整理：Joychao　www.joychao.cc

5 天驾驭 JQuery 教程（jQuery 速成教程）

【第一节】jQuery 速成 - 向 jQuery 进军！

jQuery 其实很简单，相信您看了 W3Cfuns.com 精心编写的 jQuery 速成教程后，一定会被它的优雅、轻巧吸引住，而且如果您掌握了它，定能成为提升薪水的一个资本。

本节并没有涉及到 jQuery 如何编写，只为大家解决几个常见问题，因为这也是面试中常常遇到的几个问题。如果你掌握了 jQuery 可以成为提升工资的一个资本。

初学 jQuery 的朋友，基本上都会问同一个问题“什么是 jQuery？”等类似的问题，理解这个问题对于后面的学习会起到促进作用。以下是我整理出的大家常问的几个问题：

1. 什么是 **jQuery**？
2. 什么是类库？
3. **jQuery** 与 **JavaScript** 有什么关系？
4. **JavaScript** 与 **java** 又有什么关系？
5. **JavaScript**、**jQuery**、**Ajax**、**Json** 等又是什么？区别又是什么？
6. 不会 **JS** 能不能学 **jQuery**？

我相信，很多人对这些技术有着似懂非懂的感觉，不是很了解，以上问题肯定是初学者都会产生的问题，如果你对它有兴趣或者想去驾驭它。那么请跟 KwooShung 来，向 jQuery 进军！let's GO！

Q:什么是 jQuery？

A:jQuery 是继 Prototype 之后又一个优秀的 JavaScript 类库，它由美国人 John Resig 创建，至今已吸引来自世界各地众多的 JavaScript 高手加入其团队，其宗旨是：写更少的代码，做更多的事(write less,do more)

Q:什么是类库？

A:这是程序中的一个基本概念。所谓类，可以理解为是一组语句的集合，用来描述一组具有共同属性和功能的对象。字面理解也就是说类库就是类的集合。Java 和.net 的类库意思类似。但是在 jQuery 中，只是集合了许多的方法功能集合，使我们可以通过简单的代码就能实现复杂的效果。

Q:jQuery 与 JavaScript 有什么关系？

A:jQuery 是使用 JavaScript 编写的，也就是说 JavaScript 可以任意调用，其他程序并不能很轻易的调用。就好比 JavaScript 是鸡，jQuery 是鸡蛋，鸡只能生鸡蛋。再说的明白些，jQuery 是使用 JavaScript 编写的，就好比鸡蛋是鸡生的。

如果你还不明白再看看另一个例子：

我们可以把 JavaScript 比喻成文字，假如使用文字写了两本菜谱《西餐菜谱大全》和《中餐菜谱大全》，前者是 prototype（在 jQuery 之前也是一种类库）后者是 jQuery。菜谱里面都集合了很多做菜的方法也可以叫做菜的类库。在程序里面呢就叫做类库，方法库，函数库等。

Q:JavaScript 与 Java 又有什么关系？

A:乍眼一看，它们有关系或者是同一家公司的产品，其实它们俩既不是兄弟姐妹也不是同一家公司所开发的，前者是 Netscape 网景公司开发，后者是 Sun 公司开发，2009 年 04 月 20 日，Oracle（甲骨文)宣布以 74 亿美元收购 Sun。（详细的资料可以到网上搜一下）

Q:JavaScript、jQuery、Ajax、Json 等又是什么？区别又是什么？

A:读到这里，我想不用再讲 JS 和 jQuery 了吧？如果你还认为需要讲解，那你再读读前面的几个问题，肯定就明白了，直接切入正题。

Ajax：全称为“Asynchronous JavaScript and XML”（异步 JavaScript 和 XML），它是由 JavaScript+CSS+DOM+ XMLHttpRequest 的四种技术的结合,并且 JS 是 Ajax 的核心。jQuery 将 Ajax 的实现变得更加轻松容易。Ajax 就是咱们常说的局部刷新。

JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式。易于人阅读和编写的同时也易于机器解析和生成。它基于 JavaScript 的一个子集。 Json 采用完全独立于语言的文本格式，可以简单的理解为数据存储的一种格式或交换方式。

Q:不会 JS 能不能学 jQuery？

A:我不知道大家所指的不会 JS 是什么程度，但我认为如果你一点都不懂 JS 那是很难学懂的，因为 jQuery 是 JS 的类库。另外如果你懂得其他语言，也可以很快的理解的。如果什么都不会我可以给大家一个快速掌握 JS 基础的方法，先去看 JS 的数据类型、变量的定义、函数方法的定义、函数方法的调用以及循环和判断，只要知道上面这几种非常容易学会。看着内容挺多其实很简单，最慢一天就能搞懂。

注：下节课我们还是延续《跟 KwooJan 学习 CSS》的风格，以实例为主的通俗易懂的教程，请大家做好准备！

目前大家只需要准备好记事本、类库和您的键盘鼠标就可以啦~不过还是建议大家用编辑器进行开发，便于提高开发效率以及正确率，这里 KwooShung 建议大家使用 Dreamweaver CS4，版本一定要是 CS4，因为以后我们用到它的地方还很多。

相关开发工具下载：点击此处前往
jQuery 类库下载：点击此处前往

【第二节】jQuery 速成 - 核心！

jQuery 的核心是大家必须要学习的内容，我写了一个简单的例子，大家先不要看我的源代码，看看使用 js 是否能够做得出。

【例子】
要求：
1）页面上一个按钮；
2）点击后弹出窗口，我被点击了；

JavaScript 代码如下：

```
1.  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
2.  <html xmlns="http://www.w3.org/1999/xhtml">
3.  <head>
4.  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

```
5.  <title>实例 1</title>
6.  <script>
7.  function myClick()
8.  {
9.  alert("我被点击了！ ");
10. }
11. </script>
12. </head>
13. <body>
14. <input type="submit" value="请点击我" onclick="myClick();" />
15. </body>
16. </html>
```

然而 jQuery 与此类似，但是在 HTML 页中不必填写 onclick="myClick();" 事件调用 myClick()方法，只需要引入 jQuery 类库以及自己编写的 jQuery 代码即可。
下面是我的 jQuery 代码实现的上图方法

第 2 课.rar (34.67 KB, 下载次数: 360)

看完上面的 jQuery 例子以后，可以发现其实我们什么都不用在 HTML 页面中写，只需引入相关的 js 文件即可，这样做的优点可以使我们的 HTML 页面更加简单，不需要穿插复杂的 js 代码，从而使页面与脚本完美分离，是不是很神奇？

注：例题中的<script language="javascript" charset="GB2312" src="js/demo1.js"></script>加入了 charset="GB2312"是为了使其能够正确的显示中文，因为 jQuery 是 utf-8 编码。

同学们可能会问 demo1.js 中的"$"符号表示什么呢？KwooShung 建议大家不要小看了它哟~~~因为 jQuery 的核心功能都是通过这个函数实现的，以后编写的 jQuery 代码都是构建在这个函数上的，简单的说，就是都以某种方式在使用它。不过它为什么是个函数呢，明明就一个符号而已啊？

在此案例中，我简写了代码，他们的关系如下，区分大小写：

$等同于 jQuery
$(document).ready()等同于 jQuery(document).ready()
他们在例题中的意思都等同于 js 中的
onload()方法
click()表示鼠标点击事件，此事件日后会在详细讲解

本节课，只需要你明白"$"符号是什么就 OK 了，在后面的学习中你会对他逐渐的熟悉~怎么样，是不是很有意思？

【作业】
1、在页面中使用 css 定义一个长宽均为 100px DIV 红色区域
2、鼠标点击此红色区域，弹出对话框，至于对话框什么内容随便。

【第三节】jQuery 速成 - 核心方法的使用

学过上一节课的朋友我想都知道了$符号的作用了，那么，今天我们再深入的学习一下 jQuery 的核心。

jQuery 对象访问：

each(callback)

根据我的理解，each 是 jQuery 中的一种循环机制。一般与 this 关键字配合使用。学过程序的朋友都知道程序中的循环方式有以下几种 do...while()、while()、for(expression1,expression2,expression3)以及 C#和 javaEE 中 JSTL 标签中独有的 forEach 循环。jQuery 中的 each 循环与 forEach 循环类似。具体使用方法在本节案例中讲述。

$("Element").length

表示某个对象在 HTML 页面中的数量，与 size 用法一致，此方法不带有()。

$("Element").size()

表示某个对象在 HTML 页面中的数量，与 length 用法一致。

$("Element").get()

表示获得某个元素在 HTML 页面中的集合，以数组方式构建。

$("Element").get(index)

作用同上，如果 get 方法里面带有数字则表示获得数组中的第几个元素，索引从 0 开始。

$("Element").get().reverse()

表示将获取到得 dom 元素集合构建成的数组进行反向。比如默认排序是 1,2,3 使用了此方法则为 3,2,1

$("Element").index($("Element"))

搜索 index 中所获得的元素在所匹配对象元素中的索引值（从 0 开始计数），若没有找到则返回-1。比如有 5 个 div，其中第 4 个标签的 ID 是#bar 那么$("div").index($("#bar"))所返回的索引值就是 3。

jQuery 插件机制：

```
$.extend
({
    max:function(num1,num2){return num1 > num2 ? num1:num2;}
    min:function(num1,num2){return num1 < num2? num1:num2;}
})
```

$.extend 等同于 jQuery.extend 在这里面的 max 和 min 是两个自定义的函数，并且都有 2 个参数，在方法体内进行比较。方法体内使用的是条件表达式，与 if 条件判断差不多。此条件表达式的意思是：如果 num1>num2 相比较后如果 num1 大于 num2 那么返回“true”,那么此方法返回“？”之后“：”之前的内容也就是 num1，反之是 num2。

调用的时候只要使用$.max(2,3)传入任意的两个参数，那么将返回 num2 也就是数字 3;$.min(7,8)则返回 num1 因为 num1 比 num2 小。$替换成 jQuery 完全没有问题。

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

    uncheck:function()
    {
        return this.each(function() { this.checked = false; });
    }

})
```

此方法也是一种插件的实现方法，其中 this 表示是调用者当前所指 dom 对象，比如 $("#abc").click(function(){this})这里的 this 指的就是#abc 这个 dom 对象。each 在上面已

经讲解过了。在此插件方法 extend 中定义了两个方法分别是 check 和 uncheck。
比如：
$("input[@type=checkbox]").check()表示将 input 标签的 type 属性设置为选中，其中，中括号中的内容表示如果 input 的 type 属性是 checkbox 的话，再设置为选中。
$("input[@type=radio]").uncheck()表示将 input 标签的 type 属性设置为未选中，其中，中括号中的内容表示如果 input 的 type 属性是 radio 的话，再设置为未选中。

多库共存：
有的时候我们可能在同一个页面内调用多种 js 库，比如即使用 jQuery 类库又使用 ProToType 类库，按理说没有问题，但是他们都用到了"$"符号，因此为了避免与其他库产生冲突，可以使用以下两种方法将其区别开来。

jQuery.noConflict()

使用方法，var j=noConflict();表示 j 在 jQuery 中将代替"$"符号。

jQuery.noConflict(true)

使用方法，dom.query = jQuery.noConflict(true)则表示将$和 jQuery 的控制权都交还给原来的库。比如你想要将 jQuery 嵌入一个高度冲突的环境。注意:调用此方法后极有可能导致插件失效。**因此用的时候一定要考虑清楚。**dom.query 将代表"$"符号。

【第四节】jQuery 速成 - 基本对象获取

向 jQuery 进军中的战友们，通过前面两章的学习，大家肯定对 jQuery 中的"$"函数不明不白，没关系，只要大家挺得住，咱们定能得到最终胜利。言归正传，通过下面几章的学习，一定能明白"$"函数，trust me ！

现在大家想一下在 CSS 中有哪几种选择器？

```
通用选择器：*
id 选择器：#element
类选择器：.element
标签选择器：element
```

如果你对这几种选择器不太了解，建议您到这个页面继续学习

http://bbs.cssxuexi.cn/thread-1285-1-1.html

jQuery 对象的获取也是如此，方法如下：

$("*")

表示获取所有的对象

$("#element")

等同于 document.getElementById("element");

$(".abc")

表示获得 HTML 中所有使用了 abc 这个样式的元素

$("div")

表示获得 HTML 中所有的 div 元素

以下两种是层级对象的获取，属于下节课的内容，下节课不再着重讲解

$("#a,.b,p")

表示获得 ID 是 a 的元素和使用了类样式 b 的元素以及所有的 p 元素

$("#a .b p")

表示获得了 ID 是 a 的元素所包含的使用了类样式的 b 元素中的所有的 p 元素

以下是第四课例题

第 4 课.rar (35.69 KB, 下载次数: 198)

【作业】
望大家自觉的练习 jQuery 对象的获取。

通过本节课的学习，我相信大部分同学都已经明白了"$"函数的作用了。下节课再继续讲解更深一层的内容。本节课结束，同学们下课！

【第五节】jQuery 速成 - 层级对象获取

层级对象的获取

$("Element1 Element2 Element3 Element...")

css 定义层级元素方式一样,只需要不同的元素之间有空格表示，前者父级后者子级以此类推。

$("div > input")

表示获取 div 下所有的 input。

您如果记不清到底是用“>”符号 还是 “<”符号，可以这么理解。如果你选择的元素有父子关系，那么是父亲辈儿长还是儿子辈儿长？当然是父亲辈儿长喽，因此使用“>”，那么选择的时候就是：$("父亲 > 儿子 > ..."),这下明白了吧？^_^

$("div + p")

表示匹配紧跟在 div 元素后的 p 一个元素

这个可以这么记忆，既然紧跟着，当然就得使用“+”紧密的连在一起嘛。

$("div ~ p")

表示匹配跟在 div 元素后的所有 p 元素

这个也好记忆，你看“~”符号有点像弯弯曲曲的小路，通过道路到达目的地，从路的出发点到目的地就是从前往后。当然就是选择 DIV 后面的所有 P 元素咯~

本课案例：

第 5 课.rar (35.15 KB, 下载次数: 147)

本课只需要大家记住这四种选择方式即可，同学们下课！

【第六节】jQuery 速成 - 简单对象获取

本节课主要学习对对象的另外一种获取方法，没有什么好方法，属于死记的东西，因此大家要加油咯。

$("Element:first")

获得在 HTML 页面中某种元素的第一个，比如$("div:first")表示获得第一个 div

$("Element:last")

获得在 HTML 页面中某种元素的最后一个，比如$("div:last")表示获得最后一个 div

$("Element:not(selector)")

去除所有与给定选择器匹配的元素，比如$("input:not(:checked")) 表示选择所有没有选中的复选框

$("Element:even")

获得偶数行，从 0 开始计数

$("Element:odd")

获得奇数行，从 0 开始计数

$("Element:eq(index)")

匹配一个给定索引值的元素，从 0 开始计数，比如$("div:eq(3)")表示获得 HTML 中的第 4 个 div

$("Element:gt(index)")

匹配所有大于给定索引值的元素，从 0 开始计数，比如$("p:gt(3)")表示获得比索引 3 也就是第 4 个 p 开始，之【后】所有的 p

$("Element:lt(index)")

匹配所有小于给定索引值的元素，从 0 开始计数，比如$("p:lt(3)")表示获得比索引 3 也就是第 2 个 p 开始，之【前】所有的 p

注：如果上面两个方法的大于号小于号记不清，就想想HTML标记中的[& gt ;]和[& lt ;](中括号中内容去掉空格)就行了哈。

$(":header")

匹配 h1,h2,h3...标题之类的元素

$("Element:animated")

匹配所有没有在执行动画效果中的元素（关于动画效果，在后面会讲到，在此只是一提，不必要在意，看不懂不要紧）

本节课虽没案例但有作业，我会根据大家作业的情况，再发布案例。如果没有作业提交上来，就表示大家都明白了，案例也就没有发布的必要了。作业发布在回帖中即可，编辑的时候使用代码功能发布，并且复制后可以直接执行，jQuery 在下面已经引入 OK，格式如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>第六章 jQuery 作业</title>
        <script language="javascript"
src="http://jqueryjs.googlecode.com/files/jquery-1.3.2.js"></script>
        <script language="javascript" charset="GB2312">
         此处编写 jQuery 代码
        </script>
        <style>
         如有需要，此处编写 CSS 代码
        </style>
    </head>
```

```
    <body>
    如有需要，此处编写 HTML 代码
    </body>
</html>
```

注：可能用到的方法：text()方法，用法参考上一章节中的案例。

【作业 1】

1、写一个 5 行的表格或列表

2、使用 jQuery 将第一行中填写的内容是:我是第一行

3、使用 jQuery 将最后一行中填写的内容是:我是最后一行

【作业 2】

1、写一个 10 行的表格或列表

2、使用 jQuery 将所有的偶数行均显示“偶数行”，奇数行均显示“奇数行”

【作业 3】

1、写一个 3 行的表格或列表

2、使用 jQuery 将第 2 个表格或列表写入“我是第二行”

【作业 4】

1、写一个 10 行的表格或列表

2、使用 jQuery 将第 5 行之前的表格或列表写入“我在第五行之前”

3、使用 jQuery 将第 5 行之后的表格或列表写入“我在第五行之后”

【作业 5】

1、写 h1—h6 的标题标签如下

```
<h1>h1</h1>
<h2>h2</h2><h3>h3</h3>
<h4>h4</h4>
<h5>h5</h5>
<h6>h6</h6>
```

2、使用 jQuery 将所有的 h 标签中的内容都更改成“我是 h 标题标签”

【第七节】jQuery 速成 - 内容对象的获取和对象可见性！

本节课将要学习对内容对象的获取和对象的可见性。

【内容获取】

$("Element:contains(text)")

匹配元素对象中的文本是否包含某个字母或者某个字符串（符串或串(String)是由零个或多个字符组成的有限序列。）

$("Element:empty")

获得对象元素不包含文本或子元素

$("Element:parent")

以上面相反，获得对象元素包含文本或子元素

$("Element:has(selector)")

匹配含有某个元素是否包含某个元素 比如$("p:has(span)")表示所有包含 span 元素的 p 元素

【可见性】

$("Element:hidden")

匹配所有不可见元素，包括 display:none 以及 input 的属性是 hidden 都可以匹配得到。

$("Element:visible")

与上面相反，匹配所有可见元素

课件下载

 第 7 课.rar (35.77 KB, 下载次数: 102)

【第八节】jQuery 速成 - 对象获取进阶

今天我们来学习一下其他对对象获的方法，当您学完这节课，相信你会觉得 jQuery 真的是太方便了，本节课与第五章《jQuery 速成 - 简单对象获取》一样，也是属于死记的东西。

$("Element[id]")

获得所有带有 ID 属性的元素

$("Element[attribute=KwooShung]")

获得所有某个属性为 KwooShung 的元素

$("Element[attribute!=KwooShung]")

获得所有某个属性不为 KwooShung 的元素，我想如果学过 C#和 java 的人都明白"!"在程序中表示非

$("Element[attribute^=Kwoo]")

获得所有某个属性值是以 Kwoo 开头的元素

$("Element[attribute$=Kwoo]")

获得所有某个属性值是以 Kwoo 结尾的元素

$("Element[attribute*=Kwoo]")

获得所有某个属性值包含 Kwoo 的元素

$("Element[selector1][selector2][....]")

符合属性选择器，比如$("input[id][name][value=kwooshung]")表示获得带有 ID、Name 以及 value 是 KwooShung 的 input 元素。

本课案例

第 8 课.rar (39.13 KB, 下载次数: 97)

【第九节】jQuery 速成 - 子元素的获取

到目前为止，我写的 jQuery 教程已经到了第八章了，不知大家现在对 jQuery 是否还比较陌生，如果你还很陌生的话，没关系。css 学习网也在教程的后面留下了作业或案例，希望朋友们能认真的完成作业认真的看案例。我相信大家一定能好好的驾驭这匹烈马的。

本节课相对来说比较简单，就四个函数。

$("Element:nth-child(index)")

选择父级下的第 N 个子级元素，索引从 1 开始，而 eq 函数（eq 函数会在后面学习到）从 0 开始。

├—:nth-child(even)偶数

├—:nth-child(odd)奇数

├—:nth-child(3n)表达式

├—:nth-child(2)索引

├—:nth-child(3n+1)表达式

└—:nth-child(3n+2)表达式

$("Element:first-child")

匹配父级下的第一个子级元素

$("Element:last-child")

匹配父级下的最后一个子级元素

$("Element:only-child")

匹配父级下的唯一的一个子级元素，例如 dt 在 dl 列表中唯一，那么将选择 dt

本章案例：

第 9 课.rar (40.05 KB, 下载次数: 83)

【第十节】jQuery 速成 - 表单对象的获取

我相信看到本节标题的时候都可能会问，我们表单对象获取已经学过了，之前学的都是对象的获取，表单对象的获取还有什么可学的呢？其实不是这样，本节课主要学习一下针对表单对象的另外一种获取方法，由于表单对象较多，所以本节课的内容也很多，当然喽，也属于一些死记的东西。没什么好办法。

$(:input)

只能匹配 Input 元素类型为 input button select textarea

$(:text)

匹配所有的单行文本框

$(:password)

匹配所有的密码框

$(:radio)

匹配所有的单选按钮

$(:checkbox)

匹配所有的复选框

$(:submit)

匹配所有的提交按钮

$(:image)

匹配所有的图像域，例如<input type="image" />

$(:reset)

匹配所有的重置按钮

$(:button)

匹配所有的按钮

$(:file)

匹配所有的文件上传域

$(:hidden)

匹配所有的不可见元素或者 type 为 hidden 的元素

$(:enabled)

匹配所有可用的 input 元素，比如 radio:enabled 表示匹配所有可用的单选按钮

$(:disabled)

匹配所有的不可用 input 元素，作用与上相反

$(:checked)

匹配所有选中的复选框元素

$(:selected)

匹配所有的下拉列表

由于本章较为简单，需要记忆的东西比较多，所以没有写本课案例，如有需要，请在 jQuery 群组中留言。

这节课是 jQuery 基础篇的最后一课，学完这节课，同学们就算是掌握了骑马的技巧，但是如果想要驾驭它还很难，同学们加油啊！

$("Element").attr(name)

取得第一个匹配元素的属性值，比如$("img").attr("src")

$("Element").attr({key:value,key,value,....})

表示为某一个元素一次性设置多个属性

$("Element").attr(key,value)

为某一个元素设置属性

$("Element").attr(key,function)

为所有匹配的元素设置一个计算的属性值。

$("Element").removeAttr(name)

移除某一个属性

本节案例：

第 11 课.rar (125.94 KB，下载次数: 90)

【第十二节】jQuery 速成 - 过滤

教程写到这里，也不知道大家对我的教程感觉如何，因此我希望大家顶贴，发表自己对此教程的看法。

我们今天的课程就要进入新的一章了。大家要加油

$("Element").eq(index)

取得第 n 个元素，此方法的是从 0 算起的。$("div").eq(5)表示获得此页面中的第 6 个 div

$("Element").hasClass("className")

检查当前的元素是否含有某个特定的类，如果有，则返回 true。

$("Element").filter("Expression")

筛选出与指定表达式匹配的元素集合。这个方法用于缩小匹配的范围。可用逗号分隔多个表达式。比如$("input",".Names",":last")表示筛选出最后一个使用 Names 类选择器的 input 中的最后一个。

$("Element").filter("function")

使用方法同上，Function 与《【第十一节】jQuery 速成 - 元素属性的设置与移除》中的第四个元素使用方法一致。

$("Element").is("Expression")

用一个表达式来检查当前选择的元素集合，如果其中至少有一个元素符合这个给定的表达式就返回 true。比如$("div:first[class='abc']").parent().is("body")

$("Element").map("callback")

将一组元素转换成其他数组（不论是否是元素数组）

$("Element").not("Expression")

删除与指定表达式匹配的元素，

$("Element").slice(start,end)

$("Element").slice(start,end),start (Integer) :开始选取子集的位置。第一个元素是 0.如果是负数，则可以从集合的尾部开始选起。end (Integer) : (可选) 结束选取自己的位置，如果不指定，则就是本身的结尾。

补充：案例中的 not 写错了，我又写了一个例子。

$("Element").not("Expression")

比如：HTML: Hello Hello Again
jQuery：$("p").not($("#abc")[0])
结果： Hello 案例中此效果写错了，看这个即可。

本节案例：


第 12 课.rar (126.27 KB, 下载次数: 78)

【第十三节】jQuery 速成 - 查找

$("Element").add("Expressions")

把与表达式匹配的元素添加到 jQuery 对象中。这个函数可以用于连接分别与两个表达式匹配的元素结果集。

$("Element").children("Expressions")

可以通过可选的表达式来过滤所匹配的子元素。注意：parents()将查找所有祖辈元素，而children()只考虑子元素而不考虑所有后代元素。

$("Element").contents()

表示获得某个元素的子元素内容

$("Element").find("Expressions")

表示搜索某个元素下面的某个子元素，比如$("div").find("p")等同于$("div p")

$("Element").next("Expressions")

表示获得某个元素后面的同辈元素的集合,这里的同辈不是指同种元素，而是在同一个层下的所有元素。

$("Element").nextAll("Expressions")

表示查找当前元素之后的所有元素。可以用表达式过滤

$("Element").prev("Expressions")

取得一个包含匹配的元素集合中每一个元素紧邻的前一个同辈元素的元素集合。

$("Element").prevAll("Expressions")

查找当前元素之前所有的同辈元素可以用表达式过滤。

$("Element").parent("Expressions")

取得一个包含着所有匹配元素的唯一父元素的元素集合。

$("Element").parents("Expressions")

取得一个包含着所有匹配元素的祖先元素的元素集合（不包含根元素）。可以通过一个可选的表达式进行筛选。

$("Element").siblings("Expressions")

查找当前元素之前所有的同辈元素,可以用表达式过滤。

本节课没有案例，与第六节:jQuery 速成 - 简单的对象获取规则一样，我会根据大家作业的情况，再发布案例。

```
1. <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
2. <html xmlns="http://www.w3.org/1999/xhtml">
3.     <head>
4.         <meta http-equiv="Content-Type" content="text/html;
   charset=utf-8" />
5.         <title>第十三章 jQuery 作业</title>
6.         <script language="javascript"
   src="http://jqueryjs.googlecode.com/files/jquery-1.3.2.js"></script>
7.         <script language="javascript" charset="GB2312">
8.          此处编写 jQuery 代码
9.         </script>
10.        <style>
11.         如有需要，此处编写 CSS 代码
12.        </style>
13.    </head>
14.    <body>
15.     如有需要，此处编写 HTML 代码
16.    </body>
```

【作业】
将上述所有的方法写一个例子。

【第十四节】jQuery 速成 - 串联

本节结束后，又要进入一个新的篇章咯。本节较为简单，就 2 个方法。

$("Element").andSelf()

将先前所选的加入当前元素中比如$("p a")表示选择 p 下面的所有 a 元素，如果加上$("p a").andSelf()表示选择的是 p

$("Element").end()

回到最近的一个"破坏性"操作之前。即，将匹配的元素列表变为前一次的状态。详细解释在案例中

案例下载：

第 14 课.rar (125.69 KB, 下载次数: 64)

【第十五节】jQuery 速成 - 内部插入

$("Element").append("content")

向选中的元素中追加内容

$("Element").appendTo("content")

将选中的元素追加到到另外一个元素内部，实际上，使用这个方法是颠倒了常规的$(A).append(B)的操作，即不是把 B 追加到 A 中，而是把 A 追加到 B 中。

$("Element").prepend("content")

向选中的元素中追加内容并前置

$("Element").prependTo("content")

将选中的元素追加到到另外一个元素内部并前置，实际上，使用这个方法是颠倒了常规的$(A).prepend(B)的操作，即不是把 B 前置到 A 中，而是把 A 前置到 B 中。

案例如下：


第 15 课.rar (125.67 KB，下载次数: 56)

【第十六节】jQuery 速成 - 外部插入

这节课非常的有意思，从标题上来看与上一节非常相似，而且功能也非常相似，甚至解释和案例也非常相似。当然与上节一样，用处也非常大。

$("Element").after("content")

向选中的元素后面追加内容

$("Element").before("content")

将选中的元素前面添加内容

$("Element").insertAfter("content")

使用这个方法是颠倒了常规的$(A).after(B)的操作，即不是把 B 插入到 A 后面，而是把 A 插入到 B 后面。

$("Element").insertBefore("content")

使用这个方法是颠倒了常规的$(A).before(B)的操作，即不是把 B 插入到 A 前面，而是把 A 插入到 B 前面。

案例如下：


第 16 课.rar (125.73 KB，下载次数: 48)

【第十七节】jQuery 速成 - 包裹

我相信很多人看到了这个标题会非常迷惑，包裹在这里到底是什么意思呢？其实也非常容易理解，就是把所有匹配的元素用其他元素的结构化标记包裹起来。看完这句话您估计就能理解个差不多了，那么再看看包裹的方法和案例，你就能很容易的掌握它！

$("Element").wrap("html")

所匹配的元素分别用其他元素的结构化标记包裹起来，比如$("p").wrap(" ")或者$("p").(" ")，否则无效。无效的方式有：$("p").("div")等

$("Element").wrap("elem")

同上比如$("p").wrap($("#Element"))

$("Element").wrapAll("html")

将所有匹配的元素用单个元素包裹起来，$("p").wrapAll(" ")便会将所有的 p 使用一个 div 包裹起来

$("Element").wrapAll("elem")

同上，比如$("p").wrapAll($("#Element"))

$("Element").wrapInner("html")

将每一个匹配的元素的子内容(包括文本也属于子元素)用一个 HTML 结构包裹起来
$("p").wrapInner(" ")

$("Element").wrapInner("elem")

同上，比如$("p").wrapInner($("#Element"))

案例如下：  第 17 课.rar (124.86 KB，下载次数: 45)

【第十八节】jQuery 速成 - 替换，删除和复制

本章主要讲解 jQuery 对文本替换，删除和复制的处理方法。

替换：

$("Element").replaceWith("content")

将所有匹配的元素替换成指定的 HTML 或 DOM 元素。

$("Element").replaceAll("selector")

用匹配的元素替换掉所有 selector 匹配到的元素。

删除：

$("Element").empty()

删除匹配的元素集合中所有的子节点。包括文本也属于子节点。

$("Element").remove("Expressions")

从 DOM 中删除所有匹配的元素。

复制：

$("Element").clone()

克隆匹配的 DOM 元素并且选中这些克隆的副本，就是复制的意思。

$("Element").clone("true")

元素以及其所有的事件处理并且选中这些克隆的副本。比如某个按钮带有事件，他将自己复制后，事件也可以复制。详细讲解在案例。

案例如下：

第 18 课.rar (126.14 KB, 下载次数: 100)

【第十九节】jQuery 速成 - 元素的赋值

本节课较为简单，并且下面的这 7 个方法也都已经在前面的章节的案例中使用过，因此本节课便不再做案例。如果您真的有什么问题或者觉得有必要出案例，请在到 jQuery 小组进行讨论。

HTML:

$("Element").HTML()

获得选定元素的 HTML 代码

$("Element").HTML("val")

设置指定元素的 HTML 代码，HTML 代码想怎么写就怎么写与平时在 body 中写 HTML 一样

文本：

$("Element").text()

获得指定标签中显示的文字,与 HTML 不一样

$("Element").text("val")

获得指定标签设置内容，即使是 HTML 也会按原样输出

值：

$("Element").val("val")

获得 input 的值 check,select,radio 等都能获取

$("Element").HTML("val")

设置指定的 input 的值

【第二十节】jQuery 速成 - 样式的设置与定义

今天这节课内容稍微多了一点，但是与按照章节来说，还是比较少的。另外 jQuery 可以对元素更好的设置样式，也更加方便快捷。

样式设置：

$("Element").addClass("Class")

获得选定元素的 HTML 代码,假如定义了一个类样式叫做.main{.....}，那么可以为 p 元素这么添加$("p").addClass("main")

$("Element").removeClass("Class")

与上述一样，addClass("main")表示添加样式，此方法则是移除 main 样式

$("Element").toggleClass("Class")

这个可以算是前两个方法的一个综合。简单来说就是：如果存在（不存在）就删除（添加）一个类。

样式定义：

$("Element").css("name")

表示获得某个元素的 css 样式，比如$("div").css("color")表示获得此 div 的字体颜色。

$("Element").css(name,value)

name 表示属性名称，value 表示值。如果为所有的 div 设置字体颜色为#f00 的话，只需要 $("div").css("color","#f00");

$("Element").css({name:"value",name:"value",.....})

此方法是为了能够对元素一次性设置多个样式属性，比如我想对所有的 p 设置字体颜色是红色并且背景颜色是绿色，就用到了此方法$("p").css({color: "#f00", background: "#0f0"});当然中括号中项设置几个就设置几个

元素位置：

$("Element").offset()

此方法返回两个整形属性，分别是 top 和 left，此方法只对可见元素有效。

元素宽高：

$("Element").width()

获得某个元素的宽度

$("Element").width("val")

设置某个元素的宽度

$("Element").height()

获得某个元素高度

$("Element").height("val")

设置某个元素的高度

注：以上方法缺省情况下均以像素为单位。

案例如下： 第 20 课.rar (126.32 KB, 下载次数: 85)

【第二十一节】jQuery 速成 - 页面的载入事件与事件处理

从今天开始，我们要学习 jQuery 中事件机制，如之前用过的 click()这就是事件机制中的一种。通过如此简单的开场白，相信大家也能明白些什么是事件机制了吧？如果你还不明白，没关系，你还可以这么理解，当你在元素上进行单击、双击、移入、移出都是事件。当你将鼠标移入元素表示出发了移入事件，当你点击了元素那么就触发了 click()事件。

注：红色的方法是本教程原来没有的，后来补充上的。

载入事件：

$(document).ready(function)

等同于 jQuery(document).ready(fn)等同于$

上面这个方法我相信大家应该不会陌生，因为它充当的是 js 中的 onLoad()事件，凡是做过作业或者看过我的案例的朋友都应该看到了，每当我们要是用 jQuery 进行对元素的控制的时候都需要在第一行的开始使用"$"符号，这是为了当页面载入完毕，在调用 jQuery 方法对元素进行控制。如果看到这里你还不懂，建议你再重新看一下第二节的教程**【第二节】jQuery 速成 - 核心**

事件处理：

$("Element").bind("type",[data],function)

表示为某一个元素绑定特定的事件
type:事件类型
data：返回类型（可选）
function:普通的 js 方法或者 jQuery 方法
详细应用见案例

$("Element").live("type",function)

表示为某一个元素绑定特定的事件,推荐使用此方法代替上面的 bind

$("Element").unbind([type],function)

与 bind 方法相反，删除匹配的元素所绑定的某个特定的事件
type:事件类型（可选）
function：反绑定的事件处理函数（可选）
如果以上参数均无，则表示将所匹配元素的所有事件取消绑定

$("Element").die("type",function)

表示为某一个元素解除特定的事件,推荐使用此方法代替上面的 unbind

$("Element").one("type",[data],function)

表示为某一个元素绑定一次性的特定事件
type:事件类型
data：返回类型（可选）
function:普通的 js 方法或者 jQuery 方法
此方法看似与 bind 相似，使用方法亦是如此，但是功能差距很大，因为这个事件是一次性的，如果在一个页面中不刷新，绑定的这个事件只能使用一次。

$("Element").trigger("type",[data])

在每个匹配的元素上绑定某类事件
type:事件类型
data：附加参数（可选）
比如
$("form:first").trigger("submit")表示页面中的第一个 form 表单提交。我们知道一般将按钮放在 form 中，点击此按钮才会提交他所在的 form 表单，如果使用此方法，即使按钮在表单区域之外，也同样会使其提交。另外如果这个按钮有浏览器默认的事件，它也会执行，你设置的事件也会执行。如果要阻止默认事件，那么此方法返回 false 或者使用下面的的方法都可。

更多的使用方法见案例

$("Element").triggerHandler("type",[data])

在每个匹配的元素上绑定某类事件,但不会执行浏览器默认的事件
type:事件类型
data：返回类型（可选）
使用方法如上，不同的是不会执行浏览器默认事件

案例如下：

第 21 课.rar (126.77 KB，下载次数: 82)

【第二十二节】jQuery 速成 - 鼠标事件与交互

今天为大家带来的主要是鼠标有关的事件讲解，由于交互处理方面与鼠标交互处理事件多多少少牵扯到点儿关系，因此这 2 个元素与本节合并为一节。

交互处理：

$("Element").hover(over,out)

模拟鼠标悬停的事件，当鼠标移入移出选定元素的时候分别触发 over 和 out 事件。
参数：over：function
　　out：function

$("Element").toggle(function,function)

与上面的方法雷同，当鼠标第一次点击的时候触发前者，当鼠标第二次点击的时候触发后者。

鼠标单击：

$("Element").click()
$("Element").click(function)

当鼠标点击的时候触发，具体的应用方法我想不用具体讲解了。因为大家看了我过去的所写的案例就已经有数了。

鼠标双击：

$("Element").dblclick()
$("Element").dblclick(function)

与鼠标单击一样，只不过这个是鼠标双击事件。也就是说只有鼠标在选定的元素上双击才会触发此事件。dbl 是 Double 的缩写。

鼠标点击前后：

$("Element").mousedown(function)

当鼠标点击后触发，从表面上看类似 click 事件，其实有本质上的区别。

$("Element").mouseup(function)

当鼠标点击释放的时候触发。就是鼠标点击了元素当你松开鼠标按键的时候触发。

鼠标的移动：

$("Element").mousemove(function)

当鼠标在选定的元素上来回移动的时候触发。

$("Element").mouseover(function)

当鼠标在移入选定的元素范围的时候触发。

$("Element").mouseenter(function)

当鼠标在移入选定的元素范围的时候触发。与 mouseover 有很大的区别就是它不冒泡的事件，点击子元素的时候不会触发父级元素

$("Element").mouseout(function)

当鼠标移出选定的元素范围的时候触发。

$("Element").mouseleave(function)

当鼠标移出选定的元素范围的时候触发。与 mouseenter 一样，不是冒泡事件

本节作业：

1、将以上所有的事件均做一个例子发上来。

2、作业发布在回帖中即可，编辑的时候使用代码功能发布，并且复制后可以直接执行。

```
1.  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
2.  <html xmlns="http://www.w3.org/1999/xhtml">
3.      <head>
4.          <meta http-equiv="Content-Type" content="text/html;
    charset=utf-8" />
5.          <title>第二十二节 jQuery 作业</title>
```

```
6.          <script language="javascript"
    src="http://jqueryjs.googlecode.com/files/jquery-1.3.2.js"></script>
7.          <script language="javascript" charset="GB2312">
8.           此处编写 jQuery 代码
9.          </script>
10.         <style>
11.          如有需要，此处编写 CSS 代码
12.         </style>
13.     </head>
14.     <body>
15.      如有需要，此处编写 HTML 代码
16.     </body>
17. </html>
```

【第二十三节】 jQuery 速成- 焦点事件

今天学习以下焦点事件，这些事件在网页中经常使用。比如文本框失去焦点后验证输入。建议大家本节课与下节课一起学习。

触发焦点：

$("Element").focus()

触发每一个匹配元素获得焦点事件。

$("Element").focus(function)

事件会在获得焦点的时候触发，既可以是鼠标行为，也可以是按 tab 键导航触发的行为，并且绑定一个处理方法。

失去焦点：

$("Element").blur()

触发每一个匹配元素失去焦点事件。

$("Element").blur(function)

事件会在元素失去焦点的时候触发，既可以是鼠标行为，也可以是按 tab 键离开的行为，并且绑定一个处理方法。

改变焦点：

$("Element").change()

触发每一个匹配元素改变时事件.

$("Element").change(function)

在每一个匹配元素的 change 事件中绑定一个处理函数。

本节课过于简单，因此无案例，如有需要请留言。作业在下节课一起布置。

【第二十四节】jQuery 速成 - 键盘事件

键盘按下：

$("Element").keydown()

当键盘按下的时候触发此事件。

$("Element").keydown(function)

当键盘按下的时候触发此事件，并绑定一个处理方法。

键盘敲击：

$("Element").keypress()

当键盘按下的时候触发此事件。

$("Element").keypress(function)

当键盘按下的时候触发此事件，并绑定一个处理方法。

注：虽然从表面上理解 keypress 与 keydown 是一个意思，但二者的本质区别是：系统由 keydown 返回键盘的代码，然后由 TranslateMessage 函数翻译成字符，由 keypress 返回字符值. 因此在 keydown 中返回的是键盘的代码，而 keypress 返回的是 ASCII 字符. 所以根据你的目的，如果只想读取字符，用 keypress，如果想读各键的状态，用 keydown。

键盘弹起：

$("Element").keyup()

当键盘按键释放的时候触发。

$("Element").keyup(function)

当键盘按键释放的时候触发并绑定一个处理方法。

课后作业：

注：本节课后作业可以对前几节的内容进行回顾，并且可以使大家能够快速的将讲过的方法融合在一起使用，更好的驾驭 jQuery！

作业 1：

页面上有一个文本域并有一段文字，当失去焦点的时候在旁边"比如 div span 等"显示有多少个字。

作业 2：

页面中有一个文本域，当在页面上输入文字的时候，在上方显示"正在输入"的字样，并以蓝色字体显示，停止输入的时候上方无任何文字。

```
1.  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
2.  <html xmlns="http://www.w3.org/1999/xhtml">
3.      <head>
4.          <meta http-equiv="Content-Type" content="text/html;
    charset=utf-8" />
5.          <title>第二十二节 jQuery 作业</title>
6.          <script language="javascript"
    src="http://jqueryjs.googlecode.com/files/jquery-1.3.2.js"></script>
7.          <script language="javascript" charset="GB2312">
8.           此处编写 jQuery 代码
9.          </script>
10.         <style>
11.          如有需要，此处编写 CSS 代码
12.         </style>
13.     </head>
14.     <body>
15.      如有需要，此处编写 HTML 代码
16.     </body>
17. </html>
```

【第二十五节】jQuery 速成 - 其他事件

由于我最近入职于麒麟网信息技术有限公司的关系，所以课程的进度停的大概有 1 周左右，凑这两天我会尽量将《jQuery 速成教程》结束掉。大家要有个心理准备。

这节课，我们就要结束了 jQuery 中的事件。

$("Element").load(type,function)

在元素后面绑定一个处理函数，当元素内容加载完毕后自动调用。就如同每次写 jQuery 的时候都写 jQuery，$(document).reday()或者$的方式差不多。
type：事件类型
function:函数体

$("Element").unload(function)

与上面的函数相反，在每一个匹配元素的卸载事件中绑定一个处理函数。比如页面卸载的时候弹出一个警告框。$(document).unload(function () { alert("Bye now!"); });

$("Element").resize(function)

当窗口大小发生改变的时候触发,比如$(window).resize(function(){alert("你正在试图改变窗口的大小");});

$("Element").scroll(function)

当滚动条发生改变的时候触发$(window).scroll(function(){alert("你正在试图改变滚动条");});

$("Element").select()

触发每一个匹配元素的 select 事件，这个函数会调用执行绑定到 select 事件的所有函数，包括浏览器的默认行为。比如触发所有 input 元素的 select 事件，$("input").select();

$("Element").select(function)

当用户在文本框(包括 input 和 textarea)中选中某段文本时会触发 select 事件。

$("Element").submit()

函数会调用执行绑定到 submit 事件的所有函数，包括浏览器的默认行为。可以通过在某个绑定的函数中返回 false 来防止触发浏览器的默认行为。$("form:first").submit(); 表示第一个 form 提交的时候触发。

$("Element").submit(function)

在每一个匹配元素的 submit 事件中绑定一个处理函数。

$("Element").error()

这个函数会调用所有绑定到 error 事件上的函数，包括在对应元素上的浏览器默认行为。可以通过在某个绑定的函数中返回 false 来防止触发浏览器的默认行为。error 事件通常可以在元素由于点击或者 tab 导航失去焦点时触发。

$("Element").error(function)

对于 error 事件，没有一个公众的标准。在大多数浏览器中，当页面的 JavaScript 发生错误时，window 对象会触发 error 事件;当图像的 src 属性无效时，比如文件不存在或者图像数据错误时，也会触发图像对象的 error 事件。

课后作业：

作业：

根据上面所描述的所有方法，均写一个例子出来。最好都在一个 HTML 文件里。

```
1.  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
2.  <html xmlns="http://www.w3.org/1999/xhtml">
3.      <head>
4.          <meta http-equiv="Content-Type" content="text/html;
    charset=utf-8" />
5.          <title>第二十二节 jQuery 作业</title>
6.          <script language="javascript"
    src="http://jqueryjs.googlecode.com/files/jquery-1.3.2.js"></script>
7.          <script language="javascript" charset="GB2312">
8.           此处编写 jQuery 代码
9.          </script>
10.         <style>
```

```
11.          如有需要，此处编写 CSS 代码
12.        </style>
13.    </head>
14.    <body>
15.      如有需要，此处编写 HTML 代码
16.    </body>
17. </html>
```

【第二十六节】jQuery 速成 - 动画实现的基本方法

$("Element").show()

将不可见元素设置为可见，比如某个元素被定义为 display:none;

$("Element").show(speed,[callback])

speed 为显示的速度，推荐 600 毫秒为佳
speed:时间，单位毫秒
callback：回调函数（可选）

$("Element").hide()

与第一个函数相反，将某个元素设置为隐藏，也就是设置为 display:none；

$("Element").hide(speed,[callback])

speed 为显示的速度，推荐 600 毫秒为佳
speed:时间，单位毫秒
callback：回调函数（可选）

$("Element").toggle()

切换元素的显示状态，如果元素是可见的，切换为隐藏的；如果元素是隐藏的，切换为可见的。

$("Element").toggle(speed,[callback])

speed 为显示的速度，推荐 600 毫秒为佳
speed:时间，单位毫秒
callback：回调函数（可选）

课后作业：

作业：

由于这两节课程比较简单，所以本节作业还是跟前一节一样，根据上面所描述的所有方法，均写一个例子出来。最好都在一个 HTML 文件里。

```
1.  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
2.  <html xmlns="http://www.w3.org/1999/xhtml">
3.      <head>
4.          <meta http-equiv="Content-Type" content="text/html;
    charset=utf-8" />
5.          <title>第二十二节 jQuery 作业</title>
6.          <script language="javascript"
    src="http://jqueryjs.googlecode.com/files/jquery-1.3.2.js"></script>
7.          <script language="javascript" charset="GB2312">
8.           此处编写 jQuery 代码
9.          </script>
10.         <style>
11.          如有需要，此处编写 CSS 代码
12.         </style>
13.     </head>
14.     <body>
15.      如有需要，此处编写 HTML 代码
16.     </body>
17. </html>
```

【第二十七节】jQuery 速成 - 元素的渐隐渐显

$("Element").fadeIn(speed,[callback])

通过不透明度的变化来实现所有匹配元素的淡入效果，并在动画完成后可选地触发一个回调函数。

speed:时间，单位毫秒
callback：回调函数（可选）

$("Element").fadeOut(speed,[callback])

通过不透明度的变化来实现所有匹配元素的淡出效果，并在动画完成后可选地触发一个回调函数。
speed:时间，单位毫秒
callback：回调函数（可选）

$("Element").fadeTo(speed,opacity,[callback])

把所有匹配元素的不透明度以渐进方式调整到指定的不透明度，并在动画完成后可选地触发一个回调函数。
speed:时间，单位毫秒
opacity:要调整到的不透明度值(0 到 1 之间的数字，一般小数点后面保留 2 位).
callback：回调函数（可选）

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>第二十二节 jQuery 作业</title>
        <script language="javascript"
src="http://jqueryjs.googlecode.com/files/jquery-1.3.2.js"></script>
        <script language="javascript" charset="GB2312">
         此处编写 jQuery 代码
        </script>
```

```
        <style>
          如有需要，此处编写 CSS 代码
        </style>
    </head>
    <body>
    如有需要，此处编写 HTML 代码
    </body>
</html>
```

作业：

本节作业还是跟前一节一样，根据上面所描述的所有方法，均写一个例子出来。最好都在一个 HTML 文件里。

【第二十八节】jQuery 速成 - 元素的滑动

今天是动画效果的最后一节课，内容比较难比较多，大家加油啊！

$("Element").animate(params[,duration[,easing[,callback]]])

[quote]用于创建自定义动画的函数。

这个函数的关键在于指定动画形式及结果样式属性对象。这个对象中每个属性都表示一个可以变化的样式属性（如“height”、“top”或“opacity”）。

注意：所有指定的属性必须用骆驼形式，比如用 marginLeft 代替 margin-left，如果有不懂得骆驼命名法的朋友请看三种通用 CSS 规范化命名的规则。

而每个属性的值表示这个样式属性到多少时动画结束。如果是一个数值，样式属性就会从当前的值渐变到指定的值。如果使用的是“hide”、“show”或“toggle”这样的字符串值，则会为该属性调用默认的动画形式。

params (Options) : 一组包含作为动画属性和终值的样式属性和及其值的集合

duration (String,Number) : (可选) 三种预定速度之一的字符串("slow", "normal", or "fast")或表示动画时长的毫秒数值(如：1000)

easing (String) : (可选) 要使用的擦除效果的名称(需要插件支持).默认 jQuery 提供 "linear" 和 "swing".

callback (Function) : (可选) 在动画完成时执行的函数

$("Element").animate(params,options)

同上
params (Options)： 一组包含作为动画属性和终值的样式属性和及其值的集合
options (Options)： 一组包含动画选项的值的集合。

$("Element").stop()

停止指定元素上正在运行的动画

$("Element").queue()

返回指向第一个匹配元素的队列，常与 length 配合使用；可以将其理解为数组，一个动画数组中包含了好几个效果，queue().length 表示获得当前所执行的第一个效果。

```
1.  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
2.  <html xmlns="http://www.w3.org/1999/xhtml">
3.      <head>
4.          <meta http-equiv="Content-Type" content="text/html;
    charset=utf-8" />
5.          <title>第二十九节 jQuery 作业</title>
6.          <script language="javascript"
    src="http://jqueryjs.googlecode.com/files/jquery-1.3.2.js"></script>
7.          <script language="javascript" charset="GB2312">
8.           $(
9.                  function()
10.                 {
11.                     $("#show").click
12.                     (
13.                         function()
14.                         {
```

```
15.                             var n = $("div").queue();
16.                             $("span").text("Queue length is: " +
    $("div").queue().length);
17.                         }
18.                     );
19.                     runIt();
20.                 }
21.             );
22.
23.             function runIt()
24.             {
25.                 $("div").show("slow");
26.                 $("div").animate({left:'+=200'},2000);
27.                 $("div").slideToggle(1000);
28.                 $("div").slideToggle("fast");
29.                 $("div").animate({left:'-=200'},1500);
30.                 $("div").hide("slow");
31.                 $("div").show(1200);
32.                 $("div").slideUp("normal", runIt);
33.             }
34.         </script>
35.         <style>
36.                 div { margin:3px; width:40px;
    height:40px;position:absolute; left:0px; top:30px; background:green;
    display:none; }
37.                 div.newcolor { background:blue; }
38.                 span { color:red; }
39.         </style>
40.     </head>
41.     <body>
42.         <button id="show">Show Length of Queue</button>
43.         <span></span>
```

```
44.         <div></div>
45.     </body>
46. </html>
47. [/quote]
48. $("Element")[color=blue][color=black].[color=#0000ff]queue(function)[/c
    olor][/color][/color][color=#9acd32]// 要添加进队列的函数[/color]
49. [hide]
50. <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
51. <html xmlns="http://www.w3.org/1999/xhtml">
52.     <head>
53.         <meta http-equiv="Content-Type" content="text/html;
    charset=utf-8" />
54.         <title>第二十九节 jQuery 作业</title>
55.         <script language="javascript"
    src="http://jqueryjs.googlecode.com/files/jquery-1.3.2.js"></script>
56.         <script language="javascript" charset="GB2312">
57.         $
58.         (
59.             function()
60.             {
61.                 $(document.body).click
62.                 (
63.                     function ()
64.                     {
65.                         $("div").show("slow");
66.                         $("div").animate({left:'+=200'},2000);
67.                         $("div").queue
68.                         (
69.                             function()
70.                             {
71.                                 $(this).addClass("newcolor");
```

```
72.                                 $(this).dequeue();
73.                             }
74.                         );
75.                         $("div").animate({left:'-=200'},500);
76.                         $("div").queue(function(){$(this).removeClass("n
    ewcolor");$(this).dequeue();});
77.                         $("div").slideUp();
78.                     }
79.                 )
80.             }
81.         );
82.         </script>
83.         <style>
84.                 div {margin:3px; width:40px;
    height:40px;position:absolute; left:0px; top:30px;  background:green;
    display:none; }
85.                 div.newcolor{ background:blue;}
86.         </style>
87.     </head>
88.     <body>
89.         Click here...
90.         <div></div>
91.     </body>
92. </html>
93. [/quote]
94.
95. $("Element")[color=blue][color=black].[color=#0000ff]queue(queue)[/colo
    r][/color][/color][color=#9acd32]//将匹配元素的动画队列用新的一个队列来代替
    (函数数组)[/color]
96. [indent]
97. <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html;
    charset=utf-8" />
        <title>第二十九节 jQuery 作业</title>
        <script language="javascript"
    src="http://jqueryjs.googlecode.com/files/jquery-1.3.2.js"></script>
        <script language="javascript" charset="GB2312">
        $
        (
            function()
            {
                $("#start").click
                (
                    function()
                    {
                        $("div").show("slow");
                        $("div").animate({left:'+=200'},5000);
                        $("div").queue
                        (
                            function()
                            {
                                $(this).addClass("newcolor");
                                $(this).dequeue();
                            }
                        );
                        $("div").animate({left:'-=200'},1500);
                        $("div").queue
                        (
                            function()
                            {
                                $(this).removeClass("newcolor");
```

```
                                $(this).dequeue();
                            }
                        );
                        $("div").slideUp();
                    }
                )

                $("#stop").click
                (
                    function()
                    {
                        $("div").queue("fx", []);
                        $("div").stop();
                    }
                )
            }
        );
        </script>
        <style>
                div {margin:3px; width:40px;
    height:40px;position:absolute; left:0px; top:30px;  background:green;
    display:none; }
                div.newcolor{ background:blue;}
        </style>
    </head>
    <body>
        <button id="start">Start</button>
        <button id="stop">Stop</button>
        <div></div>
    </body>
</html>
[/quote]
```

```
158.
159. $("Element")[color=blue][color=black].[color=#0000ff]dequeue()[/color][
    /color][/color][color=#9acd32]//从动画队列中移除一个队列函数[/color]
160. [hide][indent]
161. <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
162. <html xmlns="http://www.w3.org/1999/xhtml">
163.     <head>
164.         <meta http-equiv="Content-Type" content="text/html;
    charset=utf-8" />
165.         <title>第二十九节 jQuery 作业</title>
166.         <script language="javascript"
    src="http://jqueryjs.googlecode.com/files/jquery-1.3.2.js"></script>
167.         <script language="javascript" charset="GB2312">
168.         $
169.         (
170.                 function()
171.                 {
172.                     $("button").click
173.                     (
174.                         function()
175.                         {
176.                             $("div").animate({left:'+=200px'}, 2000);
177.                             $("div").animate({top:'0px'}, 600);
178.                             $("div").queue
179.                             (
180.                                 function()
181.                                 {
182.                                     $(this).toggleClass("red");
183.                                     $(this).dequeue();
184.                                 }
185.                             );
```

```
186.                         $("div").animate({left:'10px', top:'30px'},
    700);
187.                     }
188.                 );
189.             }
190.         );
191.     </script>
192.     <style>
193.         div {margin:3px; width:50px; position:absolute;height:50px;
    left:10px; top:30px; background-color:yellow; }
194.         div.red { background-color:red; }
195.     </style>
196. </head>
197. <body>
198.     <button>Start</button>
199.     <div></div>
200. </body>
201. </html>
```

【第二十九节】jQuery 速成 - 自定义动画

今天是动画效果的最后一节课，内容比较难比较多，大家加油啊！

$("Element").animate(params[,duration[,easing[,callback]]])

用于创建自定义动画的函数。
这个函数的关键在于指定动画形式及结果样式属性对象。这个对象中每个属性都表示一个可以变化的样式属性（如“height”、“top”或“opacity”）。
注意：所有指定的属性必须用骆驼形式，比如用 marginLeft 代替 margin-left，如果有不懂得骆驼命名法的朋友请看三种通用 CSS 规范化命名的规则。

而每个属性的值表示这个样式属性到多少时动画结束。如果是一个数值，样式属性就会从当前的值渐变到指定的值。如果使用的是“hide”、“show”或“toggle”这样的字符串值，则会为该属性调用默认的动画形式。
params (Options) : 一组包含作为动画属性和终值的样式属性和及其值的集合
duration (String,Number) : (可选) 三种预定速度之一的字符串("slow", "normal", or "fast")或表示动画时长的毫秒数值(如：1000)
easing (String) : (可选) 要使用的擦除效果的名称(需要插件支持).默认 jQuery 提供 "linear" 和 "swing".
callback (Function) : (可选) 在动画完成时执行的函数

$("Element").animate(params,options)

同上
params (Options)： 一组包含作为动画属性和终值的样式属性和及其值的集合
options (Options)： 一组包含动画选项的值的集合。

$("Element").stop()

停止指定元素上正在运行的动画

$("Element").queue()

返回指向第一个匹配元素的队列，常与 length 配合使用；可以将其理解为数组，一个动画数组中包含了好几个效果，queue().length 表示获得当前所执行的第一个效果。

```
1.  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
2.  <html xmlns="http://www.w3.org/1999/xhtml">
3.      <head>
4.          <meta http-equiv="Content-Type" content="text/html;
    charset=utf-8" />
5.          <title>第二十九节 jQuery 作业</title>
6.          <script language="javascript"
    src="http://jqueryjs.googlecode.com/files/jquery-1.3.2.js"></script>
7.          <script language="javascript" charset="GB2312">
8.           $(
9.                  function()
10.                 {
11.                     $("#show").click
```

```
                    (
                        function()
                        {
                            var n = $("div").queue();
                            $("span").text("Queue length is: " +
    $("div").queue().length);
                        }
                    );
                    runIt();
                }
            );

            function runIt()
            {
                $("div").show("slow");
                $("div").animate({left:'+=200'},2000);
                $("div").slideToggle(1000);
                $("div").slideToggle("fast");
                $("div").animate({left:'-=200'},1500);
                $("div").hide("slow");
                $("div").show(1200);
                $("div").slideUp("normal", runIt);
            }
        </script>
        <style>
                div { margin:3px; width:40px;
    height:40px;position:absolute; left:0px; top:30px; background:green;
    display:none; }
                div.newcolor { background:blue; }
                span { color:red; }
        </style>
    </head>
```

```
41.     <body>
42.         <button id="show">Show Length of Queue</button>
43.         <span></span>
44.         <div></div>
45.     </body>
46. </html>
```

```
1.  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
2.  <html xmlns="http://www.w3.org/1999/xhtml">
3.      <head>
4.          <meta http-equiv="Content-Type" content="text/html;
    charset=utf-8" />
5.          <title>第二十九节 jQuery 作业</title>
6.          <script language="javascript"
    src="http://jqueryjs.googlecode.com/files/jquery-1.3.2.js"></script>
7.          <script language="javascript" charset="GB2312">
8.          $
9.          (
10.             function()
11.             {
12.                 $(document.body).click
13.                 (
14.                     function ()
15.                     {
16.                         $("div").show("slow");
17.                         $("div").animate({left:'+=200'},2000);
18.                         $("div").queue
19.                         (
20.                             function()
```

```
                            {
                                $(this).addClass("newcolor");
                                $(this).dequeue();
                            }
                        );
                        $("div").animate({left:'-=200'},500);
                        $("div").queue(function(){$(this).removeClass("n
    ewcolor");$(this).dequeue();});
                        $("div").slideUp();
                    }
                )
            }
        );
        </script>
        <style>
                div {margin:3px; width:40px;
    height:40px;position:absolute; left:0px; top:30px;  background:green;
    display:none; }
                div.newcolor{ background:blue;}
        </style>
    </head>
    <body>
        Click here...
        <div></div>
    </body>
</html>
```

==========

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
```

```
4.          <meta http-equiv="Content-Type" content="text/html;
    charset=utf-8" />
5.          <title>第二十九节 jQuery 作业</title>
6.          <script language="javascript"
    src="http://jqueryjs.googlecode.com/files/jquery-1.3.2.js"></script>
7.          <script language="javascript" charset="GB2312">
8.          $
9.          (
10.             function()
11.             {
12.                 $("#start").click
13.                 (
14.                     function()
15.                     {
16.                         $("div").show("slow");
17.                         $("div").animate({left:'+=200'},5000);
18.                         $("div").queue
19.                         (
20.                             function()
21.                             {
22.                                 $(this).addClass("newcolor");
23.                                 $(this).dequeue();
24.                             }
25.                         );
26.                         $("div").animate({left:'-=200'},1500);
27.                         $("div").queue
28.                         (
29.                             function()
30.                             {
31.                                 $(this).removeClass("newcolor");
32.                                 $(this).dequeue();
33.                             }
```

```
                        );
                        $("div").slideUp();
                    }
                )

                $("#stop").click
                (
                    function()
                    {
                        $("div").queue("fx", []);
                        $("div").stop();
                    }
                )
            }
        );
        </script>
        <style>
                div {margin:3px; width:40px;
    height:40px;position:absolute; left:0px; top:30px;  background:green;
    display:none; }
                div.newcolor{ background:blue;}
        </style>
    </head>
    <body>
        <button id="start">Start</button>
        <button id="stop">Stop</button>
        <div></div>
    </body>
</html>
```

$("Element").dequeue()//从动画队列中移除一个队列函数

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html;
charset=utf-8" />
        <title>第二十九节 jQuery 作业</title>
        <script language="javascript"
src="http://jqueryjs.googlecode.com/files/jquery-1.3.2.js"></script>
        <script language="javascript" charset="GB2312">
        $
        (
                function()
                {
                    $("button").click
                    (
                        function()
                        {
                            $("div").animate({left:'+=200px'}, 2000);
                            $("div").animate({top:'0px'}, 600);
                            $("div").queue
                            (
                                function()
                                {
                                    $(this).toggleClass("red");
                                    $(this).dequeue();
                                }
                            );
                            $("div").animate({left:'10px', top:'30px'},
700);
                        }
                    );
```

```
29.                 }
30.             );
31.         </script>
32.         <style>
33.             div {margin:3px; width:50px; position:absolute;height:50px;
    left:10px; top:30px; background-color:yellow; }
34.             div.red { background-color:red; }
35.         </style>
36.     </head>
37.     <body>
38.         <button>Start</button>
39.         <div></div>
40.     </body>
41. </html>
```

【第三十节】jQuery 速成 - 浏览器种类及其特性的检测

有很多朋友跟我提建议说，青色字有点刺眼，看着很累，这次已经将注释或解释使用深绿色，随后我也会将所有的课程都修改成深绿色。这节课我们来学习一下在 jQuery 中如何检测浏览器的种类及特性。

$.support.*

此方法目前用到的不多，而且兼容性不是很好，所以暂不详解，大家可以看一下 API

$.browser.type

此方法用于返回浏览器内核标识。

type:safari/opera/msie/mozilla
type 的值分别表示苹果浏览器/opera 浏览器/IE 浏览器/火狐浏览器
注：经过本人测试，在 chrome 浏览器中 $.browser.safari 返回值为 true，因此 chrome 与 safari 是同一内核。

$.browser.version

此方法用于返回浏览器渲染引擎版本号【注意：不是浏览器的版本号】。

$.boxModel

此方法返回当前页面中浏览器是否使用标准盒模型渲染页面，boolean 类型。

案例下载： 第 30 课.rar (25.54 KB, 下载次数: 61)

【第三十一节】jQuery 速成 - 数组和对象的操作

在我们编写 JS 的时候，数组和对象是我们经常用到的，但是对于复杂的数组操作有的时候很不方便，因此 jQuery 提供了一系列的快捷简便的方法。本节课建议先下载案例在对比着学效果会好一些。

$.each(object,function)

object 填写数组对象，function 是遍历后的回调函数

$.eachnd(boolean,object1,object2,[objectN])

boolean 布尔值(true/false)。object 与 function 为必填项，此方法主要用于合并两个数组，相同的下标将有第二个数组对象中的值替换第一个数组对象中相同位置的内容

$.grep(array,function,boolean)

array 填写数组，function 是遍历后的回调函数,boolean 设置为 true 则从数组中排除 function 中符合条件的选项。

$.makeArray(object)

将【类】数组对象转换为数组对象

$.map(array,function(i))

将 array:数组通过 function:回调函数根据条件转换为一个新数组

$.inArray(value,array)

value:用于在数组中查找是否存在的条件，array 待查找的数组

$.toArray()

把 jQuery 集合中所有的 DOM 元素转换成一个数组

$.merge(object1,object2)

此方法主要用于合并两个数组，相同的下标不会被替换

$.unique(array)

删除数组中重复元素。只处理删除 DOM 元素数组，而不能处理字符串或者数字数组。

$.parseJSON(json)

解析一个【标准的】JSON 字符串，详见案例。

本节案例：  第 31 课.rar (25.85 KB, 下载次数: 84)

【第三十二节】jQuery 速成 - 工具杂项

今天这节课是第七章的最后一节可，下节课将为大家讲解 jQuery 的高级应用——Ajax 的应用。

$.noop

就是一个纯粹的空参数，主要用与船体一个空参的时候使用。

$.proxy(function,object)

设置某个时间的特定域。我感觉这个 jQuery 方法有点像把方法跟方法体（方法的实现）分离开了，通过 proxy 指定执行某个方法体。通过真正的应用，感觉这个方法真的是非常有用，返回值为 Boolean。。

$.contains(obj)

判断一个 dom 节点是否包含另一个 dom 节点，返回值为 Boolean。

$.isArray(obj)

判断一个对象是否为数组对象，返回值为 Boolean。

$.isFunction(obj)

判断一个对象是否为一个方法，返回值为 Boolean。

$.isEmptyObject(obj)

判断一个对象是否为空，返回值为 Boolean。

$.isPlainObject(obj)

判断一个对象是否仅仅只是一个纯粹的对象，返回值为 Boolean。

$.trim(string)

去除字符串的开头与结尾的空格。

$.param()

将表单元数组对象进行序列化，个人感觉在 Ajax URL 传递方面比较有用。

$.error(message)

接受一个字符串，并且直接抛出一个包含这个字符串的异常。只要针对开发人员在火狐的 fireBug 插件中使用，

案例下载


第 32 课.rar (25.29 KB，下载次数: 34)

【第三十三节】jQuery 速成 - Ajax 请求（讲解篇）

jQuery 的 Ajax 教程已经停写了 2 个月之久，对各位 CSSer 说声抱歉。现在切入正题，说一下本章的安排。

本章，一节课对大家讲解 jQuery Ajax 的函数(方法)进行说明，另一节课直接通过示例来为大家进行更详细的说明。以此类推，一节课内容和介绍，另一节课案例实战。

本节课内容稍多，肯定会有很多人一看下面那么多属性、参数、键值对就会对自己产生了疑问或者不自信，我给大家的建议是只需了解即可不必深究，下节课我们便会对今天学习的内容进行小练一把。

$.load(url,[data],[callback])

方法说明：载入远程 HTML 文件代码并插入至 DOM 中。

url:待装入 HTML 网页地址

data:(可选) 发送至服务器的 key/value(键值对) 数据。

callback: (可选) 载入成功时回调函数。过去并没有对回调函数进行说明，今天稍微的说一下，jQuery 中的回调函数意思是指当某个操作执行完毕后要执行的函数。

举例：

要求：1、随便建立两个页面，假设为 A.html 和 B.html

2、在 B 中随便写几个 DIV

3、推荐在 A.html 中定义 B 中的 DIV 样式，因为$("html").load()只载入 HTML 不会载入【外链】样式。

写法：

```
$
(
    function()
    {
        $("html").load("B.html");
    }
);
```

$.get(url,[data],[callback])

方法说明：简单的 Get 的请求，

url: 请求 HTML 网页地址

data:(可选) 发送至服务器的 key/value(键值对) 数据。

callback: (可选) 载入成功时回调函数。

$.post(url,[data],[callback])

方法说明：简单的 Post 请求，

url: 请求 HTML 网页地址

data:(可选) 发送至服务器的 key/value(键值对) 数据。

callback: (可选) 载入成功时回调函数。

$.getJSON(url,[data],[callback])

方法说明：简单的 GET 请求载入 JSON 数据

url: 请求 HTML 网页地址

data:(可选) 发送至服务器的 key/value(键值对) 数据。

callback: (可选) 载入成功时回调函数。

$.getScript(url,[callback])

方法说明：简单的 GET 请求载入 script 脚本数据并执行

url: 请求 HTML 网页地址

data:(可选) 发送至服务器的 key/value(键值对) 数据。

callback: (可选) 载入成功时回调函数。

$.Ajax(options)

//此方法只要用于返回创建的 XMLHttpRequest 对象。大家如果看了 jQuery API 对 Ajax 的参数的讲解，肯定会觉得此方法
的内容较多，其实此方法只有一个值，是 key/value(也就是我们常说的"键值对")，下面将对此方法的参数进行逐一讲解。

注意：参数 options 是可选的其中键值对也都是可选的。

async://Boolean 类型 此键值对默认情况下为 true，也就是异步请求（局部刷新）；如果设置为 false，将会变成同步请求，那么此时将会锁住浏览器，用户无法对其进行其他操作，必须等待请求完毕后才会解锁。

global (Boolean 类型) //表示是否触发全局，默认为触发（true），Ajax 的全局设置将在后面的章节讲，设置全局则表示所有的 Ajax 将能够使用此全局内容，比如所有的 Ajax 事件都触发同一个路径。

type (String) : //(默认: "GET") 请求方式 ("POST" 或 "GET"), 默认为 "GET"。注意：其它 HTTP 请求方法，如 PUT 和 DELETE 也可以使用，但仅部分浏览器支持。

cache:(Boolean 类型) //设置 false 将不会从浏览器缓存中加载信息，用于 jQuery1.2 或更高的版本。

contentType:(String 类型) //用于设置编码格式，默认为: "application/x-www-form-urlencoded")格式，一般推荐此格式。

ifModified (Boolean) : //(默认: false) 仅在服务器数据改变时获取新数据。

processData (Boolean) : // (默认: true) 默认情况下，发送的数据将被转换为对象。

timeout (Number) ://设置置请求超时时间（毫秒）。此设置为全局设置。

dataType:(String 类型) //用于设置服务器返回的数据类型，但填写的内容也是有限制的，可用值如下

├xml //设置此值服务器端将 XML 文档，如果大家对 Ajax 比较了解，我想大

家也知道 Ajax 中的 x 代表是什么了吧？因此叫做 **A**(Asynchronous)**j**(javascript)**a**(and)**x**(xml)

├html //服务器返回 HTML 格式文档,（根据个人理解，如果按照遭上面的理解 xml 表示 x 的话，那么此类型不就表示 ajah 了嘛）。

├script //服务器返回【纯文本】的脚本，不会执行或进行计算。

├json //返回 Json 格式的文档

├text //返回纯文本

└jsonp //JSONP 是一个非官方的协议，它允许在服务器端集成 Script tags 返回至客户端，通过 javascript callback 的形式实现跨域访问，比如用户想得到 ["kwoojan","KwooShung"]，如果设置了 jsonp 那么服务器将返回 callbackFunction(["kwoojan","KwooShung"])

data(String 类型) //此方法至关重要，主要用于将数据发送至服务器。格式为键值对，如 userName=CSS 学习网&Address=http://www.cssxuexi.cn/ 那么服务器接受到的 userName 相对应的值就是 CSS 学习网

url (String) :// (默认：当前页地址) 发送请求的地址，也就是你的 data 数据需要被处理的地址。

beforeSend(function) //此键值为函数方法，当你发送请求之前调用此方法。例如当用户进行了某个操作，这时页面上显示出"正在加载中，请稍后"等类似的字样，多数情况下此键值对用于给予用户友好的提示。

success (Function) //当 Ajax 请求成功时，调用此方法，一般用于解析服务器所返回的数据。

error (Function) //求失败时调用此方法。

complete (Function) //当 Ajax 请求完毕的时候调用此方法。

【第三十四节】jQuery 速成 - Ajax 请求（实战篇）

今天我们对上节课的所讲解的内容进行实战，通过案例可以使我们更加熟练的掌握 jQuery Ajax。首先我们先来对这个案例的流程进行一下简要的说明。

注意：

① 下面的内容注释较多因而显着内容稍乱，从而给我们的心理上造成此课较难的印象，其实这是没必要的，只要专心的看下去，本节课真的很简单。

② 本课的服务器提交地址虽然是 aspx 结尾，但是【并没有因为服务器语言而对大家进行了限制，比如必须会.NET 才能学习本课，因为这节内容根本没有牵扯到 ASP.NET(C#) 的编程。】无论你是否会服务器编程这都不限制你学习本课所讲的 jQuery Ajax。

③ 本课的案例分别有 JavaEE、ASP.NET、ASP 和 PHP 这四种不同的版本，可以方便的帮助大家学习。

1、当在登录/注册窗口中的用户名输入框中输入完毕，将判读用户名是否已经被注册。

2、用户可以点击注册，将用户名和密码写入数据库。

3、用户点击登录时则判断用户是否存在，密码是否正确，符合以上条件便可以成功登录。

4、登录成功，在页面内显示出前最新注册的 13 条用户记录。

流程截图：


9、天天向上
8、好好学习
7、啊唆哈
6、效果帅吧？
5、CSS学习网
4、carrie
3、Alice
2、kwoojan
1、kwooshung
登录成功后，显示
前13条用户信息
www.w3cfuns.com


用户名和密码输入不完
整无法注册或登录
当输入框失去焦点
判断用户是否注册
注册 / 登录
用户名: kwooshung
用户名已经被注册!
密 码:
注 册
登 录
点击注册则进行注册
点击登录则进行登录
www.w3cfuns.com

以下是 HTML 页面中比较重要的代码

```
<input id="userName" type="text" />//输入框 - 用户名
<input id="userPwd" type="password" />//输入框 - 密码
<p id="userNameError"></p>//用户名错误提示信息
<p id="userPwdError"></p>//密码错误提示信息
<a id="bt_Reg" href="#"></a>//注册按钮
<a id="bt_Login" href="#"></a>//登录按钮
```

判断用户名是否已经被注册

```
/*
* 下面的代码与案例中代码稍微有些不一样，因为没有判断用户名
* 或者密码是否为空等操作，下面的代码是只要失去焦点就执行。
* 当用户名点击 bt_Login 这个按钮的时候则异步调用执行服务器程序
* 从而进行判断用户名是否已经被注册
*/
$("#bt_Login").click
(
        function()
        {
                $.ajax
                ({
                        type: "POST",//使用 POST 方式进行提交
                        cache:false,//不从浏览器缓存中读取信息
                        url: "userServer.aspx",//待处理的页面
                        dataType:"json",//设置处理格式为 Json
                        data: "class=check&userName="+ $("#userName").val(),
/*
 * 这里的键值对随便写，class=check 这就是一个键值对。
 * 多个键值对使用&(and 符)连接
 * =(等号)右边为传输的内容
 * 按照平时，我们只需要传输 userName 即可
 * 由于案例中的处理页面根据 class 判断类型
 * 从而执行不同的操作（检测用户名/注册/登录）所以多写了一个
 * classclass=check 这就是一个键值对
 */

                        beforeSend :function()
                        {
/*
* 提交请求中(服务器没有返回数据或正在处理信息的时候)
* 我们给用户一个友好的提示
*/
                                $("#userNameError").text("正在验证，请稍后。");
                        },
                        success: function(data)
                        {
                                $("#userNameError").text(data.info);
```

```
/*
* data 为一个 Json 对象，由服务器段创建的一段字符串，将其返回
*  假如在用户名输入框中输入"CSS 学习网"经过服务器的判断此用户名存在
*  则拼成的字符串如下
* {"info":"此用户名已经被注册",}这就是一个最简单的 Json 格式的字符串
* $("#userNameError").text(data.info); 等同于 $("userNameError").text("此用户名已经被注册")
*/
                        },
                        error: function()
//当访问失败或者服务器出错，则直接提示访问数据失败。
                        {
                                alert("访问数据失败！");
                        }
                });
        }
);
```

由于上面的代码加了注释因而显着较乱，我们对它进行一下整理，代码如下：

```
$("#bt_Login").click
(
        function()
        {
                $.ajax
                ({
                        type: "POST",
                        cache:false,
                        url: "userServer.aspx",
                        dataType:"json",
                        data: "class=check&userName="+ $("#userName").val(),
                        beforeSend :function()
                        {
                                $("#userNameError").text("正在验证，请稍后。");
                        },
                        success: function(data)
```

```
                {
                        $("#userNameError").text(data.info);
                },
                error: function()
                {
                        alert("访问数据失败！");
                }
        });
    }
);
```

这样一看，代码是不是很简单？

由于登录与注册跟上面的检测用户名差不多，只不过传参与提交地址不一致而已，所以注册登录省略不讲，大家看案例即可。下面我们对$.getJSON(url,[data],[callback])进行实际应用，只有已经登录的用户才能显示内容。

```
$.getJSON
(
        "userList.aspx?random=" + Math.random(),
/*
* 向处理页面(userList.aspx)发送请求，对其"索要"处理结果
* 按常理来说地址应该写成 userList.aspx 从?问号开始(包括问号)
* 根本不用写后面的随即数，由于次方法从服务器的缓存中读取内容
* 【假设】数据库目前就 13 条数据且第 13 条数据是 kwooshung
* 第一次访问的时候获得的后 13 条用户也就是从 1-13，最后一条是
* kwooshung，这时突然又注册进来一条新的数据叫做 kwoojan
* 那么读取的内容就是 2-14 条，这才是最新的数据
* 当第二次请求此页面的时候还是从【缓存中读取】，页面自然
* 呈现的最新的数据还是 1-13 条的内容，最后一个注册的是 kwooshung
* 这是不对的，因为目前最后一个注册的是 kwoojan
* 所以 url 设置为不会重复的随机数，这样每次请求就是一个新 URL，
* 也就不会从缓存中读取信息了，从而每次读取信息都是最新的。
* 如果上面的例子没有读懂，没有关系，只要记住设置随机数是为了
* 防止每次读取内容都是读取缓存造成每次读取的都不是最新的数据
```

```
*/
        function(data)
        {
/*
* 我相信很多朋友都遇到此问题，不知道如何取出多条数据
* 应该怎么循环出数据。其实多条数据也很简单，只不过返回的
* Json 没有上面的验证用户是否被注册的 Json 的写法那么简单罢了
* 假如有 3 条数据如下:
* ID    userName
* 1     CSS 学习网
* 2     KwooJan
* 3     KwooShung

* 以上数据如果使用 Json 来表示便是(倒序):
* {
*         "userList":
*         [
*                 {"id":"3","userName":"KwooShung"},
*                 {"id":"2","userName":"KwooJan"},
*                 {"id":"1","userName":"CSS 学习网"}
*         ]
* }
*
* userList、ID、userName 是随便起的名字
* 目前我们只需知道多条 Json 数据按照上面的格式如何写即可。
* 看完下面的循环继续深入讲解。
*/
                for(var i=0; i < data.userList.length; i++)
                {
                        $("#content").html
                        (
                                $("#content").html()
                                + "<li>"+ data.userList.id +"、
                                "+ data.userList.userName +"</li>"
                        );
                }
/*
* 通过上面的循环我们可以看出 data 便是上面所写出的 Json 数据
```

```
* userList 是 data 中的某一组 json 的数据名称
*  那么 data.userList.length 就是获取 userList 中有多少条数据
*  我们不难看出次循环需要执行 3 次
*  我们向 ID 为 content 的 html 元素中输入 Html 内容
*
* $("#content").html
* (
*  每次输入 HTML 都将原来的内容取出并与新的数据拼接字符串并输入
*          $("#content").html()
*          + "<li>"+ data.userList.id +"、
*          "+ data.userList.userName +"</li>"
* );
*
*  最终 HTML 内容如下
* <ul id="content">
*           <li>3、KwooShung</li>
*           <li>2、KwooJan</li>
*           <li>1、CSS 学习网</li>
* </ul>
*
*/
        }
);
```

上述代码整理后如下：

```
$.getJSON
(
        "userList.aspx?random=" + Math.random(),
        function(data)
        {
                for(var i=0; i < data.userList.length; i++)
                {
                        $("#content").html
                        (
                                $("#content").html()
```

```
                    + "<li>"+ data.userList.id +"、
                    "+ data.userList.userName +"</li>"
                );
            }
        }
);
```

总结：

通过上面的讲解，KwooShung 已经为大家讲解完了 Ajax 请求的所有内容。
至于

```
$.get(url,[data],[callback])
$.post(url,[data],[callback])
$.getScript(url,[callback])
```

这四种请求方法与$.getJson(url,[data],[callback])的使用方法如出一辙，
而他们唯一的区别就是返回的数据不同。
使用方法已经在上节课中简单的讲解过，在此便不在做过多的讲解。

目前为大家提供了 JAVAEE 版本和 ASP.NET 版本的 jQuery Ajax 异步请求/调用案例。
近期将会放上 ASP 和 PHP 版本，敬请期待。

本课案例：

 ASPX_jQuery_Ajax.rar (900.11 KB, 下载次数: 68)

 JAVAEE_jQuery_Ajax.rar (2.42 MB, 下载次数: 32)

整理：Joychao www.joychao.cc