2.0"了，但是到现在为止，我们还缺少一些酷炫的效果。下一节将如谈到这些效果。

感兴趣的链接：

- jQuery AJAX Module
- jQuery API: Contains description and examples for append and all other jQuery methods
- ThickBox: A jQuery plugin that uses jQuery to enhance the famous lightbox

下一篇:http://keelsike.blogspot.com/2006/12/jqueryjquery_12.html

前一篇:http://keelsike.blogspot.com/2006/12/jqueryjquery_10.html

posted by Keel @ 6:10 PM 0 comments

2006-12-10

jQuery 中文入门指南，翻译加实例，jQuery 的起点教程(二)

三.Find me:使用选择器和事件

jQuery 提供两种方式来选择 html 的 elements，第一种是用 CSS 和 Xpath 选择器联合起来形成一个字符串来传送到 jQuery 的构造器（如：$("div > ul a")）；第二种是用 jQuery 对象的几个 methods(方法)。这两种方式还可以联合起来混合使用。

为了测试一下这些选择器，我们来试着在我们 starterkit.html 中选择并修改第一个 ordered list.

一开始，我们需要选择这个 list 本身，这个 list 有一个 ID 叫"orderedlist"，通常的 javascript 写法是 document.getElementById("orderedlist").在 jQuery 中，我们这样做：

```
$(document).ready(function() {
$("#orderedlist").addClass("red");
});
```

这里将 starterkit 中的一个 CSS 样式 red 附加到了 orderedlist 上(译者 Keel 注：

参考测试包中的 css 目录下的 core.css，其中定义了 red 样式)。因此，在你刷新了 starterkit.html 后，你将会看到第一个有序列表(ordered list)背景色变成了红色，而第二个有序列表没有变化.

现在，让我们添加一些新的样式到 list 的子节点.

```
$(document).ready(function() {
$("#orderedlist >
li").addClass("blue");
});
```

这样，所有 orderedlist 中的 li 都附加了样式 blue。

现在我们再做个复杂一点的，当把鼠标放在 li 对象上面和移开时进行样式切换，但只在 list 的最后一个 element 上生效。

```
$(document).ready(function() {
$("#orderedlist
li:last").hover(function() {
$(this).addClass("green");
}, function() {
$(this).removeClass("green");
});
});
```

还有大量的类似的 **CSS** 和 **XPath** 例子，更多的例子和列表可以在这里找到。（译者 Keel 注：入门看此文，修行在个人，要想在入门之后懂更多，所以这段话的几个链接迟早是要必看的！不会又要翻译吧...^_^!）

每一个 onXXX 事件都有效，如 onclick,onchange,onsubmit 等，都有 jQuery 等价表示方法（译者 Keel 注：jQuery 不喜欢 onXXX，所以都改成了 XXX，去掉了 on）。其他的一些事件，如 ready 和 hover,也提供了相应的方法。

你可以在 **Visual jQuery** 找到全部的事件列表，在 Events 栏目下。

用这些选择器和事件你已经可以做很多的事情了，但这里有一个更强的好东东！

```
$(document).ready(function() {
$("#orderedlist").find("li").each(function(i) {
$(this).html(
$(this).html() + " BAM! " + i );
});
});
```

find() 让你在已经选择的element中作条件查找,因此 $("#orderedlist).find("li") 就像 $("#orderedlist li"). each()一样迭代了所有的 li，并可以在此基础上作更多的处理。 大部分的方法,如 addClass(), 都可以用它们自己的 each() 。在这个例子中, html()用来获取每个 li 的 html 文本, 追加一些文字，并将之设置为 li 的 html 文本。（译者 Keel 注：从这个例子可以看到.html()方法是获取对象的 html 代码，而.html('xxx')是设置'xxx'为对象的 html 代码）

另一个经常碰到的任务是在没有被 jQuery 覆盖的 DOM 元素上 call 一些方法，想像一个在你用 AJAX 方式成功提交后的 reset：

```
$(document).ready(function() {
// use this to reset a
single form
$("#reset").click(function() {
$("#form")[0].reset();
});
});
```

（译者 Keel 注：这里作者将 form 的 id 也写成了 form，源文件有<form

id="form">，这是非常不好的写法，你可以将这个 ID 改成 form1 或者 testForm，然后用$("#form1")或者$("#testForm")来表示它，再进行测试。）

这个代码选择了所有 ID 为"form"的元素，并在其第一个上 call 了一个 reset()。如果你有一个以上的 form，你可以这样做：

```
$(document).ready(function() {
// use this to reset
several forms at once
$("#reset").click(function() {
$("form").each(function() {
this.reset();
});
});
});
```

（译者 Keel 注：请注意一定要亲自将这些代码写在 custom.js 中并在 starterkit.html 上测试效果才能有所体会！必要时要观察 starterkit.html 的 html 代码）

这样你在点击 Reset 链接后，就选择了文档中所有的 form 元素，并对它们都执行了一次 reset()。

还有一个你可能要面对的问题是不希望某些特定的元素被选择。jQuery 提供了 filter() 和 not() 方法来解决这个问题。 filter()以过滤表达式来减少不符合的被选择项, not()则用来取消所有符合过滤表达式的被选择项. 考虑一个无序的 list，你想要选择所有的没有 ul 子元素的 li 元素。

```
$(document).ready(function() {
$("li").not("[ul]").css("border", "1px solid black");
});
```

这个代码选择了所有的 li 元素，然后去除了没有 ul 子元素的 li 元素。刷新浏览器后，所有的 li 元素都有了一个边框，只有 ul 子元素的那个 li 元素例外。

（译者 Keel 注：请注意体会方便之极的 css()方法，并再次提醒请务必实际测试观察效果，比方说换个 CSS 样式呢？再加一个 CSS 样式呢？像这样：$("li").not("[ul]").css("border", "1px solid black").css("color","red");）

上面代码中的[expression] 语法是从 XPath 而来，可以在子元素和属性(elements and attributes)上用作过滤器，比如你可能想选择所有的带有 name 属性的链接：

```
$(document).ready(function() {
$("a[@name]").background("#eee");
});
```

这个代码给所有带有 name 属性的链接加了一个背景色。（译者 Keel 注：这个颜色太不明显了，建议写成$("a[@name]").background("red");）

更常见的情况是以 name 来选择链接，你可能需要选择一个有特点 href 属性的链接，这在不同的浏览器下对 href 的理解可能会不一致，所以我们的部分匹配("*=")的方式来代替完全匹配("="):

```
$(document).ready(function() {
$("a[@href*=/content/gallery]").click(function() {
// do something with
all links that point somewhere to /content/gallery
});
});
```

到现在为止，选择器都用来选择子元素或者是过滤元素。另外还有一种情况是选择上一个或者下一个元素，比如一个 FAQ 的页面，答案首先会隐藏，当问题点击时，答案显示出来，jQuery 代码如下：

```
$(document).ready(function() {
$('#faq').find('dd').hide().end().find('dt').click(function() {
var
answer = $(this).next();
if (answer.is(':visible')) {
answer.slideUp();
} else {
answer.slideDown();
}
});
});
```

这里我们用了一些链式表达法来减少代码量，而且看上去更直观更容易理解。像 '#faq' 只选择了一次，利用 end()方法，第一次 find()方法会结束(undone)，所以我们可以接着在后面继续 find('dt')，而不需要再写$('#faq').find('dt')。

在点击事件中的，我们用 $(this).next() 来找到 dt 下面紧接的一个 dd 元素，这让我们可以快速地选择在被点击问题下面的答案。
（译者 Keel 注：这个例子真是太酷了，FAQ 中的答案可以收缩！从利用 next()的思路到实现这些效果都有很多地方需要我们消化，注意 if (answer.is(':visible'))用法，注意 answer.slideUp();不懂的地方赶紧查我在最开始提到的两个必看 API 文档）

除了选择同级别的元素外，你也可以选择父级的元素。可能你想在用户鼠标移到文章某段的某个链接时，它的父级元素--也就是文章的这一段突出显示，试试这个：

```
$(document).ready(function() {
$("a").hover(function() {
$(this).parents("p").addClass("highlight");
}, function() {
$(this).parents("p").removeClass("highlight");
```

```
});
});
```

测试效果可以看到，移到文章某段的链接时，它所在的段全用上 highlight 样式，移走之后又恢复原样。

（译者 Keel 注：highlight 是 core.css 中定义的样式，你也可以改变它，注意这里有第二个 function()这是 hover 方法的特点，请在 API 文档中查阅 hover，上面也有例子说明）

在我们继续之前我们先来看看这一步： jQuery 会让代码变得更短从而更容易理解和维护，下面是$(document).ready(callback)的缩写法：

```
$(function() {
// code to execute when the DOM is ready
});
```

应用到我们的 Hello world 例子中，可以这样：

```
$(function() {
$("a").click(function() {
alert("Hello world!");
});
});
```

现在，我们手上有了这些基础的知识，我们可以更进一步的探索其它方面的东西，就从 AJAX 开始！

感兴趣的链接：

- **jQuery API documentation**
- **Visual jQuery - A categorized browsable API documentation**
- **jQuery Expressions: CSS**

- jQuery Expressions: XPath
- jQuery Expressions: Custom
- jQuery Special Events
- jQuery DOM Traversing

下一篇:http://keelsike.blogspot.com/2006/12/jqueryjquery_11.html

上一篇:http://keelsike.blogspot.com/2006/12/jqueryjquery.html

posted by Keel @ 11:01 PM **0 comments**

2006-12-08

jQuery 中文入门指南，翻译加实例，jQuery 的起点教程(一)

jQuery 出来已经这么长时间了，在 GOOGLE 搜索还是那么点东西，15 天那个系列其实只是一个介绍 jQuery 特性的文章，很多人却把它当作一个教程来翻译，甚至还只是翻译计划。至于现实用到 jQuery 的应用，就更少了，比起 Prototype 来，国内对 jQuery 的进展也太不到位了吧，呵呵：）

其实最好的入门文章(教程)在这里：
http://jquery.bassistance.de/jquery-getting-started.html ，感谢 Jörn(http://bassistance.de/)

我仔细看了这个文章，觉得跟着这个走一遍的话，jQuery 就算入门了，此文以实例为基础一步步说明了 jQuery 的工作方式。现在我将翻译（添加我的补充说明）如下。转载的话请麻烦写个回复告知。本文发布已征求原作者同意。

另外我认为在学习过程中,有两个 API 文档你要打开随时查看:
http://jquery.com/api/
http://visualjquery.com/

----------以下为原文翻译----------

jQuery 入门指南

这个指南是一个对 jQuery 库的说明，要求读者了解 DOM 的一些常识。它包括了一个简单的 Hello World 的例子，选择器和事件基础，AJAX、FX 的用法，以及如何制作 jQuery 的插件。

这个指南包括了很多代码，你可以 copy 它们，并试着修改它们，看看产生的效果。

内容提要

一.安装

一开始,我们需求一个 jQuery 的库,最新的下载可以到这里找到。

这个指南提供一个基本包含实例的包供下载.

下载: **jquery-starterkit**

(译者 Keel 注:一定要下载这个包，光看文章不实践肯定是不行的。)

下载后解压缩，然后用你最喜欢的文本编辑器打开 starterkit.html 和 custom.js

这两个文件。

(译者 Keel 注:这两个就是例子文件,所有的例子都用这两个例子作出,custom.js 写 jQuery 代码,starterkit.html 观察效果.建议用 editPlus 打开)

现在,我们就已经做好了一切准备来进行这个著名的"Hello world"例子.

二.**Hello jQuery** 在做所有事情之前,我们要让 jQuery 读取和处理文档的 DOM,必须尽可能快地在 DOM 载入后开始执行事件,所以,我们用一个 ready 事件作为处理 HTML 文档的开始.看看我们打开的 custom.js 这个文件,里面已经准备好了:

```
$(document).ready(function() {
// do stuff when DOM is ready//当文档载入后从此处开始执行代码
});
```

下面我们放一个简单的 alert 事件在一个 function 中,因为这个 alert 不需求等 DOM 完成载入,所以我们把任务稍稍变复杂一点:在点击任何一个链接时显示一个 alert.

```
$(document).ready(function() {
$("a").click(function() {
alert("Hello world!");
});
});
```

这样在你点击页面的一个链接时都会触发这个"Hello world"的提示。

(译者 Keel 注:请照此代码修改 custom.js 并保存,然后用浏览器打开 starterkit.html 观察效果。)

让我们看一下这些修改是什么含义。$("a") 是一个 jQuery 选择器(selector),在这里，它选择所有的 a 标签（**译者 Keel 注：即<a>**），$号是 jQuery

"类"(jQuery "class")的一个别称，因此$()构造了一个新的 jQuery 对象(jQuery object)。函数 click() 是这个 jQuery 对象的一个方法，它绑定了一个单击事件到所有选中的标签(这里是所有的 a 标签),并在事件触发时执行了它所提供的 alert 方法.

这里有一个拟行相似功能的代码:

```
<a href="#" onclick="alert('Hello world')">Link</a>
```

不同之处很明显,用 jQuery 不需要在每个 a 标签上写 onclick 事件,所以我们拥有了一个整洁的结构文档(HTML)和一个行为文档(JS),达到了将结构与行为分开的目的,就像我们使用 CSS 追求的一样.

下面我们会更多地了解到选择器与事件.

感兴趣的链接:

jQuery Base

jQuery Expressions

jQuery Basic Events

下 一

篇:**http://keelsike.blogspot.com/2006/12/jqueryjquery_10.html**

*posted by Keel @ **2:13 AM** **0 comments***

2006-10-11

试用了 google 的 docs,很不错!

http://docs.google.com

提供在线文档和在线电子表格服务,试用了一下,还是比较不错的!

*posted by Keel @ **9:38 PM** **0 comments***

2006-10-10

原 blog.k99k.com 暂时关闭

因为某些原因,原 blog.k99k.com 暂时关闭,域名不久会直接指向这里.

待年底,我的 CMS 系统出来后,blog.k99k.com 将重见光明!

posted by Keel @ ***5:48 PM*** **0 comments**

2006-09-28

不错的 CSS 产生阴影效果

```
<style type="text/css">
<!--
.shadowbox {
background: #ccc;
position: relative;
top: 2px;
left: 2px;
}
.shadowbox div {
background: #333;
border: 8px solid #000;/*可调*/
color: #fff;
padding: 10px;
position: relative;
top: -8px;/*可调*/
left: -8px;/*可调*/
}
-->
</style>

<div class="shadowbox">
<div>里面的 div 标签必须要有</div>
</div>
```

例子:

里面的 div 标签必须要有

*posted by Keel @ **1:10 AM** **0 comments***

CSS 切换实例解析[转]

作者：Linyupark / 2006-04-07

实例的几个文件介绍

整个例子用到三个文件(没有把 JS 分离出来,在实际运用中应该分离):

1.css 这个文件设置了 h1 的字体大小为 80px

2.css 将 h1 的字体设置为 20px

createStyleSheet.html 头部写进了 JS 代码

点这查看效果

具体代码分析

CSS 文件的内容如下,所表达的意思在上面已经说了,本身没什么特别的,只是为了突出下变化的效果:/* 1.css 的文件内容 */

```
h1{
font-family:"Trebuchet MS";
font-size:80px;
}
/* 2.css 的文件内容 */
h1{
font-family:"Trebuchet MS";
font-size:20px;
}
```

现在我们开始分析重头,createStyleSheet.html 文件中几个重要的部分:

1.导入 CSS 文件的语句

该语句中可以发现并没有定义 href 属性,因为在后面我们要根据选择不同的样式来给它添加进去,多了一个 ID 属性是为了能使用 getElementById 来将这个语句定义为一个对象来使用它(后面会涉及到):<link rel="stylesheet" id="CSSC" type="text/css">

2.JS 部分<script language="javascript">

```
var i,objCSS,cssname;
objCSS = document.getElementById("CSSC");
/*
上面定义的 i 为 CSS 文件后缀前的名称字符串,在这个实例中它的值分 1 和 2 两种情况
objCSS 则为前面提到的 CSS 连接语句的对象.
*/
function change(i){
setCookie('cssname',i,365);
objCSS.setAttribute("href",i+".css");
}
/*
以上为函数 change(i),它的职能是当有事件触发这个函数时,
设置浏览器 cookie 中 cssname 的值为 i,
并且该 cookie 过期时间为 365 天(具体的设置过程使用了函数 setCookie).
并给 objCSS 对象加上了属性 href,它的值为 i.css
*/
function checkStyle()
{
cssname=getCookie('cssname');
if (cssname!=null)
{
objCSS.setAttribute("href",cssname+".css");
}
}
/*
```

checkStyle()这个函数在页面加载的时候就执行,
其目的就是判断当前是否存在 cookie 保存的样式信息,
有的话就直接设置所保存的样式.
达成这个目的使用到了 getCookie 函数,
当返回的结果不为 null 的时候就执行设置样式的代码.
*/

```
function setCookie(c_name,value,expiredays)
{
var exdate=new Date();
exdate.setDate(expiredays);
document.cookie=c_name+ "=" +escape(value)+((expiredays==null) ? "" :
";expires="+exdate);
}
/*
setCookie 函数就起到保存信息的作用,里面包含了三个参数:
c_name 用来指定是保存名为什么的 cookie,并依据这个名称来做以后的调用
value 就是这个 cookie 实际要保存的值
sepiredays 是设置过期的时间,
在这里它还做了一个判断,如果不写这个过期的时间就表示不进行保存
*/
function getCookie(c_name)
{
if (document.cookie.length>0)
{
c_start=document.cookie.indexOf(c_name + "=");
if (c_start!=-1)
{
c_start=c_start + c_name.length+1;
c_end=document.cookie.indexOf(";",c_start);
if (c_end==-1) c_end=document.cookie.length
return unescape(document.cookie.substring(c_start,c_end));
}
```

```
}
return null
}
/*
这就是对现有 cookie 做判断的函数,为 null 就返回一个 null,
不为 null 就返回指定 cookie 的值
*/
</script>
3.html 代码 <body onload="checkStyle()"> <!--表示在页面加载时候就执行这
个函数-->
<input type="button" onclick="change(1)" value="改变成 1 样式">
<input type="button" onclick="change(2)" value="改变成 2 样式">
<h1>hahahahahhahahaah</h1>
</body>
```

jQuery 中文入门指南，翻译加实例，jQuery 的起点教程

中文版译者：Keel

此文以实例为基础一步步说明了 jQuery 的工作方式。现以中文翻译（添加我的补充说明）如下。如有相关意见或建议请麻烦到我的 BLOG 写个回复或者 EMAIL 告知。

英文原版：http://jquery.bassistance.de/jquery-getting-started.html ，感谢原文作者 Jörn Zaefferer

本文发布已征求原作者同意。

另外我认为在学习过程中,有两个 API 文档你要打开随时查看:

- **http://jquery.com/api/**
- **http://visualjquery.com/**

以下部分为原文翻译:

---

jQuery 入门指南教程

这个指南是一个对 jQuery 库的说明，要求读者了解 HTML(DOM)和 CSS 的一些常识。它包括了一个简单的 Hello World 的例子，选择器和事件基础，AJAX、FX 的用法，以及如何制作 jQuery 的插件。 这个指南包括了很多代码，你可以 copy 它们，并试着修改它们，看看产生的效果。

内容提要

8. **Next steps(下一步)**

安装

一开始,我们需要一个 jQuery 的库,最新的下载可以到这里找到。这个指南提供一个基本包含实例的包供下载.

下载：jQuery Starterkit

(译者 Keel 注:一定要下载这个包，光看文章不实践肯定是不行的。)

下载后解压缩，然后用你最喜欢的文本编辑器打开 starterkit.html 和 custom.js 这两个文件。(译者 Keel 注:这两个就是例子文件,所有的例子都用这两个例子作出,custom.js 写 jQuery 代码,starterkit.html 观察效果.建议用 editPlus 打开)

现在,我们就已经做好了一切准备来进行这个著名的"Hello world"例子.

本章的相关链接:

- **Starterkit**
- **jQuery Downloads**

Hello jQuery

在做所有事情之前,我们要让 jQuery 读取和处理文档的 DOM,必须尽可能快地在 DOM 载入后开始执行事件,所以,我们用一个 ready 事件作为处理 HTML 文档的开始.看看我们打开的 custom.js 这个文件,里面已经准备好了:

```
$(document).ready(function() {
        // do stuff when DOM is ready
});
```

放一个简单的 alert 事件在需要等 DOM 完成载入,所以我们把任务稍稍变复杂一点:在点击任何一个链接时显示一个 alert.

```
$(document).ready(function() {
        $("a").click(function() {
                alert("Hello world!");
        });
});
```

这样在你点击页面的一个链接时都会触发这个"Hello world"的提示。

(译者 Keel 注:请照此代码修改 custom.js 并保存,然后用浏览器打开 starterkit.html 观察效果。)

让我们看一下这些修改是什么含义。$("a") 是一个 jQuery 选择器(selector),在这里，它选择所有的 a 标签（译者 Keel 注：即<a>），$号是 jQuery “类”(jQuery "class")的一个别称，因此$()构造了一个新的 jQuery 对象(jQuery object)。函数 click() 是这个 jQuery 对象的一个方法，它绑定了一个单击事件到所有选中的标签(这里是所有的 a 标签),并在事件触发时执行了它所提供的 alert 方法.

这里有一个拟行相似功能的代码:

```
<a href="#" onclick="alert('Hello world')">Link</a>
```

不同之处很明显,用 jQuery 不需要在每个 a 标签上写 onclick 事件,所以我们拥有了一个整洁的结构文档(HTML)和一个行为文档(JS),达到了将结构与行为分开的目的,就像我们使用 CSS 追求的一样.

下面我们会更多地了解到选择器与事件.

本章的相关链接:

- jQuery Base
- jQuery Expressions
- jQuery Basic Events

Find me:使用选择器和事件

jQuery 提供两种方式来选择 html 的 elements,第一种是用 CSS 和 Xpath 选择器联合起来形成一个字符串来传送到 jQuery 的构造器（如：$("div > ul a")）；第二种是用 jQuery 对象的几个 methods(方法)。这两种方式还可以联合起来混合使用。

为了测试一下这些选择器,我们来试着在我们 starterkit.html 中选择并修改第一个 ordered list.

一开始,我们需要选择这个 list 本身,这个 list 有一个 ID 叫“orderedlist”，通常的 javascript 写法是 document.getElementById("orderedlist").在 jQuery 中，我们这样做:

```
$(document).ready(function() {
        $("#orderedlist").addClass("red");
});
```

这里将 starterkit 中的一个 CSS 样式 red 附加到了 orderedlist 上(译者 Keel 注：参考测试包中的 css 目录下的 core.css，其中定义了 red 样式)。因此，在你刷新了 starterkit.html 后，你将会看到第一个有序列表(ordered list)背景色变成了红色，而第二个有序列表没有变化.

现在，让我们添加一些新的样式到 list 的子节点.

```
$(document).ready(function() {
        $("#orderedlist > li").addClass("blue");
});
```

这样，所有 orderedlist 中的 li 都附加了样式"blue"。

现在我们再做个复杂一点的，当把鼠标放在 li 对象上面和移开时进行样式切换，但只在 list 的最后一个 element 上生效。

```
$(document).ready(function() {
        $("#orderedlist li:last").hover(function() {
                $(this).addClass("green");
        }, function() {
                $(this).removeClass("green");
        });
});
```

还有大量的类似的 CSS 和 XPath 例子，更多的例子和列表可以在这里找到。（译者 Keel 注：入门看此文，修行在个人，要想在入门之后懂更多，所以这段话的几个链接迟早是要必看的！不会又要翻译吧...^_^!）

每一个 onXXX 事件都有效，如 onclick,onchange,onsubmit 等，都有 jQuery 等价表示方法（译者 Keel 注：jQuery 不喜欢 onXXX，所以都改成了 XXX，去掉了 on）。其他的一些事件，如 ready 和 hover,也提供了相应的方法。

你可以在 Visual jQuery 找到全部的事件列表，在 Events 栏目下.

用这些选择器和事件你已经可以做很多的事情了，但这里有一个更强的好东东！

```
$(document).ready(function() {
        $("#orderedlist").find("li").each(function(i) {
                $(this).html( $(this).html() + " BAM! " + i );
        });
});
```

find() 让你在已经选择的 element 中作条件查找,因此 $("#orderedlist).find("li") 就像 $("#orderedlist li"). each()一样迭代了所有的 li，并可以在此基础上作更多的处理。 大部分的方法,如 addClass(),都可以用它们自己的 each() 。在这个例子中，html()用来获取每个 li 的 html 文本，追加一些文字，并将之设置为 li 的 html 文本。（译者 Keel 注：从这个例子可以看到.html()方法是获取对象的 html 代码，而.html('xxx')是设置'xxx'为对象的 html 代码）

另一个经常碰到的任务是在没有被 jQuery 覆盖的 DOM 元素上 call 一些方法,想像一个在你用 AJAX 方式成功提交后的 reset：

```
$(document).ready(function() {
        // use this to reset a single form
        $("#reset").click(function() {
                $("#form")[0].reset();
```

```
        });
});
```

（译者 Keel 注：这里作者将 form 的 id 也写成了 form，源文件有<form id="form">，这是非常不好的写法，你可以将这个 ID 改成 form1 或者 testForm，然后用$("#form1")或者$("#testForm")来表示它，再进行测试。）

这个代码选择了所有 ID 为"form"的元素，并在其第一个上 call 了一个 reset()。如果你有一个以上的 form，你可以这样做：

```
$(document).ready(function() {
        // use this to reset several forms at once
        $("#reset").click(function() {
                $("form").each(function() {
                        this.reset();
                });
        });
});
```

（译者 Keel 注：请注意一定要亲自将这些代码写在 custom.js 中并在 starterkit.html 上测试效果才能有所体会！必要时要观察 starterkit.html 的 html 代码）

这样你在点击 Reset 链接后，就选择了文档中所有的 form 元素，并对它们都执行了一次 reset()。

还有一个你可能要面对的问题是不希望某些特定的元素被选择。jQuery 提供了 filter() 和 not() 方法来解决这个问题。 filter()以过滤表达式来减少不符合的被选择项，not()则用来取消所有符合过滤表达式的被选择项. 考虑一个无序的 list，你想要选择所有的没有 ul 子元素的 li 元素。

```
$(document).ready(function() {
        $("li").not("[ul]").css("border", "1px solid black");
});
```

这个代码选择了所有的 li 元素，然后去除了没有 ul 子元素的 li 元素。刷新浏览器后，所有的 li 元素都有了一个边框，只有 ul 子元素的那个 li 元素例外。

（译者 Keel 注：请注意体会方便之极的 css()方法，并再次提醒请务必实际测试观察效果，比方说换个 CSS 样式呢？再加一个 CSS 样式呢？像这样：
$("li").not("[ul]").css("border", "1px solid black").css("color","red");）

上面代码中的[expression] 语法是从 XPath 而来，可以在子元素和属性(elements and attributes)上用作过滤器，比如你可能想选择所有的带有 name 属性的链接:

```
$(document).ready(function() {
        $("a[@name]").background("#eee");
});
```

这个代码给所有带有 name 属性的链接加了一个背景色。（译者 Keel 注：这个颜色太不明显了，建议写成$("a[@name]").background("red");）

更常见的情况是以 name 来选择链接，你可能需要选择一个有特点 href 属性的链接，这在不同的浏览器下对 href 的理解可能会不一致，所以我们的部分匹配("*=")的方式来代替完全匹配("=")：

```
$(document).ready(function() {
        $("a[@href*=/content/gallery]").click(function() {
                // do something with all links that point somewhere to
/content/gallery
        });
});
```

到现在为止，选择器都用来选择子元素或者是过滤元素。另外还有一种情况是选择上一个或者下一个元素，比如一个 FAQ 的页面，答案首先会隐藏，当问题点击时，答案显示出来，jQuery 代码如下：

```
$(document).ready(function() {
        $('#faq').find('dd').hide().end().find('dt').click(function()
{
            var answer = $(this).next();
            if (answer.is(':visible')) {
                answer.slideUp();
            } else {
                answer.slideDown();
            }
        });
});
```

这里我们用了一些链式表达法来减少代码量，而且看上去更直观更容易理解。像'#faq' 只选择了一次，利用 end()方法，第一次 find()方法会结束(undone)，所以我们可以接着在后面继续 find('dt')，而不需要再写 $('#faq').find('dt')。

在点击事件中的，我们用 $(this).next() 来找到 dt 下面紧接的一个 dd 元素，这让我们可以快速地选择在被点击问题下面的答案。

（译者 Keel 注：这个例子真是太酷了，FAQ 中的答案可以收缩！从利用 next()的思路到实现这些效果都有很多地方需要我们消化，注意 if (answer.is(':visible'))用法，注意 answer.slideUp();不懂的地方赶紧查我在最开始提到的两个必看 API 文档）

除了选择同级别的元素外，你也可以选择父级的元素。可能你想在用户鼠标移到文章某段的某个链接时，它的父级元素--也就是文章的这一段突出显示，试试这个：

```
$(document).ready(function() {
        $("a").hover(function() {
                $(this).parents("p").addClass("highlight");
        }, function() {
                $(this).parents("p").removeClass("highlight");
        });
});
```

测试效果可以看到，移到文章某段的链接时，它所在的段全用上 highlight 样式，移走之后又恢复原样。

（译者 Keel 注：highlight 是 core.css 中定义的样式，你也可以改变它，注意这里有第二个 function()这是 hover 方法的特点，请在 API 文档中查阅 hover，上面也有例子说明）

在我们继续之前我们先来看看这一步： jQuery 会让代码变得更短从而更容易理解和维护，下面是$(document).ready(callback)的缩写法：

```
$(function() {
        // code to execute when the DOM is ready
});
```

应用到我们的 Hello world 例子中，可以这样：

```
$(function() {
        $("a").click(function() {
                alert("Hello world!");
        });
});
```

现在，我们手上有了这些基础的知识，我们可以更进一步的探索其它方面的东西，就从 AJAX 开始！

本章的相关链接:

- **jQuery API documentation**
- **Visual jQuery - A categorized browsable API documentation**
- **jQuery Expressions: CSS**
- **jQuery Expressions: XPath**
- **jQuery Expressions: Custom**
- **jQuery Special Events**
- **jQuery DOM Traversing**

Rate me:使用 AJAX

在这一部分我们写了一个小小的 AJAX 应用，它能够 rate 一些东西（译 Keel 注：就是对某些东西投票），就像在 youtube.com 上面看到的一样。

首先我们需要一些服务器端代码，这个例子中用到了一个 PHP 文件，读取 rating 参数然后返回 rating 总数和平均数。看一下 rate.php 代码.

虽然这些例子也可以不使用 AJAX 来实现，但显示我们不会那么做，我们用 jQuery 生成一个 DIV 容器， ID 是"rating".

```
$(document).ready(function() {
        // generate markup
        var ratingMarkup = ["Please rate: "];
        for(var i=1; i <= 5; i++) {
                ratingMarkup[ratingMarkup.length] = "<a href='#'>" + i
+ "</a> ";
        }
        // add markup to container and applier click handlers to anchors
        $("#rating").append( ratingMarkup.join('') ).find("a").click(
function(e) {
                e.preventDefault();
                // send requests
                $.post("rate.php", {rating: $(this).html()},
function(xml) {
                        // format result
                        var result = [
                                "Thanks for rating, current average: ",
                                $("average", xml).text(),
                                ", number of votes: ",
                                $("count", xml).text()
                        ];
                        // output result
                        $("#rating").html(result.join(''));
                } );
        });
});
```

这段代码生成了 5 个链接，并将它们追加到 id 为"rating"容器中，当其中一个链接被点击时，该链接标明的分数就会以 rating 参数形式发送到 rate.php，然后，结果将以 XML 形式会从服务器端传回来，添加到容器中，替代这些链接。

如果你没有一个安装过 PHP 的 webserver，你可以看看这个在线的例子.

不使用 javascript 实现的例子可以访问 softonic.de 点击 "Kurz bewerten!"

更多的 AJAX 方法可以从这里 找到，或者看看 API 文档 下面的 AJAX filed under AJAX.

（译者 Keel 注：这个在线实例从国内访问还是比较慢的，点击后要等一会儿才能看到结果，可以考虑对它进行修改，比如加上 loading，投票后加上再投票的返回链接等。此外，这个例子中还是有很多需要进一步消化的地方，看不懂的地方请参考 API 文档。）

一个在使用 AJAX 载入内容时经常发生的问题是：当载入一个事件句柄到一个 HTML 文档时，还需要在载入内容上应用这些事件，你不得不在内容加载完成后应用这些事件句柄，为了防止代码重复执行，你可能用到如下一个 function:

```
// lets use the shortcut
$(function() {
        var addClickHandlers = function() {
                $("a.clickMeToLoadContent").click(function() {
                        $("#target").load(this.href,
addClickHandlers);
                });
        };
        addClickHandlers();
});
```

现在，addClickHandlers 只在 DOM 载入完成后执行一次，这是在用户每次点击具有 clickMeToLoadContent 这个样式的链接并且内容加载完成后.

请注意 addClickHandlers 函数是作为一个局部变量定义的，而不是全局变量(如：function addClickHandlers() {...})，这样做是为了防止与其他的全局变量或者函数相冲突.

另一个常见的问题是关于回调(callback)的参数。你可以通过一个额外的参数指定回调的方法，简单的办法是将这个回调方法包含在一个其它的 function 中:

```
// get some data
var foobar = ...;
// specify handler, it needs data as a paramter
var handler = function(data) {
  ...
};
// add click handler and pass foobar!
$('a').click( function(event) { handler(foobar); } );

// if you need the context of the original handler, use apply:
$('a').click( function(event) { handler.apply(this, [foobar]); } );
```

用到简单的 AJAX 后，我们可以认为已经非常之“web2.0”了，但是到现在为止，我们还缺少一些酷炫的效果。下一节将会谈到这些效果.

本章的相关链接:

- jQuery AJAX Module
- jQuery API: Contains description and examples for append and all other jQuery methods
- ThickBox: A jQuery plugin that uses jQuery to enhance the famous lightbox

Animate me(让我生动起来):使用 FX

一些动态的效果可以使用 show() 和 hide()来表现:

```
$(document).ready(function() {
        $("a").toggle(function() {
                $(".stuff").hide('slow');
        }, function() {
                $(".stuff").show('fast');
        });
});
```

你可以与 animate()联合起来创建一些效果,如一个带渐显的滑动效果:

```
$(document).ready(function() {
        $("a").toggle(function() {
                $(".stuff").animate({
                        height: 'hide',
                        opacity: 'hide'
                }, 'slow');
        }, function() {
                $(".stuff").animate({
                        height: 'show',
                        opacity: 'show'
                }, 'slow');
        });
});
```

很多不错的效果可以访问 interface plugin collection. 这个站点提供了很多 demos 和文档

这些效果插件是位于 jQuery 插件列表的前面的，当然也有很多其他的插件，比如我们下一章讲到的表格排序插件。

本章的相关链接:

- jQuery FX Module
- Interface plugin

Sort me(将我有序化):使用 tablesorter 插件(表格排序)

这个表格排序插件能让我们在客户端按某一列进行排序，引入 jQuery 和这个插件的 js 文件，然后告诉插件你想要哪个表格拥有排序功能。

要测试这个例子，先在 starterkit.html 中加上像下面这一行的代码：

```
<script src="lib/jquery.tablesorter.js"
type="text/javascript"></script>
```

然后可以这样调用不着:

```
$(document).ready(function() {
        $("#large").tableSorter();
});
```

现在点击表格的第一行 head 区域，你可以看到排序的效果，再次点击会按倒过来的顺序进行排列。

这个表格还可以加一些突出显示的效果，我们可以做这样一个隔行背景色（斑马线）效果:

```
$(document).ready(function() {
        $("#large").tableSorter({
                stripingRowClass: ['odd','even'],       // Class names
for striping supplyed as a array.
                stripRowsOnStartUp: true                // Strip rows on
tableSorter init.
        });
});
```

关于这个插件的更多例子和文档可以在 tablesorter 首页找到.

几乎所有的特件都是这样用的:先 include 插件的 js 文件,然后在某些元素上使用插件定义的方法,当然也有一些参数选项是可以配置的

经常更新的插件列表可以从 jQuery 官方站 on the jQuery site 找到.

当你更经常地使用 jQuery 时,你会发现将你自己的代码打包成插件是很有用处的,它能方便地让你的公司或者其他人进行重用.下一章我们将谈到如何构建一个自己的插件.

本章的相关链接:

- Plugins for jQuery
- Tablesorter Plugin

Plug me:制作自己的插件

写一个自己的 jQuery 插件是非常容易的, 如果你按照下面的原则来做, 可以让其他人也容易地结合使用你的插件.

1. 为你的插件取一个名字,在这个例子里面我们叫它"foobar".
2. 创建一个像这样的文件:jquery.[yourpluginname].js,比如我们创建一个 jquery.foobar.js
3. 创建一个或更多的插件方法,使用继承 jQuery 对象的方式,如:

```
4.    jQuery.fn.foobar = function() {
5.            // do something
    };
```

6. 可选的:创建一个用于帮助说明的函数,如:

```
7.    jQuery.fooBar = {
8.            height: 5,
9.            calculateBar = function() { ... },
10.           checkDependencies = function() { ... }
    };
```

你现在可以在你的插件中使用这些帮助函数了:

```
jQuery.fn.foobar = function() {
  // do something
  jQuery.foobar.checkDependencies(value);
  // do something else
};
```

11. 可选的 1:创建一个默认的初始参数配置,这些配置也可以由用户自行设定,如:

```
12.  jQuery.fn.foobar = function(options) {
13.          var settings = {
14.                  value: 5,
15.                  name: "pete",
16.                  bar: 655
17.          };
18.          if(options) {
19.                  jQuery.extend(settings, options);
20.          }
    };
```

现在可以无需做任何配置地使用插件了,默认的参数在此时生效:

```
$("...").foobar();
```

或者加入这些参数定义:

```
$("...").foobar({
```

```
        value: 123,
        bar: 9
    });
```

如果你 release 你的插件，你还应该提供一些例子和文档，大部分的插件都具备这些良好的参考文档.

现在你应该有了写一个插件的基础，让我们试着用这些知识写一个自己的插件.

很多人试着控制所有的 radio 或者 checkbox 是否被选中，比如：

```
$("input[@type='checkbox']").each(function() {
        this.checked = true;
        // or, to uncheck
        this.checked = false;
        // or, to toggle
        this.checked = !this.checked;
});
```

无论何时候，当你的代码出现 each 时，你应该重写上面的代码来构造一个插件，很直接地：

```
$.fn.check = function() {
        return this.each(function() {
                this.checked = true;
        });
};
```

这个插件现在可以这样用：

```
$("input[@type='checkbox']").check();
```

现在你应该还可以写出 uncheck()和 toggleCheck()了.但是先停一下，让我们的插件接收一些参数.

```
$.fn.check = function(mode) {
        var mode = mode || 'on'; // if mode is undefined, use 'on' as
default
        return this.each(function() {
                switch(mode) {
                case 'on':
                        this.checked = true;
                        break;
case 'off':
                        this.checked = false;
                        break;
case 'toggle':
                        this.checked = !this.checked;
                        break;
}
```

```
        });
};
```

这里我们设置了默认的参数,所以将"on"参数省略也是可以的,当然也可以加上"on","off", 或 "toggle",如:

```
$("input[@type='checkbox']").check();
$("input[@type='checkbox']").check('on');
$("input[@type='checkbox']").check('off');
$("input[@type='checkbox']").check('toggle');
```

如果有多于一个的参数设置会稍稍有点复杂,在使用时如果只想设置第二个参数,则要在第一个参数位置写入 null.

从上一章的 tablesorter 插件用法我们可以看到,既可以省略所有参数来使用或者通过一个 key/value 对来重新设置每个参数.

作为一个练习,你可以试着将 第四章 的功能重写为一个插件.这个插件的骨架应该是像这样的:

```
$.fn.rateMe = function(options) {
        var container = this; // instead of selecting a static container with $("#rating"), we now use the jQuery context

        var settings = {
                url: "rate.php"
                // put more defaults here
                // remember to put a comma (",") after each pair, but not after the last one!
        };

        if(options) { // check if options are present before extending the settings
                $.extend(settings, options);
        }

        // ...
        // rest of the code
        // ...

        return this; // if possible, return "this" to not break the chain
});
```

Next steps(下一步)

如果你想做更好的 javascript 开发,建议你使用一个叫 FireBug 的 firefox 插件. 它提供了断点调试(比 alert 强多了)、观察 DOM 变化等很多漂亮的功能

如果你还有未解决的问题，或者新的想法与建议，你可以使用 jQuery 的邮件列表 jQuery mailing list.

关于这个指南的任何事情，你可以写 mail 给作者或者发表评论在他的日志：blog.

关于这个指南的翻译任何事情，你可以写 mail 给我或者发表评论在我的日志：blog.

还有什么...

大大感谢 John Resig 创造了这么好的 library！ 感谢 jQuery community 为 John 提供了如此多的咖啡和其他的一切!

中文版翻译: Keel **- 最后更新: 2006-12-13**