# jQuery 入门[1] - 构造函数

http://thinhunan.cnblogs.com/archive/2008/03/05/1091816.html**jQuery 优点**

◦ 体积小（v1.2.3 15kb）
◦ 丰富的 DOM 选择器(CSS1-3 + XPath) ◦跨浏览器（IE6,FF,Safari,Opera）
◦ 链式代码
◦ 强大的事件、样式支持
◦ 强大的 AJAX 功能
◦ 易于扩展，插件丰富

jQuery 的构造函数接收四种类型的参数：

jQuery(expression,context)
jQuery(html)
jQuery(elements)
jQuery(fn)

第一种根据表达式（ID，DOM 元素名，CSS 表达式，XPath 表达式）找出文档中的元素，并组装成一个 jQuery 对象返回。

DEMO:

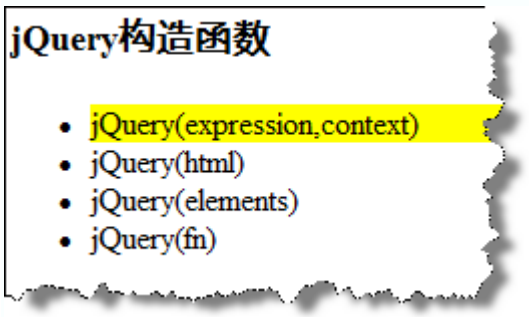
```
DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>jQuery basictitle>
    <style type="text/css">
        .selected
        {
            background-color:Yellow;
        }
    style>
    <script src="../scripts/jquery-1.2.3.intellisense.js"
type="text/javascript">script>
head>
<body>
    <h3>jQuery 构造函数 h3>
    <ul>
        <li>jQuery(expression,context)li>
        <li>jQuery(html)li>
        <li>jQuery(elements)li>
        <li>jQuery(fn)li>
    ul>
    <script type="text/javascript">
```

```
    script>
body>
html>
```

将以下 jQuery 代码加入文末的脚本块中：

```
jQuery("ul>li:first").addClass("selected");
```

页面运行效果如下：



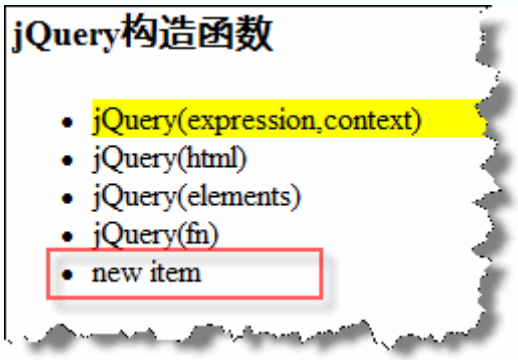

其中 jQuery()可替换为快捷方式$()，如果$被其它对象占用，可使用 jQuery.noConflict()函数取消快捷方式。

"ul>li:first"中 ul>li 表示所有位于 ul 下的 li 元素（为 CSS 表达式，XPath 方式可用 ul/li），:first 表示其中的第一个。addClass()为 jQuery 对象用来添加 CSS 样式类的函数，相反的函数为 removeClass()。

再加入以下代码：

```
$('ul').append($('
    new item '));
```

运行效果如下：

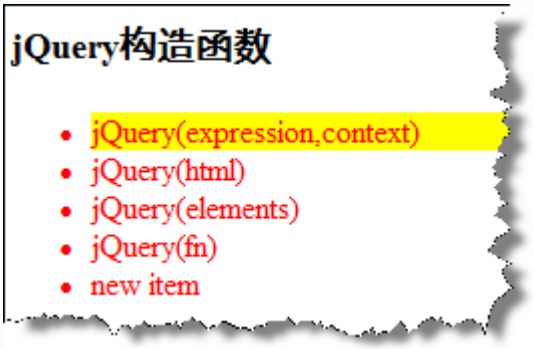



其中$('
    new item
')将其中的字符串转换为 DOM 对象，然后通过 append()函数加入 ul 对象的最后。

接下来：

```
        $(document).ready(function(){
            $('ul').css('color','red');
        });
```

则效果如：




jQuery 构造函数中还可以直接传入 DOM 对象，如 document，或 jQuery 对象（当然就没什么意义）。ready()函数为 document 添加事件处理函数，将 ul 的颜色设为红色。$(document).ready()由于应用场景众多，所以可以直接用$(fn)来代替，fn 表示处理函数。（ready 处理函数貌似在文档内容载入完成后执行，无需等待相关其它图片等资源载入完成，所以比 load 事件要更早执行，对于这点，没有具体证实）

```
$(function(){
    alert('welcome to jQuery');
});
```

以上代码的效果是页面一载入，就弹出一个对话框。

jQuery1.2 选择器

**jQuery1.2选择器**

以下的文档根据官网1.2选择器汉化，并做相应的调整及加入了部份示例。

由于实际使用中选择器在 IE 和非 IE 下会有不同的效果，请参照红色的字样。如有错误请及时联系我。

绯雨汉化：http://feiyu.asgard.cn/http://feiyu.asgard.cn

| 基本选择器 | |
|---|---|
| **#myid** | 返回:对象> |
| 匹配一个 id 为 myid 的元素。 | |
| **element** | 返回:对象>数组 |
| 匹配所有的 element 元素 | |
| **.myclass** | 返回:对象>数组 |
| 匹配所有 class 为 myclass 的元素 | |
| ***** | 返回:对象>数组 |
| 匹配所有元素。该选择器会选择文档中所有的元素，包括 html，head，body | |
| **selector1,selector2,selectorN** | 返回:对象>数组 |
| 匹配所有满足 selector1或 selector2或 selectorN 的元素 | |
| **层次选择** | |
| **elementParent elementChild** | 返回:对象>数组 |

匹配 elementParent 下的所有子元素 elementChild。例如：$("div p")选择所有 div 下的 p 元素

**elementParent > elementChild** 返回:对象>数组

匹配 elementParent 下的子元素 elementChild。例如：$("div>p")选择所有上级元素为 div 的 p 元素

**prev+next** 返回:对象>数组

匹配 prev 同级之后紧邻的元素 next。例如：$("h1+div")选择所有 div 同级之前为 h1的元素（）

**prev ~ siblings** 返回:对象>数组

匹配 prev 同级之后的元素 siblings。例如：$("h1~div")可以匹配（）

## 基本滤镜

**:first** 返回:对象>

匹配第一个元素

**:last** 返回:对象>

匹配最后一个元素

**:not(selector)** 返回:对象>数组

匹配不满足 selector 的元素

**:has(selector)** 返回:对象>数组

匹配包含满足 selector 的元素。此选择器为1.2新增

**:even** 返回:对象>数组

从匹配的元素集中取序数为偶数的元素。

**:odd** 返回:对象>数组

从匹配的元素集中取序数为奇数的元素。

**:eq(index)** 返回:对象>数组

从匹配的元素集中取第 index 个元素

**:gt(index)** 返回:对象>数组

从匹配的元素中取序数大于 index 的元素

**:lt(index)** 返回:对象>数组

从匹配的元素中取序数小于 index 的元素

**:header** 返回:对象>数组

匹配所有的标题元素，例如 h1，h2，h3.....hN。此选择器

为1.2新增

**:animated** 返回:对象>数组

匹配正在执行动画的元素。此选择器为1.2新增

**:empty** 返回:对象>数组

匹配所有没有子元素（包括文本内容）的元素

**:parent** 返回:对象>数组

匹配包含子元素（包含文本内容）的所有元素

**:contains(text)** 返回:对象>数组

匹配所有含有 text 的元素

**:hidden** 返回:对象>数组

匹配所有隐藏的元素，包含属性 type 值为 hidden 的元素

**:visible** 返回:对象>数组

匹配所有非隐藏的元素

**子元素滤镜**

**E:nth-child(index/even/odd/equation)** 返回:对象>数组

匹配所有 E 在其父元素下满足（index/even/odd/equation）条件的集合。注：下标从1开始

**E:first-child** 返回:对象>数组

匹配所有 E 在其父元素下是第一个子元素的集合。例如：HTML（"），使用$("p:first-child")，选取：

**E:last-child** 返回:对象>数组

匹配所有 E 在其父元素下是最后一个子元素的集合。例如：同上的 HTML，使用$("p:last-child")，选取：

**E:only-child** 返回:对象>数组

匹配所有 E 是其父元素的唯一子元素的集合。例如：同上的 HTML，使用$("p:only-child")，选取：

**表单滤镜**

**:input** 返回:对象>数组

匹配所有的 input、textarea、select、button

**:text** 返回:对象>数组

匹配文本域。注：在 IE 浏览器下，选择的对象是所有 type 属性为 text 的元素，在非 IE 浏览器下，选择的对象是 input 元素 type 属性为 text 的元素

**:password** 返回:对象>数组

匹配密码域。注：在 IE 浏览器下，选择的对象是所有 type 属性为 password 的元素，在非 IE 浏览器下，选择的对象是 input 元素 type 属性为 password 的元素

**:radio** 返回:对象>数组

匹配单选按钮。注：在 IE 浏览器下，选择的对象是所有 type 属性为 radio 的元素，在非 IE 浏览器下，选择的对象是 input 元素 type 属性为 radio 的元素

**:checkbox** 返回:对象>数组

匹配复选框。注：在 IE 浏览器下，选择的对象是所有 type 属性为 checkbox 的元素，在非 IE 浏览器下，选择的对象是 input 元素 type 属性为 checkbox 的元素

**:submit** 返回:对象>数组

匹配提交按钮。注：在 IE 浏览器下，选择的对象是所有 type 属性为 submit 的元素，在非 IE 浏览器下，选择的对象是 input 元素 type 属性为 submit 的元素和 button 元素 type 属性为空或为 submit 的元素

**:image** 返回:对象>数组

匹配图像域。注：在 IE 浏览器下，选择的对象是所有 type 属性为 image 的元素，在非 IE 浏览器下，选择的对象是 input 元素 type 属性为 image 的元素

**:reset** 返回:对象>数组

匹配重置按钮。注：在 IE 浏览器下，选择的对象是所有 type 属性为 reset 的元素，在非 IE 浏览器下，选择的对象是 input 或 button 元素 type 属性为 reset 的元素

**:button** 返回:对象>数组

匹配按钮。注：在 IE 浏览器下，选择的对象是所有 type 属性为 button 的元素和元素名为 button 的元素，在非 IE 浏览器下，选择的对象是 input 元素 type 属性为 button 的元素和元素名为 button 的元素

**:file** 返回:对象>数组

匹配文件域。注：在 IE 浏览器下，选择的对象是所有 type 属性为 file 的元素，在非 IE 浏览器下，选择的对象是 input 元素 type 属性为 file 的元素

**:enabled** 返回:对象>数组

匹配所有可用的元素。注：即:not(:disabled)，参考:disabled 的注释

**:disabled** 返回:对象>数组

匹配所有禁用的元素。注：在非 IE 浏览器下，选择的对象是禁用的表单元素

**:checked** 返回:对象>数组

匹配所有被选中的表单。注：在 IE 浏览器下，选择的对象是含有 checked 属性的所有元素

**:selected** 返回:对象>数组

匹配所有选择的表单。注：在 IE 浏览器下，选择的对象是含有 selected 属性的所有元素

**属性滤镜**

**[attribute]** 返回:对象>数组

匹配拥有 attribute 属性的元素

**[attribute=value]** 返回:对象>数组

匹配属性 attribute 为 value 的元素

**[attribute!=value]** 返回:对象>数组

匹配属性 attribute 不为 value 的元素

**[attribute^=value]** 返回:对象>数组

匹配属性 attribute 的值以 value 开始的元素

**[attribute$=value]** 返回:对象>数组

匹配属性 attribute 的值以 value 结尾的元素

**[attribute*=value]** 返回:对象>数组

匹配属性 attribute 的值包含 value 的元素

**[selector1][selector2][selectorN]** 返回:对象>数组

匹配满足属性选择器 selector1、selector2、selectorN 的元素

**热榜**

这个热榜展示了最常用选择器的排名，数据是根据使用 jQuery 的著名网站分析得来的。热榜中的选择器已被归类（如，'div span'和'ul li'都是'tag tag'形式）。红色部分表示与 W3C 规范不兼容。% Used 一栏表示选择器使用百分比（# of Uses 一栏表示选择器使用次数）。只用过一次的选择器没在热榜中罗列。

| ***Selector*** | **% Used** | **# of Uses** |
|---|---|---|
| #id | 51.290 % | 1431 |
| .class | 13.082 % | 365 |
| tag | 6.416% | 179 |
| tag.class | 3.978% | 111 |
| #id tag | 2.151% | 60 |
| tag#id | 1.935% | 54 |
| #id:visible | 1.577% | 44 |
| #id .class | 1.434% | 40 |

| | | |
|---|---|---|
| .class .class | 1.183% | 33 |
| * | 0.968% | 27 |
| #id tag.class | 0.932% | 26 |
| #id:hidden | 0.789% | 22 |
| tag[name=value] | 0.645% | 18 |
| .class tag | 0.573% | 16 |
| [name=value] | 0.538% | 15 |
| tag tag | 0.502% | 14 |
| #id #id | 0.430% | 12 |
| #id tag tag | 0.358% | 10 |
| tag[name$=value] | 0.323% | 9 |
| #id:checkbox | 0.323% | 9 |
| #id #id:selected | 0.287% | 8 |
| .class tag.class | 0.287% | 8 |
| tag#id > tag | 0.287% | 8 |
| tag, tag | 0.251% | 7 |
| tag.class tag | 0.251% | 7 |
| tag .class | 0.215% | 6 |
| :radio | 0.215% | 6 |
| #id, #id, #id | 0.215% | 6 |
| #id .class tag | 0.215% | 6 |
| :text | 0.215% | 6 |
| tag, tag, tag | 0.215% | 6 |
| .class .class .class | 0.215% | 6 |
| #id tag[name=value] | 0.179% | 5 |
| :checkbox | 0.179% | 5 |
| tag[name*=value] | 0.179% | 5 |
| #id, #id | 0.179% | 5 |
| tag.class tag.class tag | 0.143% | 4 |
| tag.class .class | 0.143% | 4 |
| tag:selected | 0.143% | 4 |
| .class .class .class tag | 0.143% | 4 |
| .class.class | 0.143% | 4 |
| tag:file | 0.143% | 4 |
| tag, tag[name=value] | 0.143% | 4 |
| #id, tag[name$=value] | 0.143% | 4 |
| tag[name!=value] | 0.143% | 4 |
| .class .class tag + .class | 0.108% | 3 |
| .class .class tag:gt(2) | 0.108% | 3 |
| tag:submit, tag:image, tag:submit | 0.108% | 3 |

| | | |
|---|---|---|
| tag.class tag.class tag.class tag.class tag.class tag.class | 0.108% | 3 |
| #id tag tag.class tag.class tag | 0.108% | 3 |
| :input | 0.108% | 3 |
| tag.class tag.class | 0.108% | 3 |
| .class .class tag:has(tag[name^=value]) | 0.108% | 3 |
| #id tag.class tag | 0.108% | 3 |
| tag:eq(0) | 0.108% | 3 |
| #id:input | 0.108% | 3 |
| tag#id tag#id tag | 0.108% | 3 |
| .class, .class | 0.108% | 3 |
| tag:eq(1) | 0.108% | 3 |
| tag#id tag | 0.108% | 3 |
| .class tag#id | 0.072% | 2 |
| #id tag:first | 0.072% | 2 |
| #id tag tag[name!=value] | 0.072% | 2 |
| .class #id tag tag | 0.072% | 2 |
| tag#id tag.class | 0.072% | 2 |
| tag:filled | 0.072% | 2 |
| tag:first | 0.072% | 2 |
| .class tag tag.class | 0.072% | 2 |
| #id tag.class tag tag.class | 0.072% | 2 |
| tag.class tag.class tag.class | 0.072% | 2 |
| #id #id #id tag.class | 0.072% | 2 |
| tag[name$=value], #id | 0.072% | 2 |
| tag tag tag | 0.072% | 2 |
| :submit, tag:image | 0.072% | 2 |
| .class .class tag | 0.072% | 2 |
| [name=value]:first | 0.072% | 2 |
| .class .class, #id | 0.072% | 2 |
| #id .class tag.class | 0.072% | 2 |
| .class #id .class | 0.072% | 2 |
| #id #id tag tag | 0.072% | 2 |
| tag[name] | 0.072% | 2 |
| tag, tag, tag, tag | 0.072% | 2 |
| tag#id tag#id tag.class | 0.072% | 2 |
| tag:unchecked | 0.072% | 2 |
| #id .class.class | 0.072% | 2 |
| #id tag.class tag > tag | 0.072% | 2 |
| .class tag tag tag | 0.072% | 2 |

| | | |
|---|---|---|
| tag.class:first | 0.072% | 2 |
| .class tag.class tag | 0.072% | 2 |
| tag#id tag.class:first | 0.072% | 2 |
| #id tag.class tag.class tag | 0.072% | 2 |
| .class tag tag | 0.072% | 2 |
| #id .class tag tag | 0.072% | 2 |
| #id tag tag#id | 0.072% | 2 |
| tag.class > tag | 0.072% | 2 |
| #id .class * | 0.072% | 2 |
| :input:visible | 0.072% | 2 |
| #id .class .class | 0.072% | 2 |
| #id > tag > tag > tag > tag > tag | 0.072% | 2 |
| #id tag.class tag tag:gt(0) | 0.072% | 2 |
| .class, .class, .class | 0.072% | 2 |
| #id #id * | 0.072% | 2 |
| #id > *:not(#id) | 0.072% | 2 |
| #id tag[name^=value] | 0.072% | 2 |
| .class tag.class | 0.072% | 2 |
| tag:blank | 0.072% | 2 |

## *jQuery 入门[2] - 选择器*

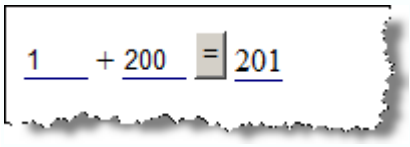
http://thinhunan.cnblogs.com/archive/2008/03/05/1091816.htmljQuery 之所以令人爱不释手，在于其强大的选择器表达式令 DOM 操作优雅而艺术。
jQuery 的选择符支持 id,tagName,css1-3 expressions,XPath，参见：
http://docs.jquery.com/Selectors

### DEMO:

```
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>
    <title>Selectortitle>
    <script src="../scripts/jquery-1.2.3.intellisense.js"
type="text/javascript">script>
head>
<body>
    <input value="1" /> +
    <input value="2" />
    <input type="button" value="=" />
```

```
    <label> label>
    <script type="text/javascript">
        $("input[type='button']").click(function(){
            var i = 0;
            $("input[type='text']").each(function(){
                i += parseInt($(this).val());
            });
            $('label').text(i);
        });
        $('input:lt(2)')
            .add('label')
            .css('border','none')
            .css('borderBottom','solid 1px navy')
            .css({'width':'30px'});
    script>
body>
html>
```

运行效果如下：




代码分解：

$("input[type='button']")用于找到 type 属性为 button 的 input 元素（此为 CSS 表达式，IE7才开始支持，所以在 IE6中通常用 jQuery 的这种表达式代替 CSS 代码设置样式）。click()函数为 button 添加 click 事件处理函数。

在 button_click 时，$("input[type='text']")找出所有输入框，each()函数遍历找出来的数组中的对象的值，相加后设到 label 中。

```
        $('input:lt(2)')
            .add('label')
```

两行代码意为：所有 input 中的前面两个（lt 表示序号小于）再加上 label 对象合并成一个 jQuery 对象。

```
            .css('border','none')
            .css('borderBottom','solid 1px navy')
            .css({'width':'30px'});
```

以上三行代码都是针对之前的 jQuery 对象设置 CSS 样式，如果一次需要设置多个 CSS 值，可用另一种形式，如：

```
            .css({'border':'none','borderBottom':'solid 1px
navy','width':'30px'});
```

css()函数如果只传一个字符串参数，则为取样式值，比如 css('color')为取得当前 jQuery 对象的样式属性 color 的值。jQuery 对象有多种这样的函数，比如，val('')为设 value，val()为取 value，text('text')为设 innerText，text()为取得 innerText，此外还有 html()，用于操作 innerHTML，而 click(fn)/click()，change(fn)/change()……系统函数则为事件的设置处理函数与触发事件。

由于多数 jQuery 对象的方法仍返回当前 jQuery，所以 jQuery 代码通常写成一串串的，如上面的

```
.css('border','none')
.css('borderBottom','solid 1px navy')
.css({'width':'30px'});
```

，而不需要不断重复定位对象，这是 jQuery 的链式特点，后面文章还会有补充。

## *jQuery 入门[3] - 事件*

http://thinhunan.cnblogs.com/archive/2008/03/05/1091816.htmljQuery 对事件的支持主要包括：

bind()--为事件绑定处理程序，如：

```
$("p").bind("mouseenter mouseleave", function(e){
$(this).toggleClass("over");
});
```

unbind()--注销绑定在事件上的处理程序，如：$(document).unbind('ready');，如不给参数，则清除所有事件处理程序。

```
$("#unbind").click(function () {
$("#theone").unbind('click', aClick);
});
```

trigger()--触发某类事件。

```
$("button:first").trigger('click');
```

triggerHandler()--触发某类事件，但不触发默认的事件处理逻辑，比如 a 的定向。

```
$("input").triggerHandler("focus");
```

one()——为事件绑定只能被触发一次的处理程序。

```
$("div").one("click", function(){
});
```

ready()/click()/change()/toggle(fn,fn)/dblclick()……各种常规事件的快捷方式，xxx(fn)为绑定处理程序，xxx()为触发事件

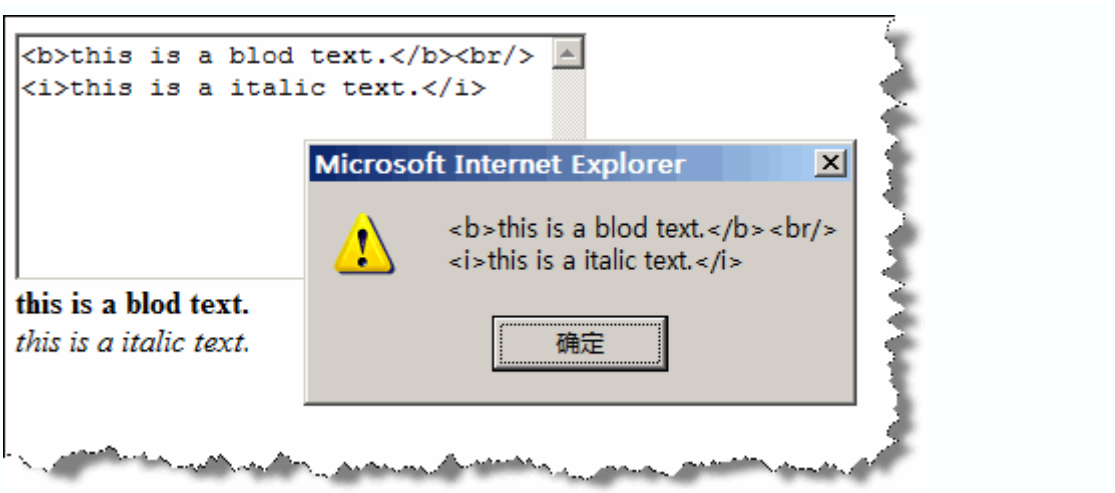
jQuery 1.2的事件支持命名空间，

```
$("div").bind("click",function(){ alert("hello"); });

$("div").bind("click.plugin",function(){ alert("goodbye"); });

$("div").trigger("click!");// alert("hello") only
```

## DEMO:

```
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>
    <title>Eventstitle>
    <script src="../scripts/jquery-1.2.3.intellisense.js"
type="text/javascript">script>
    <style type="text/css">
        textarea
        {
            height: 118px;
            width: 280px;
        }
    style>
    <script type="text/javascript">
        $(function(){
            $('textarea').bind('propertychange',function(){
                $('#result').html($('textarea').val())
            }
            ).bind('change',function(){
                alert($('textarea').val());
            });
        });

    script>
head>
<body>
    <textarea>textarea>
    <div id='result'>div>
body>
html>
```

运行效果如下：

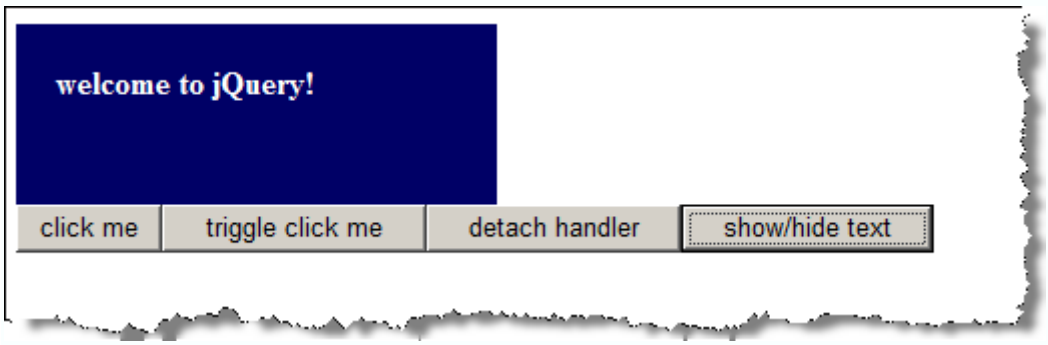



# jQuery 入门[4] - 链式代码

http://thinhunan.cnblogs.com/archive/2008/03/05/1091816.htmljQuery 另一个很令人惬意的地方是，一般的代码都是一行一行写，jQuery 的代码可以一串一串写。
这一点，在前面的文章中已经介绍过了。
直接来一个 Demo:

```
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>
    <title>chainning codetitle>
    <script src="../scripts/jquery-1.2.3.intellisense.js"
type="text/javascript">script>
    <script type="text/javascript">
        $(function(){
            //添加四个按钮
            $('').appendTo($('body'));
            //找出所有按钮
            $('input[type="button"]')
                .eq(0)//找到第一个按钮
                    .click(function(){
                        alert('you clicked me!');
                    })
                .end().eq(1)//返回所有按钮，再找到第二个
                    .click(function(){
                        $('input[type="button"]:eq(0)').trigger('click');
                    })
                .end().eq(2)//返回所有按钮，再找到第三个
```

```
                    .click(function(){
                        $('input[type="button"]:eq(0)').unbind('click');
                    })
                .end().eq(3)//返回所有按钮，再找到第四个
                    .toggle(function(){
                        $('.panel').hide('slow');
                    },function(){
                        $('.panel').show('slow');
                    });
        });
    script>
    <style type="text/css">
        .panel
        {
            padding: 20px;
            background-color: #000066;
            color: #FFFFFF;
            font-weight: bold;
            width: 200px;
            height: 50px;
        }
    style>
head>
<body>
    <div class="panel">welcome to jQuery!div>
body>
html>
```




现在，链式代码已经成为 jQuery 非常流行的一个特点了，在使用链条方式写代码时，可能会用到 eq()/filter()……（reference:http://docs.jquery.com/Traversing）等方法变化 jQuery 对象的对应范围，然后，又可以用 end()函数将范围复原到原来的状况。

需要注意的是，有几个函数并不返回 jQuery 对象，所以链条就不能继续下去，比如 get()就不能像 eq()那样用。

# jQuery 入门[5] - AJAX

http://thinhunan.cnblogs.com/archive/2008/03/05/1091816.htmljQuery 为 AJAX 提供了非常丰富的支持，参见 Ajax
其中最基本当属$ajax()，通过不同的参数，这个方法可以灵活支持各种 AJAX 应用场景。
如：

```
$.ajax({
url: "test.html",
cache: false,
success: function(html){
$("#results").append(html);
}
});
```

完整参数列表参见：options
当然，常用的应该是这些：

load()--直接将 AJAX 请求结果作为 jQuery 对象内容
$.get()--用 get 方式请求
$.post()--用 post 方式提交
ajaxStart()/ajaxComplete()/ajaxError()......--全局的 ajax 事件响应

## DEMO

建一个 GenericHandler 作 AJAX 请求服务端:CubeHandler.ashx

```
<%@ WebHandler Language="C#" Class="CubeHandler" %>

using System;
using System.Web;

public class CubeHandler : IHttpHandler {

    public void ProcessRequest (HttpContext context) {
        context.Response.ContentType = "text/plain";
        int number = 0;
        int.TryParse(context.Request.Params["number"], out number);
        context.Response.StatusCode = 200;
        context.Response.Cache.SetCacheability(HttpCacheability.NoCache);
        context.Response.Write(string.Format("{0} cubed is
{1}",number,Math.Pow(number, 3)));
    }

    public bool IsReusable {
        get {
```

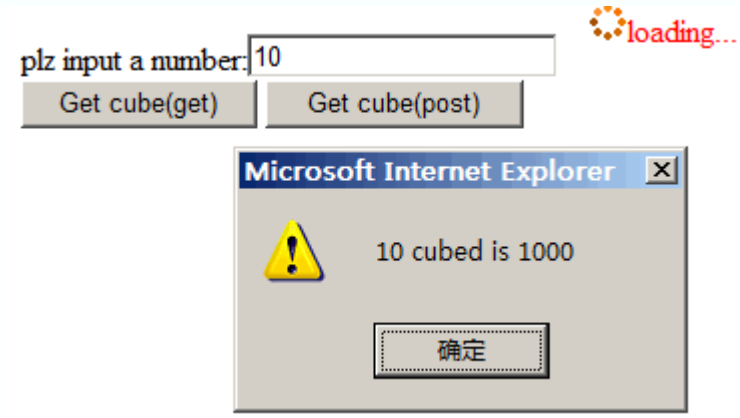
```
            return true;
        }
    }
}
```

因为用的是 Request.Params,所以这个 handler 能同时支持 get 和 post，

```
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>
    <title>ajaxtitle>
    <script src="../scripts/jquery-1.2.3.intellisense.js"
type="text/javascript">script>
    <script type="text/javascript">
        $(function(){
            //设置指示器
            $('#divIndicator').ajaxStart(function(){$(this).show()})
                              .ajaxSuccess(function(){$(this).hide()})
                              .ajaxError(function(msg){$(this).hide();alert(msg
);});
            //ajax get 请求
            $('#btnGetCubeInGet').click(function(){
                var number = $('#txtNumber').val();
                $.get('CubeHandler.ashx?number='+number,function(result){
                  alert(result);
                });
            });

            //ajax post 提交
            $('#btnGetCubeInPost').click(function(){
                var number = $('#txtNumber').val();
                $.get('CubeHandler.ashx',{'number':number},function(result){
                    alert(result);
                });
            });
        });
    script>
    <style type="text/css">
        .indicator
        {
            color: #FF0000;
            position: absolute;
            top: 0px;
            right: 0px;
```

```
            display: none;
        }
    style>
head>
<body>
    <div id="divIndicator" class="indicator">

        <img src="indicator.gif" />loading···div>
    plz input a number:<input id="txtNumber" />
    <input type="button" id="btnGetCubeInGet" value="Get cube(get)" />
    <input type="button" id="btnGetCubeInPost" value="Get cube(post)" />
body>
html>
```

点击按钮后的效果：




# *jQuery 入门[6] - 动画*

jQuery 直接各种动画，常见的被封装成各种方法，如 show()/hide()/slideDown()/fadeIn()等等，参见：Effects

最灵活的则属于 animate( params, [duration], [easing], [callback] )方法，参见：animate
其中 params 为动画的运行结果，可以为各种样式属性，jQuery 将在 duration 指定的时间内，将对象的当前状态渐变为 params 参数指定的值。如：

```
    $("#go").click(function(){
      $("#block").animate({
        width: "70%",
        opacity: 0.4,
        marginLeft: "0.6in",
        fontSize: "3em",
```

```
        borderWidth: "10px"
      }, 1500 );
    });
```

params 也可以是一些相对数据：

```
    $("#right").click(function(){
      $(".block").animate({"left": "+=50px"}, "slow");
    });

    $("#left").click(function(){
      $(".block").animate({"left": "-=50px"}, "slow");
    });
```

通过 stop()函数可将对象再在执行的动画暂停。选择符:animated 可以判断对象是否处在动画运行状态。
以下为来自官网的一个例子：

```
DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
                    "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
  <script src="http://code.jquery.com/jquery-latest.js">script>

  <script>
  $(document).ready(function(){

    $("#show").click(function () {
      var n = $("div").queue("fx");
      $("span").text("Queue length is: " + n.length);
    });
    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);
    }
    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>
<body>
  <button id="show">Show Length of Queuebutton>
  <span>span>
  <div>div>
body>
html>
```

效果为不断变化的一个方块，因为最后一个动画$("div").slideUp("normal", runIt)执行后又
调用 runIt 方法，所以动画不断循环。