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 J?rn 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

5. Animate me(让我生动起来):使用 FX
6. Sort me(将我有序化):使用 tablesorter 插件(表格排序)
7. Plug me:制作您自己的插件
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(f
unction(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:

```
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 l: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();
```

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