

MooTools 1.2 介绍

原文地址: [30 Days of Mootools 1.2 Tutorials - Day 1 - Intro to the Library](#)

请尊重个人劳动, 转载请注明出处: <http://ooboy.net>, 译者: Fdream

这里是 [《MooTools 1.2 系列教程目录》](#)。

有人最近要求我们写一个关于 MooTools 1.2 的 30 天的教程, 这似乎也是个很不错的主意, 于是我们决定现在就开始。在这些教程中, 我们假设用户没有任何 MooTools 或者是 JavaScript 经验, 但是至少有基本的 HTML 和 CSS 知识。

MooTools 1.2 JavaScript 库介绍

[MooTools 1.2](#) 是个强大的轻量级的 JavaScript 库, 专门为减轻 Web 中交互性 JavaScript 开发。在某种程度上, 你可以认为 MooTools 是 CSS 的扩展。例如, CSS 可以让你在鼠标移上去时发生改变。JavaScript 允许你接触更多的时间 (点击事件、鼠标悬停事件、键盘事件……), MooTools 让这一切变得非常容易。

另外, MooTools 还有各种各样的非常好的扩展, 可以让你不只是改变一个元素的属性, 还可以让你有 "morph" (变形) 或者 "tween" (补间动画) 属性, 让你有能力去创建动画效果, 就像你在我的导航菜单上看到的一样 (Fdream 注: 原作者的, [我的首页](#) 也有)。

这只是一个例子, MooTools 可以让你做更多的事情。在接下来的 30 天里, 我们将深入 MooTools 库, 探索从数组 (Array) 和函数 (Function) 到 Fx.Slide, 以及其他捆绑插件的每一个东西。

引用 MooTools 1.2

首先, 下载并引用 MooTools 1.2 核心库。

1. 下载 [MooTools 1.2 核心库](#)
2. 把 MooTools 1.2 核心库上传到你的服务器或者工作区
3. 在你的 HTML 文档头部 head 标记之内链接 MooTools 1.2 核心库

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. <script src="mootools-1.2-core.js" type="text/javascript"></script>
```

(Fdream 注: 现在 MooTools 1.2 下载下来后, 默认的后缀名是 ".txt", 请更改后缀为 ".js"。)

在 Head 标签之内添加 Script 标签

现在，你已经在你的页面中应用了 MooTools 了，你还需要一个地方来写你的代码。这里有两种选择：

1. 把下面的代码写在你的 head 标签之内，你的代码写在 script 标记之内：

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

1. `<script type="text/javascript">`
2. `//Mootools code goes here`
3. `</script>`

2. 在外部建立一个 JavaScript 文件，然后在页面头部链接此文件：

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

1. `<script src="myJavaScriptFile.js" type="text/javascript"></script>`

在这里，你可以使用任何一种方式。我通常把 domready 事件中调用的方法放在 script 标记之间，而我的函数放在外部文件中。

把代码放在 domready 中

MooTools 的方法必须在 domready 事件中调用。

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

1. `window.addEvent('domready', function() {`
2. `exampleFunction();`
3. `});`

(Fdream 注：不完全是这样，但是可以保证你的 JavaScript 代码尽可能少地出错。顺便说一下 domready 事件：当页面的 HTML 代码（不包括图片、flash 等等，只是代码）下载完成时，此时会触发 domready 事件。这样可以在页面完全下载完成（包括图片、flash 等都下载完成）之前执行你的脚本，从而避免因一张大图要下很长时间而导致脚本不能执行，从而出现异常。)

把代码放在一个函数中

你仍然可以在 domready 之外创建你的函数，然后在 domready 中调用它：

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var exampleFunction = function() {  
2.     alert('hello')  
3. };  
4.  
5. window.addEvent('domready', function() {  
6.     exampleFunction();  
7. });
```

关于库的详细介绍

在这第一讲中,我们会仔细地看了一下这个库架构的一些关键组件,然后粗略地看一下其他基本功能。

Core (核心)

核心 (core) 部分包含 MooTools 库的一些公共函数 (Function) 来完成一些常见的任务,也加强了许多原有功能 (后面会有详细介绍)。下面的内容只是作为 MooTools 功能的一些例子,并不能替代您阅读 [MooTools 的文档](#)。

- 检查一个值 (如果没有值或者为 0 则返回 false) - `$chk(value)`;
- 返回两个值之间的一个随机整数 - `$random(min, max)`;
- 可以更容易地检测[浏览器、浏览器的引擎及浏览器的能力](#)

(Fdream 注: 第一个描述有误, `$chk(value)` 只是检查一个值是不是已经定义或者已经赋值, 为 0 时会返回 true, 只有 undefined 或者 null 时返回 false。)

Native (本地对象)

在库的这一部分也包含了一些公共工具,可以让你很容易地操作数组 (Array, 值或者对象的简单列表)、函数 (Function)、数值 (Number)、字符串 (String)、哈希对象 (Hash) 和事件 (Event)。这里是本地对象中的一些工具特性:

- 对数组中的每个元素执行一段脚本 - `.each()`;
- 得到数组中的最后一个元素 - `.getLast()`;
- 每个 x 毫秒触发一个事件 - `.periodical()`;
- 对小数取整 - `.round()`;
- 把 rgb 转换为十六进制 (HEX) - `.rgbToHex()`;

Class (类)

一个 JavaScript 类 (相对于 CSS 的类), 是一个功能可以重复使用的对象。若要更多地了解 MooTools 类, 你可以看看 Valerio 的这篇简单介绍的文章

([MooTools 类——怎样使用它们](#))。我也同时推荐 David Walsh 的 [MooTools 类模板](#)。

Element (元素)

MooTools 库的 Element 类提供了一些非常有用的功能。通过这个类，你可以选择 DOM 元素、操控他们的属性和位置、改变他们的 CSS 风格。这里是 MooTools 提供的一些非常强大的处理 DOM 元素的工具：

- 选择所有有相同 ID 或者 CSS 类名的 DOM 元素 - `.getElements()`;
- 给一个元素添加一个 CSS 类 - `.addClass()`;
- 取得一个元素的属性值 - `.getProperty()`;
- 改变一个元素的属性值 - `.setProperty()`;
- 取得一个元素的样式属性值 - `.getStyle()`;
- 改变一个元素的样式属性值 - `.setStyle()`;
- 取得一个元素的坐标位置 - `.getCoordinates()`;

(Fdream 注：不推荐在一个页面中有多个相同 ID，最好不要出现，在一些浏览器下很容易出现不可预见的错误。)

Utilities (实用工具)

实用工具 (Utilities) 提供了更多精良的选择逻辑，包括 domready 事件、可以管理 AJAX 调用的工具、可以轻松管理 cookie 的工具，甚至还有”swiff“功能，可以提供 JavaScript 接口给 ActionScript。

FX (效果)

这可能是 MooTools 最有趣的部分了。通过 Fx (效果)，你可以创建”Tween“ (补间动画) 和”morph“ (形变动画) 效果，从而让你的 DOM 对象动起来。

- 在两个样式属性值之间创建一个动画变形 (比如让一个 div 平缓地变大) - `var myFx = new Fx.Tween(element);`
- 在多个不同的属性值之间创建一个动画变形 (比如在让一个 div 平缓变大的过程中，让它的边框越来越粗，同时变换它的背景颜色) - `var myFx = new Fx.Morph(element);`

Request (请求)

包含一些可以轻松处理 JavaScript XMLHttpRequest (AJAX) 功能的工具。为了减轻整个请求/响应 (request/response) 带来的痛苦，Request 对象还有一些专门用来处理 HTML 和 [JSON 对象 \(JavaScript 对象表示法\)](#) 的扩展。

Plugins (插件)

MooTools 插件扩展了核心功能，可以轻松地为你的 web 项目添加高级 UI 功能。插件列表如下：

- Fx.Slide
- Fx.Scroll
- Fx.Elements
- Drag
- Drag.Move
- Color
- Group
- Hash.Cookie
- Sortables
- Tips
- SmoothScroll
- Slider
- Scroller
- Assets
- Accordion

全局概览

在开始下一讲之前，花一点时间全面地看一下 MooTools 的文档。可能有些地方你看不太懂，不要管它，尽管通读它，然后吸收那些你懂的。在接下来的 29 天中，我们将逐步深入这个库特定的部分，然后把 MooTools 分解成一些容易消化的小部分，但是首先，一定要好好看一下整个目录。

更多学习

[一个压缩包，包含了你开始起步所需要的所有东西](#)

包括一个 MooTools 1.2 核心库、一个简单的 HTML 文件、一个用来写你的函数的外部的 JavaScript 文件、一个 CSS 样式表文件。这个 HTML 文件已经写了详细的注释，并包含有 domready 事件。

其他的 MooTools 教程

同时，这里列出了一些其他[帮助你开始的的 MooTools 教程。](#)

MooTools 1.2 备忘录

这里是一份很好的 MooTools 1.2 功能的归纳表，我才给自己打印了一份，正在找地方把它给挂起来。也许我应该顺便拜访一下打印机，然后让它们给我提供一张海报大小的：）。不管怎样，这里是 [MooTools 1.2 备忘录的链接](#)。

Mootools 论坛

如果你想和其他人讨论 MooTools，检查代码示例或者深入讨论某个问题，你可以来这里。这才刚刚开张，但是正在逐渐热起来：[MooTools 1.2 论坛](#)。

Mootools 1.2 教程(2)——DOM 选择器

原文地址：[30 Days of Mootools 1.2 Tutorials - Day 2 - Selectors](#)

请尊重个人劳动，转载请注明来源：<http://ooboy.net>，译者：Fdream

这里是[《MooTools 1.2 系列教程目录》](#)。

如果你还没有准备好，请先阅读上一篇[《Mootools 1.2 教程\(1\)——MooTools 介绍》](#)。我们讲了怎么引用 MooTools 1.2 以及怎么在 domready 里面调用你的脚本。

今天开始本系列教程的第 2 讲。在这一讲中，我们会学习几种选择 HTML 元素的方法。在许多方面，这是 MooTools 用得最多最基本的。毕竟，要创建一个基于 HTML 元素的交互性用户体验，你必须首先把它们掌握在手中。

基本的方法

`$()`;

`$`函数是 MooTools 中基本的选择器。你可以通过它来根据一个 ID 选择 DOM 元素。

参考代码：[\[复制代码\]](#) [\[保存代码\]](#)

1. `// 选择 ID 为 "body_wrap" 的元素`
2. `$('body_wrap');`

参考代码：[\[复制代码\]](#) [\[保存代码\]](#)

1. `<div id="body_wrap">`
2. `</div>`

`.getElement()`;

`.getElement()`；扩展了 `$` 方法，可以让你简化你的选择操作。例如，你可以通过 `$` 方法来选择 ID 为 "body_wrap" 的元素，然后选择第一个子节点。`.getElement()`；只选择一个元素，如果有多个符合要求的元素则返回第一个

元素。如果你给 `.getElement()` 方法一个 CSS 类名 作为参数，你就会得到第一个有这个 CSS 类名的元素，而不是函数所有元素的数组。要选择多个元素，则可以使用下面的 `.getElements()` 方法。

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

1. // 选择 ID 为 "body_wrap" 的元素下面的第一个链接
2. `$('#body_wrap').getElement('a');`
- 3.
4. // 选择 ID 为 "body_wrap" 的元素下面的 ID 为 "special_anchor" 的元素
5. `$('#body_wrap').getElement('#special_anchor');`
- 6.
7. // 选择 ID 为 "body_wrap" 的元素下面第一个 CSS 类名为 "special_anchor_class" 的元素
8. `$('#body_wrap').getElement('.special_anchor_class');`

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

1. `<div id="body_wrap">`
2. `anchor`
3. `another anchor`
4. `special anchor`
5. `special anchor`
6. `another special anchor`
7. `</div>`

`$$()`;

`$$` 函数可以让你快速选择多个元素，并组成一个数组（一种你可以操作、获取和以任何方式重新排序的列表）。你可以通过标签名（如 `div`、`a`、`img` 等）、或者 ID 或者是他们的各种组合来选择多个元素。就像一个读者指出的那样，你可以[用\\$\\$做很多事情](#)，远远超出我们这里所介绍的。

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

1. // 选择这个页面中的所有 `div`
2. `$$('div');`
- 3.
4. // 选择 ID 为 "id_name" 的元素和所有的 `div`
5. `$('#id_name', 'div');`

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

1. `<div>`
2. `<div>a div</div>`
3. `a span`
4. `</div>`

`.getElements()`;

`.getElements()`;和`.getElement()`;非常类似, 不过它返回所有符合要求的元素, 并组成一个数组。你可以想使用`.getElement()`;方法那样使用`.getElements()`;。

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

1. `// 选择 ID 为 "body_wrap" 的元素下面的所有链接`
2. `$('#body_wrap').getElements('a');`
- 3.
4. `// 选择 ID 为 "body_wrap" 的元素下面的所有 CSS 类名为 "special_anchor_class" 的子元素`
5. `$('#body_wrap').getElements('.special_anchor_class');`

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

1. `<div id="body_wrap">`
2. `anchor`
3. `another anchor`
4. `special anchor`
 ``
5. `another special anchor`
6. `</div>`

用运算符包含和排除结果

运算符

MooTools 1.2 支持几种运算符, 可以让你进一步精简你的选择操作。你可以在`.getElements()`;中使用这些运算符来包含或者排除特定的结果。MooTools 支持 4 种运算符, 每一种都可以用来通过名字 (name) 选择一个 input 元素。

- `=` : 等于

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

1. `//选择 name 为 "phone_number" 的 input 元素`
2. `$('#body_wrap').getElements('input[name=phone_number]');`

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

1. // 选择 name 以” phone “开头的 input 元素
2. \$('body_wrap').getElements('input[name^=phone]');

- \$= : 以……结束

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

1. // 选择 name 以数字（number）结束的 input 元素
2. \$('body_wrap').getElements('input[name\$=number]');

- != : 不等于

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

1. // 选择名字不等于” address “的 input 元素
2. \$('body_wrap').getElements('input[name!=address]');

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

1. <div id="body_wrap">
2. <input name="address" type="text" />
3. <input name="phone_number" type="text" /> <!-- 上面的所有示例代码都将选中这个元素 -->
4. </div>

（Fdream注：input 在这里只是作为一个例子，你同样可以使用这种方式选择其他元素，比如 div、a 等等。）

要使用运算符，你必须首先指定元素的类型（比如这里的 input），然后指定你要过滤的属性（比如这里的 name），再加上你的 运算符，最后选择你的过滤字符串。

基于元素顺序的选择器

even（偶数选择）

通过这个简单的选择器，你可以选择序号为偶数的元素。**注意：**这个选择器从 0 开始计数，因此第一个元素是偶数序号的。

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

1. // 选择序号为偶数的 div
2. \$\$('div:even');

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

1. `<div>Even</div><!-- 上面的代码将选中这个元素 -->`
2. `<div>Odd</div>`
3. `<div>Even</div><!-- 上面的代码将选中这个元素 -->`
4. `<div>Odd</div>`

odd (奇数选择)

和 even 一样, 只不过它选择序号为奇数的元素。

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

1. `// 选择所有序号为奇数的 div`
2. `$$('div:odd');`

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

1. `<div>Even</div>`
2. `<div>Odd</div><!-- 上面的代码将选中这个元素 -->`
3. `<div>Even</div>`
4. `<div>Odd</div><!-- 上面的代码将选中这个元素 -->`

.getParent();

通过 .getParent(); 方法, 你可以得到一个元素的父元素 (parent)。

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

1. `// 选择 ID 为 "child_id" 的元素的父元素`
2. `$('child_id').getParent();`

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

1. `<div id="parent_id"> <!-- 上面的脚本将返回这个元素-->`
2. `<div id="child_id">Even</div>`
3. `</div>`

代码举例

任何 MooTools UI 开发都是从选择器开始的。这里是一些非常简单的例子, 演示了怎么去使用选择器操作 DOM 元素。

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. // 设置所有 span 的背景颜色为#eee
2. $$('span').setStyle('background-color', '#eee');
3. // 设置所有序号为奇数的 span 的背景色为#eee
4. $$('span:odd').setStyle('background-color', '#eee');
5.
6. // 设置 ID 为 body_wrap 的元素下的所有 CSS 类名为.middle_spans 的
   span 的背景色为#eee
7. $('body_wrap').getElements('.middle_spans').setStyle('background-color', '#eee');
```

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. <div id="body_wrap">
2.     <span>Even</span>
3.     <span class="middle_spans">Odd</span>
4.     <span class="middle_spans">Even</span>
5.     <span>Odd</span>
6. </div>
```

[下载 zip 包并尝试一下](#)

这个 zip 包中包含了一个简单的 html 文件、MooTools 1.2 核心库、一个外部 js 文件和上面你所看到的例子。

更多学习……

这并不意味着这是 MooTools 1.2 的选择器的全部功能列表，这仅仅只是帮助你入门，告诉你 MooTools 给你提供了什么功能。要学习有关选择器的更多东西，请参考下面的文档：

- 这里有非常多的有关[元素 \(Element\)](#) 选择器的文档
- 顺便也可以看一下[选择器 \(Selectors\)](#)

MooTools Blog 上有关\$\$选择器的文章

这是 mootools.net 上非常好的一篇有关[\\$\\$选择器和介绍它的变化多端](#)的 blog 文章。通过这个选择器你可以做多到你无法相信的事情，这篇文章很值得一读。

Slickspeed 选择器

这里有别人针对 MooTools 做的一个实验，测量不同的库在选择元素时到底有多快。这对于它本身来说很 cool，不过[这些选择器](#)的例子非常有价值。这里所有的选择器特性都可以通过 \$\$ 方法实现。

W3C 选择器

MooTools 也可以让你利用伪选择器的力量（就像上面的 Slickspeed 所证明的）。这里是 W3C 的[一篇关于选择器的文章](#)，一定值得读一遍（如果只有选择器的列表对你有帮助的话）。我不确定 MooTools 的 \$\$ 选择器是不是支持这个页面上的每一个单独选择器，但是至少是绝大部分。欢迎大家告诉我有关这方面的更多信息。

数组使用简介

原文地址：[30 Days of Mootools 1.2 Tutorials - Day 3 - Intro to Using Arrays](#)

请尊重个人劳动，转载请注明出处：<http://ooboy.net>，译者：Fdream

如果你还没有准备好，请阅读以前的教程，这里是[《MooTools 1.2 系列教程目录》](#)。

在上一篇教程——[《Mootools 1.2 教程\(2\)——DOM 选择器》](#)中，我们介绍了一下选择器，其中有很多方法就会返回数组（一个你可以对其中内容进行多种操作的特殊列表）。今天，我们在来看看如何使用数组来管理 DOM 元素。

基本方法

`.each()`;

在处理数组时，`.each()` 方法是你最好的朋友。它提供了一种很容易的方法来遍历数组的每个元素，如果有需要还可以对其中的元素进行任何逻辑处理。例如，我们可以假设你需要为页面中的每个 div 对象调用 `alert` 方法：

参考代码：[\[复制代码\]](#) [\[保存代码\]](#)

```
1. $$('div').each(function() {  
2.     alert('a div');  
3. });
```

如果使用下面的 HTML 代码，上面的 JavaScript 代码将弹出两个 `alert` 对话框，每个 div 一个。

参考代码：[\[复制代码\]](#) [\[保存代码\]](#)

```
1. <div>One</div>  
2. <div>Two</div>
```

`.each()` 方法不需要你使用 `$$` 方法。创建一个数组的另一种方式（就像我们昨天讲到过的）是使用 `.getElements()` 方法。

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. $('body_wrap').getElements('div').each(function() {  
2.     alert('a div');  
3. });
```

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. <div id="body_wrap">  
2.     <div>One</div>  
3.     <div>Two</div>  
4. </div>
```

还有另外一种方法来完成这个相同的任务，就是把这个数组赋值给一个变量，然后对那个变量使用 `.each()` 方法:

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. // 首先你需要通过语句" var VARIABLE_NAME "来声明一个变量  
2. // 然后用等于运算符" = "来给这个变量赋值  
3. // 在这个例子中，是一个包含 div 元素的数组  
4. var myArray = $('body_wrap').getElements('div');  
5.  
6. // 现在你就可以把这个变量当数组选择器使用了  
7. myArray.each(function() {  
8.     alert('a div');  
9. });
```

最后，如果你可能想把你的函数从选择器中独立出来。我们会在明天的关于使用函数的教程中更深入地讲解这个问题。不过，现在我们可以创建一个非常简单的示例:

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var myArray = $('body_wrap').getElements('div');  
2.  
3. // 要创建一个函数，你可以像刚才一样声明一个变量，然后给它命名  
4. // 在等号后面使用" function() "来声明这个变量为一个函数  
5. // 最后，在 { 和 } 之间写入你的函数代码  
6. var myFunction = function() {  
7.     alert('a div');  
8. };  
9.  
10. // 现在你就可以在 .each(); 方法里面调用刚才的函数了  
11. myArray.each(myFunction);
```

注意：当你像刚才那样在`.each()`;. 方法里面调用函数时，你不需要给函数名加上引号。

复制一个数组

`$A`

MooTools 提供了一个简单的方式——通过`$A` 函数来复制一个数组。让我们像刚才那样使用变量创建一个数组：

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

1. `// 创建你的数组变量`
2. `var myArray = $('body_wrap').getElements('div');`

复制一个数组（创建该数组的副本）：

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

1. `// 建立一个新的变量名，命名为” myCopy “，然后把” myArray “的副本赋值给它`
2. `var myCopy = $A(myArray);`

从数组中获取指定的元素

`.getLast()`;

`.getLast()` ;方法返回数组中最后一个元素。首先我们建立一个数组：

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

1. `var myArray = $('body_wrap').getElements('div');`

现在我们可以从这个数组中获取最后一个元素：

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

1. `var lastElement = myArray.getLast();`

变量 `lastElement` 现在的值就是数组 `myArray` 中的最后一个元素了。

`.getRandom()`;

和`.getLast()` ;一样，不过它随机从数组中取得一个元素：

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var randomElement = myArray.getRandom();
```

变量 randomElement 现在的值就是从数组 myArray 中随机选取的一个元素了。

向数组中添加一个元素

`.include();`

通过这个方法, 你可以给数组添加另外一个元素。只要把元素选择器传给 `.include();` 方法, 它就会包含到你的数组中。我们使用下面的 HTML 代码:

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. <div id="body_wrap">
2.     <div>one</div>
3.     <div>two</div>
4.     <span id="add_to_array">add to array</span>
5. </div>
```

我们可以像以前那样调用 "body_wrap" 下面的所有 div 一样来创建一个数组:

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var myArray = $('body_wrap').getElements('div');
```

要把另外一个元素添加到这个数组中, 首先你需要把这个元素赋值给一个变量, 然后使用 include 方法:

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. // 首先把你的元素赋值给一个变量
2. var newToArray = $('add_to_array');
3.
4. // 然后把它添加到数组
5. myArray.include(newToArray);
```

现在, 这个数组就同时包含 div 和 span 元素了。

`.combine();`

和 `.include();` 方法一样, 不过它可以让你把一个数组添加到一个已经存在的数组中, 而不用担心有重复的内容。假设我们现在从下面的 HTML 中取得了两个数组:

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. <div id="body_wrap">
2.     <div>one</div>
3.     <div>two</div>
4.     <span class="class_name">add to array</span>
5.     <span class="class_name">add to array, also</span>
6.     <span class="class_name">add to array, too</span>
7. </div>
```

我们可以这样建立两个数组:

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. // 就像我们以前那样建立你的数组
2. var myArray= $('body_wrap').getElements('div');
3.
4. // 然后建立一个所有 CSS 类名为.class_name 的元素数组
5. var newArrayToArray = $('.class_name');
```

现在我们可以使用.combine();方法来合并两个数组,这个方法会自己处理重复的元素,因此我们不需要处理:

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. // 把数组 newArrayToArray 合并到数组 myArray 中
2. myArray.combine(newArrayToArray );
```

现在 myArray 就包含了 newArrayToArray 中的所有元素。

代码示例

数组可以让你遍历包含所有项目的列表,并对每个元素执行相同的代码。在这个例子中,注意变量”item“作为当前元素的替代符的使用。

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. // 创建一个数组,这个数组包含”body_wrap“里面所有 CSS 类名
   为.class_name 的元素
2. var myArray = $('body_wrap').getElements('.class_name');
3.
4. // 首先建立一个要添加到数组中的元素
5. var addSpan = $('addtoarray');
6. // 然后建立一个要合并的数组
```



```

7. var addMany = $$('.addMany');
8.
9. // 现在我们把元素 addSpan 加入到数组中
10. myArray.include(addSpan);
11. // 然后合并数组 addMany 到 myArray 中
12. myArray.combine(addMany);
13.
14. // 建立一个需要对数组中的每个元素都要执行的函数
15. var myArrayFunction = function(item) {
16. // item 现在指向数组中的当前元素
17. item.setStyle('background-color', '#eee');
18. }
19.
20. // 现在对数组中的每个项目调用 myArrayFunction 函数
21. myArray.each(myArrayFunction);

```

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```

1. <div id="body_wrap">
2.   <div class="class_name">one</div><!-- this has gray backgro
   und -->
3.   <div>two</div>
4.   <div class="class_name">three</div><!-- this has gray backg
   round -->
5.   <span id="addtoarray">add to array</span> <!-- this has gr
   ay background -->
6.   <br /><span class="addMany">one of many</span> <!-- this h
   as gray background -->
7.   <br /><span class="addMany">two of many</span> <!-- this h
   as gray background -->
8. </div>

```

延伸学习

这个教程并没有打算涵盖你能对数组做的全部事情, 但是希望能够给你一个参考, 告诉你 MooTools 提供了一些什么功能。要学习更多关于数组的东西, 请仔细阅读这些内容:

- [文档中的数组部分](#)
- 这个页面中有许多[关于 JavaScript 数组的信息](#)

[下载一个包含你开始所需要的所有东西的 zip 包](#)

包括一个简单的 html 文件、MooTools 1.2 核心库、一个外部 JavaScript 文件、一个 css 文件和上面的所有例子。

函数和 MooTools 1.2

请尊重个人劳动，转载请注明出处：<http://ooboy.net>，译者：Fdream

如果你还没有准备好，请阅读以前的教程，这里是 [《MooTools 1.2 系列教程目录》](#)。

今天开始 MooTools 系列教程的第 4 讲。如果你还没有看过上一讲，请先查看上一篇教程[《MooTools 1.2 教程 \(3\)——数组使用简介》](#)。今天我们先不讲 MooTools，而是讲一讲 JavaScript 中的函数（function）的基本知识。

但是，为了符合 MooTools 这个主题，你需要知道在哪里该使用 MooTools 的函数。此前，我们已经在我们的所有示例代码中，把代码都放在 domready 方法中。当我们需要把它放在 domready 的外面时，我们使用了函数（function）。如果你在 domready 里面调用函数之前，函数并不会被执行。

一般来说，一种比较好的方式是尽可能地把你的函数代码都放在页面之外的某一个地方，然后通过 JavaScript 应用来调用它们。当你只是写代码玩玩，可能把所有的东西写在一个页面上更容易一些。在这个教程中，我们使用下面的约定：

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. <script type="text/javascript">
2. /*
3.  * 函数定义写在这里
4. */
5.
6. window.addEvent('domready', function() {
7.     /*
8.     * 函数调用写在这里
9.     */
10. });
11.</script>
```

所有的例子都遵循这个格式，当页面载入的时候（load）执行函数代码。在每个函数的下面，都有一个相应的按钮，你可以点击它们，然后看到结果。这是通过使用 MooTools 的事件处理来完成的，明天我们将会讲到这个。

函数基础

在 JavaScript 中，有几种方式来定义函数，由于我们的主题是讲解 MooTools，因此我们将选择 MooTools 的首选方式。下面的示例是一个函数定义的开始。我们什么了一个变量，并命名为 `simple_function`，并把这个变量定义为一个函数：

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var simple_function = function() {
```

然后我们给这个函数增加了一个 `alert` 语句，当函数被调用的时候就会执行：

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. alert('This is a simple function');
```

最后，我们以一个花括号结束这个函数的定义：

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. }
```

这个关闭花括号看起来是一件非常简单的事情，但是很多时候要追踪这个问题却是一件很痛苦的事情。因此，适度地强迫对函数定义的关闭符号进行检查是个不错的主意。

在下面的例子中，我们把它们组合起来了。在声明这个函数之后，我们在页面加载后的 `domready` 事件里面添加了对这个函数的调用。可以点击例子下面的按钮查看调用函数 `simple_function()` 后的结果。

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. // 定义 simple_function 为一个函数
2. var simple_function = function() {
3.     alert('This is a simple function');
4. }
5.
6. window.addEvent('domready', function() {
7.     // 当页面加载后调用 simple_function
8.     simple_function();
9. });
```

单个参数

虽然你有很多代码可以轻松地随时调用，这已经很有用了，但是如果你可以给它传递参数（信息）进行处理，这将会更有用。要在函数中使用参数，你需要在 `function` 后面的括号中添加一个变量，就像这样：

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var name_of_function = function(name_of_the_parameter) {  
2.     /* 函数代码写在这里 */  
3. }
```

一旦你这样做了, 在这个函数内部就可以使用这个变量了。尽管你可以给参数取任何你想要的名字, 但是让参数名更有意义是个不错的选择。举个例子来说, 如果你要 传递一个小镇的名字, 可能你把参数命名为 `town_name` 比其他更模糊的名字要好一些 (比如 `user_input`)。

在下面的例子中, 我们定义了一个只带有一个参数的函数, 并在弹出对话框中显示这个变量。请注意, 信息的第一部分被单引号包含起来了, 而参数没有。当你要把参数和硬编码的字符串连接在一起, 你需要用加号 (+) 运算符把他们连接起来, 就像下面一样:

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var single_parameter_function = function(parameter) {  
2.     alert('the parameter is : ' + parameter);  
3. }  
4.  
5. window.addEventListener('domready', function() {  
6.     single_parameter_function('this is a parameter');  
7. });
```

多个参数

JavaScript 没有限制在一个函数中可以定义的参数的个数。一般来说, 要让传给函数的参数个数尽可能地 少, 这会使代码更具可读性。函数中定义多个参数使用逗号分割, 其它行为这和单个参数函数一样。下面的示例中的函数带有两个数字, 并把它们的和赋值给第三个数字, 就像这样:

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var third_number = first_number + second_number;
```

这里加号 (+) 运算符的使用和把这些结果连接成字符串略有一些不同:

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. alert(first_number + " plus " + second_number + " equals " + third_number);
```

虽然这个初一看起来可能有些混乱，但是实际上却非常简单。如果你在两个数字之间使用加号 (+)，你就是把它们加在一起。如果你在任意组合的数字和字符串之间使用加号 (+)，那么就是把所有的东西作为字符串连接起来。

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var two_parameter_function = function(first_number, second_number) {  
2.     // 取得 first_number 和 second_number 相加的和  
3.     var third_number = first_number + second_number;  
4.  
5.     // 显示结果  
6.     alert(first_number + " plus " + second_number + " equals "  
7.         + third_number);  
8. }  
9. window.addEventListener('domready', function() {  
10. two_parameter_function(10, 5);  
11. });
```

返回值

在一个弹出对话框中显示一个函数的执行结果可能很有用，但是有些时候你可能需要在其他地方用到这个结果。要完成这个任务，你需要使用函数中的 return 功能。下面的示例代码中，函数和上面的示例一样，不过这里不是弹出一个对话框，而是返回两个数字相加后的结果：

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. return third_number;
```

你会发现，我们也在 domready 中做了更多的事情。为了显示这个结果，我们把这个函数的返回值赋值给了一个名称为 return_value 的参数，然后把它显示在弹出对话框中。

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var two_parameter_returning_function = function(first_number, second_number) {  
2.     var third_number = first_number + second_number;  
3.     return third_number;  
4. }  
5. window.addEventListener('domready', function() {  
6.     var return_value = two_parameter_returning_function(10, 5);
```

```
7.     alert("return value is : " + return_value);
8. });
```

把函数作为参数

如果你看看 MooTools 的 domready 里面我们包装的东西, 你会注意到我们把一个函数作为参数传递进去了:

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. window.addEvent('domready', function() {
2.     /* 函数代码 */
3. });
```

一个像这样把函数作为一个参数传递进去的函数称为匿名函数:

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. function() {
2.     /* 函数代码 */
3. }
```

在第一篇教程的评论中, Boomstix 指出了在 domready 中不使用匿名函数的一种替代方式。这种方式就是这样的:

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. // 建立一个要在 domready 时调用的函数
2. var domready_function() {
3.     /* 函数代码 */
4. }
5.
6. // 把函数指定到 domready 事件
7. window.addEvent('domready', domready_function);
```

我不知道这两种方式在性能和功能性上的任何明显差别, 因此我认为这基本上只是一个风格习惯而已。我们会继续坚持我们的方式, 如果有任何人知道这些差别请告诉我们。

代码示例

为了刺激你明天的食欲 (和弥补今天对 MooTools 的缺少), 我写了一个没有什么意义的函数, 可以让你随意改变这个页面的背景:

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var changeColor = function() {
2.     // 用来从输入框中获得颜色值
3.     // (请参考:
4.     // http://docs.mootools.net/Element/Element#Element:get)
5.     var red   = $('red').get('value');
6.     var green = $('green').get('value');
7.     var blue  = $('blue').get('value');
8.
9.     // 确保每一个东西都是整数
10.    // (请参考:
11.    // http://docs.mootools.net/Native/Number#Number:toInt)
12.    red   = red.toInt();
13.    green = green.toInt();
14.    blue  = blue.toInt();
15.
16.    // 确保每一个数字都在 1 到 255 之间
17.    // 如果有需要则取整
18.    // (请参考:
19.    // http://docs.mootools.net/Native/Number#Number:limit)
20.    red   = red.limit(1, 255);
21.    green = green.limit(1, 255);
22.    blue  = blue.limit(1, 255);
23.
24.    // 取得十六进制代码
25.    // (请参考:
26.    // http://docs.mootools.net/Native/Array/#Array:rgbToHex)
27.    var color = [red, green, blue].rgbToHex();
28.
29.    // 设置为该页面的背景色
30.    // (请参考:
31.    // http://docs.mootools.net/Element/Element.Style#Element:setStyle)
32.    $('body_wrap').setStyle('background', color);
33.
34. }
35.
36. var resetColor = function() {
37.     // 重新设置页面的背景色为白色
38.     // (请参考:
39.     // http://docs.mootools.net/Element/Element.Style#Element:setStyle)
40.     $('body_wrap').setStyle('background', '#fff');
```

```
41. }
42.
43. window.addEvent(' domready', function() {
44.     // 为按钮添加点击事件（明天我们会讲这个）
45.     // （请参考：
46.     // http://docs.mootools.net/Element/Element.Event#Element:ad
        ddEvent)
47.     $('change').addEvent('click', changeColor);
48.     $('reset').addEvent('click', resetColor);
49. });
```

红 (R)

绿 (G)

蓝 (B)

延伸学习...

[下载包含你学习所需要的所有东西的 zip 包](#)

包含 MooTools 1.2 核心库、一个外部 JavaScript 文件、一个简单的 html 页面和一个 css 文件。

更多关于 JavaScript 函数的内容

[JavaScript 函数上的 Quirksmode（怪异模式）](#)

我没有很好的关于 JavaScript 函数的资源，如果有人知道的话请告诉我。

有关示例的文档

- [Utilities/DomReady](#)
- [Number.toInt\(\)](#)
- [Number.limit\(\)](#)
- [Array.rgbToHex\(\)](#)
- [Element.setStyle\(\)](#)
- [Element.addEvent\(\)](#)

原文地址: [30 Days of MooTools 1.2 Tutorials - Day 5 - Event Handling](#)

MooTools 1.2 中的事件处理

请尊重个人劳动, 转载请注明出处: <http://ooboy.net>, 译者: Fdream

如果你还没有准备好, 请阅读以前的教程, 这里是 [《MooTools 1.2 系列教程目录》](#)。

今天我们开始第五讲, 在上一讲 ([《MooTools 1.2 教程\(4\)——函数》](#)) 中, 我们学习了在 MooTools 1.2 中建立和使用函数的几种不同方式。下一步就是理解事件了。和选择器类似, 事件也是建立互动界面的一个重要部分。一旦你掌握了一个元素, 你需要去决定 什么行为来触发什么效果。先把效果留着以后在讲, 我们首先看一看中间步骤和一些常见的事件。

左键单击事件

左键单击事件是 web 开发中最常见的事件。超链接识别点击事件, 然后把你带到另外一个 URL 地址。MooTools 能够识别其他 DOM 元素上的点击事件, 在设计和功能上给了你极大的灵活性。给一个元素添加一个点击事件的第一步:

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. // 通过$('id_name') 取得一个元素
2. // 用.addEvent 添加事件
3. // ('click')定义了事件的类型
4. $('id_name').addEvent('click', function(){
5.     // 在这里添加点击事件发生时你要执行的任何代码
6.     alert('this element now recognizes the click event');
7. });
```

你也可以把这个函数从.addEvent();独立出来来完成相同的事情:

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var clickFunction = function(){
2.     // 在这里添加事件发生时你要执行的任何代码
3.     alert('this element now recognizes the click event');
4. }
5.
6. window.addEvent('domready', function() {
7.     $('id_name').addEvent('click', clickFunction);
8. });
```

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. <body>
2.     <div id="id_name"> <!-- this element now recognizes the click event -->
3.     </div>
4. </body>
```

注意: 和超链接识别点击事件一样, MooTools 的点击事件实际上也是识别 “mouse up”, 意味着当你鼠标松开是发生, 而不是鼠标按下去的时候发生。这就给了用户一个机会去改变他们的主意——只要在松开之前把鼠标的指针从点击的元素上移开就可以了。

鼠标进入和离开事件

当鼠标停留在一个链接元素上时, 超级链接还识别 “hover” 事件。通过 MooTools, 你可以给其他的 DOM 元素也添加一个悬停事件。通过把这个事件分为鼠标进入和鼠标离开事件, 你可以更加灵活地根据用户的行为来操控 DOM。

和以前一样, 我们要做的第一件事就是把一个事件附加到一个元素:

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var mouseEnterFunction = function() {
2.     // 在这里添加事件发生时你要执行的任何代码
3.     alert('this element now recognizes the mouse enter event');
4. }
5.
6. window.addEventListener('domready', function() {
7.     $('id_name').addEventListener('mouseenter', mouseEnterFunction);
8. });
```

鼠标离开事件也是同样的, 这个事件在鼠标指针离开一个元素时发生。

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var mouseLeaveFunction = function() {
2.     // 在这里添加事件发生时你要执行的任何代码
3.     alert('this element now recognizes the mouse leave event');
4. }
5.
6. window.addEventListener('domready', function() {
7.     $('id_name').addEventListener('mouseleave', mouseLeaveFunction);
8. });
```

删除一个事件

总有一些时候，你一旦不再需要那些事件，于是你需要从一个元素上删除一个事件。删除一个事件和添加一个事件一样容易，甚至连结构都是类似的。

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

1. // 和前一个示例一样
2. // 只不过把.addEvent 换成了.removeEvent
3. \$('id_name').removeEvent('mouseleave', mouseLeaveFunction);

textarea 或者 input 中的按键事件

MooTools 也可以让你识别文本域（textarea）和文本框（input）中的按键事件。其语法和我们上面看到的类似：

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

1. var function = keydownEventFunction () {
2. alert('This textarea can now recognize keystroke events');
3. };
- 4.
5. window.addEvent('domready', function() {
6. \$('myTextarea').addEvent('keydown', keydownEventFunction);
7. });

上面的代码将会识别任何按键。要针对一个特定的按键，我们需要添加一个参数，然后使用一个 if 语句：

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

1. // 注意函数括号中的“event”参数
2. var keyStrokeEvent = function(event) {
3. // 下面的代码是说：
4. // 如果按下的键为“k”，则做下面的事
5. if (event.key == "k") {
6. alert("This tutorial has been brought you by the letter k.")
7. };
8. }
- 9.
10. window.addEvent('domready', function() {
11. \$('myInput').addEvent('keydown', keyStrokeEvent);

```
12. });
```

如果需要其他的控制，比如“shift”键和“control”见，语法略有一点不同：

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var keyStrokeEvent = function(event) {
2.     // 下面代码是说：
3.     // 如果按下的键是“shift”，则做下面的事
4.     if (event.shift) {
5.         alert("You pressed shift.")
6.     };
7. }
8.
9. window.addEventListener('domready', function() {
10.     $('myInput').addEventListener('keydown', keyStrokeEvent);
11. });
```

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. <div id="body_wrap">
2.     <input id="myInput" type="text" />
3. </div>
```

示例

这里是上面我们写过的一些可以执行的代码：

注意：你可以在单击示例上面试一下，不过不是在上面松开鼠标，而是把鼠标一直按着从区块上离开，然后再松开。注意一下它没有触发点击事件。

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var keyStrokeEvent = function(event) {
2.     // 下面的代码是说：
3.     // 如果按下的键为“k”，则做下面的事
4.     if (event.key == 'k') {
5.         alert("This Mootorial was brought to you by the letter
        'k.'")
6.     };
7. }
8.
9. var mouseLeaveFunction = function() {
10.     // 在这里添加事件发生时你要执行的任何代码
```

```

11.     alert('this element now recognizes the mouse leave event');
12. }
13.
14. var mouseEnterFunction = function() {
15.     // 在这里添加事件发生时你要执行的任何代码
16.     alert('this element now recognizes the mouse enter event');
17. }
18.
19. var clickFunction = function() {
20.     // 在这里添加事件发生时你要执行的任何代码
21.     alert('this element now recognizes the click event');
22. }
23.
24. window.addEventListener('DOMContentLoaded', function() {
25.     $('click').addEventListener('click', clickFunction);
26.     $('enter').addEventListener('mouseenter', mouseEnterFunction);
27.     $('leave').addEventListener('mouseleave', mouseLeaveFunction);
28.     $('keyevent').addEventListener('keydown', keyStrokeEvent);
29. });

```

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```

1. <div id="click" class="block">左键单击 (Click) </div><br />
2. <div id="enter" class="block">鼠标进入(Mouse Enter)</div><br />
3. <div id="leave" class="block">鼠标离开(Mouse Leave)</div><br />
4. <input id="keyevent" type="text" value="请输入字符'k'" />

```

左键单击 (Click)

鼠标进入 (Mouse Enter)

鼠标离开 (Mouse Leave)

请输入字符'

更多学习……

[下载一个包含你开始所需要的所用东西的 zip 包](#)

包含 MooTools 1.2 核心库、一个外部 JavaScript 文件、一个简单的 html 页面和一个 css 文件。

更多关于事件的资料

MooTools 给了你更多的关于事件的控制方法，比我们这里讲得要多得多。要学习更多内容，请查看下面几个链接：

- MooTools 文档中的 [Events 部分](#)
- MooTools 文档中的 [Element.Events 部分](#)
- 还有，阅读一下 w3school 网站上[关于 JavaScript 事件的内容](#)

通过 Mootools 1.2 来操纵 HTML DOM 元素

请尊重个人劳动，转载请注明出处：<http://ooboy.net>

如果你还没有准备好，请阅读以前的教程，这里是 [《MooTools 1.2 系列教程目录》](#)。

我们已经学习过如何来选取 DOM 元素，怎么创建数组，怎么创建函数，怎么把事件添加到元素，今天我们来深入地学习一下如果操纵 HTML 元素。通过 MooTools 1.2，你可以添加新元素到一个 HTML 页面中，也可以删除元素，以及改变任何样式或者元素参数，这些都非常容易。

基本方法

`.get()`;

这个工具可以让你获取元素的属性（property）。元素的属性是组成一个 HTML 元素的各种不同部分，例如 src、value、name 等等。使用 `.get()` 方法非常简单：

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

1. `// 下面这行将返回 id 为 “id_name” 的元素的 html 标记名 (div、a、span……)`
2. `$('#id_name').get('tag');`

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

1. `<div id="body_wrap">`
2. `Element <!-- 上面的代码将返回 “span” -->`

3. </div>

你可以使用 `.get()` 方法获得更多属性，而不只是 html 标记名：

- id
- name
- value
- href
- src
- class (如果有多个 CSS 类名，则将返回全部 CSS 类名)
- text (一个元素的文本内容)
- 等等...

`.set()`;

`.set()` 方法和 `.get()` 方法一样，不过不是获得一个值，而是设置一个值。当和事件联合使用时比较有用，通过这个方法你可以在页面加载之后改变一些属性值。

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

1. // 这将设置 id 为 id_name 的元素链接地址为
 “http://www.google.com”
2. `$('#id_name').set('href', 'http://www.google.com');`

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

1. `<div id="body_wrap">`
2. `<!-- 上面的代码将改变链接地址为`
 `“http://www.google.com” -->`
3. `Search Engine</`
 `a>`
4. `</div>`

`.erase()`;

通过 `.erase()` 方法，你可以清除一个元素的属性值。它和前面两个方法类似。选取元素，然后选择你要清除的属性。

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

1. // 这讲移除 id 为 id_name 的元素的 href 属性
2. `$('#id_name').erase('href');`

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. <div id="body_wrap">
2.     <!-- 上面的代码将清除链接地址 -->
3.     <a href="http://www.yahoo.com">Search Engine</a>
4. </div>
```

移动元素

`.inject()`;

要移动页面上一个已经存在的元素，你可以使用 `.inject()` 方法。和我们看到的其它方法类似，它用起来也非常简单，可以在你的用户界面上给你更多操控权。要使用 `.inject()` 方法，首先要设置一些包含元素变量：

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var elementA = $('elemA');
2. var elementB = $('elemB');
3. var elementC = $('elemC');
```

上面的代码把下面这个 HTML 分别赋值给了不同的变量，这样用 MooTools 来操作时会比较简单。

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. <div id="body_wrap">
2.     <div id="elemA">A</div>
3.     <div id="elemB">B</div>
4.     <div id="elemC">C</div>
5. </div>
```

现在，要改变这些元素的顺序，我们可以通过四种方式来使用 `.inject()` 方法。我们可以把元素注入到：

- 底部 (bottom, 默认)
- 顶部 (top)
- 在某个元素的前面 (before)
- 在某个元素的后面 (after)

bottom 和 top 将把这个元素注入到一个选中元素的内部，在元素内最底部或者最顶部。相对地，before 和 after 将把一个元素注入到另外一个元素的顶部或者底部，但是不是注入到元素内部。

因此，让我们把元素顺序改变为 A-C-B。由于我们不需要把一个元素注入到另外一个元素的内部，我们可以使用 before 或者 after。

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

1. // 下面这句话的意思是: 把元素 C 放到元素 B 之前
2. `elementC.inject(elementB, 'before');`
- 3.
4. // 下面这句话的意思是: 把元素 B 放到元素 C 之后
5. `elementB.inject(elementC, 'after');`

创建一个新元素

`new Element`

你可以使用“`new Element`”构造器来创建一个行的 HTML 元素。这和写一个正常的 HTML 元素非常类似,只不过你需要调整一下语法,以便能够在 MooTools 下正常运行:

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

1. // 首先命名一个变量并声明一个“`new Element`”
2. // 然后定义元素的类型 (`div`、`a`、`span...`)
3. `var newElementVar = new Element('div', {`
4. // 在这里设置元素的所有属性
5. `'id': 'id_name',`
6. `'text': 'I am a new div',`
7. `'styles': {`
8. // 在这里设置元素的所有样式参数
9. `'width': '200px',`
10. `'height': '200px',`
11. `'background-color': '#eee',`
12. `'float': 'left'`
13. `}`
14. `});`

现在你就有一个元素了,你可以通过我们刚才学的 `inject()` 方法把这个元素放在页面上的某个位置。我们从下面这个简单的 HTML 开始:

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

1. `<div id="body_wrap">`
2. `<div id="content_id">Some div content</div>`
3. `</div>`

现在,我们把 ID 为 `content_id` 的元素转换为一个变量:

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var bodyWrapVar = $('body_wrap');
```

和我们刚才学的一样, 我们可以把我们创建的这个元素注入到当前的 HTML 中:

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. // 这句话的意思是说: “把 newElementVar 注入到 bodyWrapVar 内部, 并  
   放置到顶部”  
2. newElementVar.inject(bodyWrapVar, 'top');
```

这个代码最终可能是这样的:

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. <div id="body_wrap">  
2.     <!-- 这个元素被注入到内部顶部 -->  
3.     <div id="id_name">I am a new div</div>  
4.     <div id="content_id">Some div content</div>  
5. </div>
```

示例

为了这个例子, 我们来创建一个表单, 可以让你添加一个行元素到你的 HTML 页面。首先, 建立一些文本框和按钮。

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. <div id="body_wrap">  
2.     ID: <input id="id_input" name="id" />  
3.     text: <input id="text_input" name="text" />  
4.     <button id="new_div">创建一个新的 div</button>  
5. </div>
```

现在, 我们来用 MooTools 写 JavaScript 来实现让这个 HTML 表单可以插入一个新的元素到你的页面中。首先, 我们先给这个按钮添加一个事件, 并写一个函数来包含我们的代码:

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var newDiv = function() {  
2.     // 我们将把 “添加一个新元素” 的代码放在这里  
3. };  
4.
```

```
5. window.addEvent('domready', function() {
6.     $('new_div').addEvent('click', newDiv);
7. });
```

下一件事我们要做的就是指定我们要处理的变量。要使用输入表单中的数据，我们需要使用.get()方法：

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var idValue = $('id_input').get('value');
2. var textValue = $('text_input').get('value');
```

现在，上面代码中的变量idValue和textValue就包含了它们指定的输入表单的值。由于我们需要在用户点击“创建一个新的div”按钮时获得输入框的值，我们需要把上面的代码放在newDiv();这个函数中。如果我们需要在这个函数外面获得这个值，我们需要在页面加载时获得，而不是点击时。

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var newDiv = function() {
2.     var idValue = $('id_input').get('value');
3.     var textValue = $('text_input').get('value');
4. };
5.
6. window.addEvent('domready', function() {
7.     $('new_div').addEvent('click', newDiv);
8. });
```

接下来，我们需要获得我们新元素要插入到的元素：

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var newDiv = function() {
2.     var idValue = $('id_input').get('value');
3.     var textValue = $('text_input').get('value');
4.     var bodyWrapVar = $('newElementContainer');
5. };
6.
7. window.addEvent('domready', function() {
8.     $('new_div').addEvent('click', newDiv);
9. });
```

我们已经有了我们的输入表单的值了，现在我们可以创建一个新元素了：

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```

1. var newDiv = function() {
2.     var idValue = $('id_input').get('value');
3.     var textValue = $('text_input').get('value');
4.     var bodyWrapVar = $('newElementContainer');
5.
6.     var newElementVar = new Element('div', {
7.         // 这将设置这个元素的 id 为 idValue 的值
8.         'id': idValue,
9.         // 这将设置这个元素的文本为 textValue 的值
10.        'html': textValue
11.    });
12.};
13.
14.window.addEventListener('domready', function() {
15.    $('new_div').addEventListener('click', newDiv);
16.});

```

剩下我们要做的就是把这个新元素插入到我们的页面中了：

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```

1. var newDiv = function() {
2.     var bodyWrapVar = $('newElementContainer');
3.     var idValue = $('id_input').get('value');
4.     var textValue = $('text_input').get('value');
5.
6.     var newElementVar = new Element('div', {
7.         'id': idValue,
8.         'text': textValue
9.    });
10.
11.    // 下面这句是说：“把 newElementVar 插入到 bodyWrapVar 的
    内部顶部”
12.    newElementVar.inject(bodyWrapVar, 'top');
13.};
14.
15.var removeDiv = function() {
16.    // 这将删除内部的 html 值（就是 div 标记类的所有东西）
17.    $('newElementContainer').erase('html');
18.}
19.
20.window.addEventListener('domready', function() {
21.    $('new_div').addEventListener('click', newDiv);
22.    $('remove_div').addEventListener('click', removeDiv);
23.});

```

代码效果演示

ID:

text:

创建一个新 div 删除所有新 div

你可以试试在 id 文本框中输入: ilovemilk

更多学习...

一定要花一些时间看一些 MooTools 文档中的 Elements 这一节:

- [Element](#) 这一节包含了我们这里讲到的大多数内容，还有很多其它内容
- [Element.style](#) 可以给你在元素样式属性上更多的控制权（有些东西我们将在以后的教程中深入讲解）
- [Element.dimentions](#) 包含了处理位置、坐标、尺寸大小等东西的工具

设置和获取样式表属性

请尊重个人劳动，转载请注明出处: <http://ooboy.net>, 译者: Fdream

如果你还没有准备好开始这一讲，请参考这一系列的教程，这里是[《MooTools 1.2 系列教程目录》](#)。

欢迎开始这一系列的教程的第七讲。今天，我们来看一下如何通过 MooTools 1.2 和我们以前几讲中的内容来操作样式，这将给你在 UI 上带来很大的控制权。处理样式非常简单，不过今天我们要做一些调整。例如，我们要介绍键值对（key-value pair）对象。我们也会讲到在 domready 之外来传递变量，就像我们在[关于函数的那一讲](#)中学到的一样。从这里开始，我们会开始慢慢提高难度，介绍一些必要的编程概念。如果你是 JavaScript 新手或者第一次开始学 MooTools，请确保你在明白了前面的教程，你可以随意地问我任何问题。

基本方法

```
.setStyle();
```

这个函数可以允许你设置一个元素的样式属性。我们在前面的一些例子中已经使用过了。你要做的就是把它放在你的 选择器之后，那么它将改变一个元素或者多个元素的样式属性。

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. // 定义你的选择器
2. // 添加.setStyle 方法
3. // 指定样式属性和值
4. $('body_wrap').setStyle('background-color', '#eeeeee');
5. $$('.class_name').setStyle('background-color', '#eeeeee');
```

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. <div id="body_wrap">
2.     <div class="class_name"></div>
3.     <div class="class_name"></div>
4.     <div class="class_name"></div>
5.     <div class="class_name"></div>
6. </div>
```

.getStyle();

同样,这个方法就像它的字面意思一样。`.getStyle();`将返回一个元素的一个属性值。

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. // 首先, 建立一个变量来保存这个样式属性值
2. var styleValue = $('body_wrap').getStyle('background-color');
```

如果我们在上面的例子中运行这个代码, 那么它将返回“#eeeeee”给变量 `styleValue`。

设置和获取多个样式表属性

.setStyles();

`.setStyles();`就像你所想象的那样, 可以让你一次给一个元素或者一个元素数组设置多个属性值。为了能够同时设置多个样式表属性值, `.setStyles();`的语法略有一点不同。

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. // 还是从你的选择器开始, 然后在后面加上.setStyles({
2. $('body_wrap').setStyles({
3.     // 下面的格式为: 'property': 'value',
4.     'width': '1000px',
5.     'height': '1000px',
```

```
6.      // 特别注意：最后一个属性没有逗号
7.      // 如果有逗号，将不能跨浏览器
8.      'background-color': '#eeeeee'
9.  });
```

注意：实际上，属性选择器也可以不需要单引号，除非属性名中有连接符“-”，比如在“background-color”中，为了保持简单，给每个属性选择器都加上单引号更容易一些。

同时也要注意：属性值可能更灵活多变一些（比如“100px”或者“#eeeeee”）。除了字符串（一个只有字母的串，我们会在以后的教程中更深入地讲解这个），你也可以传入数字（这可能在大多数情况下会被解释为 px）或者变量而不需要引号：

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1.  // 这个把变量 firstBackgroundColor 的值设置为字符串（STRING）
   ' #eeeeee'
2.  var firstBackgroundColor = ' #eeeeee';
3.
4.  // 你可以把一个变量传递给另外一个变量
5.  // 这使得变量 backgroundColor 的值也等于字符串（string）' #eeeeee'
6.  var backgroundColor = firstBackgroundColor;
7.
8.  // 这个把变量 divWidth 的值设置为数字（NUMBER）500
9.  var divWidth = 500;
10.
11. $('body_wrap').setStyles({
12.    // 在这种情况下，变量名是不需要用引号包围起来的
13.    'width': divWidth,
14.    // 数字也一样，不需要引号包围
15.    'height': 1000,
16.    // 另外一个变量
17.    'background-color': backgroundColor,
18.    // 字符串就是用单引号引起来的一系列字符组成的串
19.    'color': 'black'
20. });
```

.getStyles();

这个方法可以让你一次获得多个样式属性。这个和我们看到的上面的略有一些不同，因为它包含了多个数据集，每个数据集有一个键（key，属性名）和一个值（value，属性值）。这个数据集叫做对象，.getStyles();方法可以非常容易地把多个属性值放入这些对象中，并可以很简单地把它们取回来。

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. // 首先为你的对象定义一个变量
2. // 然后创建一个选择器
3. // 然后把.getStyles 添加到你的选择器
4. // 然后创建一个用逗号分隔开的样式属性列表
5. // 确保每个属性都在一个单引号中
6. var bodyStyles = $('body_wrap').getStyles('width', 'height', 'background-color');
7.
8. // 首先我们创建一个对象来保存这个属性值
9. // 然后通过指定的属性的名（键）来得到一个值
10. // 把属性名放在两个方括号[]之间
11. // 并确保属性名已经用单引号引起来了
12. var bgStyle = bodyStyles['background-color'];
```

如果在我们的 CSS 文件中有这样的样式定义：

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. #body_wrap {
2.     width: 1000px;
3.     height: 1000px;
4.     background-color: #eeeeee;
5. }
```

那么变量 bgStyle 将包含值 “#eeeeee”。

注意：如果你要从你的样式表对象中取得一个单独的属性，首先取得一个对象变量（在这个例子中是“bodyStyles”），然后使用方括号和单引号（['']），最后填入属性名 key（如 width 或者 background-color）。就这么简单！

代码示例

在这个例子中，我们将使用我们刚才在上面学到的一些方法来获取和设置样式。在注意样式属性操作用法的同时，也要特别注意它本身的结构。为了把我们的函数从 domready 中独立出来，我们需要把那些变量传递到 domready 事件的函数中。我们通过给 domready 里面的函数传递一个参数来实现这个。注意点击（click）事件使用了匿名方法——这可以让我们从 domready 事件中把变量传递到外面的函数中。如果你第一遍没有看懂，请不要着急，下面的例子可能会让这些更清楚更明白一些：

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)


```
1. // 这里是一些函数
2.
3. // 注意这个函数有一个参数: "bgColor"
4. // 这个是从 domready 事件中传递过来的
5. var seeBgColor = function(bgColor) {
6.     alert(bgColor);
7. }
8.
9. var seeBorderColor = function(borderColor) {
10.    alert(borderColor);
11.}
12.
13.// 我们把 playDiv 传递给这个函数, 从而可以操作这个元素
14.// 也可以让我们避免重复地使用选择器
15.// 在处理复杂的选择器时很有用
16.var seeDivWidth = function(playDiv) {
17.    // 我们再次开始获得样式属性
18.    // 和我们在 domready 中用的 getStyles 独立开来
19.    // 因为我们想使用当前的值
20.    // 这可以保持 width 是准确的
21.    // 即使它在 domready 事件之后被改变了
22.    var currentDivWidth = playDiv.getStyle('width');
23.    alert(currentDivWidth);
24.}
25.
26.var seeDivHeight = function(playDiv) {
27.    var currentDivHeight = playDiv.getStyle('height');
28.    alert(currentDivHeight);
29.}
30.
31.var setDivWidth = function(playDiv) {
32.    playDiv.setStyle('width', '50px');
33.}
34.
35.var setDivHeight = function(playDiv) {
36.    playDiv.setStyle('height', '50px');
37.}
38.
39.// 注意, 在这个时候, 这个变量可以取任何名称
40.// 它会传递任何值, value 或者 element 或者你的任何东西
41.// 它将会取代任何在 domready 里面传过来的东西
42.var resetSize = function(foo) {
43.    foo.setStyles({
44.        'height': 200,
```

```
45.         'width': 200
46.     });
47. }
48.
49. window.addEventListener('domready', function() {
50.     // 因为我们要多次使用这个选择器，所以我们把它赋值给一个变量
51.     var playDiv = $('playstyles');
52.
53.     // 这里我们创建了一个包含几个属性的对象
54.     var bodyStyles = playDiv.getStyles('background-color', 'border-bottom-color');
55.
56.     // 你可以通过调用属性名来获得样式值然后传递给一个变量
57.
58.     var bgColor = bodyStyles['background-color'];
59.
60.     // 这里我们使用了一个匿名函数，从而我们可以把参数传递给
        domready 外面的函数
61.     $('bgcolor').addEventListener('click', function() {
62.         seeBgColor(bgColor);
63.     });
64.
65.     $('border_color').addEventListener('click', function() {
66.         // 除了可以把一个样式属性传递给一个变量
67.         // 你还可以在这里直接调用
68.         seeBorderColor(bodyStyles['border-bottom-color']);
69.     });
70.
71.     $('div_width').addEventListener('click', function() {
72.         seeDivWidth(playDiv);
73.     });
74.
75.     $('div_height').addEventListener('click', function() {
76.         seeDivHeight(playDiv);
77.     });
78.
79.     $('set_width').addEventListener('click', function() {
80.         setDivWidth(playDiv);
81.     });
82.
83.     $('set_height').addEventListener('click', function() {
84.         setDivHeight(playDiv);
85.     });
86.
```

```
87.    $('reset').addEvent('click', function() {  
88.        resetSize(playDiv);  
89.    });  
90.});
```

这里是 HTML 代码:

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. <div id="playstyles"> </div>  
2.    <br />  
3.    <button id="bgcolor">See background-color</button>  
4.    <button id="border_color">See border-bottom-color</button><  
    br /><br />  
5.    <button id="div_width">See width</button>  
6.    <button id="div_height">See height</button><br /><br />  
7.    <button id="set_width">Set weight to 50px</button>  
8.    <button id="set_height">Set height to 50px</button><br /><b  
    r />  
9.    <button id="reset">Reset size</button>
```

这里是 CSS 代码

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. #playstyles {  
2.    width: 200px  
3.    height: 200px  
4.    background-color: #eeeeee  
5.    border: 3px solid #dd97a1  
6. }
```

查看 background-color 查看 border-bottom-color

查看 width 查看 height

设置 width 为 50px 设置 height 为 50px

恢复大小

更多学习...

[下载一个包含你开始所需要的所用东西的 zip 包](#)

包含 MooTools 1.2 核心库，一个外部 JavaScript 文件，一个简单的 HTML 页面和一个 CSS 文件。

更多关于样式表的内容

要学习更多关于样式表的内容，请查阅 MooTools 文档中的 [Element.Style](#) 部分。

输入过滤——第一部分（数字）

请尊重个人劳动，转载请注明出处：http://ooboy.net，译者：Fdream

如果你还没有准备好开始这一讲，请参考这一系列的教程，这里是[《MooTools 1.2 系列教程目录》](#)。

今天我们来看看 MooTools 是怎样使得过滤用户输入变得非常轻松。我们今天将讲一些基本的数字过滤，明天再更深入地讲讲字符串过滤。

注意：JavaScript 中的输入过滤只是为了保证（客户端）代码顺利执行，并不能替代服务器端的字符串过滤来保护你的应用程序不被注入攻击。

在第四讲的最后的一个例子中，我们从文本输入框获取 RGB 值，然后使用它们来改变页面背景色，今天我们首先来看看那个例子的部分代码，并以此展开我们这一讲。

rgbToHex()

从技术上讲，rgbToHex() 方法实际上是属于 [Array 集合](#) 的。由于它是一个来处理数字的数组方法，我们今天来学习一下这个方法。从功能上来讲，rgbToHex() 使用起

来很简单：

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. function changeColor(red_value, green_value, blue_value) {  
2.     var color = [red_value, green_value, blue_value].rgbToHex()  
3.     ;  
4.     alert('Converts to : ' + color);  
5. }
```

这很正常很完美，因为红色、绿色和蓝色的值都是数字。试试，如果当你传入了一些其他意外的东西：

在这个结果的最后你看到了一个“NaN”，NaN 代表不是一个数字（Not a Number）。如果你把颜色的值作为硬编码写在代码里面，这种情况可能不会出现。但是如果你是从一个输入表单获得的这

个值，那么你很可能会碰到这样的情况，你需要去处理这样一些不符合要求的输入值。

toInt()

因此，现在我们需要一种方式确保传给 rgbToHex() 方法的参数都是数字——这里就需要使用 toInt() 方法了。toInt() 是另一个相对简单的函数。你可以在一个变量上调用它，那么它将尽可能地将它转换成一个整数。

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var toIntDemo = function(make_me_a_number) {  
2.     var number = make_me_a_number.toInt();  
3.     alert ('Best Attempt : ' + number);  
4. }
```

正如你说看到的，toInt() 方法并不能处理所有你可以想到的情况，不过幸亏有了 MooTools 里面另外一个很酷的方法叫做 \$type()，我们也可以很好地处理那个问题。

\$type()

\$type() 是另外一个来自 MooTools 的令人不可思议的简单和有用的东西。它可以检查你传入的无论什么变量，然后返回一个字符串，告诉你这个变量是什么类型：

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var checkType = function(variable_to_check) {  
2.     var variable_type = $type(variable_to_check);  
3.     alert("Variable is a : " + variable_type);  
4. }
```

那里还有许多\$type()方法可以检测的类型——你可以在这个[Core.\\$type\(\)文档](#)中找到一个完整的列表。不过现在，我们真正关心的是怎么检测整数。如果我们在

toIntDemo()方法中使用\$type()方法，那么我们就可以很容易地处理那些toInt()不能处理的输入了：

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var toIntDemo = function(make_me_a_number) {
2.     //Try to make the input number
3.     var number = make_me_a_number.toInt();
4.
5.     //If That didn't work, set number to 0
6.     if ($type(number) != 'number') {number = 0;}
7.     alert('Best Attempt : ' + number);
8. }
```

当我们把它们和 changeColor()方法组合起来，我们就可以得到一个几乎接近完美的解决方案了：

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var changeColor_2 = function(red_value, green_value, blue_value
2. ) {
3.     //Try to make sure everything is an integer
4.     red_value = red_value.toInt();
5.     green_value = green_value.toInt();
6.     blue_value = blue_value.toInt();
7.
8.     //Set default values on anything thats Not a Number
9.     if ($type(red_value) != 'number') {red_value = 0;}
10.    if ($type(green_value) != 'number') {green_value = 0;}
11.    if ($type(blue_value) != 'number') {blue_value = 0;}
12.
13.    //Calculate hex value
14.    var color = [red_value, green_value, blue_value].rgbToHex()
15.    ;
16.    alert('Converts to : ' + color);
17. }
```

最后一个方法中传给 `rgbToHex()` 方法的数字超过了 RGB 允许值 0-255 的范围，这个值还是被忠实地转换成了它的十六进制值。不幸的是，这意味着我们接受了一个超过那个范围的数字，我们将不能得到一个有效的十六进

制颜色值。幸运的是，MooTools 中哎哟另外一个方法，我们可以用来处理这个问题。

limit()

MooTools 中的 [limit\(\)](#) 方法也是非常简单直接的。你可以在一个数字上面调用这个方法，传入一个这个数字允许的最小值和一个允许的最大值作为参

数，它会自动地进行舍入处理。你还需要牢记这一点：`limit` 方法需要传入整数参数，因此一般在使用 `limit` 方法之前先对你要指定为数字的东西（或者其他在[数字集合](#)（

[Number Collection](#)）里面的东西）使用 `toInt()` 方法。

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var limitDemo = function(number_to_limit) {
2.     //Do our best to get an integer
3.     number_to_limit = number_to_limit.toInt();
4.
5.     //Get the limited value
6.     var limited_number = number_to_limit.limit(0, 255);
7.     alert("Number Limited To : " + limited_number);
8. }
```

示例代码

把上面的方法和我们刚才的 `changeColor()` 方法混合起来试试：

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var changeColor = function(red_value, green_value, blue_value) {
2.     //Try to make sure everything is an integer
3.     red_value    = red_value.toInt();
4.     green_value  = green_value.toInt();
5.     blue_value   = blue_value.toInt();
6.
7.     //Set default values on anything thats Not a Number
8.     if ($type(red_value) != 'number') {red_value = 0;}
```

```
9.     if ($type(green_value) != 'number') {green_value = 0;}
10.    if ($type(blue_value)  != 'number') {blue_value = 0;}
11.
12.    //Limit Everything to the RGB Scale (0 - 255)
13.    red_value   = red_value.limit(0, 255);
14.    green_value = green_value.limit(0, 255);
15.    blue_value  = blue_value.limit(0, 255);
16.
17.    //Calculate hex value
18.    var color = [red_value, green_value, blue_value].rgbToHex()
    ;
19.    alert('Converts to : ' + color);
20. }
```

更多学习

[下载一个包含你开始所需要的全部东西的 zip 包](#)

- [标准的数字（Number）处理功能函数](#)
- [Mootools 的数字（Number）处理功能函数](#)
- [Mootools 的数组（Array）处理功能函数](#)

输入过滤第二部分（字符串）

请尊重个人劳动，转载请注明出处：<http://ooboy.net>， 译者：Fdream

字符串函数

如果你还没有准备好开始这一讲，请参考这一系列的教程，这里是[《 MooTools 1.2 系列教程目录 》](#)。

今 天我们来看一看 MooTools 给我们提供的额外的一些处理字符函数。这只是 MooTools 字符串处理中的一部分，并不包含一些神秘的函数（比如 `toCamelCase()`）和使用正则表达式处理字符串的函数。我们会在以后另外用一讲来将一下正则表达式的基本知识和在 MooTools 下的使用。

在开始之前，我想先花一点时间来看一下字符串函数是怎么调用的。在我的例子中，我是在字符串变量上面直接调用这个方法的，就像下面的这样：

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var my_text_variable = "Heres some text";
2. // 结果 字符串变量 方法名
3. var result_of_function = my_text_variable.someStringFunction();
```

但是我这样写只是为了能够更清楚地解释它,你应该了解到这些字符串函数也可以直接在字符串上调用,而不需要声明一个变量,就像这样:

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var result_of_function = "Heres some text".someStringFunction()
;
```

注意一下,这个方式在 MooTools 中的数字处理函数也同样有效:

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. // 注意一下用法,是括号中的数字
2. // 而不是单引号引起来的字符串
3. var limited_number = (256).limit(1, 100);
```

还有,我想再次强调一遍:用 JavaScript 对输入过滤并不能在数据发送到服务器之前对其进行安全过滤。你在 JavaScript 中写的所有的一切都可以被你的网页浏览者看到、操控和禁止。我们将在以后讲 MooTools 的 Request 类时,对 PHP 的过滤技术进行一些简单的探讨。同时,继续保持原来要在服务器端做的任何与安全相关的事情,不要依赖 JavaScript。

trim()

trim 函数提供了一个简单直接的方式来去掉任何你想处理的字符串两端的空白字符。

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. // 这是我们要 trim 的字符串
2. var text_to_trim = " \nString With Whitespace ";
3. // trim 后的字符串是"String With Whitespace"
4. var trimmed_text = text_to_trim.trim();
```

如果你还没有见过 \n, 其实这只是一个换行符而已。你可以在一个字符串中使用它来把字符串分割成多行。trim 方法把换行符也当作一个空白符,因此它也会把换行符去掉。trim 方法唯一不做的一件特别的事情就是:它并不会去掉一个字符串里面的任何多余的空白字符。下面的这个例子展示了 trim 是怎样处理字符串里面的换行符的:

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var trimDemo = function() {
2.     // 设置我们要修剪的字符串
3.     var text_to_trim = '        \ntoo        much        whi
    tespace\n        ';
4.
5.     // 对其进行修剪
6.     var trimmed_text = text_to_trim.trim();
7.
8.     // 显示结果
9.     alert('Before Trimming : \n' +
10.         ' |-' + text_to_trim + '-| \n\n' +
11.         'After Trimming : \n' +
12.         ' |-' + trimmed_text + '-|');
13. }
```

clean()

为了更有意义,你也许不需要去掉一个字符串里的所有空白符。幸运的是,对于那些你觉得坚强的空白字符,MooTools 慷慨地为你提供了 clean() 方法。

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. // 这是我们要修剪的字符串
2. var text_to_clean = "        \nString        \nWith        Lots \n \n
    More        \nWhitespace \n        ";
3. // clean 以后的字符串是"String With Lots More Whitespace"
4. var cleaned_text = text_to_trim.clean();
```

clean() 方法与 trim() 方法有一点很大的不同。它把你传入的字符里面的空格全部提取出来,而不只是头部和尾部的空白字符。它们意味着字符串中的任何多于一个的空白字符和任何换行符和制表符(tab)。对比一下修剪的结果,我们看看到底是什么意思:

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var cleanDemo = function() {
2.     // 设置我们要修剪的字符串
3.     var text_to_clean = '        too\n        much\n
    whitespace        ';
4.
5.     // clean 该字符串
6.     var cleaned_text = text_to_clean.clean();
```

```

7.
8.    // 显示结果
9.    alert('Before Cleaning : \n' +
10.        ' |- ' + text_to_clean + ' -|\n\n' +
11.        'After Cleaning : \n' +
12.        ' |- ' + cleaned_text + ' -|');
13.}

```

contains()

和 trim() 以及 clean() 方法类似, contains() 方法做一件很简单的事情, 没有任何其他的花架子。它检查一个字符串去看它是否包含一个你要查找的字符串, 如果找到了要查找的字符串就返回 true, 如果没有找到就返回 false。

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```

1. // 我们要在这个字符串里面查找
2. var string_to_match = "Does this contain thing work?";
3.
4. // 找'contain', did_string_match 为 true
5. var did_string_match = string_to_match.contains('contain');
6.
7. // 找'propane', did_string_match 为 false
8. did_string_match = string_to_match.contains('propane');

```

这个方法可以在各种情况下派上用场, 当你和其他工具, 如我们在第三讲中讲到的 Array.each() 函数配合使用时, 你可以用相对较少的代码来完成一些稍微复杂的任务。举个例子, 如果我们把一系列单词放进一个数组, 然后一个一个地遍历, 我们可以用较少的代码在一个文本的相同区域中寻找多个单词:

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```

1. string_to_match = "string containing whatever words you want to
   try to match";
2.    word_array = ['words', 'to', 'match'];
3.    // 把数组中的每一个单词作为变量传进去
4.    word_array.each(function(word_to_match) {
5.        // 寻找当前的单词
6.        if (string_to_match.contains(word_to_match)) {
7.            alert('found ' + word_to_match);
8.        };
9.    });

```

我们把它放进一个 textbox 中，加一点想象，你就可以拥有你自己的脏词（或者其他任何）检测器。

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var containsDemo = function() {
2.     // 把我们要禁止的词放进一个数组
3.     var banned_words = ['drat', 'goshdarn', 'fiddlesticks', 'kumquat'];
4.
5.     // 获得文本域中的内容
6.     var textarea_input = $('textarea_1').get('value');
7.
8.     // 枚举过滤词中的每一个词
9.     banned_words.each(function(banned_word) {
10.        // 在文本域内容中查找当前的过滤词
11.        if (textarea_input.contains(banned_word)) {
12.            // 告诉用户不能使用那个单词
13.            alert(banned_word + ' is not allowed');
14.        };
15.    });
16. }
```

Banned Words : drat goshdarn fiddlesticks kumquat

substitute()

substitute() 是一个非常强大的工具。我们今天只是讲一下一些关于它的基本知识，substitute 的更多强大的功能来自于它的正则表达式的使用，我们会在后面稍微讲一下。然而，仅仅使用这些基本功能你就可以做很多事情了。

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. // 这是要使用 substitute 方法的文本模板
2. // 注意，要替代的部分都是用花括号括起来的部分
3. var text_for_substitute = "One is {one}, Two {two}, Three is {three}.";
4.
5. // 这个对象包含了要替换的规则
6. // 没有用引号引起来的部分是搜索项
7. // 用引号引起来的部分是用来替换搜索项的句子
8. var substitution_object = {
9.     one    : 'the first variable',
```

```

10.    two    : 'always comes second',
11.    three  : 'getting sick of bronze..'
12.    };
13.
14. // 在 text_for_substitute 上调用 substitute 方法
15. // 把 substitution_object 作为参数传入
16. // 把替换结果赋值给变量 new_string
17. var new_string = text_for_substitute.substitute(substitution_ob
    ject);
18.
19. // new_string 现在的值为
    "One is the first variable,  Two always comes second, Three is
    getting sick of bronze..."

```

事实上你并不需要创建一个 substitution_object 对象来使用 substitute 方法，如果你觉得它不合适的话，下面的方法也同样可以实现：

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```

1. // 建立要替换的字符串
2. var text_for_substitute = "{substitute_key} and the original te
    xt";
3. // 把要替换的对象作为参数传给 substitute 方法
4. var result_text = text_for_substitute.substitute({substitute_ke
    y : 'substitute_value'});
5. // result_text 现在就是 "substitute_value and the original text"

```

你可以通过这个方法做得更多更深入一点，你可以用从一个 DOM 对象中获得值的函数调用来作为替换项的值，这也是可以的。

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```

1. var substituteDemo = function() {
2.     // 从 textfield 中获得原始的 文本
3.     var original_text = $('substitute_span').get('html');
4.
5.     // 用文本框中的值替换 textfield 中的值
6.     var new_text = original_text.substitute({
7.         first  : $('first_value').get('value'),
8.         second : $('second_value').get('value'),
9.         third  : $('third_value').get('value'),
10.    });
11.
12.    // 用新的文本替换 span 中的内容
13.    $('substitute_span').set('html', new_text);

```

```

14.
15.
16.    // 禁用 substitute 按钮
17.    // 并启用 reset 按钮
18.    $('simple_substitute').set('disabled', true);
19.    $('simple_sub_reset').set('disabled', false);
20. }
21.
22. var substituteReset = function() {
23.    // 创建一个变量来保存原有的文本
24.    var original_text = "|- {first} -- {second} -- {third} -|";
25.
26.    // 用原有的文本来替换 span 中的内容
27.    $('substitute_span').set('html', original_text);
28.
29.    // 禁用 reset 按钮
30.    // 并启用 substitute
31.    $('simple_sub_reset').set('disabled', true);
32.    $('simple_substitute').set('disabled', false);
33. }

```

|- {first} -- {second} -- {third} -|
first_value

second_value

third_value

在今天结束之前，有一个很小的提示，如果你在一个字符串上调用 `substitute` 方法，并且不为要替换的关键字提供一个键/值对 (key /value pair) 对象，那么它将只是简单地删除掉花括号里面的内容。因此，如果你需要保留花括号里面的字符串，请注意不要使用这个方法。举个例子，如下：

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. (" {one} some stuff {two} some more stuff").substitute({one : 'substitution text'});
```

这将返回 “substitution text some stuff some more stuff”。

更多学习

[下载一个包含你开始所需要的 zip 包](#)

- [String 上的怪异模式](#) (this guy is amazing)
- [JavaScript 字符串函数参考](#)
- [MooTools 字符串文档](#)

MooTools 1.2 的渐变

请尊重个人劳动，转载请注明出处：<http://ooboy.net>，译者：Fdream

如果你还没有准备好开始这一讲，请参考这一系列的教程，这里是[《MooTools 1.2 系列教程目录》](#)。

今天我们开始第三讲，我们今天主要看一下 [Fx.Tween](#)。和我们看到的其他的 MooTools 函数一样，这些方法使用起来都非常简单，但是功能都很强大。Tween 可以让你添加那些极“炫”的效果——可以很平滑地发生形变动画，从而改善你的用户体验。

Tween 的快捷方法

我们通常都从“基本知识”入手，不过 MooTools 1.2 只为渐变 (tween) 提供了这样极其出色的快捷方法，它们使用起来都极其简单以至于我忍不住要从这里开始。

```
.tween();
```

一个**渐变 (tween)** 是一个在两个样式属性值之间的平滑的变化。举个例子，通过渐变 (tween) 你可以把 div 的宽度 (width) 平滑地由 100 像素变化为 300 像素。

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. // 创建一个新的函数
2. var tweenerFunction = function() {
```

```

3.      // 选中你要使用渐变的元素
4.      // 然后加上.tween
5.      // 最后声明你要变化到的属性和值
6.      $('tweener').tween('width', '300px');
7.  }
8.
9.  window.addEvent('domready', function() {
10.     // 在这里给一个按钮添加一个事件
11.     // 当点击时初始化这个渐变
12.     // 并调用我们的渐变函数
13.     $('tween_button').addEvent('click', tweenerFunction);
14. });

```

相应于上面的代码，我们的HTML代码看起来大概应该是这样的：

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```

1. <div id="tweener"></div>
2. <button id="tween_button">300 width</button>

```

.fade();

这个方法可以让你平滑地调整一个元素的不透明度（opacity）。这个使用起来和上面的例子几乎一模一样：

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```

1. // 创建一个新函数
2. var tweenFadeFifty = function() {
3.     // 在这里创建你的选择器
4.     // 然后添加.fade
5.     // 最后声明一个0到1之间的值（0代表不可见，1代表完全可见）
6.     $('tweener').fade('.5');
7. }
8.
9. window.addEvent('domready', function() {
10.     // 在这里给按钮添加一个事件
11.     // 点击时初始化这个渐变
12.     // 并调用我们的渐变函数
13.     $('tween_fade_fifty').addEvent('click', tweenFadeFifty);
14. });

```

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```

1. <div id="tweener">

```



```
2. <button id="tween_fade_fifty">Fade to fifty percent opacity</button>
```

.highlight();

醒目 (highlight) 是一个目标非常明确 (也极其有用) 的渐变快捷方法, 它提供了两个功能:

1. 使用它来平滑变化到一种不同的背景色
2. 直接设置一个不同的背景色, 然后平滑变化到另外一个背景色

这些在创建用户反馈时非常有用。例如, 你可以在一个东西被选中时让某个元素闪一下, 或者你改变一下颜色, 然后当它保存或者关闭时再闪一下。这有非常多的选择, 而且非常简单易用。在这个例子中, 让我们创建两个 div, 然后分别添加两种类型的醒目 (highlight) 方法:

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. // 创建一个函数
2. var tweenHighlight = function(event) {
3.     // 这里我们使用了 event.target, 这是从这个函数中传过来的参数
4.     // 这个意思是指“触发事件的目标 (来源)”
5.     // 由于这个效果应用到触发事件的元素上面
6.     // 因此我们不需要再创建选择器
7.     // 注意: addEvent 将会自动把 event 对象作为参数传入这个调用函数
8.     // ... 非常方便
9.     event.target.highlight('#eaeal6');
10. }
11.
12. // 创建一个函数
13. // 你需要传入一个参数
14. // 由于这个函数是在一个事件 (event) 里面被调用的
15. // 这个函数将会自动传入 event 对象
16. // 然后你就可以通过 .target 来引用这个元素
17. // (event 的目标元素)
18. var tweenHighlightChange = function(item) {
19.     // 这里我们不是使用一个颜色, 而是添加了一个逗号来分隔第二个
20.     // 这将设置第一个颜色, 然后变化到第二个颜色
21.     item.target.highlight('#eaeal6', '#333333');
22. }
23.
24. window.addEvent('domready', function() {
25.     $('tweener').addEvent('mouseover', tweenHighlight);
```

```
26.     $('tweenerChange').addEvent('mouseover', tweenHighlightChan  
    ge);  
27. });
```

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. <div id="tweener"></div>  
2. <div id="tweenerChange"></div>
```

快捷方法示例

这里是一些效果的快捷方法的在线的示例。你可以按不同顺序点击这些按钮，然后注意一下它们的变化：

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var tweenerFunction = function() {  
2.     $('tweener').tween('width', '300px');  
3. }  
4.  
5. var tweenerGoBack = function() {  
6.     $('tweener').tween('width', '100px');  
7. }  
8.  
9. // .fade 也可以接受'out'和'in'作为参数，相当于0和1  
10. var tweenFadeOut = function() {  
11.     $('tweener').fade('out');  
12. }  
13.  
14. var tweenFadeFifty = function() {  
15.     $('tweener').fade('.5');  
16. }  
17.  
18. var tweenFadeIn = function() {  
19.     $('tweener').fade('in');  
20. }  
21.  
22. var tweenHighlight = function(event) {  
23.     event.target.highlight('#eaeal6');  
24. }  
25.  
26. window.addEvent('domready', function() {  
27.     $('tween_button').addEvent('click', tweenerFunction);  
28.     $('tween_reset').addEvent('click', tweenerGoBack);  
29.     $('tween_fade_out').addEvent('click', tweenFadeOut);
```

```

30.    $('tween_fade_fifty').addEvent('click', tweenFadeFifty);
31.    $('tween_fade_in').addEvent('click', tweenFadeIn);
32.    $('tweener').addEvent('mouseover', tweenHighlight);
33. });

```

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```

1. <div id="tweener"></div><br />
2. <button id="tween_button">300 width</button>
3. <button id="tween_reset">100 width</button>
4. <button id="tween_fade_out">Fade Out</button>
5. <button id="tween_fade_fifty">Fade to 50% opacity</button>
6. <button id="tween_fade_in">Fade In</button>

```

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```

1. #tweener {
2.     height: 100px
3.     width: 100px
4.     background-color: #3399CC
5. }

```

把鼠标移上去可以看到第一种类型的醒目效果。



```

300 width
100 width
Fade Out
Fade to 50% opacity
Fade In

```

更多渐变 (Tween)

创建一个新的渐变

如果你想更灵活多变和更多地控制你的变化效果,你可能想创建一个新的形变动画来替代那些快捷方式。注意使用“new”来创建一个新的 Fx.Tween 的实例:

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```

1. window.addEvent('domready', function() {
2.     // 首先我们把要变化的元素赋值给一个变量
3.     var newTweenElement = $('newTween');

```

```

4.
5.         // 现在我们创建一个动画事件，然后把这个元素作为参数传入
6.         var newTween = new Fx.Tween(newTweenElement);
7.     });

```

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```

1. <!-- 这个是我们应用渐变效果的元素 -->
2. <div id="newTween"></div>
3.
4. <!-- 这个是启动渐变效果的按钮 -->
5. <button id="netTween_set">Set</div>

```

通过渐变设置样式

一旦你从一个元素创建了一个新的渐变，你可以轻松地通过 `.set()` 方法设置一个样式属性。这个和 `.setStyle()` 方法一样。

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```

1. var newTweenSet = function() {
2.     // 注意"this"的使用
3.     // 学习如何使用"this"
4.     this.set('width', '300px');
5. }

```

就像我们以前学习的，我们想要把我们的函数从 `domready` 事件中独立出来，但是在这个例子中，我们想要在 `domready` 事件中创建一个渐变，然后在外 部引用它。我们已经掌握了一种在 `domready` 之外传递参数的方法（通过创建一个匿名函数并传递一个参数），今天我们要学习一种 Moo 化的更好的方式来 传递渐变对象（这并不是说匿名函数在任何时候都不再合适）。

`.bind()`;

通过 `.bind()`，我们可以让一个函数里面的“this”等同于参数：

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```

1. // 首先我们给上面我们看到的那个按钮添加一个点击事件
2. // 然后，不只是仅仅调用这个函数
3. // 我们调用这个函数并且添加".bind()"
4. // 然后我们替换掉任何我们要传递给函数的
5. // 现在，在这个"newTweenSet"函数内部，"this"将指向"newTween"
6. $('netTween_set').addEvent('click', newTweenSet.bind(newTween))
;

```

因此，现在我们再看看上面的这个函数，“this”现在就等同于“newTween”了（就是我们的 tween 对象）。

现在我们把这些东西放在一起看看：

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. // 这里是我们的函数
2. var newTweenSet = function() {
3.     // 由于我们使用了 bind，现在“this”就指向了“newTween”
4.     // 因此，这里的相当于是：
5.     // newTween.set('width', '300px')
6.     this.set('width', '300px');
7. }
8.
9. window.addEventListener('domready', function() {
10.    // 首先把我们的元素赋值给一个变量
11.    var newTweenElement = $('newTween');
12.
13.    // 然后我们 new 一个 Fx.Tween，然后赋值给一个变量
14.    var newTween = new Fx.Tween(newTweenElement);
15.
16.    // 现在添加我们的事件，并绑定 newTween 和 newTweenSet
17.    $('netTween_set').addEventListener('click', newTweenSet.bind(newTween));
18. });
```

启动一个渐变效果

现在，我们已经设置好了我们所有的渐变对象，我们的函数在 domready 事件之外，我们也学习了如何通过 .set(); 方法设置一个样式表属性，我们来看看实际的渐变。启动一个渐变非常简单，和 .fade(); 非常类似，总共有两种方式来使用 .start(); 方法：

1. 让一个属性值从当前值变化到另外一个值
2. 先设置一个属性值，然后变化到另外一个值

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. // 这将让宽度（width）从当前已经存在的值变化到 300px
2. newTween.start('width', '300px');
3.
4. // 这将首先设置宽度（width）为 100px，然后变化到 300px
5. newTween.start('width', '100px', '300px');
```

现在，你就可以在一个函数内部通过使用 `.start()` 方法来启动这个渐变了，如果你使用了在函数上通过 `.bind()` 方法绑定了 “newTween”，你可以使用 “this”。

以上这些就是到现在为止全部的渐变（tween）了……

尽管如此，关于渐变效果仍然有许多可以讲的。例如，如果你想一次同时 “渐变” 多个样式表属性，你可以使用 `.morph()` 方法。你还可以使用过渡效果库（transition）来改变它们进行过渡。不过这篇教程已经足够长了，因此我们把这些留在以后再讲。

更多学习……

[下载一个包含你开始所需要的东西的 zip 包](#)

包含一个 MooTools 1.2 的库，上面的例子，一个外部 JavaScript 文件，一个简单的 HTML 文件和一个 CSS 文件。

和以前一样，你最好的资源是 MooTools 1.2 的文档。

- 关于 [.tween\(\)](#) 方法的信息
- 还有，浏览一下 [.morph\(\)](#) 方法和 [transitions](#) 库

MooTools 1.2 的 Fx.Morph、Fx 选项和 Fx 事件

请尊重个人劳动，转载请注明出处：<http://ooboy.net>，译者：Fdream

如果你还没有准备好开始这一讲，请参考这一系列的教程，这里是 [《MooTools 1.2 系列教程目录》](#)。

今天，我们继续探索一下这个库的 Fx 部分，我们将学习如何使用 Fx.Morph（它从本质上可以让你同时渐变多个样式表属性），然后我们再检查一下应用到 Fx.Tween 和 Fx.Morph 的一些 Fx 选项，最后我们将看看如何使用 Fx 事件，譬如 “onComplete” 和 “onStart”。通过这些选项 和事件，我们可以获得更好的控制权来控制形变动画。

Fx.Morph

创建一个新的 Fx.Morph

初始化一个新的形变和创建一个新的渐变很类似，除了你要指定多个样式属性以外。

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. // 首先，把我们的元素赋值给一个变量
2. var morphElement = $('morph_element');
3.
4. // 现在，我们创建一个新的形变
5. var morphObject = new Fx.Morph(morphElement);
6.
7. // 现在我们可以设置样式属性，就像 Fx.Tween 一样
8. // 不过我们这里可以同时设置多个样式属性
9. morphObject.set({
10.     'width': 100,
11.     'height': 100,
12.     'background-color': '#eeeeee'
13. });
14.
15. // 我们也可以像启动一个渐变一样来启动我们的形变
16. // 不过我们这里要同时放置多个属性值
17. morphObject.start({
18.     'width': 300,
19.     'height': 300,
20.     'background-color': '#d3715c'
21. });
```

上面这些就是全部的内容了，包括创建、设置和启动一个形变。

为了让这个更合理一些，我们应该创建我们的变量，把我们的函数独立出来，并创建一些事件来控制这个事情：

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var morphSet = function() {
2.     // 这里我们可以像 Fx.Tween 一样设置样式属性
3.     // 不过在这里我们可以同时设置多个样式属性
4.     this.set({
5.         'width': 100,
6.         'height': 100,
7.         'background-color': '#eeeeee'
8.     });
9. }
10.
11. var morphStart = function() {
```

```

12.    // 我们也可以像启动一个渐变一样启动一个形变
13.    // 不过现在我们可以同时设置多个样式属性
14.    this.start({
15.        'width': 300,
16.        'height': 300,
17.        'background-color': '#d3715c'
18.    });
19. }
20.
21.
22. var morphReset = function() {
23.    // 设置为最开始的值
24.    this.set({
25.        'width': 0,
26.        'height': 0,
27.        'background-color': '#ffffff'
28.    });
29. }
30.
31. window.addEventListener('domready', function() {
32.    // 首先，把我们的元素赋值给一个变量
33.    var morphElement = $('morph_element');
34.
35.    // 现在，我们创建我们的形变
36.    var morphObject = new Fx.Morph(morphElement);
37.
38.    // 在这里我们给按钮添加点击事件
39.    // 并且绑定 morphObject 和这个函数
40.    // 从而可以在上面的函数中使用“this”
41.    $('morph_set').addEventListener('click', morphSet.bind(morphObject)
    );
42.    $('morph_start').addEventListener('click', morphStart.bind(morphObject));
43.    $('morph_reset').addEventListener('click', morphReset.bind(morphObject));
44. });

```

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```

1. <div id="morph_element"></div>
2. <button id="morph_set">Set</button>
3. <button id="morph_start">Start</button>
4. <button id="morph_reset">reset</button>

```


Set Start reset

Fx 选项 (Options)

下面的选项都可以被 Fx.Tween 和 Fx.Morph 接受。它们都非常容易实现，而且可以给你非常多的控制权来控制你的效果。要使用这些选项，请使用下面的语法：

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. // 建立你的渐变或者形变
2. // 然后在大括号{ }之间设置你的选项
3. var morphObject = new Fx.Morph(morphElement, {
4.     // 首先是选项的名字
5.     // 然后后面跟一个冒号 (:)
6.     // 然后定义你的选项
7.     duration: 'long',
8.     transition: 'sine:in'
9. });
```

fps (每秒帧数, frames per second)

这个选项决定了这个动画每秒的帧数。默认值是 50，可以接受数字和值为数字的变量。

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. // 建立你的渐变或者形变
2. // 然后在大括号{ }之间设置你的选项
3. var morphObject = new Fx.Morph(morphElement, {
4.     fps: 60
5. });
6.
7. // 或者这样
8. var framesPerSecond = 60;
9.
10. var tweenObject = new Fx.Tween(tweenElement, {
11.     fps: framesPerSecond
12. });
```

unit (单位)

这个选项设置了数字的单位。例如，你的 100 是指 100 个像素 (px)、百分比还是 em？

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. // 建立你的渐变或者形变
2. // 然后在大括号{ }之间设置你的选项
3. var morphObject = new Fx.Morph(morphElement, {
4.     unit: '%'
5. });
```

link (连接)

link 选项提供了一种方式可以让你管理多个启动效果的函数调用。例如，如果你有一个鼠标移上去（mouseover）的效果，你是希望每次用户移上去都启动这个效果吗？或者是，如果一个人把鼠标移上去两次，它应该忽略第二个响应还是应该把它们串连起来，然后等第一次调用完成以后再第二次调用这个效果？

link 又 三个设置：

- “ignore”（默认）——在一个效果没有完成之前忽略任何启动新效果的调用
- “cancel”——如果有另外一个效果调用，则放弃当前的效果，转而处理新的效果调用
- “chain”——链可以让你把效果像“链条”一样把效果连接起来，把这些调用进行堆栈，然后逐一调用这些效果，直到完成

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. // 建立你的渐变或者形变
2. // 然后在大括号{ }之间设置你的选项
3. var morphObject = new Fx.Morph(morphElement, {
4.     link: 'chain'
5. });
```

duration (持续时间)

duration 可以让你定义这个动画的持续时间。持续事件和速度是不一样的，因此如果你想让一个对象在一秒内移动 100 个像素，那么它将比一个每秒移动 1000 个像素的对象要慢。你可以输入一个数字（以毫秒为单位）、一个值为数字的变量或者三个快捷方式：

- “short” =250ms
- “normal” =500ms（默认）
- “long” =1000ms

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. // 建立你的渐变或者形变
```

```
2. // 然后在大括号{ }之间设置你的选项
3. var morphObject = new Fx.Morph(morphElement, {
4.     duration: 'long'
5. });
6.
7. // 或者这样
8. var morphObject = new Fx.Morph(morphElement, {
9.     duration: 1000
10.});
```

transition (过渡效果)

最后一个选项: transition, 可以让你决定过渡类型。例如, 它是不是一个平滑的过渡或者它应该先慢慢开始然后加速直到结束。看看这些在 MooTools 的核心库里可以用的过渡效果:

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var tweenObject = new Fx.Tween(tweenElement, {
2.     transition: 'quad:in'
3. });
```

注意: 第一个过渡条在开始时触发了一个红色的醒目效果, 在结束时触发了一个橙色的醒目效果。看看下面是怎么使用 Fx 的事件的。

这上面 30 个过渡类型可以分成十组:

- Quad
- Cubic
- Quart
- Quint
- Expo
- Circ
- Sine
- Back
- Bounce
- Elastic

每一个组都有三个选项:

- Ease In
- Ease Out
- Ease In Out

Fx 的事件

Fx 的事件使得你在动画效果的执行过程中，在不同的点执行一些代码。在创建用户反馈信息时这会很有用，这也给了你另一层控制权来控制你的渐变和形变：

- onStart——当 Fx 开始时触发
- onCancel——当 Fx 取消时触发
- onComplete——当 Fx 完成时触发
- onChainComplete——当 Fx 链完成时触发

当你建立一个渐变或者形变时，你可以设置这其中的一个事件，就像你设置一个或多个选项一样，不过不是设置一个值，而是设置一个函数：

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. // 首先我们把一个新的 Fx.Tween 赋值给一个变量
2. // 然后定义我们要渐变的元素
3. quadIn = new Fx.Tween(quadIn, {
4.     // 这里是一些选项
5.     link: 'cancel',
6.     transition: 'quad:in',
7.
8.     // 这里是一些事件
9.     onStart: function(passes_tween_element) {
10.        // 这些事件都会传递渐变的对象
11.        // 因此当动画开始时
12.        // 这里我们调用一个“highlight”效果
13.        passes_tween_element.highlight('#C54641');
14.    },
15.
16.    // 注意这个逗号是怎样始终出现在每个事件和选项之间的
17.    // 但是最后一个事件或者选项后面没有
18.    onComplete: function(passes_tween_element) {
19.        // 在结束时，我们再应用一个 highlight 效果
20.        passes_tween_element.highlight('#C54641');
21.    }
22. });
```

示例

为了生成上面的变形代码，我们可以用一种我们在这个系列的教程中还没有见过的方式来重用我们的函数。这上面所有的变形元素都使用了两个函数，一个当鼠标进入时渐变淡出，另外一个在当鼠标离开时渐变返回：

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. // 这是我们在鼠标进入时调用的函数
2. // 宽度渐变到 700px
3. var enterFunction = function() {
4.     this.start('width', '700px');
5. }
6.
7. // 这是我们在鼠标离开时调用的函数
8. // 宽度渐变回 300px
9. var leaveFunction = function() {
10.    this.start('width', '300px');
11.}
12.
13.window.addEventListener('domready', function() {
14.    // 这里我们把一些元素赋值给变量
15.    var quadIn = $('quadin');
16.    var quadOut = $('quadout');
17.    var quadInOut = $('quadinout');
18.
19.    // 然后我们创建三个渐变元素
20.    // 分别对应上面的三个变量
21.    quadIn = new Fx.Tween(quadIn, {
22.        link: 'cancel',
23.        transition: Fx.Transitions.Quad.easeIn,
24.        onStart: function(passes_tween_element) {
25.            passes_tween_element.highlight('#C54641');
26.        },
27.        onComplete: function(passes_tween_element) {
28.            passes_tween_element.highlight('#E67F0E');
29.        }
30.    });
31.
32.    quadOut = new Fx.Tween(quadOut, {
33.        link: 'cancel',
34.        transition: 'quad:out'
35.    });
36.
37.    quadInOut = new Fx.Tween(quadInOut, {
38.        link: 'cancel',
39.        transition: 'quad:in:out'
40.    });
41.
42.    // 现在我们添加鼠标进入和鼠标离开事件
```

```

43.    // 注意.addEvents 的使用
44.    // 则和.addEvent 的使用类似
45.    // 不过你可以通过下面的模式添加多个事件
46.    $('quadin').addEvents({
47.    // 首先，你要说明是什么事件，并把事件用单引号引起来
48.    // 然后后面跟一个冒号 (:)
49.    // 最后放置你的函数
50.    // 在这个例子中，函数 banding 到这个渐变对象
51.    'mouseenter': enterFunction.bind(quadIn),
52.    'mouseleave': leaveFunction.bind(quadIn)
53.    });
54.
55.    $('quadout').addEvents({
56.    // 注意我们这里是怎样重复使用这个函数的
57.    'mouseenter': enterFunction.bind(quadOut),
58.    'mouseleave': leaveFunction.bind(quadOut)
59.    });
60.
61.    $('quadinout').addEvents({
62.    // 我们这里也使用了那些同样的函数
63.    // 不过每次我们都应用一个事件到不同的元素
64.    // 并且绑定不同的渐变
65.    'mouseenter': enterFunction.bind(quadInOut),
66.    'mouseleave': leaveFunction.bind(quadInOut)
67.    });

```

更多学习……

你可以通过 Fx 库里面的工具来获得对效果更细致的控制权。请一定要阅读一下文档中的 [Fx 这一节](#)，还有 [tween](#)、[morph](#) 和 [transitions](#)。

[下载一个包含你开始所需要的东西的 zip 包](#)

包括这个页面上的实例，MooTools 1.2 核心库，一个外部的 JavaScript 文件，一个外部的 CSS 文件或者一个简单的 html 文件。

使用 MooTools 1.2 中的 Drag.Move 来实现拖放

请尊重个人劳动，转载请注明出处：<http://ooboy.net>，译者：Fdream

如果你还没有准备好开始这一讲，请参考这一系列的教程，这里是 [《MooTools 1.2 系列教程目录》](#)。

今天我们开始第十二讲，今天我们将仔细看一下 Drag.Move——一个很强大的 MooTools 类，它可以让你给你的 web 应用添加拖放功能。它的使用和我们见过的其他的插件类似：首先你使用“new”关键字来创建一个 Drag.Move 对象并赋值给一个变量，然后你再定义你的选项和事件。这就是全部要做的事情，不过你一定要注意一下下面的例子中描述的 IE 的 CSS 怪异现象。

基本用法

Drag.Move

创建你自己的拖动对象非常的容易。稍微看一下下面的例子就行了。注意一下我们是怎么把我们的 Drag.Move 对象的选项和事件从我们的 Drag 选项和事件中分离出来的。Drag.Move 类扩展了 Drag 类，因此它可以接受 Drag 类的选项和事件。今天我们并不打算特别地讲一讲 Drag 类，不过我们还是要研究一下一些有用的选项和事件。看一下下面的代码，然后学习一下其中的细节。

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var myDrag = new Drag.Move(dragElement, {
2.     // Drag.Move 的选项
3.     droppables: dropElement,
4.     container: dragContainer,
5.     // Drag 的选项
6.     handle: dragHandle,
7.     // Drag.Move 的事件
8.     // Drag.Move 事件会传递拖动的元素，
9.     // 还有可接纳拖动元素的元素（droppable）
10.    onDrop: function(el, dr) {
11.        // 显示拖动到可接纳元素的元素的 id
12.        alert(dr.get('id'));
13.    },
14.    // Drag 事件
15.    // Drag 事件传递拖动的元素
16.    onComplete: function(el) {
17.        alert(el.get('id'));
18.    }
19.});
```

在这里我们稍微打断一下……

Drag.Move 选项

Drag.Move 选项有两个很重要的元素：

- **droppables**——设置可接纳的（draggable）元素的选择器（这个元素将会注册拖动相关的事件）
- **container**——设置拖动元素的容器（可以保证元素一直在容器内）

设置这个选项非常的容易：

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. // 这里我们通过 id 定义了一个元素
2. var dragElement = $('drag_element');
3. // 这里我们通过 class 定义了一组元素
4. var dropElements = $('.drag_element');
5.
6. var dragContainer = $('drag_container');
7. // 现在创建我们的 Drag.Move 对象
8. var myDrag = new Drag.Move(dragElement , {
9.     // Drag.Move 选项
10.    // 把我们上面定义的 draggable 赋值给 droppables
11.    droppables: dropElements ,
12.    // 把我们的容器元素变量赋值给容器
13.    container: dragContainer
14.});
```

现在你的可接受拖动元素的元素就包含进来了，你就有了一个可以接受拖放元素的类。

Drag.Move 事件

这个事件可以让你在不同的点去触发一个函数，比如当你开始拖动一个对象或者你准备放下它。每一个 Drag.Move 事件都将传递拖动元素和接受拖动元素的元素（我们一直叫做 draggable）作为参数。

- **onDrop**——这个事件将在一个可拖动的元素放到一个接受拖动元素的元素里面时触发。
- **onLeave**——这个事件将在一个可拖动的元素离开一个接受拖动元素的元素时触发。
- **onEnter**——这个事件将在一个可拖动的元素进入一个接受拖动元素的元素时触发。

这些事件中的每一个事件都将调用一个函数，每个函数都将在相应的事件触发时调用。

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var dragContainer = $('drag_container');
```



```

2.
3. var myDrag = new Drag.Move(dragElement , {
4.     // Drag.Move 选项
5.     droppables: dropElements ,
6.     container: dragContainer ,
7.     // Drag.Move 事件
8.     // Drag.Move 函数将传递可拖动的元素（这个例子中是'el'）
9.     // 还有接受拖动元素的元素（这个例子中是'dr'）
10.    onDrop: function(el, dr) {
11.        // 下面这句话的意思是：
12.        // 如果你拖动的元素不是到了可以接受拖动元素的元素的范围
    内
13.        if (!dr) {
14.            // 什么都不做
15.        }
16.        // 否则（从逻辑上讲，
17.        // 如果你拖动的那个的元素到了可接受拖动元素的元素范围
    内）
18.        // 做这一件事件
19.        else {
20.            // 在这里做一些事情
21.        };
22.    },
23.    onLeave: function(el, dr) {
24.        // 这个事件将在拖动的元素离开可接受拖动对象的元素时触发
25.    },
26.    onEnter: function(el, dr) {
27.        // 这个事件将在拖动的元素进入可接受拖动对象的元素时触发
28.    }
29.});

```

一些 Drag 事件和选项

对于 Drag，有许多选项和事件，不过这里我们只看一小部分。

snap——选项

snap 选项可以让你设置用户的鼠标至少移动多少个像素后开始拖动。默认是 6，你额可以设置为任何数字或者值为数字的变量。很明显，这里有一些合理的限制（比如设置 snap 为 1000 将毫无用处），但是这在定制你的用户体验时将会派上用场。

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```

1. var myDrag = new Drag.Move(dragElement , {

```

```
2.      // Drag 选项
3.      snap: 10
4.  });
```

handle——选项

handle 可以给你的拖动元素添加一个控制对象。这个控制对象将成为唯一的可以接受“抓取”（拖动）的元素，从而允许你使用其他的元素做一些其他的事情。要设置一个控制对象，只需调用这个元素就可以了。

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1.  // 这里我们使用了一个类选择器建立了一个数组
2.  // 这将使得我们很轻易地添加多个控制对象，如果我们决定要有多个可
    接受拖动元素的元素
3.  var dragHandle = $('drag_handle');
4.  var myDrag = new Drag.Move(dragElement , {
5.      // Drag 选项
6.      handle: dragHandle
7.  });
```

onStart——事件

onStart 和它名字一样，当开始拖动时触发这个事件。如果你设置了一个很大的 snap，这个事件将不会触发直到鼠标离开元素有指定的 snap 值那么远。

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1.  var myDrag = new Drag.Move(dragElement , {
2.      // Drag 选项
3.      // Drag 选项将把拖动的元素作为参数传递
4.      onStart: function(el) {
5.          // 在这里放置开始拖动时你要做的任何事情
6.      }
7.  });
```

onDarg——事件

这个 onDrag 事件，将会在你拖动一个元素时连续地触发。

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1.  var myDrag = new Drag.Move(dragElement , {
2.      // Drag 选项
3.      // Drag 选项将把拖动的元素作为参数传递
```

```
4.     onDrag: function(el) {
5.         // 在这里放置开始拖动时你要做的任何事情
6.     }
7. });
```

onComplete——事件

最后是 onComplete 事件，将在你放下一个拖动元素时触发，而不管你是不是把它放到了一个可以接受拖动元素的元素内部。

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var myDrag = new Drag.Move(dragElement , {
2.     // Drag 选项
3.     // Drag 选项将把拖动的元素作为参数传递
4.     onComplete: function(el) {
5.         // 在这里放置开始拖动时你要做的任何事情
6.     }
7. });
```

代码示例

让我们把刚才的这些代码以一种方式组合起来，当不同的事件触发时，我们突出显示不同的内容，并且我们使用上面我们看到的选项来配置我们的 Drag.Move 对象：

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. window.addEvent('domready', function() {
2.     var dragElement = $('drag_me');
3.     var dragContainer = $('drag_cont');
4.     var dragHandle = $('drag_me_handle');
5.     var dropElement = $$('.draggable');
6.     var startEl = $('start');
7.     var completeEl = $('complete');
8.     var dragIndicatorEl = $('drag_ind');
9.     var enterDrop = $('enter');
10.    var leaveDrop = $('leave');
11.    var dropDrop = $('drop_in_droppable');
12.
13.    var myDrag = new Drag.Move(dragElement, {
14.        // Drag.Move 选项
15.        droppables: dropElement,
16.        container: dragContainer,
```

```

17.    // Drag 选项
18.    handle: dragHandle,
19.    // Drag.Move 事件
20.    onDrop: function(el, dr) {
21.        if (!dr) { }
22.
23.        else {
24.            dropDrop.highlight('#FB911C'); //橙色闪烁
25.            el.highlight('#fff'); //白色闪烁
26.            dr.highlight('#667C4A'); //绿色闪烁
27.        };
28.    },
29.    onLeave: function(el, dr) {
30.        leaveDrop.highlight('#FB911C'); //橙色闪烁
31.    },
32.    onEnter: function(el, dr) {
33.        enterDrop.highlight('#FB911C'); //橙色闪烁
34.    },
35.    // Drag 事件
36.    onStart: function(el) {
37.        startEl.highlight('#FB911C'); //橙色闪烁
38.    },
39.    onDrag: function(el) {
40.        dragIndicatorEl.highlight('#FB911C'); //橙色闪烁
41.    },
42.    onComplete: function(el) {
43.        completeEl.highlight('#FB911C'); //橙色闪烁
44.    }
45.    });
46.});

```

注意一下 CSS: 在 IE 中, 为了能够适合地注册 Drag.Move 的容器, 你需要在下面的 CSS 中明确地指出它的位置。最重要的一点是你需要记住设置容器的位置为 “position: relative”, 而设置可拖动的元素的位置为 “position: absolute”, 然后一定要设置可拖动元素的 left 和 top 属性。现在, 如果你正在为其他浏览器构建并且遵循此规则, 你可以忽略这一部分:

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```

1. /* 下面这个定义通常是不错的主意 */
2. body {
3.     margin: 0
4.     padding: 0
5. }
6.

```

```
7. /* 确保可拖动的元素有“position: absolute” */
8. /* 并设置开始时的 left 和 top 属性 */
9. #drag_me {
10.     width: 100px
11.     height: 100px
12.     background-color: #333
13.     position: absolute
14.     top: 0
15.     left: 0
16. }
17.
18.
19. #drop_here {
20.     width: 200px
21.     height: 200px
22.     background-color: #eee
23. }
24.
25. /* 确保拖动的容器有 “position: relative” */
26. #drag_cont {
27.     background-color: #ccc
28.     height: 600px
29.     width: 500px
30.     position: relative
31.     margin-top: 100px
32.     margin-left: 100px
33. }
34.
35. #drag_me_handle {
36.     width: 100%
37.     height: auto
38.     background-color: #666
39. }
40.
41. #drag_me_handle span {
42.     display: block
43.     padding: 5px
44. }
45.
46.
47. .indicator {
48.     width: 100%
49.     height: auto
50.     background-color: #0066FF
```

```

51.     border-bottom: 1px solid #eee
52. }
53.
54. .indicator span {
55.     padding: 10px
56.     display: block
57. }
58.
59. .draggable {
60.     width: 200px
61.     height: 200px
62.     background-color: blue
63. }

```

现在再建立我们的 HTML：

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```

1. <div id="drag_cont">
2.     <div id="start" class="indicator"><span>拖动开始
   </span></div>
3.     <div id="drag_ind" class="indicator"><span>拖动中
   </span></div>
4.     <div id="complete" class="indicator"><span>拖动结束
   </span></div>
5.     <div id="enter" class="indicator"><span>进入了 Droppable 元
   素</span></div>
6.     <div id="leave" class="indicator"><span>离开了 Droppable 元
   素</span></div>
7.     <div id="drop_in_droppable" class="indicator"><span>放进了
   Droppable 元素</span></div>
8.     <div id="drag_me">
9.     <div id="drag_me_handle"><span>控制对象</span></div>
10.    </div>
11.
12.    <div id="drop_here" class="draggable"> </div>
13.</div>

```

拖动开始

拖动中

拖动结束

进入了 Droppable 元素

离开了 Droppable 元素

放进了 Droppable 元素

控制对象

更多学习……

这里是文档中一些相关的章节：

- [Drag](#)
- [Drag.Move](#)

[下载一个包含你开始所需要的所有东西的 zip 包](#)

包含 MooTools 1.2 核心库、MooTools 1.2 扩展库，一个包含你的函数的外部 JavaScript 文件，一个定义你的样式的外部 CSS 文件，一个简单的 HTML 文件和上面的例子。

MooTools 和正则表达式

请尊重个人劳动，转载请注明出处：<http://ooboy.net>，译者：Fdream

如果你还没有准备好开始这一讲，我建议你先看这一系列的教程的前面一部分，这里是 [《MooTools 1.2 系列教程目录》](#)。

今天我们将先简要地看一下正则表达式，然后再看一下 MooTools 提供一些让正则表达式更容易使用的功能。如果你还不熟悉怎么使用正则表达式（regular expression (regex)），我强烈建议你花一定的时间好好看一下这篇文章中的一些链接，尤其是文章结尾“更多学习”部分的链接。我们今天只是讲一讲正则表达式最基本的用法，正则表达式能做的远远超过我们今天所讲的内容。

基本用法

test() 方法

它的简单在于，一个正则表达式可以是一个你想要匹配的简单字符串。尽管 JavaScript 本身已经为 [RegExp 对象](#) 提供了它自己的 [test\(\) 方法](#)，MooTools 的 test() 方法更好用一些，在 JavaScript 中使用正则表达式也更轻松一些。

对于初学者，我们先看一下 test() 方法最简单的用法，在一个大的字符串中查找特定的字符串：

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. // 我们要在这个字符串中查找
2. var string_to_test = "Match anything in here";
3.
4. // 我们要查找的正则表达式
5. var regular_expression = "anything";
6.
7. // 应用正则表达式，返回 true 或者 false
8. var result = string_to_test.test(regular_expression);
9.
10. // result 现在为 true
```

这和 `contains()` 函数的行为基本类似，不过 `contains` 是按照完整的单词查找，而正则表达式匹配任何它出现的地方。举个例子，在下面的这个实例中，`contains()` 方法将不返回 `true`，而 `test()` 方法将返回 `true`。（Fdream 注：经 taoyu3781212 的提醒，这个说法是不正确的。实际上，`contains()` 方法可以指定两个参数，第一个参数是要查找的字符串，第二个是分隔字符串，只有当指定第二个参数时，`contains()` 方法才会返回 `false`，这个实际上就是 `array` 的 `contains()` 方法。）

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var string_to_match = "anything else";
2.
3. // 返回 true
4. string_to_match.contains('nything')
5.
6. // 返回 false
7. string_to_match.contains('nything', ' ')
8.
9. // 返回 true
10. string_to_match.contains('anything')
11.
12. // 返回 true
13. string_to_match.test('nything');
```

另外要注意的是，除非你明确指定，正则表达式是大小写敏感的（区分大小写），因此你包含“Match”的字符串中查找“match”将返回 `false`。你可以在下面的例子中试一试：

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var regex_demo = function() {
2.     var test_string = $('regex_1_value').get('value');
3.     var regex_value = $('regex_1_match').get('value');
4.     var test_result = test_string.test(regex_value);
```



```

5.
6.     if(test_result){
7.         $('regex_1_result').set('html', "matched");
8.     }
9.     else {
10.        $('regex_1_result').set('html', "didn't match");
11.    }
12.}

```

注意，在正则表达式中有一些特殊字符，你需要小心使用。如果你把这些字符中的任何一个输入到下面的正则表达式文本框中将会产生错误，这个时候你需要刷新这个页面才能继续下面的演示例子。

- . * + ? ^ \$ { } () | [] / \

要测试的字符串:

正则表达式

忽略大小写

在很多情况下，你不需要关心你要匹配的项的大小写。如果你不想要一个正则表达式对大小写敏感，你可以在调用 `test()` 方法时添加一个参数 “i”：

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```

1. // 我们要在这个字符串中查找
2. var string_to_test = "IgNorE CaSe";
3.
4. // 返回 false
5. string_to_test.test("ignore");
6.
7. // 返回 true
8. string_to_test.test("ignore", "i");

```

从技术上讲，你可以传递多个参数给 `test()` 方法，但是由于 JavaScript 现在仅仅只支持 3 个正则表达式参数（其中 2 个在 `test()` 方法中默认启用），这个期间内你可能仅仅只能使用参数 “i”。你可以继续测试一下大小写匹配的不同：

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```

1. var regex_demo = function() {
2.     // 从输入文本框中得到要测试的字符串

```

```

3.     var test_string = $('regex_2_value').get('value');
4.
5.     // 从输入文本框中得到正则表达式
6.     var regex_value = $('regex_2_match').get('value');
7.
8.     // 如果我们需要忽略大小写
9.     var regex_param = "";
10.    if ($('regex_2_param').checked) {
11.        regex_param = "i";
12.    }
13.
14.    // 运行 test() 方法并得到结果
15.    var test_result = test_string.test(regex_value, regex_param
    );
16.
17.    // 更新结果显示区域
18.    if (test_result) {
19.        $('regex_2_result').set('html', "matched");
20.    }
21.    else {
22.        $('regex_2_result').set('html', "didn't match");
23.    }
24.}

```

要测试的字符串:

正则表达式

忽略大小写 ☐

有趣的事情

现在我们已经学会了简单的匹配, 我们可以开始看一下正则表达式更令人印象深刻的一些方面。这里不会涵盖一切可能与正则表达式相关的东西——我们将挑选一些更直接更有用的功能。

使用^从字符串开始匹配

正则表达式的“^”运算符允许你在一行字符串的开头匹配, 而不管字符的后面有没有相应的匹配。把它放在你要匹配的正则表达式的开头, 就像下面这样:

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. // 我们要测试的字符串
2. var string_to_test = "lets match at the beginning"
3.
4. // 测试这个字符串是不是以 lets 开头, 返回 true
5. var is_true = string_to_test.match("^lets");
6. 和你期待的一样, 如果这个表达式不是在字符串的开头, 这个测试将返回
   false:
7. // 我们要测试的字符串
8. var string_to_test = "lets match at the beginning";
9.
10. // 测试这个字符串是不是以 match 开头, 返回 false
11. var is_false = string_to_test.match("^match");
```

继续测试下面的:

要测试的字符串:

正则表达式

忽略大小写 ☐

使用\$匹配字符串的结尾

“\$”运算符的功能和“^”的功能类似, 但是有两点不一样:

1. 它匹配一个字符串的结尾而不是开头
2. 它放在正则表达式的结尾而不是开头

除此之外, 它的所有功能和你期待的一样:

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. // 我们要测试的字符串
2. var string_to_test = "lets match at the end";
3.
4. // 测试这个字符串是不是以 end 结尾, 返回 true
5. var is_true = string_to_test.match("end$");
6.
7. // 测试这个字符串是不是以 the 结尾, 返回 false
8. var is_false = string_to_test.match("the$");
```

通过联合使用这两个运算符, 你可以做一个很干净的测试: 你可以检查一个字符串是不是只包含你要匹配的表达式内容而没有任何其他东西。

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. // 我们要测试的字符串
2. var string_to_test = "lets match everything";
3.
4. // 测试这个字符串是不是完全和"lets match everything"一样, 返回
   true
5. var is_true = string_to_test.match("^lets match everything$");
6.
7. // 测试这个字符串是不是完全和"lets everything"一样, 返回 false
8. var is_false = string_to_test.match("^lets everything$");
```

要测试的字符串:

正则表达式

忽略大小写 ☐

字符集

字符集是另外一个正则表达式工具, 可以让你匹配多个特定的字符(A 或者 Z), 以及一系列的字符 (A 到 Z)。据个例子, 如果你想测试一个字符串中是否包含单词 moo 或者 boo, 通过字符集, 你可以在一个正则表达式的方括号[]内放置这两个字符来实现:

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. // 测试 moo 用的字符串
2. var first_string_to_test = "cows go moo";
3.
4. // 测试 boo 用的字符串
5. var second_string_to_test = "ghosts go boo";
6.
7. // 这匹配第一个字符串而不匹配第二个字符串
8. var returns_true = first_string_to_test.test("moo");
9. var returns_false = second_string_to_test("moo");
10.
11. // 这匹配第二个字符串而不匹配第一个字符串
12. returns_false = first_string_to_test.test("boo");
13. returns_true = second_string_to_test.test("boo")
14.
15. // 这同时匹配第一个和第二个字符串
16. returns_true = first_string_to_test("[mb]oo");
```

```
17.returns_true = second_string_to_test("[mb]oo");
```

要测试的字符串一:

要测试的字符串二:

正则表达式

忽略大小写 ☐

为了匹配一系列的字符，你可以单独拿出这一系列字符的开头一个字符和最后一个字符，然后把它们用一个连接符（-）连接起来。你可以通过这种方式定义一系列的数字或者字符：

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var string_to_test = "b or 3";
2. // 匹配 a, b, c, 或者 d, 返回 true
3. string_to_test.test("[a-d]");
4.
5. // 匹配 1, 2, 3, 4, 或者 5. 返回 true.
6. string_to_test.test("[1-5]");
```

如果你想在多个字符集中匹配，你可以把你的字符集放在一个方括号[]中，然后用“|”运算符隔开。

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var string_to_test = "b or 3";
2. // 匹配 a 到 d 或者 1 到 5, 返回 true
3. string_to_test.test([ "[a-d] | [1-5] ]");
```

要测试的字符串一:

要测试的字符串二:

正则表达式

忽略大小写 ☐

escapeRegExp() 方法

当你看到正则表达式建立的方法时，你可能觉得要匹配一些特殊字符非常的困难。举个实际的例子，如果你要在一个字符串中查找 “[stuff-in-here]” 或者 “\$300” 时怎么办？你可以通过手动地在每个你要忽略的特殊字符前面添加 ‘\’ 来实现。

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. // 我们要匹配的字符串，注意[, ]、-和$
2. var string_to_match = "[stuff-in-here] or $300";
3.
4. // 不正确的匹配方式
5. string_to_match.test("[stuff-in-here]");
6. string_to_match.test("$300");
7.
8. // 正确的匹配方式
9. // 注意[, ]、-和$前面的\
10. string_to_match.test("\[stuff\-in\-here\]");
11. string_to_match.test("\$300");
```

这往往是处理正则表达式头痛的地方，尤其是你对它们没有完全熟悉的时候。作为参考，正则表达式中需要转义的特殊字符包括：

- . * + ? ^ \$ { } () | [] / \

幸运的是，MooTools 提供了 `escapeRegExp()` 函数，可以确保你的正则表达式被正确地转义。这是另外一个字符串函数，因此你只需要在你开始查找之前，在你匹配的正则表达式字符串上调用这个方法就行了。

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. // 我们要转义的字符串
2. var unescaped_regex_string = "[stuff-in-here]";
3.
4. // 转义这个字符串
5. var escaped_regex_string = unescaped_regex_string.escapeRegExp(
6. );
7. // 转义后的字符串是 "\[stuff\-in\-here\]"
```

注意，这意味着你要在正则表达式中使用的任何特殊字符都必须在转义之后再添加上去：

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. // 需要转义的字符串
```

```
2. var unescaped_regex_string = "[stuff-in-here] ";
3. // 转义这个字符串，从开头匹配
4. var escaped_regex_string = "^" + unescaped_regex_string.escapeRegExp();
5. // escaped_regex_string 现在就是 "^\\[stuff\\-in\\-here\\]"
```

继续在下方的例子中测试使用 `escapeRegExp()` 和不使用的区别：


参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var regex_demo = function() {
2.     // 获取要测试的字符串
3.     var test_string_1 = $('regex_7_value_1').get('value');
4.
5.     // 获取要使用的正则表达式
6.     var regex_value = $('regex_7_match').get('value');
7.
8.     // 检查我们是不是要转义正则表达式
9.     if ($('regex_7_escape').checked) {
10.        // 如果是的，我们则进行转义
11.        regex_value = regex_value.escapeRegExp();
12.    }
13.
14.    // 检查一下我们是不是要忽略大小写
15.    var regex_param = "";
16.    if ($('regex_7_param').checked) {
17.        regex_param = "i";
18.    }
19.
20.    // 运行测试
21.    var test_result_1 = test_string_1.test(regex_value, regex_param);
22.    if (test_result_1) {
23.        $('regex_7_result_1').set('html', "matched");
24.    }
25.    else {
26.        $('regex_7_result_1').set('html', "didn't match");
27.    }
28.}
```

要测试的字符串一：

正则表达式

对正则进行转义 

忽略大小写 

记住，你可能因为使用了没有转义的特殊字符而使演示例子不能正常运行，因此当示例不能运行的时候请不要感到奇怪，因为你一直都在玩这些东西。

更多学习

[下载一个包含你开始所需要的所有东西的 zip 包](#)

[Regular-Expressions.info](#) 是一个很好的参考和学习的地方——一个值得花一些时间浏览的网站。对于那些熟悉 Perl 或者熟悉各种语言差异的人，[Robert 的 Perl 教程](#)中的关于正则表达式这一节则对一些基本概念解释得非常的好。同样，Stephen Ramsay 已经写了一个关于[Unix 正则表达式的教程](#)，用一种非常清楚和直接了当的方式讲解了其中的一些概念。

另外一个不错的地方是[正则表达式库](#)，它们有数不清的正则表达式例子来完成各种各样的常见任务。最后，如果你有勇气，你应该花一些时间来看一下 Mozilla 的[JavaScript 正则表达式参考手册](#)。这可能非常的多，但是极其有用。如果你想看一下 MooTools 这边关于正则的内容，可以看一下[test\(\) 函数的文档](#)。

MooTools 1.2 中的定时器和 Hash 对象

请尊重个人劳动，转载请注明出处：<http://ooboy.net>，译者：Fdream

如果你还没有准备好开始这一讲，请参考这一系列的教程，这里是[《MooTools 1.2 系列教程目录》](#)。

在今天的教程中，我们将关注两块内容：第一个就是`.periodical()`方法，然后我们再对 hash 做一个简介。定时器能比它表面看起来做更多的事情——定时能定期地触发一个函数。另一方面，hash 则是键值对（key/value）的集合。如果你对 hash 还不熟悉现在也不要着急——我们今天就会做一个快速简要的介绍，并且会提供一些延伸阅读的相关链接。就像 MooTools 中的所有东西一样，一旦你看到它的正确用法，它使用起来就非常的简单，并且 不可思议的有用。

`.periodical()` 函数

基本用法

使用这个方法你唯一要做的就是在一个函数的结尾添加`.periodical()`;, 那样你的函数就会定时地触发。和以前的一样, 有几个东西你是需要定义的。对于初学者, 你需要定义一个你需要使用定时器的函数, 还有你需要它多久触发一次 (以毫秒为单位)。

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var periodicalFunction = function(){
2.     alert('again');
3. }
4.
5. window.addEvent('domready', function() {
6.     // 结尾的数字决定了这个函数触发的时间间隔, 以毫秒为单位
7.     var periodicalFunctionVar = periodicalFunction.periodical(100);
8. });
```

内置的.bind()方法

`.periodical()` 方法包含了一个非常好的特性——它可以自动地绑定第二个参数。举个例子, 如果你想从 `domready` 的外面传递一个参数, 你只需要把它作为第二个参数传进去, 你就可以把它绑定到你要定期触发的函数上了。

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. window.addEvent('domready', function() {
2.     // 给一个变量赋值
3.     var passedVar = $('elementID');
4.     // 现在 periodicalFunction 就可以使用"this"来引用"passedVar"
5.     var periodicalFunctionVar = periodicalFunction.periodical(100, passedVar);
6. });
```

停止一个定时触发的函数

`$clear()`

一旦你初始化了一个定时触发的函数 (就像我们上面所做的那样), 如果你想停止它, 你可以使用`$clear()`方法, 它使用起来非常简单:

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

1. // 我们传递那个我们使用了定时器函数的定时器变量
2. \$clear(periodicalFunctionVar);

代码示例

为把这所有的连贯起来，我们就用我们目前学过的一些东西（也有一些是没有学过的）来创建一个定时器。首先，建立一个定时器的 HTML 页面，我们还需要一个开始按钮，一个停止按钮，还有一个重置按钮。另外，我们还要创建一个条形块，它可以随着时间慢慢变长。最后，我们还需要一个地方来显示当前已经运行的时间。

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

1. <button id="timer_start">start</button>
2. <button id="timer_stop">pause</button>
3. <button id="timer_reset">reset</button>
- 4.
5. <div id="timer_bar_wrap">
6. <div id="timer_bar"> </div>
7. </div>
- 8.
- 9.
10. <div id="timer_display">0</div>

现在该是 MooTools 的代码了：

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

1. var timerFunction = function(){
2. // 每次当这个函数被调用时
3. // 变量 currentTime 就会增加一
4. // 同时要注意一下"this.counter"的使用
5. // "this"是 hash
6. // 而"counter"是 key
7. var currentTime = this.counter++;
8. // 这里我们改变显示时间的 div 里面的内容
9. \$('timer_display').set('text', currentTime);
10. // 这里改变样式表的 width 属性，可以轻松地创建一个时间进度条
11. \$('timer_bar').setStyle('width', currentTime);
12. }
- 13.
14. window.addEvent('domready', function() {
15. // 这是一个简单的 hash 对象
16. // 只有一个键值对（key/value pair）

```

17. var currentCounter = new Hash({counter: 0});
18. // 我们初始化我们的定时器并传入和绑定 hash 变量
19. var simpleTimer = timerFunction.periodical(100, currentCounter);
20.
21. // 由于我们不想在 onload 的时候就启动定时器
22. // 因此我们在这里要停止这个定时器
23. $clear(simpleTimer);
24.
25. // 在开始按钮上添加一个事件
26. // 在这里再次启动这个定时器
27. $('timer_start').addEvent("click", function(){
28.     simpleTimer = timerFunction.periodical(100, currentCounter);
29. });
30.
31. // 在这里清除定时器
32. // 并是时间条闪亮一下
33. $('timer_stop').addEvent("click", function(){
34.     $clear(simpleTimer);
35.     $('timer_bar').highlight('#EFE02F');
36. });
37.
38. $('timer_reset').addEvent("click", function(){
39.     // 重置按钮首先清除这个定时器
40.     $clear(simpleTimer);
41.     // 然后把 counter 设为 0
42.     // 这个稍后再详细讲
43.     currentCounter.set('counter', 0);
44.     //
45.     $('timer_display').set('text', currentCounter.counter);
46.     $('timer_bar').setStyle('width', 0);
47. });
48. });

```

0

Hash 快速入门

创建一个 hash

在上面的例子中,可能有一些东西是你从来没有见过的。首先,我们使用了 hash。hash 是一个由键值对 (key/value) 组成的集合,它和一个包含许多对象的数组类似,不过这些对象都只有一个属性。我们先来看一下如何建立一个 hash:

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var hashVar = new Hash({
2.     'firstKey': 0,
3.     'secondKey': 0
4. });
```

你需要把键 (key) 放在左边, 而值 (value) 放在右边 (除了那些对 hash 很熟悉的人外, 我们只讲一些关于 hash 最基本的东西, 我们会在以后讲类时再来讲 hash 的存储功能)。不管怎样, 使用和这类似的系统还是有很好处的。首先, 你可以在一个对象中存储多个集合, 存取变得容易得多, 对于组织复杂的数据组织起来。

.set()方法和.get()方法

你也可以在 hash 中使用你熟悉的 .set() 和 .get() 方法。当你需要设置的时候, 你声明一个键 (key), 然后是你设置的值。当你需要获取的时候, 你值需要声明你要获取的键 (key) 就行了。就这么简单。

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. // 还是使用上面的 hash
2. // 这里我们设置 firstKey 的值为 200
3. hashVar.set('firstKey', 200);
4.
5. // 这里我们获取 firstKey 的值, 现在是 200
6. var hashValue = hashVar.get('firstKey');
```

你可以可以通过引用 hash. 键名来获取一个值:

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var hashValue = hashVar.firstKey;
2. // 上面的和下面的一样
3. var hashValue = hashVar.get('firstKey');
```

添加一个新的键值对到 hash 中

.extend();方法

你可以通过 .extend() 方法来添加一个或者多个新的键值对 (key/value pair) 集合到 hash 中。首先, 我们要创建一个新的键值对对象。

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

1. // 这是一个普通的对象
2. // 它包含键值对 (key/value pairs)
3. // 但是没有 hash 的功能
4. var genericObject = {
5. 'third': 450,
6. 'fourth': 89
7. };

如果我们要把这个集合加入到我们已经存在的 hash 中，我们只需要使用 `.extend()` 方法来扩展 hash 就行了：

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

1. //现在 hashVar 包含了 genericObject 中的所有键值对
2. hashVar.extend(genericObject);

注意：任何重复的键都将会被后面的设置覆盖掉。因此，如果你在原始的 hash 中有 `"firstKey": "letterA"` 这样一对，然后你又扩展了一对 `"firstKey": "letterB"`，这样你在 hash 中的读取结果将是 `"firstKey": "letterB"`。

合并两个 hash

`.combine()` ; 方法

这个方法可以让你合并两个 hash 对象，如果有重复的键将保留原始的值。其余的则和 `.extend()` 方法一样。

从 hash 中删除一个键值对

`.erase()` ; 方法

我们已经见过这个方法一次了。它的工作就和你期望的那样。你声明一个 hash，然后在后面副加上 `.erase()`；，最后你再把“键”（key）放在括号里面。

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

1. hashVar.erase('firstKey');

hash 和 `.each()` 方法

hash 和 `.each()` 方法又一种特别的关系，当你遍历一个 hash 的时候，遍历的函数将把“值”（value）作为第一个参数传递，而把“键”（key）作为第二个参

数传递——这和你在数组上使用 `.each` 的时候一样，它把每个“项”（item）作为第一个参数。

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. hashVar.each(function(value, key) {  
2.     // 这将为 hash 中的每一个键值对弹出一个对话框  
3.     alert(key + ":" + value);  
4. });
```

更多学习

我们目前为止要讲的关于 hash 的内容就这么多了。由于这个系列教程中我们会更深入的学习，在以后我们将找一些机会来讲有关 hash 的更多功能。但是现在，如果你是初学者，我们只是希望你能对 hash 有一个基本的概念。很快我们就要讲解类（class）了，那个时候所有的东西才会串连起来。同时，阅读一下文档中[有关 hash 的这一节](#)。

[下载一个包含你开始所需要的所有东西的 zip 包](#)

包括 MooTools 1.2 的核心库，上面的示例，一个外部的 JavaScript 文件，一个简单的 HTML 页面和一个 CSS 文件。

MooTools 1.2 的滚动条（Slider）

请尊重个人劳动，转载请注明出处：<http://ooboy.net>，译者：Fdream

如果你还没有准备好开始这一讲，请参考这一系列的教程，这里是[《MooTools 1.2 系列教程目录》](#)。

到现在为止，初始化这些 MooTools 插件对象就会开始变得越来越熟悉。滚动条（Slider）没有任何不同，你要创建一个新的滚动条，定义滚动条和滑块相关的元素，然后设置你的选项，再创建一些回调事件的控制函数。尽管滚动条（Slider）遵循这个熟悉的模式，但是任然还有一点特殊的地方。

基本用法

创建一个新的滚动条（Slider）对象

我们首先从 HTML 和 CSS 开始。基本的想法是创建一个滚动条的 div，因此是一个长的长方形（长度取决于我们用滚动条做什么），还有一个子元素作为滑块。

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. <div id="slider"><div id="knob"></div></div>
```

我们的 CSS 也可以这么简单。只需要设置宽、高，还有背景颜色。

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. #slider {  
2.     width: 200px  
3.     height: 20px  
4.     background-color: #0099FF  
5. }  
6.  
7. #knob {  
8.     width: 20px  
9.     height: 20px  
10.    background-color: #993333  
11. }
```

现在，我们可以创建我们的新滚动条对象了。要初始化滚动条，首先要把你的相关元素赋值给一些变量，然后使用“new”来创建一个滚动条 Slider 对象，这和我们以前创建 tween、morph 和 drag.move 时一样：

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. // 把元素赋值给变量  
2. var sliderObject = $('slider');  
3. var knobObject = $('knob');  
4.  
5. // 创建一个新的 slider 对象  
6. // 首先定义 slider 元素  
7. // 然后定义滑块元素  
8. var SliderObject = new Slider(sliderObject, knobObject, {  
9.     // 这里是你的选项  
10.    // 稍后我们会仔细讲一下这些选项  
11.    range: [0, 10],  
12.    snap: true,  
13.    steps: 10,
```

```

14.  offset: 0,
15.  wheel: true,
16.  mode: 'horizontal',
17.  // 当 step 的值改变时将触发 onchange 事件
18.  // 它会把当前的 step 作为参数传入
19.  onChange: function(step){
20.      // 在这里放置 onchange 时要执行的代码
21.      // 你可以引用 step
22.  },
23.  // 当用户拖动滑块时触发 ontick 事件
24.  // 它会传递当前的位置（相对于父元素的位置）
25.  onTick: function(pos){
26.      // 这是必需的，用以调整滑块的位置
27.      // 我们会在下面详细解释这个
28.      this.knob.setStyle('left', pos);
29.  },
30.  // 当拖动停止时触发
31.  onComplete: function(step){
32.      // 当完成时要执行的代码
33.      // 你可以引用 step
34.  }
35. });

```

Slider 的选项

Snap:（默认为 false），可以是一个 true 或者 false 值。这决定了滑块是不是以最小单元格移动

Offset:（默认是 0），这是滑块相对于开始的位置。你可以对此做一个试验。

Range:（默认是 false），这是一个非常有用的选项。你可以设置一个数字范围，会依照此数字和你的步数（step）触发 onchange 事件。例如：如果你设置的范围是[0, 200]，而且你设置的 step 值为 10，那么每次 onchange 的 step 的值将是 20。这个范围也是负数，例如[-10, 0]，这个数字在做反向的滚动条时会非常有用（下面有示例）。

Wheel:（默认是 false），如果设置这个参数为 true，这个滚动条将会识别鼠标滚轮事件。当使用鼠标滚轮时，你肯定需要调整 range 参数，以保证鼠标滚轮事件的行为不是相反的（同样，后面会有例子）。

Steps:（默认是 100），默认值为 100 非常有用，因为它可以很容易地作为百分比使用。当然，你也可以以你的理由设置任意多步（这是可以的）。

Mode: (默认是“horizontal”), 这个参数定义了滚动条是水平滚动还是垂直滚动。当然了, 要从水平滚动转化为垂直滚动还需要一些其它步骤。

回调事件

onChange: 当 step 改变时, 触发这个事件。同时传递参数“step”。可以从下面的例子中看到它是什么时候触发的。

onTick: 当控制点的位置发生改变时触发这个事件。同时传递参数“position”。可以从下面的例子中看到它是什么时候触发的。

onComplete: 当控制点释放时触发这个事件。传递参数“step”。可以从下面的例子中看到它是什么时候触发的。

代码示例

让我们建立一个示例, 以便看看它们的效果。

.set()方法: 看一看按钮上的事件, 看是怎么使用.set()方法的。它使用起来非常简单: 调用 slider 对象, 附加.set, 然后是你想要滚动的步数(step)。

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. window.addEventListener('domready', function() {
2.   var SliderObject = new Slider('slider', 'knob', {
3.     // 选项
4.     range: [0, 10],
5.     snap: false,
6.     steps: 10,
7.     offset: 0,
8.     wheel: true,
9.     mode: 'horizontal',
10.
11.    // 回调事件
12.    onChange: function(step){
13.      $('change').highlight('#F3F825');
14.      $('steps_number').set('html', step);
15.    },
16.    onTick: function(pos){
17.      $('tick').highlight('#F3F825');
18.      $('knob_pos').set('html', pos);
19.      // 这一行是必需的 (水平滚动使用 left)
20.      this.knob.setStyle('left', pos);
```

```

21.
22. },
23. onComplete: function(step){
24.     $('complete').highlight('#F3F825')
25.     $('steps_complete_number').set('html', step);
26.     this.set(step);
27. }
28. });
29.
30. var SliderObjectV = new Slider('sliderv', 'knobv', {
31.     range: [-10, 0],
32.     snap: true,
33.     steps: 10,
34.     offset: 0,
35.     wheel: true,
36.     mode: 'vertical',
37.     onTick: function(pos){
38.         // 这一行是必需的（垂直滚动使用 top）
39.         this.knob.setStyle('top', pos);
40.     },
41.     onChange: function(step){
42.         $('stepsV_number').set('html', step*-1);
43.     }
44. });
45.
46. // 设置垂直滚动的从 0 开始
47. // 否则的话是从顶部开始
48. SliderObjectV.set(0);
49.
50. // 设置滚动条从 7 开始
51. $('set_knob').addEvent('click', function(){ SliderObject.set(7)});
52.
53. });

```

onChange

passes the step you are on:

onTick

passes the knob position:

onComplete

passes the current step:



注意在垂直滚动的例子中，我们不仅仅只是把“mode”改成了“vertical”，我们还改变了 onTick 事件中的.setStyle(); 方法中的“left”属性为“top”属性。另外，看一下我们是怎样设置“range”从-10 开始，然后到 0 的。然后，我们在 onChange 事件中显示当前的数字，我们把这个值乘了-1，正好和位置相反。这完成了两件事情：一是让我们从 10 到 0 改变这个值，0 在最底部。但是这个可能设置 rang 为从 10 到 0，从而导致鼠标滚轮事件变得相反。这就是我们的第二个原因——鼠标滚轮读取值，而不是你要控制的方向，因此要让鼠标滚轮正确地读取滚动条并且从底部的 0 开始的值的唯一方式就是做这一点改变。

注意：至于 onTick 事件中“top”和“left”的使用，我不确定这是不是 MooTools 中的“规则”。这只是我让它们正确运行的一种方法，我很有兴趣听到一些其他的清楚的说法。

更多学习

和以前一样，请参考文档中的 [sliders 一节](#)。

[下载一个包含你开始所需要的所有东西的 zip 包](#)

包括 MooTools 1.2 的核心库和扩展库，还有一个外部的 JavaScript 文件，一个简单的 HTML 页面和一个 CSS 文件和上面的示例。

MooTools 1.2 的整理排序类 Sortables

原文地址：[30 Days of Mootools 1.2 Tutorials - Day 16 - Sortables and Intro to Methods](#)

请尊重个人劳动，转载请注明出处：<http://ooboy.net>，译者：Fdream

如果你还没有准备好开始这一讲，请参考这一系列的教程，这里是[《MooTools 1.2 系列教程目录》](#)。

从今天开始，我们将开始讲解“更多”（more）库里面的更多插件。Sortables 是一个非常强大的插件，能够真正地扩大你的用户界面设计的选择面。Sortables 类还提供了包括一个名叫“serialize”的优秀方法，通过这个方法你额可以把这些元素的 id 作为数组输出——对于服务器端的开发 非常有用。接下来，我们看看如何创建一个新的排序项集合，还有一定要看一下最后的演示实例。

基本知识

创建一个新的 Sortable 对象

首先，我们要把我们要排序的元素赋值给变量。对于 Sortables 来说，如果你想要多个列表之间的元素能够在相互之间拖拽，你需要把这些元素全部都放在一个数组中，就像这样：

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var sortableListsArray = $('#listA, #listB');
```

这样就可以把两个 ul 的 id 放到一个数组里面了。我们现在就可以从这个数组创建一个新的 sortable 对象了：

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var sortableLists = new Sortables(sortableListsArray);
```

我们假设使用的是下面的 HTML：

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. <ul id="listA">
2.   <li>Item A1</li>
3.   <li>Item A2</li>
4.   <li>Item A3</li>
5.   <li>Item A4</li>
6. </ul>
7.
8. <ul id="listB">
9.   <li>Item B1</li>
10.  <li>Item B2</li>
11.  <li>Item B3</li>
12.  <li>Item B4</li>
13. </ul>
```

我们的 sortable 列表最后看起来大概应该是这样的：

- Item A1
- Item A2
- Item A3
- Item A4

- Item B1
- Item B2
- Item B3

- Item B4

Sortable 选项

如果你想完全定义你自己的 sortable 列表，你就需要使用这些选项。

constrain

默认——false

这个选项决定了你的 sortable 列表元素是否可以在多个 ul 之间拖动。

例如，如果你在一个 sortable 对象中有两个 ul，你可以通过设置选项 “constrain:true” 来“限制”（constrain）列表的元素只允许在它们的父节点 ul 之内移动。

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var sortableLists = new Sortables(sortableListsArray, {  
2.   constrain: false // 默认为 false  
3. });
```

clone

默认——false

克隆（clone）选项允许你添加一个“clone”的元素跟随你的鼠标移动，而把原始的元素留在原地不动。你可以从下面的例子中看看如何使用 clone 选项：

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var sortableLists = new Sortables(sortableListsArray, {  
2.   clone: true // 默认为 false  
3. });
```

handle

默认——false

handler 选项可以接受一个元素作为拖动的控制器。如果你要保持你的列表中的

文本可以被选中或者保留 li 的其他行为，使用这个参数则非常方便。默认参数为 false 则会使得整个元素（li）成为控制器。

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var handleElements = $$('.handlesClass');
2. var sortableLists = new Sortables(sortableListsArray, {
3.   handle: handleElements // 默认为 false
4. });
```

opacity

默认——1

不透明度（opacity）选项可以让你调整排序元素。如果你使用了一个 clone 的副本，opacity 将作用于这个排序元素，而不是更随你鼠标的那个副本。

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var sortableLists = new Sortables(sortableListsArray, {
2.   opacity: 1 // 默认为 1
3. });
```

revert

默认——false

复原（revert）参数可以接受“false”或者 Fx 的选项值。如果你给 revert 参数设置了 Fx 的选项，那么当元素放置到一个位置时会应用相应的 Fx 设置。例如，你可以设置“duration:long”，那么当你松开鼠标时，那个克隆的对象将会在这个时间之内返回到它的位置。如果要看 revert 的效果，可以看看下面的例子：

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var sortableLists = new Sortables(sortableListsArray, {
2.   revert: false // 默认为 false
3. });
4.
5. // 你也可以设置为 Fx 选项
6. var sortableLists = new Sortables(sortableListsArray, {
7.   revert: {
8.     duration: 50
9.   }
```

```
10. });
```

snap

默认——4

snap 参数允许你设置鼠标必需拖动多少个像素之后，元素才会被拖动。

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var sortableLists = new Sortables(sortableListsArray, {  
2.     snap: 10 // 用户需要拖动 10px 来开始拖动这个拖动列表  
3. });
```

Sortable 事件

sortable 事件非常好也非常简单易用。每一个都会传递当前拖动的元素（如果你使用了 clone 元素，不是那个 clone 的元素，而是原始的元素）。

- onStart——当拖动开始时触发（当 snap 触发以后）
- onSort——当项目改变排序以后触发
- onComplete——当你把一个元素放下以后触发

我们会在后面再仔细看这些事件（你可以在后面的例子中看到效果）。

Sortable 方法

尽管我们已经使用过很多方法了，但是我们从来没有详细讲过。方法本质上还是一些函数，不过它们是属于某一个类的。不过等我们在讲类的时候，我们会第二次再建立一个通用的概念。这个插件（和我们讲过的其他插件一样），全部都遵循一个类似的模式——使用“new”初始化一个插件，定义一个或者多个选择器参数，定义你的选项，添加一些事件（和建立新的 sortable 和 tween 类似）。这个模式是类的基础。一个类最基础的就是允许你保存一些选项和函数，从而可以重复使用它们。方法就是一个类里面一些特定的函数。实例的.set()和.get()方法则是 element 的属性扩展方法。在 Fx.Tween 中，.start()就是一个方法。为了更清晰的理解，我们看看 sortable 的方法。

.detach();

通过.detach();方法，你可以剥离（detach）所有的控制器，从而使得整个列表都不可以拖动。这对于禁用拖动非常有用。

.attach();

这个方法将把控制器关联到排序项目，可以在使用 `.detach()` ; 方法后再次启动排序功能。

.addItems();

这个方法可以让你添加新的项目到你的排序列表中。这个意思是说，你有一个排序列表，用户可以向里面添加新的项目，一旦你添加了一个新的项目，你就需要在那个新的项目上启动排序功能。

.removeItems();

这个方法可以让你从已有的排序列表中删除一些元素。当你需要锁定排序列表中的一些特殊的项目不让它参与排序时非常有用。

.addLists();

除了添加一个新项到一个已经存在的排序列表中，你也许还想添加一个新的列表到排序列表中。`.addLists()` ; 方法可以让你添加多个列表，这使得添加多个排序对象变得真正容易。

.removeLists();

可以让你从排序对象中移除整个整个列表。当你需要锁定一些特殊的列表时，这个很有用。你可以移除一个列表，保留下来的其他项目则可以继续排序，但是会锁定这个移除的列表。

.serialize();

这个排序功能非常优秀，不过如果你想处理这些数据怎么办？`.serialize()` ; 方法将依照它们的顺序返回包含这些项目 `id` 的数组。你可以通过索引值来选择你要获取数据的列表。

方法的影响力远远超过我们这里所涵盖的内容，如果你是新手，那就让这做为一个简单的概念介绍吧，我们会在后面的教程中更深入地讨论方法和类。

代码示例

下面的示例使用了一些选项，全部的事件和上面描述的全部方法。希望这个代码有自解释性，不多注释里又更多说明。记住，下面所有的事情都必需在 domready 事件里面。

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var sortableListsArray = $('#numberlist, #letterlist');
2. var sortableLists = new Sortables(sortableListsArray, {
3.     // 当我移动的时候，复制一个副本跟随鼠标移动
4.     clone: true,
5.     // 定义拖动控制器（柄，把手）的 css 类名
6.     handle: '.handle',
7.     // 在拖动之后，允许你使用特效让它回到某个位置
8.     revert: {
9.         // 接受 Fx 选项
10.        duration: 50
11.    },
12.    // 决定拖动元素的不透明度，而不是跟随鼠标的副本
13.    opacity: .5,
14.
15.    onStart: function(el){
16.        // 传递的是你正在拖动的元素
17.        $('start_ind').highlight('#F3F865');
18.        el.highlight('#F3F865');
19.    },
20.    onSort: function(el) {
21.        // 传递的是你正在拖动的元素
22.        $('sort_ind').highlight('#F3F865');
23.    },
24.    onComplete: function(el) {
25.        // 传递的是你正在拖动的元素
26.        $('complete_ind').highlight('#F3F865');
27.        var listOne = sortableLists.serialize(0);
28.        var listTwo = sortableLists.serialize(1);
29.        $('numberOrder').set('text', listOne).highlight('#F3F865'); ;
30.        $('letterOrder').set('text', listTwo).highlight('#F3F865'); ;
31.    }
32. }).detach(); // 禁用控制器，因此你必需点击按钮才能让它们可以拖动
33.
34. var addListoSort = $('addListTest');
35.
36. $('addListButton').addEvent('click', function(){
37.     sortableLists.addLists(addListoSort);
38. });
39.
```

```

40. $('#removeListButton').addEvent('click', function(){
41.     sortableLists.removeLists(addListoSort);
42. });
43.
44. $('#enable_handles').addEvent('click', function(){
45.     sortableLists.attach();
46. });
47.
48. $('#disable_handles').addEvent('click', function(){
49.     sortableLists.detach();
50. });
51.
52. var itemOne = $('one');
53.
54. $('#add_item').addEvent('click', function(){
55.     sortableLists.addItem(itemOne);
56. });
57.
58. $('#remove_item').addEvent('click', function(){
59.     sortableLists.removeItem(itemOne);
60. });

```

控制器默认是没有启用的（仔细看一下上面的代码）。要开始拖动排序，你需要点击“启用排序”按钮。

```

onStart
onSort
onComplete

```

Top List Order

- 1
- 2
- 3
- 4

- 5
- 6
- 7

Bottom List Order

- A
- B
- C
- D
- E
- F
- G

- add A
- add B
- add C
- add D
- add E
- add F
- add G

更多学习

参考阅读文档中[有关 sortable 的这一节](#)。

[下载一个包含你开始所需要的所有东西的 zip 包](#)

包括 MooTools 1.2 的核心库和扩展（更多）库，上面的示例，一个外部的 JavaScript 文件，一个简单的 HTML 页面和一个 CSS 文件。

Mootools 1.2 手风琴（Accordion）教程

原文地址：[30 Days of Mootools 1.2 Tutorials - Day 17 - Accordion](#)

请尊重个人劳动，转载请注明出处：<http://ooboy.net>，译者：Fdream

如果你还没有准备好开始这一讲，请参考这一系列的教程，这里是[《MooTools 1.2 系列教程目录》](#)。

继续我们的“更多”（More）库里面的插件教程，今天我们来学习一下可能是最流行最受欢迎的插件——手风琴。创建和配置一个基本的手风琴很简单，但是你一定要认真看完全文，因为更多的高级选项可能有一些复杂。

基础知识

创建一个新的手风琴

要创建一个新的手风琴，你需要选择一些成对的元素——包含标题和内容。因此，首先，需要给每一个标题和每一个内容块分别指定一个 css 类名：

参考代码：[\[复制代码\]](#) [\[保存代码\]](#)

1. `<h3 class="togggers">Toggle 1</h3>`
2. `<p class="elements">Here is the content of toggle 1</p>`
3. `<h3 class="togggers">Toggle 2</h3>`
4. `<p class="elements">Here is the content of toggle 2</p>`

现在，我们选择所有 css 类名为“togggers”和所有 css 类名为“elements”的元素，并把它们赋值给变量，然后初始化一个手风琴对象。

参考代码：[\[复制代码\]](#) [\[保存代码\]](#)

1. `var toggles = $$('.togggers');`
2. `var content = $$('.elements');`
- 3.
4. `// 创建你的对象变量`

5. // 使用“new”创建一个新的手风琴对象
6. // 设置开关（toggle）数组
7. // 设置内容数组
8. var AccordionObject = new Accordion(toggles, content);

手风琴的默认设置给你的效果可能是这样的：

Toggle 1

Here is the content of toggle 1

Toggle 2

Here is the content of toggle 2

Toggle 3

Here is the content of toggle 3

选项

当然，如果你想要手风琴默认效果以外的东西，你需要调整一下选项。在这里我们将逐个讲解。

display

默认为 0

这个选项决定了当页面加载后哪个元素会显示出来。默认值为 0，因此第一个元素会显示出来。如果要设置为其他元素，只需要设置为另外一个元素的索引值（为整数）就可以了。和“show”不一样，“display”将会使用渐变动画让元素展开。

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

1. var AccordionObject = new Accordion(toggles, content {
2. display: 0 // 默认为 0
3. });

show

默认为 0

和 “display” 非常类似，“show” 决定了当页面加载后那个元素将会展开，不过它没有渐变动画，它只是在页面加载后显示出来，而不使用任何渐变动画。

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var AccordionObject = new Accordion(toggles, content {  
2.     display: 0 // 默认为 0  
3. });
```

height

默认为 true

当设置为 true，内容显示时，将有高度渐变动画效果。这和你上面看到的一样，是一个标准的手风琴设置。

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var AccordionObject = new Accordion(toggles, content {  
2.     height: true // 默认为 true  
3. });
```

width

默认为 false

和 “height” 类似，不过不是使用高度渐变动画来显示内容，而是使用宽度渐变动画来显示内容。如果你把 “width” 选项和其他我们看到的标准设置一起使用，各个标题开关之间的距离将保持不变，这个距离完全基于内容的高度。内容的 div 将会从左到右，宽度逐渐变宽来显示内容。

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var AccordionObject = new Accordion(toggles, content {  
2.     width: false // 默认为 false  
3. });
```

opacity

默认为 true

这个选项设置了当你隐藏或者显示内容时，是否使用不透明度渐变效果。由于我们在上面使用了默认设置，所以你可以看到效果。

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var AccordionObject = new Accordion(toggles, content {  
2.     opacity: true // 默认为 true  
3. });
```

fixedHeight

默认为 false

要 设置一个固定的高度，你可以在这里设置一个整数（例如，你可以设置 100，从而可以使得内容的高度为 100px）。如果你准备设置的高度小于内容的高度，在这里你需要在 CSS 中设置一下内容的 overflow 属性。和你可能的期望一样，当你使用“show”选项时，当页面第一次载入时，它是不会生效（被注册）的。

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var AccordionObject = new Accordion(toggles, content {  
2.     fixedHeight: false // 默认为 false  
3. });
```

fixedWidth

默认为 false

和上面的“fixedHeight”类似，如果你给了这个选项一个整数，这将设置它的宽度。

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var AccordionObject = new Accordion(toggles, content {  
2.     fixedWidth: false // 默认为 false  
3. });
```

alwaysHide

默认为 false

这个选项可以让你给标题增加一个开关。通过把这个选项设置为 true，当你点击一个内容已经展开的标题时，它将关闭这个内容块，但是不会展开任何元素。你马上就可以在下面的例子中看到。

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var AccordionObject = new Accordion(toggles, content {  
2.   alwaysHide: false // 默认为 false  
3. });
```

事件

onActive

当你开关一个元素时触发这个事件。他将会传递这个开关控制元素和内容元素，还有开关状态作为参数。

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var AccordionObject = new Accordion(toggles, content {  
2.   onActive: function(toggler, element) {  
3.     toggler.highlight('#76C83D'); // 绿色  
4.     element.highlight('#76C83D');  
5.   }  
6. });
```

onBackground

当 ige 元素开始隐藏时触发这个事件，它将传递所有其他正则关闭的元素作为参数，而不是展开的元素。

参考代码： [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var AccordionObject = new Accordion(toggles, content {  
2.   onBackground: function(toggler, element) {  
3.     toggler.highlight('#DC4F4D'); // 红色  
4.     element.highlight('#DC4F4D');  
5.   }
```


6. });

onComplete

这是一个标准的 onComplete 事件。它传递一个包含内容元素的变量。这里又一种更好的方式获取这些东西，如果有人知道，可以作个记录。

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. var AccordionObject = new Accordion(toggles, content {
2.   onComplete: function(one, two, three, four){
3.     one.highlight('#5D80C8'); // 蓝色
4.     two.highlight('#5D80C8');
5.     three.highlight('#5D80C8');
6.     four.highlight('#5D80C8');
7.   }
8. });
```

方法

.addSection();

通过这个方法，你可以在中间添加一节（一个标题/内容元素对）。这个我们见过的许多其他方法一样。首先，我们引用一个手风琴对象，在后面加上.addSection，然后你可以调用标题的 id、内容的 id，最后给它指定一个位置——这个新元素要出现的位置（0 是第一个位置）。

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

```
1. AccordionObject.addSection('togglesID', 'elementsID', 2);
```

注意：当你通过这种方式添加一节，虽然它会在索引值为 2 的地方显示，但是它的真实索引应该是最后一个索引值加 1。如果你在一个数组中有 5 个项，然后你添加了第六个，它的索引值则为 5，而不管你通过.addSection();方法把它添加在了什么地方。

.display();

这个方法可以让你展开一个指定的元素。你可以通过它的索引值来选择这个元素（如果你添加了一个新的元素对，你又想展开它们，你需要使用一个新的索引值）。

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

1. `AccordionObject.display(5);` // 这将显示你新增加的元素

实例演示

这里我们有一个全功能的手风琴，使用了上面我们看到的所有事件和方法，还有非常多的选项。仔细阅读下面的代码和代码内相关的注释，看看它们是怎么使用的。

参考代码: [\[复制代码\]](#) [\[保存代码\]](#)

1. // 把开关元素和内容元素赋值给两个变量
2. `var toggles = $$('.toggles');`
3. `var content = $$('.elements');`
- 4.
5. // 建立一个对象变量
6. // 使用 `new` 创建一个新的手风琴对象
7. // 设置开关数组
8. // 设置内容数组
9. // 设置相关选项和事件
10. `var AccordionObject = new Accordion(toggles, content, {`
11. // 当页面载入后
12. // 这将显示 (`show`) 内容元素 (对应索引的元素)
13. // 没有任何渐变动画，只是展开
14. // 同时注意：如果你使用了“`fixedHeight`”，
15. // 当页面第一次载入时，`show` 选项将不会起作用
16. // “`show`”选项会覆盖“`display`”选项
17. `show: 0,`
- 18.
- 19.
20. // 当页面载入后
21. // 这将显示 (`display`) 内容元素 (对应索引的元素)
22. // 内容展开时将有渐变动画
23. // 如果同时设置了 `display` 选项和 `show` 选项
24. // `show` 选项将覆盖 `display` 选项
25. // `display: 0,`
- 26.
27. // 默认为 `true`
28. // 这将创建一个垂直的手风琴
29. `height : true,`
- 30.
31. // 这是水平手风琴参数使用的
32. // 由于牵涉到 `CSS`，这个会比较复杂

```
33. // 我们在后面的某一讲中再讲一下?
34. width : false,
35.
36. // 默认为 true
37. // 这将会给内容一个不透明度的渐变效果
38. opacity: true,
39.
40. // 默认为 false, 也可以是一个整数 -
41. // 将固定所有内容区块的高度
42. // 需要设置 css 中的 overflow 规则
43. // 如果使用了"show", 在页面第一次载入时不会生效
44. fixedHeight: false,
45.
46. // 可以为 false 或者一个整数
47. // 和上面的 fixedHeight 类似,
48. // 不过这是针对水平手风琴的设置
49. fixedWidth: false,
50.
51. // 可以让你开关一个展开的项
52. alwaysHide: true,
53.
54. // 标准的 onComplete 事件
55. // 为每一个内容块元素传递一个变量
56. onComplete: function(one, two, three, four, five){
57.     one.highlight('#5D80C8'); // 蓝色
58.     two.highlight('#5D80C8');
59.     three.highlight('#5D80C8');
60.     four.highlight('#5D80C8');
61.     five.highlight('#5D80C8'); // 这是添加的节
62.     $('complete').highlight('#5D80C8');
63. },
64.
65. // 这个事件将在你开关一个元素时触发
66. // 将会传递正在打开的开关元素
67. // 和内容元素
68. onActive: function(toggler, element) {
69.     toggler.highlight('#76C83D'); // 绿色
70.     element.highlight('#76C83D');
71.     $('active').highlight('#76C83D');
72. },
73.
74. // 这个事件将在一个元素开始隐藏时触发
75. // 将会传递所有正在关闭的元素
76. // 或者没有展开的元素
```

```

77.   onBackground: function(toggler, element) {
78.       toggler.highlight('#DC4F4D'); // 红色
79.       element.highlight('#DC4F4D');
80.       $('background').highlight('#DC4F4D');
81.   }
82. });
83.
84. $('add_section').addEvent('click', function(){
85.     // 新增加的节是成对的
86.     // （包括开关的 id 和相关的内容的 id）
87.     // 后面是它们要放置的位置的索引
88.     AccordionObject.addSection('togglersID', 'elementsID', 0);
89. });
90.
91. $('display_section').addEvent('click', function(){
92.     // 将决定哪个元素在页面第一次载入时显示
93.     // 将覆盖 display 选项的设置
94.     AccordionObject.display(4);
95. });

```

这里你可以看到事件是什么时候触发的。

```

onComplete
onActive
onBackground

```

可以试试用下面的按钮添加一对内容。

Toggle 1

Here is the content of toggle 1 Here is the content of toggle 1 Here is the content of toggle 1 Here is the content of toggle 1 Here is the content of toggle 1 Here is the content of toggle 1 Here is the content of toggle 1

Toggle 2

Here is the content of toggle 2

Toggle 3

Here is the content of toggle 3

Toggle 4

Here is the content of toggle 4

Toggle Add

Here is the content of toggle 4

要注意的地方

- `.display` 可以识别索引 (`index`)，不过如你使用了 `addSection` 方法，那么这一节将使用最后一个索引
- 如果你使用了“`fixedHeight`”选项，在“`show`”选项下不会起任何作用，但是在“`display`”选项下会有作用

更多手风琴选项、事件和方法

手风琴 (`Accordion`) 类继承自 `Fx.Element` 类，而 `Fx.Element` 类又继承自 `Fx` 类。你可以使用这些类的各种选项来优化你的手风琴 (`Accordion`)。虽然它看起来是个很简单的东西，但是手风琴是一个非常有用和强大的插件。我非常乐意看到有人利用这个做出的任何效果。

更多学习

参考文档中的 [accordion 这一节](#)，还有 [Fx.Elements](#) 和 [Fx](#) 这两节。你还可以看看 [MooTools 官方 demo 中的 accordion 的使用](#)。

[下载一个包含你开始所需要的所有东西的 zip 包](#)

包括 MooTools 1.2 的核心库和扩展 (更多) 库，上面的示例，一个外部的 JavaScript 文件，一个简单的 HTML 页面和一个 CSS 文件。

AJAX

最近使用脚本框架 mootools 做网站，以下是处理异步更新的代码：

如果没有下载该框架可以到 <http://www.mootools.net/> 下载

JS1:

```
<script type="text/javascript">

    window.addEvent("domready", function() {

        $("send").addEvent("click", function() {

            var
            url="broadcastmore.aspx?areaid="+$("areaidvalue").value+"&say="+escape($("say").value);

            // escape() 是处理编码函数，没有它传替中文时会出乱码

            new Ajax(url, {method:'post', onComplete:function() {

                $("Content").innerHTML=this.response.text;

                alert('发表成功!');

            }}).request();

        });

    });

</script>
```

HTML1:

```
<div class="main_zh">

    <div id="Content">

    </div>

    <div class="mk">

        <div class="titlezb">

            <div class="titleleft">
```

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">

    <title>mootools ajax 使用示例</title>

    <script type="text/javascript">
```

```

src="mootools-release-1.11.js"></script>
<script type="text/javascript">
window.addEvent("domready",function(){

    $('btnSent').addEvent('click',function(){
        if($('txtContent').innerText==''){

            alert('发送内容不能空! ');

            return;
        }
        var url='Default2.aspx';
        var postData=$("postMessage").toQueryString();
        new Ajax(url,{method:'post',onComplete:function(){
            $('messageBox').innerHTML+= this.response.text;
        }
        }).request(postData);
    });

});
</script>
</head>
<body>
    <div style="margin:auto;text-align:center; ">
        <div style=" width:650px; height:700px;">
            <div id="messageBox"></div>
            <hr />
            <form id="postMessage" method="post"
name="postMessage" runat="server">
                <div>
                    <ul>

                        <li style="list-style-type: none;">请输

入您的网名:  &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<input ID="txtName"

runat="server" Width="243px" value="填写网名"

runat="server"/>

                        </li>
                    </ul>
                    <ul>

                        <li style="list-style-type: none;">请输

入要发送的内容: <textarea ID="txtContent" runat="server"

```



```

Height="75px" Width="259px" value="填写内容" cols="20"
rows="4"></textarea></li>
        <span style="color:Red;">*</span>
    </ul>
    <ul>
        <li style="list-style-type:
none;"><input type="button" ID="btnSent" runat="server"
value="发送" /></li>
    </ul>
</div>
</div>
</body>
</html>

```

.cs 文件:

```

protected void Page_Load(object sender, EventArgs e)
{
    if (IsPostBack)
    {
        if (!String.IsNullOrEmpty(txtName.Value)
&& !String.IsNullOrEmpty(txtContent.Value.Trim()))
        {
            string name = txtName.Value.Trim();
            string content = txtContent.Value.Trim();

            string msg = "<div><ul><li>" + name + "说:
" + content + "</li></ul></div>";

            Response.Clear();
            Response.Write(msg);
            Response.End();
        }
        else if
( !String.IsNullOrEmpty(txtContent.Value.Trim())&&String.
IsNullOrEmpty(txtName.Value) )
        {
            string name = "游客";

```

```

        string content = txtContent.Value.Trim();
        string msg = "<div><ul><li>" + name + "说:"
" + content + "</li></ul></div>";
        Response.Clear();
        Response.Write(msg);
        Response.End();
    }
}
}

```

示例三:

HTML-JAVASCRIPT:

```

<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="responserequest.aspx.cs" Inherits="responsereq
uest" %>

```

```

<!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 runat="server">

```

```

    <title>无标题页</title>

```

```

    <script type="text/javascript"
src="mootools-release-1.11.js"></script>

```

```

    <script type="text/javascript">
        window.addEvent("domready",function(){
            $('myForm').addEvent('submit', function(e) {
                new Event(e).stop();
                if($("message").value==""){
                    alert('请输入要发送的消息');
                    return;
                }
            })
            this.send({
                onComplete: function() {
                    var
request=Json.evaluate(this.response.text);
                    $("messagebox").innerHTML=request.msg;

```

```

        $("namebox").innerHTML=request.name;

        alert('发送成功! ');

    }
    });
});

});

</script>
</head>
<body>
<form id="myForm" action="responserequest.aspx"
method="post">
    <div id="namebox"></div>
    <div id="messagebox"></div>
    <input type="text" id="message"
name="message" runat="server" />

    <input type="submit" name="btn" id="btn" value="发送"/>
</form>

</body>
</html>

```

。CS 文件:

```

using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

public partial class responserequest : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        string msg = Request["message"];
    }
}

```

```

        string name = "mimengjiangnan:";
        if (!String.IsNullOrEmpty(msg))
        {
            Response.Clear();
            Response.Write(string.Concat("{name:'", name, "
',msg:'", msg, "'}"));
            Response.End();
        }
    }
}

```

mootools 1.1 的 ajax 请求基类 XHR 创建 XMLHttpRequest 对象的方法如下:

```

setTransport: function() {
    this.transport = (window.XMLHttpRequest) ? new XMLHttpRequest() :
(window.ie ? new ActiveXObject('Microsoft.XMLHTTP') : false);
    return this;
}

```

在 IE 下,用的是最早版本的 XMLHTTP ActiveX 控件(随 IE5.0 发布),问题是随着 MSXML 库后续版本的发布,也发布了 XMLHTTP 的几个新版本,每个新版本都更加稳定、速度更快,因此应该确保在客户端使用最新的可用版本。

IE XMLHTTP ActiveX 的签名包括:

Microsoft.XMLHTTP

MSXML2.XMLHTTP

MSXML2.XMLHTTP.3.0

MSXML2.XMLHTTP.4.0

MSXML2.XMLHTTP.5.0

在这点上,我看到的一个 ajax js 库 AjaxTK-4.5.0 无疑要好上许多:

```

if (window.XMLHttpRequest) {
    AjaxLoader.__createXHR = function() { return new XMLHttpRequest(); };
}
else if (ActiveXObject) {
    (function() {
        var vers = ["MSXML2.XMLHTTP.4.0", "MSXML2.XMLHTTP.3.0",

```

```
"MSXML2.XMLHTTP", "Microsoft.XMLHTTP"];
    for (var i = 0; i < vers.length; i++) {
        try {
            new ActiveXObject(vers[i]);
            AjaxLoader.__createXHR = function() { return new
ActiveXObject(vers[i]); };
            break;
        }
        catch (e) {
            // ignore
        }
    }
    })();
}
```

这可能也是造成偶尔出现 IE Crash 问题和页面载入速度有点慢的原因之一

mootools-1.2 学习笔记之 ajax 基本操作

在 mootools 中封装了如下三个类来进行 ajax 调用，它们是：**Request**，**Request.JSON**，**Request.HTML**。

分别用于普通的 XMLHttpRequest 请求，json 数据，html 页面数据。

注：另外本人在 ajaxhelper 基础上加入了对用户控件调用的支持，并使用 proxy 支持对 ashx 的 ajax 调用。

下面的代码分别演示了这三种方式：

```
<div id="result">Waiting for the request to happen.</div>
```

```
<div id="gallery"></div>
```

```
<input id="loadTxt" type="button" value="loadTxt" />
```

```
<input id="loadJson" type="button" value="loadJson" />
```

```
<input id="loadHtml" type="button" value="loadHtml" />
```

```
<script language="javascript" type="text/javascript">
```

```
/******Request******/
```

```
$( 'loadTxt' ).addEvent( 'click', function( e ) {  
    var req = new Request( { url: 'http://localhost:21087/data/data.txt',  
        method: 'get',  
        evalScripts: true,  
        onSuccess: function( responseText ) {  
            $( 'result' ).set( 'text', responseText );  
        },  
        //Our request will most likely succeed, but just in case, we'll add an  
        //onFailure method which will let the user know what happened.  
        onFailure: function() {  
            $( 'result' ).set( 'text', 'The request failed.' );  
        }  
    } ).send();  
});
```

```
/******Request.JSON******/
```

```
var images_path = 'http://localhost:21087/images/';  
var gallery = $( 'gallery' );  
var addImages = function( images ) {  
    images.each( function( image ) {  
        var el = new Element( 'div', { 'class': 'preview' } );  
        var name = new Element( 'h3', { 'html': image.name } ).inject( el );  
        var desc = new Element( 'span', { 'html': image.description } ).inject( name, 'after' );  
        var img = new Element( 'img', { 'src': images_path + image.src } ).inject( desc, 'after' );  
    } );  
};
```

```

    var footer = new Element('span').inject(img, 'after');

    if (image.views > 50 && image.views < 250) footer.set({ 'html': 'popular', 'class': 'popular' });
    else if (image.views > 250) footer.set({ 'html': 'SUPERpopular', 'class': 'SUPERpopular' });
    else footer.set({ 'html': 'normal', 'class': 'normal' });

    el.inject(gallery);
  });
};

$('loadJson').addEvent('click', function(e) {
  e.stop();
  var request = new Request.JSON({
    url: 'http://localhost:21087/data/data.json',
    method: 'get',
    onComplete: function(jsonObj) {
      addImages(jsonObj.previews);
    }
  }).send();
});

/*****Request.HTML*****/
$('loadHtml').addEvent('click', function(e) {
  var req = new Request.HTML({ url: 'http://localhost:21087/data/data.htm',
    method: 'get',
    onSuccess: function(html) {
      //Clear the text currently inside the results div.
      $('result').set('text', '');
      //Inject the new DOM elements into the results div.
      $('result').adopt(html);
    },
    //Our request will most likely succeed, but just in case, we'll add an
    //onFailure method which will let the user know what happened.
    onFailure: function() {
      $('result').set('text', 'The request failed.');
```

```
</body>
</html>
```

下面是其演示效果：

请求成功
这是一些HTML 内容. 很简洁，不是吗?

Blue Earth

A blue version of Earth. Click to see popular

FireFox

A chubby user guy. normal

HTML

JSON

因为在.net 开发下经常使用控件，特别是用户控件，所以我在 ajaxhelper 基础上做了一些修改，使

mootools 支持用户控件的调用，当然使用的机制还是在请求的 aspx 中使用 `base.LoadControl` 方法。

其代码如下：


```

private void Page_Load(object sender, EventArgs e)
{
    //AjaxTemplate 为请求的 ascx 控制
    if (base.Request.Params["AjaxTemplate"] != null)
    {
        try
        {
            this.AjaxCallBackForm.Controls.Add(
                base.LoadControl(base.Request.Params["AjaxTemplate"].ToLower().EndsWith(".ascx")
? ascxpath +
                base.Request.Params["AjaxTemplate"] : (ascxpath + base.Request.Params["AjaxTempla
te"] + ".ascx"))));
        }
        catch { }
    }
}

```

而其 ajaxhelper 的 js 代码修改如下：

```

var AjaxProxyUrl = new String("http://localhost:21087/ajax.aspx?");

var AjaxHelper =
{
    Updater : function ajaxTemplate, output, params, onComplete)
    {
        if (typeof output == 'string')
        {
            output = $(output);
        }

        var FormatContent = function(str)
        {
            var content = new String(str);
            var prefix = new String("<!--AjaxContent-->");
            return content.substring(content.indexOf(prefix, 0) + prefix.length, content.indexOf('</for

```

```

m>', 0));
    }

    new
    ({url: AjaxProxyUrl + params + '&AjaxTemplate=' + ajaxTemplate,
     method:'get',
     evalScripts:true,
     onSuccess: function(responseText)
     {
         if (output != null)
             { output.set('text', FormatContent(responseText)); }
         if (onComplete != null)
             { onComplete(FormatContent(responseText)) }
     },
     onFailure: function() {
         output.set('text', 'The request failed.');
```

另外我还做了一个 ajax 调用 ashx 的例子，基本上是 proxy 方式，使用 webclient 调用请求的 [ashx](#) 链接，其代码

如下所示：

```

public class proxy : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        base.Response.ContentType = "text/plain";
        string remoteurl = base.Request.QueryString["remoteurl"];
        string output = null;
        if (remoteurl != null)
        {
            try
            {
```

```

WebClient webclient = new WebClient();

webclient.Headers.Add("User-Agent",

                        "Mozilla/4.0 (compatible; MSIE 6.0
; Windows NT 5.1; .NET CLR 1.1.4322)");


StreamReader streamReader = new StreamReader(webclient.OpenRead(remoteurl), E
ncoding.UTF8);
output = streamReader.ReadToEnd().Trim();
streamReader.Close();

Response.Write(ToJavaScriptString(output));
}
catch (Exception ex)
{
    Response.Write(ToJavaScriptString(ex.Message.Trim()));
}
finally
{
    Response.End();
}
}

public static string ToJavaScriptString(string str)
{
    return str.Replace("\n", "").Replace("\r", "").Replace("@
\", @\"\\").

    Replace("'", @"\' ").Replace("\"", "\\\"").Replace
("/ ", @"/ ");

}
}

```

其 js 调用与其它 mootools 并无什么区别，主要是 url 中进行了相应的参数绑定(很偷懒):

```
$('#loadAshx').addEvent('click', function(e) {  
  
    new Request({url: 'http://localhost:21087/proxy.aspx?  
  
        remoteurl=http://localhost:21087/usercontrol/AshxSample.ashx?  
username=daizhj',  
  
        method:'get',  
        evalScripts:true,  
        onSuccess: function(responseText) {  
            $('result').set('text', responseText);  
        },  
        onFailure: function() {  
            $('result').set('text', 'The request failed.');        }  
    }).send();  
});
```

好了，今天的内容就到这里了。

```

1. var xmlHttpRequest;
2.     //创建 XMLHttpRequest 对象
3.     function createXMLHttpRequest() {
4.         if(window.ActiveXObject) {      // IE, //如果浏览器支持
window.ActiveXObject 对象
5.             xmlHttpRequest = new ActiveXObject("MSXML2.XMLHTTP.3.0"
);
6.             var MSXML = ['MSXML2.XMLHTTP.5.0', 'MSXML2.XMLHTTP.
4.0', 'MSXML2.XMLHTTP.3.0', 'MSXML2.XMLHTTP', 'Microsoft.XMLHTT
P'];
7.             try {
8.                 xmlHttpRequest= new ActiveXObject("Msxml2.XMLHTTP")
;
9.             }
10.            catch (e) {
11.                try{
12.                    xmlHttpRequest = new ActiveXObject("Microsoft.X
MLHTTP");
13.                }
14.                catch (e) {}
15.            }
16.
17.        }else if(window.XMLHttpRequest){      // Mozilla, Safari,
...
18.            xmlHttpRequest = new XMLHttpRequest();
19.        }
20.    }
21.
22. function getXmlSend(flag, id) {
23.     IDflag=flag;
24.     createXMLHttpRequest();
25.     var url="/.../xx.jsp?rand=" + Math.random() + "&id=
"+id+"&flag="+flag;
26.     xmlHttpRequest.open("GET", url, true);
27.     //xmlHttpRequest.setRequestHeader('Content-Type', 'appl
ication/x-www-form-urlencoded;charset=UTF-8');
28.     xmlHttpRequest.onreadystatechange = showResult; //异步
调用 showResult 方法
29.     xmlHttpRequest.send(null); // 开始发起浏览请
求, Mozilla 必须加 null
30. /*

```

```

31. 同步的做法是：屏蔽掉上面
    xmlHttpRequest.onreadystatechange = showResult;同时
    xmlHttpRequest.open("GET",url,false);
32. 接着直接在 http_request.send(null);下面获得结果
33. var returntxt=unescape(http_request.responseText);
34. */
35.      }
36.
37. function showResult() {
38.             if(xmlHttpRequest.readyState == 4) {
39.                     if(xmlHttpRequest.status == 200) {
40. alert(xmlHttpRequest.responseText);
41.             // 更新对应的 HTML 元素里面显示的内容
42. //do something
43. }
44.             }
45.     }

```

首先是 [jsp](#) 页面和脚本，为了方便写在一个里面

这是一个很常见的检测用户名是否存在的功能

这里用的是 struts

```

<%@ page contentType="text/html; charset=GBK" %>
<html>
<head>
<title>
ajax
</title>
</head>
<body bgcolor="#ffffff">
<h1>
<input name="username" type="text" maxlength="20" />
<input id="chk-name-btn" type="button" value="检测帐号"
onclick="testName(' <%=request.getContextPath()%>') " />
<div id="view_name"></div>
</h1>

```

```

</body>
</html>
<script language="javascript">
if (window.ActiveXObject && !window.XMLHttpRequest) {
window.XMLHttpRequest=function() {
return new
ActiveXObject((navigator.userAgent.toLowerCase().indexOf('msie 5') !=
-1) ? 'Microsoft.XMLHTTP' : 'Msxml2.XMLHTTP');
};
} //取得 XMLHttpRequest 对象
function testName(path) {
//path 是取得系统路径
var view_name=document.getElementById("view_name");
var req=new XMLHttpRequest();
if (req) {
req.onreadystatechange=function() {
if (req.readyState==4 && req.status==200) { //判断状态，4 是已发送，200
已完成
if(req.responseText==0) {
view_name.style.color='green';
view_name.innerHTML='该用户名可以正常使用';
} else if(req.responseText==1) {
view_name.style.color='red';
view_name.innerHTML='该用户名已经被使用';
} else {
view_name.style.color='red';
view_name.innerHTML='该用户名含有非法字符!';
}
}
}
req.open('POST', path+'ajax.do');//struts
//req.open('POST', path+'ajax.servlet');//servlet
//req.open('POST', path+'ajax.action');//webwork

req.setRequestHeader('Content-Type',
'application/x-www-form-urlencoded');
req.send(""); //发送参数如果有参数 req.send("username="+user_name); 用
request 取得
}
}
</script>

```

这个 jsp 页面并没有取得用户名，就是演示一下。还有<div>可以换 span，具体作用问美工吧。

接受 [ajax](#) 请求的 action。

```
import org.apache.struts.action.*;
import javax.servlet.http.*;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.PrintWriter;

/**
 * <p>Title:AjaxAction </p>
 */

public class AjaxAction extends Action {
    public ActionForward execute(ActionMapping mapping,
        ActionForm form,
        HttpServletRequest request,
        HttpServletResponse response)
        throws Exception {
        PrintWriter out = response.getWriter();
        out.print(1); // ajax 取得都是字符的输出。如果数据量大的话，还可以用 xml
        来发送和接受
        return null;
    }
}
```

struts-config. [xml](#)

```
<action type="test.whw.upload.AjaxAction" validate="false"
scope="request" path="/ajax"/>
```

如果是 servlet

web. [xml](#)

```
<servlet>
<servlet-name>AjaxServlet </servlet-name>
<servlet-class>servlet.AjaxServlet </servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>AjaxServlet </servlet-name>
```



```
<url-pattern>/AjaxServlet .servlet</url-pattern>
</servlet-mapping>
```

AjaxServlet.java

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class AjaxServlet extends HttpServlet {
    Logger log = Logger.getLogger(this.getClass());
    public void doGet(HttpServletRequest request, HttpServletResponse
    response)
    throws ServletException, IOException {
        response.setContentType("text/xml; charset=GBK");
        PrintWriter out = response.getWriter();
        out.print(2);
    }
}

//Process the HTTP Post request
public void doPost(HttpServletRequest request, HttpServletResponse
    response)
    throws ServletException, IOException {
    doGet(request, response);
}

//Process the HTTP Put request
public void doPut(HttpServletRequest request, HttpServletResponse
    response)
    throws ServletException, IOException {
}

//Process the HTTP Delete request
public void delete(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, IOException {
}

//Clean up resources
public void destroy() {
}
```

```
}
```

```
-----  
-----
```

如果是 webwork

xwork. [xml](#)


```
<action name="ajax" class="com.whw.upload.action.webwork.AjaxAction"  
method="ajax"/>
```

AjaxAction.java

```
import java.io.PrintWriter;  
  
public class AjaxAlbumAction extends ActionSupport implements Action{  
  
    public void ajax() throws IOException {  
        PrintWriter pw = ServletActionContext.getResponse().getWriter();  
        ServletActionContext.getResponse().setContentType("text/html;charset=  
GBK");  
        pw.print(1);  
        pw.close();  
    }  
  
}
```

本例子在 winXPsp2、JB9、Eclipse3, j2sdk1.4.1 、Tomcat5、Tomcat4.1 都能运行

一、发送请求的 html 文件

Html 代码 

1. <html><head><title>ajax+jsp 域名查询</title>

```

2. <meta http-equiv="Content-Type" content="text/html; charset=gb2
   312">
3. <script language="javascript">
4.
5.
6.     var XMLHttpRequest;
7.         //创建 XMLHttpRequest 对象
8.         function createXMLHttpRequest() {
9.
10.             if(window.XMLHttpRequest) { //Mozilla 浏览器
11.                 XMLHttpRequest = new XMLHttpRequest();
12.             }
13.             else if (window.ActiveXObject) { // IE 浏览器
14.                 try {
15.                     XMLHttpRequest = new ActiveXObject("Msxml2.XMLHTTP"
16. );
17.                 } catch (e) {
18.                     try {
19.                         XMLHttpRequest = new ActiveXObject("Microsoft.X
20. MLHTTP");
21.                     } catch (e) {}
22.                 }
23.             }
24.         //发送请求函数
25.         function sendRequest() {
26.             document.getElementById("comments").innerHTML ="
27. 正在查询,请您稍等.....";
28.             createXMLHttpRequest();
29.             var name=document.getElementById("words").value;
30.
31.             var url = "domainQuery1.jsp?words="+name;
32.             XMLHttpRequest.open("GET", url, true);
33.             XMLHttpRequest.onreadystatechange = processResponse;//指定
34. 响应函数
35.             XMLHttpRequest.send(null); // 发送请求
36.         }
37.         // 处理返回信息函数
38.         function processResponse() {
39.             if (XMLHttpRequest.readyState == 4) { // 判断对象状态
40.                 if (XMLHttpRequest.status == 200) { // 信息已经成功返
41. 回, 开始处理信息
42.                     Display();
43.                 } else { //页面不正常

```


```

39.             window.alert("您所请求的页面有异常。");
40.         }
41.     }
42. }
43. function Display() {
44.     var msg=XMLHttpRequest.responseText;
45.     document.getElementById('comments').value=msg;
46. }
47. </script></head>
48. <body bgColor=#ccccca3>
49.
50. 请输入域名名称:
    <input type="text" value="cctv.com" id="words" name="words" size="69">
51.
52. <input type="submit" value="查询
    " id=Submit1 name=Submit1 onClick="sendRequest()" ">
53.
54. <TEXTAREA id="comments" name="comments" readonly rows=15 cols=7
    6></TEXTAREA>
55.
56. </body></html>

```

二、处理 ajax 请求的 jsp 文件:

domainQuery1.jsp

Java 代码 

```

1. <%@ page import="java.io.*" %>
2. <%@ page import="java.util.*" %>
3. <%@ page import="java.net.HttpURLConnection" %>
4. <%@ page import="java.net.URL" %>
5. <%@ page import="org.jdom.Document" %>
6. <%@ page import="org.jdom.Element" %>
7. <%@ page import="org.jdom.input.*" %>
8.
9. <%@ page contentType="text/html; charset=gb2312" %>
10. <%
11.
12.

```

```

13. //设置输出信息的格式及字符集
14.     response.setContentType("text/html; charset=gb2312");
15.     response.setHeader("Cache-Control", "no-cache");
16.     String param =request.getParameter("words");
17.     String domainmsg=null;
18.     URL url = null;
19.     BufferedReader in = null;
20.     InputStreamReader isr = null;
21.     InputStream is = null;
22.     HttpURLConnection huc = null;
23.     try
24.     {
25.         url = new URL("http://now.net.cn/domain/domaincheck
.php?query="+param);
26.         huc = (HttpURLConnection)url.openConnection();
27.
28.         is = huc.getInputStream();
29.         isr = new InputStreamReader(is);
30.         in = new BufferedReader(isr);
31.         String line = null;
32.         StringBuffer s = new StringBuffer(333);
33.         while((line = in.readLine()) != null) {
34.             line=line.trim();
35.             s.append(line);
36.         }
37.         domainmsg=s.toString();
38.         SAXBuilder builder = new SAXBuilder();
39.         Document doc = null;
40.         Reader in1= new StringReader(domainmsg);
41.         doc = builder.build(in1);
42.         Element root = doc.getRootElement();
43.
44.         String ss=root.getChild("result").getChildText("msg
");
45.         out.print(ss);
46.     }catch(Exception e){
47.         e.printStackTrace();
48.     }
49.     finally
50.     {
51.         try
52.         {
53.             huc.disconnect();
54.             is.close();

```

```

55.            isr.close();
56.            in.close();
57.        }
58.        catch(Exception e)
59.        {
60.            e.printStackTrace();
61.        }
62.    }
63.    out.close();
64.%>

```

搞了这么多年 DELPHI，看着形势不搞 JAVA 就落伍了。最近公司搞的全是 JAVA 项目。我就狼吞虎咽，一个月把 STRUTS,SPRING,JSF,AJAX 等东西给吃下了，有的现在还在打嗝呢。

好在现在发现 AJAX 是最好学的。

本例子参考了《征服 AJAX WEB2.0 开发技术详解》

这个例子用途：判断输入的身份证号码是否正确，这里能够检查出长度错误，日期错误，验证码错误。

先做一个普通网页 ajaxtest.jsp

```

<%@ page contentType="text/html; charset=gb2312" language="java" import="java.sql.*"
errorPage="" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<jsp:useBean id="bb" scope="page" class="lh.LHDBconnection"/>
<title>无标题文档</title>
</head>

<body>

<form name="form1" method="post" action="">
<table width="546" border="1" bordercolor="#6699CC" bordercolordark="#003399">
<tr>
<td>身份证号码</td>
<td><input name="idno" type="text" id="idno">
<input name="Submit3" type="button" value=" 检 查 是 否 正 确 "
onClick="javascript:checkidno();">
</td>
</tr>
</table>

```

```
</form>
```

```
<script type="text/javascript" language="javascript">  
<!--
```

```
function checkidno(){
```

```
//本来一句话就可以，这里啰嗦一堆都是为了解决浏览器兼容性问题.
```

```
var xmlhttp;
```

```
try{
```

```
    xmlhttp=new ActiveXObject('Msxml2.XMLHTTP');
```

```
} catch(e){
```

```
    try{
```

```
        xmlhttp=new ActiveXObject('Microsoft.XMLHTTP');
```

```
    } catch(e){
```

```
        try{
```

```
            xmlhttp=new XMLHttpRequest();
```

```
        }catch(e){ }
```

```
    }
```

```
}
```

```
var rrr="";
```

```
var idno=document.form1.idno.value;
```

```
//这里不用刷新整个页面，就可以直接调用jsp文件了，太方便了
```

```
xmlhttp.open("get","../comm/checkidno.jsp?_idno="+idno,true);
```

```
xmlhttp.onreadystatechange=function(){
```

```
    if (xmlhttp.readyState==4)
```

```
        //xmlhttp.status==404 代表 没有发现该文件
```

```
    if (xmlhttp.status==200)
```

```
    {
```

```
        //alert(xmlhttp.status);
```

```
        rrr=xmlhttp.responseText;
```

```
        //if(rrr=="ok") {alert("ok") ; else alert(rrr);
```

```
        //alert(rrr);
```

```
    } else
```

```
    {
```

```
        alert("发生错误: "+xmlhttp.status);
```

```
    }
```

```
}
```

```
xmlhttp.send(null);
```

```

alert(rrr);

return false;
}

//-->
</script>
</body>
</html>

```

然后是一个被调用的 JSP 文件 checkidno.jsp
 这个验证有点复杂，你可以简化一下，当然里面也可以进行数据库操作

```

<% @      page      contentType="text/html;      charset=gb2312"      language="java"
import="java.sql.*,java.util.*" errorPage="" %>
<%

String errorMessage="";
String idno=request.getParameter("_idno");
System.out.println("hello");
//out.write("hello");

Calendar now=GregorianCalendar.getInstance();
int iyear=now.get(Calendar.YEAR);
String year=Integer.toString(iyear);
String bird="";
String birthday="";

if (idno.length()!=15&&idno.length()!=18) { errorMessage=errorMessage+"身份证号长度错误！";}
else
{
    String addr=idno.substring(0,6);
    String last="";

    if (idno.length()==15) {
        bird=idno.substring(6,12);
        birthday="19"+bird.substring(0,2)+"-"+bird.substring(2,4)+"-"+bird.substring(4,6);
        last=idno.substring(12,15);

        if      ((Integer.parseInt("19"+bird.substring(0,2))<=1900)
|| (Integer.parseInt("19"+bird.substring(0,2))>=2000) ) errorMessage=errorMessage+"年份错误！";

        if      ((Integer.parseInt(bird.substring(2,4))<1)

```



```

|| (Integer.parseInt(bird.substring(2,4))>12) ) errorMessage=errorMessage+"月份部分错误! ";
                                if ((Integer.parseInt(bird.substring(4,6))<1)
|| (Integer.parseInt(bird.substring(4,6))>31) ) errorMessage=errorMessage+"日期部分错误! ";
    }

    if (idno.length()==18) {
        bird=idno.substring(6,14);
        birthday=bird.substring(0,4)+"-"+bird.substring(4,6)+"-"+bird.substring(6,8);
        last=idno.substring(14,17);
        //System.out.println("birthday==" +bird.substring(4,6));

                                if ((Integer.parseInt(bird.substring(0,4))<=1900)
|| (Integer.parseInt(bird.substring(0,4))>=iyear) ) errorMessage=errorMessage+"年份错误! ";
                                if ((Integer.parseInt(bird.substring(4,6))<1)
|| (Integer.parseInt(bird.substring(4,6))>12) ) errorMessage=errorMessage+"月份部分错误! ";
                                if ((Integer.parseInt(bird.substring(6,8))<1)
|| (Integer.parseInt(bird.substring(6,8))>31) ) errorMessage=errorMessage+"日期部分错误! ";

    }

    //ResultSet sqlrst=bb.executeQuery("select * from hr_idnosource where
ccodeid='"+idno.substring(0,5)+"'");

    if (errorMessage.equals("")){
        try {
            // 如果输入日期不是 8 位的,判定为 false.
            // || !birthday.matches("[0-9]{8}")
            if (null == birthday || "".equals(birthday) ) {
                errorMessage=errorMessage+"日期非法! ";
            }

            //1978-02-06
            System.out.println("test:="+birthday+" "+birthday.substring(8, 10));
            iyear = Integer.parseInt(birthday.substring(0, 4));
            int month = Integer.parseInt(birthday.substring(5, 7)) - 1;
            int day = Integer.parseInt(birthday.substring(8,10));
            Calendar calendar = GregorianCalendar.getInstance();
            // 当 Calendar 处于 non-lenient 模式时, 如果其日历字段中存在任何不一致
            性, 它都会抛出一个异常。
            calendar.setLenient(false);
            calendar.set(Calendar.YEAR, iyear);
            calendar.set(Calendar.MONTH, month);

```

```

        calendar.set(Calendar.DATE, day);
        // 如果日期错误,执行该语句,必定抛出异常.
        calendar.get(Calendar.YEAR);
    } catch (IllegalArgumentException e) {
        errorMessage=errorMessage+"非法日期! ";
    }
}

```

//验证码

```

if (idno.length()==18){

    String newidno=addr+bird+last ;
    int sum=
        Integer.parseInt(newidno.substring(0,1))*7
        +Integer.parseInt(newidno.substring(1,2))*9
        +Integer.parseInt(newidno.substring(2,3))*10
        +Integer.parseInt(newidno.substring(3,4))*5
        +Integer.parseInt(newidno.substring(4,5))*8
        +Integer.parseInt(newidno.substring(5,6))*4
        +Integer.parseInt(newidno.substring(6,7))*2
        +Integer.parseInt(newidno.substring(7,8))*1
        +Integer.parseInt(newidno.substring(8,9))*6
        +Integer.parseInt(newidno.substring(9,10))*3
        +Integer.parseInt(newidno.substring(10,11))*7
        +Integer.parseInt(newidno.substring(11,12))*9
        +Integer.parseInt(newidno.substring(12,13))*10
        +Integer.parseInt(newidno.substring(13,14))*5
        +Integer.parseInt(newidno.substring(14,15))*8
        +Integer.parseInt(newidno.substring(15,16))*4
        +Integer.parseInt(newidno.substring(16,17))*2 ;

    sum=sum % 11;

    String r="";
    if (sum==0)   r="1" ;
    if (sum==1)   r="0";
    if (sum==2)   r="X";
    if (sum==3)   r="9";
    if (sum==4)   r="8";
    if (sum==5)   r="7";
    if (sum==6)   r="6";
    if (sum==7)   r="5";
    if (sum==8)   r="4";
    if (sum==9)   r="3";
    if (sum==10)  r="2";
}
}

```

```

        //r:=newidno+r ;
        if (idno.substring(17,18).equals(r)){ }else {errorMessage=errorMessage+"验证码
错误! ";}

    }

}

}

if (errorMessage.equals("")){errorMessage="ok";}
//out.write(errorMessage);
//这个就是返回值
out.print(errorMessage);

%>

```

运行 ajaxtest.jsp, 点<检查是否正确>,系统就可以自动调用 checkidno.jsp, 然后返回判断结果。

可能有点 BUG, 因时间关系, 还没有解决, 欢迎大家批评指正。

注意, 这里并没有刷新 ajaxtest.jsp 这个整个页面, 这就是 AJAX 技术的价值所在。

AJAX 就是 JAVASCRIPT 的升级版, 所以很好学。

mootools 使用心得之 ajax 类 (2007-08-09 13:44:42)

标签: [it/科技](#) [mootools](#) [ajax](#) [javascript](#)

分类: [javascript](#)

mootools 是一个非常优秀的 javascript 的库; 有些地方跟 prototype 颇有相似 (指按面向对象做 js); 以前我也用过 prototype; 但觉得它太繁琐; 后来仿照 prototype 的思想直接用 js 代码编程更简洁; 但我现在决定以后用 mootools 库; 原因是 1: mootools 和 prototype 相比, mootools 使用更方便, 代码更精简; 2: mootools 展现的效果更接近于 flash (以前我觉得这是 js 不如 flash 重要的缺点);

我用的是 mootools1.11 版本(下载地址: <http://www.mootools.net/download>)。

ajax 类的使用:

```
var ajax = new Ajax(url, options);
```

```
ajax.request();
```

url 是提交到后台处理的路径。

options 有 2 个经常使用的参数: { onComplete: handleFun, data:postArgs }

```
function handleFun(req){
```

```
    alert(req);
```

```
}
```

onComplete 参数指向处理返回的文本函数; data (老版本用 postBody 名) 是按 doPost 方式提交的字符串。

下面用 ajax 的 2 种方法提交 form 表单（相当于点击 form 中的 submit 按钮）：

```
<form action="test.php" id="form1">
<input name="user_name" />
<input name="user_id" />
<input type="button" onclick="ajaxSubmit();" />
</form>
```

方法 1:

```
function ajaxSubmit(){
    var postArgs = $('form1').toQueryString();
    new Ajax('test.php',
        {data:postArgs,onComplete: handleFun}).request();
}
```

方法 2:

```
function ajaxSubmit(){
    var postArgs = $('form1').toQueryString();
    $('form1').send({onComplete:showResponse,data:postArgs});
}
```

说明:

\$()函数相当于 js 的 document.getElementById();
toQueryString()函数是得到 form 中提交元素的字符串。

注意这两个例子区别:

用例 1 时无需在 form 中指定 action,而必须指定 Ajax 中的 url
例 2 则必须指定 form 中 action, 而无需指定 Ajax 中的 url
另外在 test.php 中接受参数一律用 \$_POST[""] 接受即可