jQuery 教程

jQuery 是一个 JavaScript 库。

jQuery 极大地简化了 JavaScript 编程。

jQuery 很容易学习。

本章节的每一篇都包含了在线实例

通过本站的在线编辑器，你可以在线运行修改后的代码，并查看运行结果。

实例

```
$(document).ready(function(){ $("p").click(function(){ $(this).hide(); }); });
```

尝试一下 »

点击 "尝试一下" 按钮查看运行结果。

jQuery 实例

在本教程中，您将通过教程以及许多在线实例，学到如何通过使用 jQuery 应用 JavaScript 效果。

- jQuery 在线实例

---

jQuery 测验

在菜鸟教程上测试你的 jQuery 技能!

- jQuery 测验

jQuery 参考手册

在本教程中，你将学习到更多的jquery的使用方法。

- jQuery 参考手册
- jQuery API 手册

jQuery 简介

---

jQuery 库可以通过一行简单的标记被添加到网页中。

您需要具备的基础知识

在您开始学习 jQuery 之前，您应该对以下知识有基本的了解：

- HTML
- CSS
- JavaScript

如果您需要首先学习这些科目，请在我们的 首页 查找这些教程。

什么是 jQuery？

jQuery是一个JavaScript函数库。

jQuery是一个轻量级的"写的少，做的多"的JavaScript库。

jQuery库包含以下功能：

- HTML 元素选取
- HTML 元素操作
- CSS 操作
- HTML 事件函数
- JavaScript 特效和动画
- HTML DOM 遍历和修改
- AJAX
- Utilities

提示： 除此之外，Jquery还提供了大量的插件。

为什么使用 jQuery？

目前网络上有大量开源的 JS 框架, 但是 jQuery 是目前最流行的 JS 框架，而且提供了大量的扩展。

很多大公司都在使用 jQuery， 例如:

- Google
- Microsoft
- IBM
- Netflix

jQuery是否适用于所有浏览器？

jQuery 团体知道JS在不同浏览器中存在着大量的兼容性问题，目前jQuery兼容于所有主流浏览器, 包括Internet Explorer 6!

jQuery 安装

网页中添加 jQuery

可以通过多种方法在网页中添加 jQuery。 您可以使用以下方法：

- 从 jquery.com 下载 jQuery 库
- 从 CDN 中载入 jQuery, 如从 Google 中加载 jQuery

下载 jQuery

有两个版本的 jQuery 可供下载：

- Production version - 用于实际的网站中，已被精简和压缩。
- Development version - 用于测试和开发（未压缩，是可读的代码）

以上两个版本都可以从 jquery.com 中下载。

jQuery 库是一个 JavaScript 文件，您可以使用 HTML 的 <script> 标签引用它：

<head> <script src="jquery-1.10.2.min.js"></script> </head>

提示： 将下载的文件放在网页的同一目录下，就可以使用jQuery。

您是否很疑惑为什么我们没有在 <script> 标签中使用 type="text/javascript" ？

在 HTML5 中，不必那样做了。JavaScript 是 HTML5 以及所有现代浏览器中的默认脚本语言！

替代方案

如果您不希望下载并存放 jQuery，那么也可以通过 CDN（内容分发网络） 引用它。

BootCDN、百度、又拍云、新浪、谷歌和微软的服务器都存有 jQuery 。

如果你的站点用户是国内的，建议使用百度、又拍云、新浪等国内CDN地址，如果你站点用户是国外的可以使用谷歌和微软。

注：本站实例均采用菜鸟教程 CDN 库。

如需从 BootCDN、又拍云、新浪、谷歌或微软引用 jQuery，请使用以下代码之一：

BootCDN:

<head> <script src="https://cdn.bootcss.com/jquery/1.10.2/jquery.min.js"> </script> </head>

尝试一下 »

百度 CDN:

<head> <script src="https://apps.bdimg.com/libs/jquery/2.1.4/jquery.min.js"> </script> </head>

尝试一下 »

又拍云 CDN:

<head> <script src="http://upcdn.b0.upaiyun.com/libs/jquery/jquery-2.0.2.min.js"> </script> </head>

尝试一下 »

新浪 CDN:

<head> <script src="http://lib.sinaapp.com/js/jquery/2.0.2/jquery-2.0.2.min.js"> </script> </head>

尝试一下 »

Google CDN:

<head> <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js"> </script> </head>

尝试一下 »

不大推荐使用Google CDN来获取版本，因为Google产品在中国很不稳定。

Microsoft CDN:

<head> <script src="https://ajax.aspnetcdn.com/ajax/jquery/jquery-1.9.0.min.js"></script> </head>

尝试一下 »

使用 BootCDN、百度、又拍云、新浪、谷歌或微软的 jQuery，有一个很大的优势：

许多用户在访问其他站点时，已经从百度、又拍云、新浪、谷歌或微软加载过 jQuery。所以结果是，当他们访问您的站点时，会从缓存中加载 jQuery，这样可以减少加载时间。同时，大多数 CDN 都可以确保当用户向其请求文件时，会从离用户最近的服务器上返回响应，这样也可以提高加载速度。

jQuery 语法

通过 jQuery，您可以选取（查询，query） HTML 元素，并对它们执行"操作"（actions）。

---

jQuery 语法

jQuery 语法是通过选取 HTML 元素，并对选取的元素执行某些操作。

基础语法： **$(*selector*).*action*()**

- 美元符号定义 jQuery
- 选择符（selector）"查询"和"查找" HTML 元素
- jQuery 的 action() 执行对元素的操作

实例:

- $(this).hide() - 隐藏当前元素
- $("p").hide() - 隐藏所有 <p> 元素
- $("p.test").hide() - 隐藏所有 class="test" 的 <p> 元素
- $("#test").hide() - 隐藏所有 id="test" 的元素

你对 CSS 选择器熟悉吗？

jQuery 使用的语法是 XPath 与 CSS 选择器语法的组合。在本教程接下来的章节，您将学习到更多有关选择器的语法。

文档就绪事件

您也许已经注意到在我们的实例中的所有 jQuery 函数位于一个 document ready 函数中：

$(document).ready(function(){ // 开始写 jQuery 代码... });

这是为了防止文档在完全加载（就绪）之前运行 jQuery 代码，即在 DOM 加载完成后才可以对 DOM 进行操作。

如果在文档没有完全加载之前就运行函数，操作可能失败。下面是两个具体的例子：

- 试图隐藏一个不存在的元素
- 获得未完全加载的图像的大小

提示：简洁写法（与以上写法效果相同）：

$(function(){ // 开始写 jQuery 代码... });

以上两种方式你可以选择你喜欢的方式实现文档就绪后执行 jQuery 方法。

jQuery 选择器

---

jQuery 选择器允许您对 HTML 元素组或单个元素进行操作。

---

jQuery 选择器

jQuery 选择器允许您对 HTML 元素组或单个元素进行操作。

jQuery 选择器基于元素的 id、类、类型、属性、属性值等"查找"（或选择）HTML 元素。 它基于已经存在的 CSS 选择器，除此之外，它还有一些自定义的选择器。

jQuery 中所有选择器都以美元符号开头：$()。

元素选择器

jQuery 元素选择器基于元素名选取元素。

在页面中选取所有 <p> 元素:

$("p")

实例

用户点击按钮后，所有 <p> 元素都隐藏：

实例

```
$(document).ready(function(){ $("button").click(function(){ $("p").hide(); }); });
```

尝试一下 »

#id 选择器

jQuery #id 选择器通过 HTML 元素的 id 属性选取指定的元素。

页面中元素的 id 应该是唯一的，所以您要在页面中选取唯一的元素需要通过 #id 选择器。

通过 id 选取元素语法如下：

$("#test")

实例

当用户点击按钮后，有 id="test" 属性的元素将被隐藏：

实例

```
$(document).ready(function(){ $("button").click(function(){ $("#test").hide(); }); });
```

尝试一下 »

.class 选择器

jQuery 类选择器可以通过指定的 class 查找元素。

语法如下：

$(".test")

实例

用户点击按钮后所有带有 class="test" 属性的元素都隐藏：

实例

```
$(document).ready(function(){ $("button").click(function(){ $(".test").hide(); }); });
```

尝试一下 »

更多实例

语法	描述	实例
$("*")	选取所有元素	在线实例
$(this)	选取当前 HTML 元素	在线实例
$("p.intro")	选取 class 为 intro 的 <p> 元素	在线实例
$("p:first")	选取第一个 <p> 元素	在线实例
$("ul li:first")	选取第一个 元素的第一个 元素	在线实例
$("ul li:first-child")	选取每个 元素的第一个 元素	在线实例
$("[href]")	选取带有 href 属性的元素	在线实例
$("a[target='_blank']")	选取所有 target 属性值等于 "_blank" 的 <a> 元素	在线实例
$("a[target!='_blank']")	选取所有 target 属性值不等于 "_blank" 的 <a> 元素	在线实例
$(":button")	选取所有 type="button" 的 <input> 元素 和 <button> 元素	在线实例
$("tr:even")	选取偶数位置的 <tr> 元素	在线实例
$("tr:odd")	选取奇数位置的 <tr> 元素	在线实例

独立文件中使用 jQuery 函数

如果您的网站包含许多页面，并且您希望您的 jQuery 函数易于维护，那么请把您的 jQuery 函数放到独立的 .js 文件中。

当我们在教程中演示 jQuery 时，会将函数直接添加到 <head> 部分中。不过，把它们放到一个单独的文件中会更好，就像这样（通过 src 属性来引用文件）：

实例

```
<head> <script src="http://cdn.static.runoob.com/libs/jquery/1.10.2/jquery.min.js"> </script> <script
src="my_jquery_functions.js"></script> </head>
```

jQuery 事件

---

jQuery 是为事件处理特别设计的。

什么是事件？

页面对不同访问者的响应叫做事件。

事件处理程序指的是当 HTML 中发生某些事件时所调用的方法。

实例：

- 在元素上移动鼠标。
- 选取单选按钮
- 点击元素

在事件中经常使用术语"触发"（或"激发"）例如： "当您按下按键时触发 keypress 事件"。

常见 DOM 事件：

鼠标事件	键盘事件	表单事件	文档/窗口事件
click	keypress	submit	load
dblclick	keydown	change	resize
mouseenter	keyup	focus	scroll
mouseleave		blur	unload

---

jQuery 事件方法语法

在 jQuery 中，大多数 DOM 事件都有一个等效的 jQuery 方法。

页面中指定一个点击事件：

```
$("p").click();
```

下一步是定义什么时间触发事件。您可以通过一个事件函数实现：

```
$("p").click(function(){ // 动作触发后执行的代码!! });
```

常用的 jQuery 事件方法

$(document).ready()

$(document).ready() 方法允许我们在文档完全加载完后执行函数。该事件方法在 jQuery 语法 章节中已经提到过。

click()

click() 方法是当按钮点击事件被触发时会调用一个函数。

该函数在用户点击 HTML 元素时执行。

在下面的实例中，当点击事件在某个 <p> 元素上触发时，隐藏当前的 <p> 元素：

实例

```
$("p").click(function(){ $(this).hide(); });
```

尝试一下 »

dblclick()

当双击元素时，会发生 dblclick 事件。

dblclick() 方法触发 dblclick 事件，或规定当发生 dblclick 事件时运行的函数：

实例

```
$("p").dblclick(function(){ $(this).hide(); });
```

尝试一下 »

mouseenter()

当鼠标指针穿过元素时，会发生 mouseenter 事件。

mouseenter() 方法触发 mouseenter 事件，或规定当发生 mouseenter 事件时运行的函数：

实例

```
$("#p1").mouseenter(function(){ alert('您的鼠标移到了 id="p1" 的元素上!'); });
```

尝试一下 »

mouseleave()

当鼠标指针离开元素时，会发生 mouseleave 事件。

mouseleave() 方法触发 mouseleave 事件，或规定当发生 mouseleave 事件时运行的函数：

实例

```
$("#p1").mouseleave(function(){ alert("再见，您的鼠标离开了该段落。"); });
```

尝试一下 »

mousedown()

当鼠标指针移动到元素上方，并按下鼠标按键时，会发生 mousedown 事件。

mousedown() 方法触发 mousedown 事件，或规定当发生 mousedown 事件时运行的函数：

实例

```
$("#p1").mousedown(function(){ alert("鼠标在该段落上按下！"); });
```

尝试一下 »

mouseup()

当在元素上松开鼠标按钮时，会发生 mouseup 事件。

mouseup() 方法触发 mouseup 事件，或规定当发生 mouseup 事件时运行的函数：

实例

```
$("#p1").mouseup(function(){ alert("鼠标在段落上松开。"); });
```

尝试一下 »

hover()

hover()方法用于模拟光标悬停事件。

当鼠标移动到元素上时，会触发指定的第一个函数(mouseenter);当鼠标移出这个元素时，会触发指定的第二个函数(mouseleave)。

实例

```
$("#p1").hover( function(){ alert("你进入了 p1!"); }, function(){ alert("拜拜! 现在你离开了 p1!"); } );
```

尝试一下 »

focus()

当元素获得焦点时，发生 focus 事件。

当通过鼠标点击选中元素或通过 tab 键定位到元素时，该元素就会获得焦点。

focus() 方法触发 focus 事件，或规定当发生 focus 事件时运行的函数：

实例

```
$("input").focus(function(){ $(this).css("background-color","#cccccc"); });
```

尝试一下 »

blur()

当元素失去焦点时，发生 blur 事件。

blur() 方法触发 blur 事件，或规定当发生 blur 事件时运行的函数：

实例

```
$("input").blur(function(){ $(this).css("background-color","#ffffff"); });
```

尝试一下 »

---

jQuery 效果- 隐藏和显示

隐藏、显示、切换，滑动，淡入淡出，以及动画，哇哦！

点击 显示/隐藏面板

实例

jQuery hide()
简单的jQuery hide()方法演示。

jQuery hide()
另一个hide()实例。演示如何隐藏文本。

---

jQuery hide() 和 show()

通过 jQuery，您可以使用 hide() 和 show() 方法来隐藏和显示 HTML 元素：

实例

```
$("#hide").click(function(){ $("p").hide(); }); $("#show").click(function(){ $("p").show(); });
```

尝试一下 »

语法：

$(*selector*).hide(*speed,callback*);

$(*selector*).show(*speed,callback*);

可选的 speed 参数规定隐藏/显示的速度，可以取以下值："slow"、"fast" 或毫秒。

可选的 callback 参数是隐藏或显示完成后所执行的函数名称。

下面的例子演示了带有 speed 参数的 hide() 方法：

实例

```
$("button").click(function(){ $("p").hide(1000); });
```

尝试一下 »

---

jQuery toggle()

通过 jQuery，您可以使用 toggle() 方法来切换 hide() 和 show() 方法。

显示被隐藏的元素，并隐藏已显示的元素：

实例

```
$("button").click(function(){ $("p").toggle(); });
```

尝试一下 »

语法：

$(*selector*).toggle(*speed,callback*);

可选的 speed 参数规定隐藏/显示的速度，可以取以下值："slow"、"fast" 或毫秒。

可选的 callback 参数是隐藏或显示完成后所执行的函数名称。

jQuery 效果 - 淡入淡出

通过 jQuery，您可以实现元素的淡入淡出效果。

点击展示 **淡入/淡出** 面板

实例

jQuery fadeIn()
演示 jQuery fadeIn() 方法。

jQuery fadeOut()
演示 jQuery fadeOut() 方法。

jQuery fadeToggle()
演示 jQuery fadeToggle() 方法。

jQuery fadeTo()
演示 jQuery fadeTo() 方法。

jQuery Fading 方法

通过 jQuery，您可以实现元素的淡入淡出效果。

jQuery 拥有下面四种 fade 方法：

- fadeIn()
- fadeOut()
- fadeToggle()
- fadeTo()

jQuery fadeIn() 方法

jQuery fadeIn() 用于淡入已隐藏的元素。

语法：

$(*selector*).fadeIn(*speed,callback*);

可选的 speed 参数规定效果的时长。它可以取以下值："slow"、"fast" 或毫秒。.

可选的 callback 参数是 fading 完成后所执行的函数名称。

下面的例子演示了带有不同参数的 fadeIn() 方法：

实例

```
$("button").click(function(){ $("#div1").fadeIn(); $("#div2").fadeIn("slow"); $("#div3").fadeIn(3000); });
```

尝试一下 »

jQuery fadeOut() 方法

jQuery fadeOut() 方法用于淡出可见元素。

语法：

$(*selector*).fadeOut(*speed,callback*);

可选的 speed 参数规定效果的时长。它可以取以下值："slow"、"fast" 或毫秒。

可选的 callback 参数是 fading 完成后所执行的函数名称。

下面的例子演示了带有不同参数的 fadeOut() 方法：

实例

```
$("button").click(function(){ $("#div1").fadeOut(); $("#div2").fadeOut("slow"); $("#div3").fadeOut(3000); });
```

尝试一下 »

---

jQuery fadeToggle() 方法

jQuery fadeToggle() 方法可以在 fadeIn() 与 fadeOut() 方法之间进行切换。

如果元素已淡出，则 fadeToggle() 会向元素添加淡入效果。

如果元素已淡入，则 fadeToggle() 会向元素添加淡出效果。

语法：

$(*selector*).fadeToggle(*speed,callback*);

可选的 speed 参数规定效果的时长。它可以取以下值："slow"、"fast" 或毫秒。

可选的 callback 参数是 fading 完成后所执行的函数名称。

下面的例子演示了带有不同参数的 fadeToggle() 方法：

实例

```
$("button").click(function(){ $("#div1").fadeToggle(); $("#div2").fadeToggle("slow"); $("#div3").fadeToggle(3000); });
```

尝试一下 »

---

jQuery fadeTo() 方法

jQuery fadeTo() 方法允许渐变为给定的不透明度（值介于 0 与 1 之间）。

语法：

$(*selector*).fadeTo(*speed,opacity,callback*);

必需的 speed 参数规定效果的时长。它可以取以下值："slow"、"fast" 或毫秒。

fadeTo() 方法中必需的 opacity 参数将淡入淡出效果设置为给定的不透明度（值介于 0 与 1 之间）。

可选的 callback 参数是该函数完成后所执行的函数名称。

下面的例子演示了带有不同参数的 fadeTo() 方法：

实例

```
$("button").click(function(){ $("#div1").fadeTo("slow",0.15); $("#div2").fadeTo("slow",0.4); $("#div3").fadeTo("slow",0.7); });
```

尝试一下 »

jQuery 效果 - 滑动

---

jQuery 滑动方法可使元素上下滑动。

点击这里，隐藏/显示面板

实例

jQuery slideDown()
演示 jQuery slideDown() 方法。

jQuery slideUp()
演示 jQuery slideUp() 方法。

jQuery slideToggle()
演示 jQuery slideToggle() 方法。

---

jQuery 滑动方法

通过 jQuery，您可以在元素上创建滑动效果。

jQuery 拥有以下滑动方法：

- slideDown()
- slideUp()
- slideToggle()

---

jQuery slideDown() 方法

jQuery slideDown() 方法用于向下滑动元素。

语法：

$(*selector*).slideDown(*speed,callback*);

可选的 speed 参数规定效果的时长。它可以取以下值："slow"、"fast" 或毫秒。

可选的 callback 参数是滑动完成后所执行的函数名称。

下面的例子演示了 slideDown() 方法：

实例

```
$("#flip").click(function(){ $("#panel").slideDown(); });
```

尝试一下 »

---

jQuery slideUp() 方法

jQuery slideUp() 方法用于向上滑动元素。

语法：

$(*selector*).slideUp(*speed,callback*);

可选的 speed 参数规定效果的时长。它可以取以下值："slow"、"fast" 或毫秒。

可选的 callback 参数是滑动完成后所执行的函数名称。

下面的例子演示了 slideUp() 方法：

实例

```
$("#flip").click(function(){ $("#panel").slideUp(); });
```

尝试一下 »

---

jQuery slideToggle() 方法

jQuery slideToggle() 方法可以在 slideDown() 与 slideUp() 方法之间进行切换。

如果元素向下滑动，则 slideToggle() 可向上滑动它们。

如果元素向上滑动，则 slideToggle() 可向下滑动它们。

$(*selector*).slideToggle(*speed,callback*);

可选的 speed 参数规定效果的时长。它可以取以下值："slow"、"fast" 或毫秒。

可选的 callback 参数是滑动完成后所执行的函数名称。

下面的例子演示了 slideToggle() 方法：

实例

```
$("#flip").click(function(){ $("#panel").slideToggle(); });
```

尝试一下 »

jQuery 效果- 动画

---

jQuery animate() 方法允许您创建自定义的动画。

jQuery 动画实例

jQuery

---

jQuery 动画 - animate() 方法

jQuery animate() 方法用于创建自定义动画。

语法：

$(*selector*).animate({*params*},*speed*,*callback*);

必需的 params 参数定义形成动画的 CSS 属性。

可选的 speed 参数规定效果的时长。它可以取以下值："slow"、"fast" 或毫秒。

可选的 callback 参数是动画完成后所执行的函数名称。

下面的例子演示 animate() 方法的简单应用。它把 元素往右边移动了 250 像素：

实例

```
$("button").click(function(){ $("div").animate({left:'250px'}); });
```

尝试一下 »

默认情况下，所有 HTML 元素都有一个静态位置，且无法移动。
如需对位置进行操作，要记得首先把元素的 CSS position 属性设置为 relative、fixed 或 absolute！

---

jQuery animate() - 操作多个属性

请注意，生成动画的过程中可同时使用多个属性：

实例

```
$("button").click(function(){ $("div").animate({ left:'250px', opacity:'0.5', height:'150px', width:'150px' }); });
```

尝试一下 »

可以用 animate() 方法来操作所有 CSS 属性吗？

是的，几乎可以！不过，需要记住一件重要的事情：当使用 animate() 时，必须使用 Camel 标记法书写所有的属性名，比如，必须使用 paddingLeft 而不是 padding-left，使用 marginRight 而不是 margin-right，等等。

同时，色彩动画并不包含在核心 jQuery 库中。

如果需要生成颜色动画，您需要从 jquery.com 下载 颜色动画 插件。

---

jQuery animate() - 使用相对值

也可以定义相对值（该值相对于元素的当前值）。需要在值的前面加上 += 或 -=：

实例

```
$("button").click(function(){ $("div").animate({ left:'250px', height:'+=150px', width:'+=150px' }); });
```

尝试一下 »

---

jQuery animate() - 使用预定义的值

您甚至可以把属性的动画值设置为 "show"、"hide" 或 "toggle"：

实例

```
$("button").click(function(){ $("div").animate({ height:'toggle' }); });
```

尝试一下 »

---

jQuery animate() - 使用队列功能

默认地，jQuery 提供针对动画的队列功能。

这意味着如果您在彼此之后编写多个 animate() 调用，jQuery 会创建包含这些方法调用的"内部"队列。然后逐一运行这些 animate 调用。

实例 1

```
$("button").click(function(){ var div=$("div"); div.animate({height:'300px',opacity:'0.4'},"slow");
div.animate({width:'300px',opacity:'0.8'},"slow"); div.animate({height:'100px',opacity:'0.4'},"slow");
div.animate({width:'100px',opacity:'0.8'},"slow"); });
```

尝试一下 »

下面的例子把 元素往右边移动了 100 像素，然后增加文本的字号：

实例 2

```
$("button").click(function(){ var div=$("div"); div.animate({left:'100px'},"slow"); div.animate({fontSize:'3em'},"slow"); });
```

尝试一下 »

jQuery 停止动画

jQuery stop() 方法用于在动画或效果完成前对它们进行停止。

停止滑动

点击这里，向上/向下滑动面板

实例

jQuery stop() 滑动
演示 jQuery stop() 方法。

jQuery stop() 动画(带参数)
演示 jQuery stop() 方法

jQuery stop() 方法

jQuery stop() 方法用于停止动画或效果，在它们完成之前。

stop() 方法适用于所有 jQuery 效果函数，包括滑动、淡入淡出和自定义动画。

语法：

$(*selector*).stop(*stopAll,goToEnd*);

可选的 stopAll 参数规定是否应该清除动画队列。默认是 false，即仅停止活动的动画，允许任何排入队列的动画向后执行。

可选的 goToEnd 参数规定是否立即完成当前动画。默认是 false。

因此，默认地，stop() 会清除在被选元素上指定的当前动画。

下面的例子演示 stop() 方法，不带参数：

实例

```
$("#stop").click(function(){ $("#panel").stop(); });
```

尝试一下 »

jQuery Callback 方法

Callback 函数在当前动画 100% 完成之后执行。

---

jQuery 动画的问题

许多 jQuery 函数涉及动画。这些函数也许会将 *speed* 或 *duration* 作为可选参数。

例子：*$("p").hide("slow")*

speed 或 *duration* 参数可以设置许多不同的值，比如 "slow", "fast", "normal" 或毫秒。

实例

以下实例在隐藏效果完全实现后回调函数:

使用 callback 实例

```
$("button").click(function(){ $("p").hide("slow",function(){ alert("段落现在被隐藏了"); }); });
```

尝试一下

以下实例没有回调函数，警告框会在隐藏效果完成前弹出：

没有 callback(回调)

```
$("button").click(function(){ $("p").hide(1000); alert("段落现在被隐藏了"); });
```

在线实例 »

jQuery - 链(Chaining)

通过 jQuery，可以把动作/方法链接在一起。

Chaining 允许我们在一条语句中运行多个 jQuery 方法（在相同的元素上）。

---

jQuery 方法链接

直到现在，我们都是一次写一条 jQuery 语句（一条接着另一条）。

不过，有一种名为链接（chaining）的技术，允许我们在相同的元素上运行多条 jQuery 命令，一条接着另一条。

提示： 这样的话，浏览器就不必多次查找相同的元素。

如需链接一个动作，您只需简单地把该动作追加到之前的动作上。

下面的例子把 css()、slideUp() 和 slideDown() 链接在一起。"p1" 元素首先会变为红色，然后向上滑动，再然后向下滑动：

实例

```
$("#p1").css("color","red").slideUp(2000).slideDown(2000);
```

尝试一下 »

如果需要，我们也可以添加多个方法调用。

提示：当进行链接时，代码行会变得很长。不过，jQuery 语法不是很严格；您可以按照希望的格式来写，包含换行和缩进。

如下书写也可以很好地运行：

实例

```
$("#p1").css("color","red") .slideUp(2000) .slideDown(2000);
```

尝试一下 »

jQuery 会抛掉多余的空格，并当成一行长代码来执行上面的代码行。

jQuery - 获取内容和属性

jQuery 拥有可操作 HTML 元素和属性的强大方法。

jQuery DOM 操作

jQuery 中非常重要的部分，就是操作 DOM 的能力。

jQuery 提供一系列与 DOM 相关的方法，这使访问和操作元素和属性变得很容易。

DOM = Document Object Model（文档对象模型）

DOM 定义访问 HTML 和 XML 文档的标准：

"W3C 文档对象模型独立于平台和语言的界面，允许程序和脚本动态访问和更新文档的内容、结构以及样式。"

jQuery - 设置内容和属性

设置内容 - text()、html() 以及 val()

我们将使用前一章中的三个相同的方法来设置内容：

- text() - 设置或返回所选元素的文本内容
- html() - 设置或返回所选元素的内容（包括 HTML 标记）
- val() - 设置或返回表单字段的值

下面的例子演示如何通过 text()、html() 以及 val() 方法来设置内容：

实例

```
$("#btn1").click(function(){ $("#test1").text("Hello world!"); }); $("#btn2").click(function(){ $("#test2").html("<b>Hello world!</b>"); });
$("#btn3").click(function(){ $("#test3").val("RUNOOB"); });
```

尝试一下 »

---

text()、html() 以及 val() 的回调函数

上面的三个 jQuery 方法：text()、html() 以及 val()，同样拥有回调函数。回调函数有两个参数：被选元素列表中当前元素的下标，以及原始（旧的）值。然后以函数新值返回您希望使用的字符串。

下面的例子演示带有回调函数的 text() 和 html()：

实例

```
$("#btn1").click(function(){ $("#test1").text(function(i,origText){ return "旧文本: " + origText + " 新文本: Hello world! (index: " + i + ")"; }); }); $("#btn2").click(function(){ $("#test2").html(function(i,origText){ return "旧 html: " + origText + " 新 html: Hello <b>world! </b> (index: " + i + ")"; }); });
```

尝试一下 »

设置属性 - attr()

jQuery attr() 方法也用于设置/改变属性值。

下面的例子演示如何改变（设置）链接中 href 属性的值：

实例

```
$("button").click(function(){ $("#runoob").attr("href","http://www.runoob.com/jquery"); });
```

尝试一下 »

attr() 方法也允许您同时设置多个属性。

下面的例子演示如何同时设置 href 和 title 属性：

实例

```
$("button").click(function(){ $("#runoob").attr({ "href" : "http://www.runoob.com/jquery", "title" : "jQuery 教程" }); });
```

尝试一下 »

---

attr() 的回调函数

jQuery 方法 attr()，也提供回调函数。回调函数有两个参数：被选元素列表中当前元素的下标，以及原始（旧的）值。然后以函数新值返回您希望使用的字符串。

下面的例子演示带有回调函数的 attr() 方法：

实例

```
$("button").click(function(){ $("#runoob").attr("href", function(i,origValue){ return origValue + "/jquery"; }); });
```

尝试一下 »

jQuery - 添加元素

通过 jQuery，可以很容易地添加新元素/内容。

添加新的 HTML 内容

我们将学习用于添加新内容的四个 jQuery 方法：

- append() - 在被选元素的结尾插入内容
- prepend() - 在被选元素的开头插入内容
- after() - 在被选元素之后插入内容
- before() - 在被选元素之前插入内容

---

jQuery append() 方法

jQuery append() 方法在被选元素的结尾插入内容（仍然该元素的内部）。

实例

```
$("p").append("追加文本");
```

尝试一下 »

---

jQuery prepend() 方法

jQuery prepend() 方法在被选元素的开头插入内容。

实例

```
$("p").prepend("在开头追加文本");
```

尝试一下 »

通过 append() 和 prepend() 方法添加若干新元素

在上面的例子中，我们只在被选元素的开头/结尾插入文本/HTML。

不过，append() 和 prepend() 方法能够通过参数接收无限数量的新元素。可以通过 jQuery 来生成文本/HTML（就像上面的例子那样），或者通过 JavaScript 代码和 DOM 元素。

在下面的例子中，我们创建若干个新元素。这些元素可以通过 text/HTML、jQuery 或者 JavaScript/DOM 来创建。然后我们通过 append() 方法把这些新元素追加到文本中（对 prepend() 同样有效）：

实例

```
function appendText() { var txt1="<p>文本。</p>"; // 使用 HTML 标签创建文本 var txt2=$("<p></p>").text("文本。"); // 使用 jQuery 创建文本 var txt3=document.createElement("p"); txt3.innerHTML="文本。"; // 使用 DOM 创建文本 text with DOM $("body").append(txt1,txt2,txt3); // 追加新元素 }
```

尝试一下 »

---

jQuery after() 和 before() 方法

jQuery after() 方法在被选元素之后插入内容。

jQuery before() 方法在被选元素之前插入内容。

实例

$("img").after("在后面添加文本"); $("img").before("在前面添加文本");

尝试一下 »

通过 after() 和 before() 方法添加若干新元素

after() 和 before() 方法能够通过参数接收无限数量的新元素。可以通过 text/HTML、jQuery 或者 JavaScript/DOM 来创建新元素。

在下面的例子中，我们创建若干新元素。这些元素可以通过 text/HTML、jQuery 或者 JavaScript/DOM 来创建。然后我们通过 after() 方法把这些新元素插到文本中（对 before() 同样有效）：

实例

function afterText() { var txt1="I "; // 使用 HTML 创建元素 var txt2=$("").text("love "); // 使用 jQuery 创建元素 var txt3=document.createElement("big"); // 使用 DOM 创建元素 txt3.innerHTML="jQuery!"; $("img").after(txt1,txt2,txt3); // 在图片后添加文本 }

尝试一下 »

jQuery - 删除元素

通过 jQuery，可以很容易地删除已有的 HTML 元素。

删除元素/内容

如需删除元素和内容，一般可使用以下两个 jQuery 方法：

- remove() - 删除被选元素（及其子元素）
- empty() - 从被选元素中删除子元素

---

jQuery remove() 方法

jQuery remove() 方法删除被选元素及其子元素。

实例

```
$("#div1").remove();
```

尝试一下 »

---

jQuery empty() 方法

jQuery empty() 方法删除被选元素的子元素。

实例

```
$("#div1").empty();
```

尝试一下 »

过滤被删除的元素

jQuery remove() 方法也可接受一个参数，允许您对被删元素进行过滤。

该参数可以是任何 jQuery 选择器的语法。

下面的例子删除 class="italic" 的所有 <p> 元素：

实例

```
$("p").remove(".italic");
```

尝试一下 »

jQuery - 获取并设置 CSS 类

通过 jQuery，可以很容易地对 CSS 元素进行操作。 切换 CSS 类

---

jQuery 操作 CSS

jQuery 拥有若干进行 CSS 操作的方法。我们将学习下面这些：

- addClass() - 向被选元素添加一个或多个类
- removeClass() - 从被选元素删除一个或多个类
- toggleClass() - 对被选元素进行添加/删除类的切换操作
- css() - 设置或返回样式属性

实例样式表

下面的样式表将用于本页的所有例子：

.important { font-weight:bold; font-size:xx-large; } .blue { color:blue; }

---

jQuery addClass() 方法

下面的例子展示如何向不同的元素添加 class 属性。当然，在添加类时，您也可以选取多个元素：

实例

$("button").click(function(){ $("h1,h2,p").addClass("blue"); $("div").addClass("important"); });

尝试一下 »

您也可以在 addClass() 方法中规定多个类：

实例

$("button").click(function(){ $("body div:first").addClass("important blue"); });

尝试一下 »

---

jQuery removeClass() 方法

下面的例子演示如何在不同的元素中删除指定的 class 属性：

实例

$("button").click(function(){ $("h1,h2,p").removeClass("blue"); });

尝试一下 »

---

jQuery toggleClass() 方法

下面的例子将展示如何使用 jQuery toggleClass() 方法。该方法对被选元素进行添加/删除类的切换操作：

实例

```
$("button").click(function(){ $("h1,h2,p").toggleClass("blue"); });
```

尝试一下 »

---

jQuery css() 方法

我们将在下一章讲解 jQuery css() 方法。

jQuery css() 方法

---

jQuery css() 方法

css() 方法设置或返回被选元素的一个或多个样式属性。

返回 CSS 属性

如需返回指定的 CSS 属性的值，请使用如下语法：

css("*propertyname*");

下面的例子将返回首个匹配元素的 background-color 值：

实例

```
$("p").css("background-color");
```

尝试一下 »

设置 CSS 属性

如需设置指定的 CSS 属性，请使用如下语法：

css("*propertyname*","*value*");

下面的例子将为所有匹配元素设置 background-color 值：

实例

```
$("p").css("background-color","yellow");
```

尝试一下 »

设置多个 CSS 属性

如需设置多个 CSS 属性，请使用如下语法：

css({"*propertyname*":"*value*","*propertyname*":"*value*",...});

下面的例子将为所有匹配元素设置 background-color 和 font-size：

实例

```
$("p").css({"background-color":"yellow","font-size":"200%"});
```

尝试一下 »

jQuery 尺寸

通过 jQuery，很容易处理元素和浏览器窗口的尺寸。

---

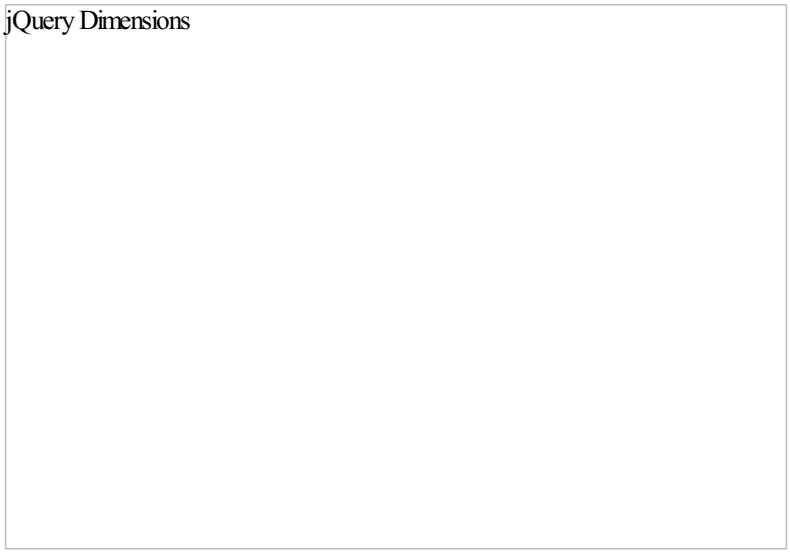
jQuery 尺寸方法

jQuery 提供多个处理尺寸的重要方法：

- width()
- height()
- innerWidth()
- innerHeight()
- outerWidth()
- outerHeight()

---

jQuery 尺寸



---

jQuery width() 和 height() 方法

width() 方法设置或返回元素的宽度（不包括内边距、边框或外边距）。

height() 方法设置或返回元素的高度（不包括内边距、边框或外边距）。

下面的例子返回指定的 元素的宽度和高度：

实例

```
$("button").click(function(){ var txt=""; txt+="div 的宽度是: " + $("#div1").width() + "</br>"; txt+="div 的高度是: " + $("#div1").height(); $("#div1").html(txt); });
```

尝试一下 »

---

jQuery innerWidth() 和 innerHeight() 方法

innerWidth() 方法返回元素的宽度（包括内边距）。

innerHeight() 方法返回元素的高度（包括内边距）。

下面的例子返回指定的 元素的 inner-width/height：

实例

```
$("button").click(function(){ var txt=""; txt+="div 宽度，包含内边距: " + $("#div1").innerWidth() + "</br>"; txt+="div 高度，包含内边距: " + $("#div1").innerHeight(); $("#div1").html(txt); });
```

尝试一下 »

---

jQuery outerWidth() 和 outerHeight() 方法

outerWidth() 方法返回元素的宽度（包括内边距和边框）。

outerHeight() 方法返回元素的高度（包括内边距和边框）。

下面的例子返回指定的 元素的 outer-width/height：

实例

```
$("button").click(function(){ var txt=""; txt+="div 宽度，包含内边距和边框: " + $("#div1").outerWidth() + "</br>"; txt+="div 高度，包含内边距和边框: " + $("#div1").outerHeight(); $("#div1").html(txt); });
```

尝试一下

jQuery 遍历

什么是遍历？

jQuery 遍历，意为"移动"，用于根据其相对于其他元素的关系来"查找"（或选取）HTML 元素。以某项选择开始，并沿着这个选择移动，直到抵达您期望的元素为止。

下图展示了一个家族树。通过 jQuery 遍历，您能够从被选（当前的）元素开始，轻松地在家族树中向上移动（祖先），向下移动（子孙），水平移动（同胞）。这种移动被称为对 DOM 进行遍历。

图示解析：

- 元素是 <ul> 的父元素，同时是其中所有内容的祖先。
- 元素是 <li> 元素的父元素，同时是 的子元素
- 左边的 <li> 元素是 的父元素， 的子元素，同时是 的后代。
- 元素是 <li> 的子元素，同时是 <ul> 和 的后代。
- 两个 <li> 元素是同胞（拥有相同的父元素）。
- 右边的 <li> 元素是 <b> 的父元素， 的子元素，同时是 的后代。
- <b> 元素是右边的 <li> 的子元素，同时是 <ul> 和 的后代。

祖先是父、祖父、曾祖父等等。后代是子、孙、曾孙等等。同胞拥有相同的父。

遍历 DOM

jQuery 提供了多种遍历 DOM 的方法。

遍历方法中最大的种类是树遍历（tree-traversal）。

下一章会讲解如何在 DOM 树中向上、下以及同级移动。

jQuery 遍历 - 祖先

祖先是父、祖父或曾祖父等等。

通过 jQuery，您能够向上遍历 DOM 树，以查找元素的祖先。

向上遍历 DOM 树

这些 jQuery 方法很有用，它们用于向上遍历 DOM 树：

- parent()
- parents()
- parentsUntil()

---

jQuery parent() 方法

parent() 方法返回被选元素的直接父元素。

该方法只会向上一级对 DOM 树进行遍历。

下面的例子返回每个 元素的的直接父元素：

实例

```
$(document).ready(function(){ $("span").parent(); });
```

尝试一下 »

---

jQuery parents() 方法

parents() 方法返回被选元素的所有祖先元素，它一路向上直到文档的根元素 (<html>)。

下面的例子返回所有 元素的所有祖先：

实例

```
$(document).ready(function(){ $("span").parents(); });
```

尝试一下 »

您也可以使用可选参数来过滤对祖先元素的搜索。

下面的例子返回所有 元素的所有祖先，并且它是 <ul> 元素：

实例

```
$(document).ready(function(){ $("span").parents("ul"); });
```

尝试一下 »

---

jQuery parentsUntil() 方法

parentsUntil() 方法返回介于两个给定元素之间的所有祖先元素。

下面的例子返回介于 与 元素之间的所有祖先元素：

实例

```
$(document).ready(function(){ $("span").parentsUntil("div"); });
```

尝试一下 »

jQuery 遍历 - 后代

后代是子、孙、曾孙等等。

通过 jQuery，您能够向下遍历 DOM 树，以查找元素的后代。

向下遍历 DOM 树

下面是两个用于向下遍历 DOM 树的 jQuery 方法：

- children()
- find()

---

jQuery children() 方法

children() 方法返回被选元素的所有直接子元素。

该方法只会向下一级对 DOM 树进行遍历。

下面的例子返回每个 元素的所有直接子元素：

实例

```
$(document).ready(function(){ $("div").children(); });
```

尝试一下 »

您也可以使用可选参数来过滤对子元素的搜索。

下面的例子返回类名为 "1" 的所有 <p> 元素，并且它们是 的直接子元素：

实例

```
$(document).ready(function(){ $("div").children("p.1"); });
```

尝试一下 »

---

jQuery find() 方法

find() 方法返回被选元素的后代元素，一路向下直到最后一个后代。

下面的例子返回属于 后代的所有 元素：

实例

```
$(document).ready(function(){ $("div").find("span"); });
```

尝试一下 »

下面的例子返回 的所有后代：

实例

```
$(document).ready(function(){ $("div").find("*"); });
```

尝试一下 »

jQuery 遍历 - 同胞(siblings)

同胞拥有相同的父元素。

通过 jQuery，您能够在 DOM 树中遍历元素的同胞元素。

在 DOM 树中水平遍历

有许多有用的方法让我们在 DOM 树进行水平遍历：

- siblings()
- next()
- nextAll()
- nextUntil()
- prev()
- prevAll()
- prevUntil()

---

jQuery siblings() 方法

siblings() 方法返回被选元素的所有同胞元素。

下面的例子返回 <h2> 的所有同胞元素：

实例

```
$(document).ready(function(){ $("h2").siblings(); });
```

尝试一下 »

您也可以使用可选参数来过滤对同胞元素的搜索。

下面的例子返回属于 <h2> 的同胞元素的所有 <p> 元素：

实例

```
$(document).ready(function(){ $("h2").siblings("p"); });
```

尝试一下 »

---

jQuery next() 方法

next() 方法返回被选元素的下一个同胞元素。

该方法只返回一个元素。

下面的例子返回 <h2> 的下一个同胞元素：

实例

```
$(document).ready(function(){ $("h2").next(); });
```

尝试一下 »

---

jQuery nextAll() 方法

nextAll() 方法返回被选元素的所有跟随的同胞元素。

下面的例子返回 <h2> 的所有跟随的同胞元素：

实例

```
$(document).ready(function(){ $("h2").nextAll(); });
```

尝试一下 »

---

jQuery nextUntil() 方法

nextUntil() 方法返回介于两个给定参数之间的所有跟随的同胞元素。

下面的例子返回介于 <h2> 与 <h6> 元素之间的所有同胞元素：

实例

```
$(document).ready(function(){ $("h2").nextUntil("h6"); });
```

尝试一下 »

---

jQuery prev(), prevAll() & prevUntil() 方法

prev(), prevAll() 以及 prevUntil() 方法的工作方式与上面的方法类似，只不过方向相反而已：它们返回的是前面的同胞元素（在 DOM 树中沿着同胞之前元素遍历，而不是之后元素遍历）。

jQuery 遍历- 过滤

缩小搜索元素的范围

三个最基本的过滤方法是：first(), last() 和 eq()，它们允许您基于其在一组元素中的位置来选择一个特定的元素。

其他过滤方法，比如 filter() 和 not() 允许您选取匹配或不匹配某项指定标准的元素。

---

jQuery first() 方法

first() 方法返回被选元素的首个元素。

下面的例子选取首个 元素内部的第一个 <p> 元素：

实例

```
$(document).ready(function(){ $("div p").first(); });
```

尝试一下 »

---

jQuery last() 方法

last() 方法返回被选元素的最后一个元素。

下面的例子选择最后一个 元素中的最后一个 <p> 元素：

实例

```
$(document).ready(function(){ $("div p").last(); });
```

尝试一下 »

---

jQuery eq() 方法

eq() 方法返回被选元素中带有指定索引号的元素。

索引号从 0 开始，因此首个元素的索引号是 0 而不是 1。下面的例子选取第二个 <p> 元素（索引号 1）：

实例

```
$(document).ready(function(){ $("p").eq(1); });
```

尝试一下 »

---

jQuery filter() 方法

filter() 方法允许您规定一个标准。不匹配这个标准的元素会被从集合中删除，匹配的元素会被返回。

下面的例子返回带有类名 "url" 的所有 <p> 元素：

实例

```
$(document).ready(function(){ $("p").filter(".url"); });
```

尝试一下 »

---

jQuery not() 方法

not() 方法返回不匹配标准的所有元素。

提示：not() 方法与 filter() 相反。

下面的例子返回不带有类名 "url" 的所有 <p> 元素：

实例

```
$(document).ready(function(){ $("p").not(".url"); });
```

尝试一下 »

jQuery - AJAX 简介

AJAX 是与服务器交换数据的技术，它在不重载全部页面的情况下，实现了对部分网页的更新。

jQuery AJAX 实例

使用 jQuery AJAX 修改文本内容

获取外部内容

尝试一下 »

什么是 AJAX?

AJAX = 异步 JavaScript 和 XML（Asynchronous JavaScript and XML）。

简短地说，在不重载整个网页的情况下，AJAX 通过后台加载数据，并在网页上进行显示。

使用 AJAX 的应用程序案例：谷歌地图、腾讯微博、优酷视频、人人网等等。

您可以在我们的 jQuery Ajax 参考手册学会 jQuery Ajax 的具体应用。

您可以在我们的 AJAX 教程中学到更多有关 AJAX 的知识。

关于 jQuery 与 AJAX

jQuery 提供多个与 AJAX 有关的方法。

通过 jQuery AJAX 方法，您能够使用 HTTP Get 和 HTTP Post 从远程服务器上请求文本、HTML、XML 或 JSON - 同时您能够把这些外部数据直接载入网页的被选元素中。

如果没有 jQuery，AJAX 编程还是有些难度的。

jQuery - AJAX load() 方法

jQuery load() 方法

jQuery load() 方法是简单但强大的 AJAX 方法。

load() 方法从服务器加载数据，并把返回的数据放入被选元素中。

语法：

$(selector).load(URL,data,callback);

必需的 *URL* 参数规定您希望加载的 URL。

可选的 *data* 参数规定与请求一同发送的查询字符串键/值对集合。

可选的 *callback* 参数是 load() 方法完成后所执行的函数名称。

这是示例文件（"demo_test.txt"）的内容：

<h2>jQuery AJAX 是个非常棒的功能！</h2> 这是段落的一些文本。

下面的例子会把文件 "demo_test.txt" 的内容加载到指定的 元素中：

实例

$("#div1").load("demo_test.txt");

尝试一下 »

也可以把 jQuery 选择器添加到 URL 参数。

下面的例子把 "demo_test.txt" 文件中 id="p1" 的元素的内容，加载到指定的 元素中：

实例

$("#div1").load("demo_test.txt #p1");

尝试一下 »

可选的 callback 参数规定当 load() 方法完成后所要允许的回调函数。回调函数可以设置不同的参数：

- *responseTxt* - 包含调用成功时的结果内容
- *statusTXT* - 包含调用的状态
- *xhr* - 包含 XMLHttpRequest 对象

下面的例子会在 load() 方法完成后显示一个提示框。如果 load() 方法已成功，则显示"外部内容加载成功！"，而如果失败，则显示错误消息：

实例

$("button").click(function(){ $("#div1").load("demo_test.txt",function(responseTxt,statusTxt,xhr){ if(statusTxt=="success") alert("外部内容加载成功!"); if(statusTxt=="error") alert("Error: "+xhr.status+": "+xhr.statusText); }); });

尝试一下 »

jQuery - AJAX get() 和 post() 方法

---

jQuery get() 和 post() 方法用于通过 HTTP GET 或 POST 请求从服务器请求数据。

HTTP 请求：GET vs. POST

两种在客户端和服务器端进行请求-响应的常用方法是：GET 和 POST。

- *GET* - 从指定的资源请求数据
- *POST* - 向指定的资源提交要处理的数据

GET 基本上用于从服务器获得（取回）数据。注释：GET 方法可能返回缓存数据。

POST 也可用于从服务器获取数据。不过，POST 方法不会缓存数据，并且常用于连同请求一起发送数据。

如需学习更多有关 GET 和 POST 以及两方法差异的知识，请阅读我们的 HTTP 方法 - GET 对比 POST。

---

jQuery $.get() 方法

$.get() 方法通过 HTTP GET 请求从服务器上请求数据。

语法：

$.get(*URL*,*callback*);

必需的 *URL* 参数规定您希望请求的 URL。

可选的 *callback* 参数是请求成功后所执行的函数名。

下面的例子使用 $.get() 方法从服务器上的一个文件中取回数据：

实例

```
$("button").click(function(){ $.get("demo_test.php",function(data,status){ alert("数据: " + data + "\n状态: " + status); }); });
```

尝试一下 »

$.get() 的第一个参数是我们希望请求的 URL（"demo_test.php"）。

第二个参数是回调函数。第一个回调参数存有被请求页面的内容，第二个回调参数存有请求的状态。

提示： 这个 PHP 文件 ("demo_test.php") 类似这样：

demo_test.php 文件代码：

```
<?php echo '这是个从PHP文件中读取的数据。'; ?>
```

---

jQuery $.post() 方法

$.post() 方法通过 HTTP POST 请求向服务器提交数据。

语法：

$.post(*URL,data,callback*);

必需的 *URL* 参数规定您希望请求的 URL。

可选的 *data* 参数规定连同请求发送的数据。

可选的 *callback* 参数是请求成功后所执行的函数名。

下面的例子使用 $.post() 连同请求一起发送数据：

实例

```
$("button").click(function(){ $.post("/try/ajax/demo_test_post.php", { name:"菜鸟教程", url:"http://www.runoob.com" }, function(data,status){ alert("数据: \n" + data + "\n状态: " + status); }); });
```

尝试一下 »

$.post() 的第一个参数是我们希望请求的 URL ("demo_test_post.php")。

然后我们连同请求（name 和 url）一起发送数据。

"demo_test_post.php" 中的 PHP 脚本读取这些参数，对它们进行处理，然后返回结果。

第三个参数是回调函数。第一个回调参数存有被请求页面的内容，而第二个参数存有请求的状态。

提示： 这个 PHP 文件 ("demo_test_post.php") 类似这样：

demo_test_post.php 文件代码:

```
<?php $name = isset($_POST['name']) ? htmlspecialchars($_POST['name']) : ''; $url = isset($_POST['url']) ? htmlspecialchars($_POST['url']) : ''; echo '网站名: ' . $name; echo "\n"; echo 'URL 地址: ' .$url; ?>
```

jQuery - noConflict() 方法

如何在页面上同时使用 jQuery 和其他框架？

---

jQuery 和其他 JavaScript 框架

正如您已经了解到的，jQuery 使用 $ 符号作为 jQuery 的简写。

如果其他 JavaScript 框架也使用 $ 符号作为简写怎么办？

其他一些 JavaScript 框架包括：MooTools、Backbone、Sammy、Cappuccino、Knockout、JavaScript MVC、Google Web Toolkit、Google Closure、Ember、Batman 以及 Ext JS。

其中某些框架也使用 $ 符号作为简写（就像 jQuery），如果您在用的两种不同的框架正在使用相同的简写符号，有可能导致脚本停止运行。

jQuery 的团队考虑到了这个问题，并实现了 noConflict() 方法。

---

jQuery noConflict() 方法

noConflict() 方法会释放对 $ 标识符的控制，这样其他脚本就可以使用它了。

当然，您仍然可以通过全名替代简写的方式来使用 jQuery：

实例

```
$.noConflict(); jQuery(document).ready(function(){ jQuery("button").click(function(){ jQuery("p").text("jQuery 仍然在工作!"); }); });
```

尝试一下 »

您也可以创建自己的简写。noConflict() 可返回对 jQuery 的引用，您可以把它存入变量，以供稍后使用。请看这个例子：

实例

```
var jq = $.noConflict(); jq(document).ready(function(){ jq("button").click(function(){ jq("p").text("jQuery 仍然在工作!"); }); });
```

尝试一下 »

如果你的 jQuery 代码块使用 $ 简写，并且您不愿意改变这个快捷方式，那么您可以把 $ 符号作为变量传递给 ready 方法。这样就可以在函数内使用 $ 符号了 - 而在函数外，依旧不得不使用 "jQuery"：

实例

```
$.noConflict(); jQuery(document).ready(function($){ $("button").click(function(){ $("p").text("jQuery 仍然在工作!"); }); });
```

尝试一下 »

JSONP 教程

本章节我们将向大家介绍 JSONP 的知识。

Jsonp(JSON with Padding) 是 json 的一种"使用模式"，可以让网页从别的域名（网站）那获取资料，即跨域读取数据。

为什么我们从不同的域（网站）访问数据需要一个特殊的技术(JSONP)呢？这是因为同源策略。

同源策略，它是由Netscape提出的一个著名的安全策略，现在所有支持JavaScript 的浏览器都会使用这个策略。

JSONP 应用

1. 服务端JSONP格式数据

如客户想访问 : http://www.runoob.com/try/ajax/jsonp.php?jsonp=callbackFunction。

假设客户期望返回JSON数据：["customername1","customername2"]。

真正返回到客户端的数据显示为: callbackFunction(["customername1","customername2"])。

服务端文件jsonp.php代码为：

jsonp.php 文件代码

<?php header('Content-type: application/json'); //获取回调函数名 $jsoncallback = htmlspecialchars($_REQUEST ['jsoncallback']); //json数据 $json_data = '["customername1","customername2"]'; //输出jsonp格式的数据 echo $jsoncallback . "(" . $json_data . ")"; ?>

2. 客户端实现 callbackFunction 函数

<script type="text/javascript"> function callbackFunction(result, methodName) { var html = '
'; for(var i = 0; i < result.length; i++) { html += '
' + result[i] + '
'; } html += '
'; document.getElementById('divCustomers').innerHTML = html; } </script>

页面展示

客户端页面完整代码

<!DOCTYPE html> <html> <head> <meta charset="utf-8"> <title>JSONP 实例</title> </head> <body> <script type="text/javascript">
function callbackFunction(result, methodName) { var html = '
'; for(var i = 0; i < result.length; i++) { html += '
' + result[i] + '
'; } html += '
'; document.getElementById('divCustomers').innerHTML = html; }
</script> <script type="text/javascript" src="http://www.runoob.com/try/ajax/jsonp.php?jsoncallback=callbackFunction"></script> </body> </html>

---

jQuery 使用 JSONP

以上代码可以使用 jQuery 代码实例：

<!DOCTYPE html> <html> <head> <meta charset="utf-8"> <title>JSONP 实例</title> <script src="http://cdn.static.runoob.com/libs/jquery/1.8.3/jquery.js"></script> </head> <body> <script>
$.getJSON("http://www.runoob.com/try/ajax/jsonp.php?jsoncallback=?", function(data) { var html = '
'; for(var i = 0; i < data.length; i++) { html += '
' + data[i] + '
'; } html += '
'; $('#divCustomers').html(html); });
</script> </body> </html>

jQuery 实例

你想增进 jQuery 技能吗？

---

jQuery 选择器

$(this).hide()

演示 jQuery 的 hide() 函数，隐藏当前的 HTML 元素。

$("p").hide()

演示 jQuery 的 hide() 函数，隐藏所有 <p> 元素。

$(".test").hide()

演示 jQuery 的 hide() 函数，隐藏所有 class="test" 的元素。

$("#test").hide()

演示 jQuery 的 hide() 函数，隐藏 id="test" 的元素。

---

jQuery 事件

jQuery click()
演示 jQuery jQuery click() 事件.

jQuery dblclick()
演示 jQuery dblclick() 事件。

jQuery mouseenter()
演示 jQuery mouseenter() 事件。

jQuery mouseleave()
演示 jQuery mouseleave() 事件。

jQuery mousedown()
演示 jQuery mousedown() 事件。

jQuery mouseup()
演示 jQuery mouseup() 事件。

jQuery hover()
演示 jQuery hover() 事件。

jQuery focus() 和 blur()
演示 jQuery focus() 和 blur() 事件。

实例解析

---

jQuery 隐藏/显示

jQuery hide()
演示 jQuery hide() 方法。

jQuery hide() 和 show()
演示jQuery hide() 和 show() 方法。

jQuery toggle()
jQuery toggle() 用于切换 hide() 和 show() 方法。

jQuery hide()
另外一个隐藏文本的实例。

实例解析

---

jQuery 淡入淡出

jQuery fadeIn()
演示 jQuery fadeIn() 方法。

jQuery fadeOut()
演示 jQuery fadeOut() 方法。

jQuery fadeToggle()
演示 jQuery fadeToggle() 方法。

jQuery fadeTo()
演示 jQuery fadeTo() 方法。

实例解析

---

jQuery 滑动

jQuery slideDown()
演示 jQuery slideDown() 方法。

jQuery slideUp()
演示 jQuery slideUp() 方法。

jQuery slideToggle()
演示 jQuery slideToggle() 方法。

实例解析

---

jQuery 动画

jQuery animate()
演示简单的 jQuery animate() 方法。

jQuery animate() - 设置多个css属性
演示通过 jQuery animate() 方法 改变样式。

jQuery animate() - 使用相关值
演示如何在 jQuery animate() 方法中使用相关值。

jQuery animate() - 使用预定义值
演示通过 animate() 方法预定义 "hide", "show", "toggle" 值。

jQuery animate()
演示更多 jQuery animate() 方法实例

jQuery animate()
演示更多 jQuery animate() 方法实例 (多个 animate() 回调).

实例 解析

---

jQuery 停止动画

jQuery stop() 滑动
演示 jQuery stop() 方法。

jQuery stop() 动画 (带参数)
演示 jQuery stop() 方法。

实例解析

---

jQuery HTML 获取 和 属性

jQuery text() 和 html() - 获取文本和内容
使用jQuery text() 和 html() 方法获取内容。

jQuery val() - 获取值
使用jQuery val() 方法获取表单的字段值。

jQuery attr() - 获取属性值
使用jQuery attr() 方法获取属性值。

实例解析

---

jQuery HTML 设置内容和属性

jQuery text(), html(), 和 val() - 设置内容
使用 jQuery text(), html() 和 val() 方法设置内容 。

jQuery text() 和 html() - 设置内容并使用回调函数
使用 text() 和 html() 设置内容并使用回调函数

jQuery attr() - 设置属性值
使用 jQuery attr() 方法设置属性值 。

jQuery attr() - 设置 多个属性值
使用jQuery attr() 方法设置多个属性值。

jQuery attr() - 设置属性值并使用回调函数
设置属性值 + 并使用回调函数调用attr().

实例解析

---

jQuery HTML 添加元素/内容

jQuery append()
在选取元素的末尾添加内容

jQuery prepend()
在选取元素的开头添加内容

jQuery append() -插入多个元素
创新新的 text/HTML 元素, jQuery 和 JavaScript/DOM。添加在新元素文本后。

jQuery after() 和 before()
在选取元素的前后添加 HTML　元素。

jQuery after() - 插入多个元素
创新新的 text/HTML 元素,jQuery和 JavaScript/DOM。在选取元素的末尾插入新元素。

实例解析

---

jQuery HTML 移除元素/内容

jQuery remove()
移除选取的元素

jQuery empty()
移除选取元素的所有子元素

jQuery remove() - 使用参数
过滤元素并移除

实例解析

---

jQuery Get 和 设置 CSS 类

jQuery addClass()
不同元素添加 class 属性

jQuery addClass() - 多个类
使用 addClass() 方法添加多个类

jQuery removeClass()
移除指定元素的类

jQuery toggleClass()
在选取的元素切换（添加/删除）类

实例解析

---

jQuery css() 方法

jQuery css() - 返回 CSS 属性
返回第一个匹配元素的css属性值

jQuery css() - 设置 CSS 属性
设置 所有配置元素指定的 CSS 属性

jQuery css() - 设置 CSS 属性
设置多个匹配元素的 CSS 属性

实例解析

---

jQuery 尺寸

jQuery - 返回 width() 和 height()
返回指定元素的 width 和 height

jQuery - 返回 innerWidth() 和 innerHeight()
返回指定元素的 inner-width/height

jQuery - 返回 outerWidth() 和 outerHeight()
返回指定元素的 outer-width/height

jQuery - 返回 outerWidth(true) 和 outerHeight(true)
返回指定元素的 outer-width/height (包含外边框)

jQuery - 返回 width() 和 height() of document 和 window
返回 HTML 文档和窗口的 width 和 height

jQuery - 设置 width() 和 height()
设置指定元素的 width 和 height

实例解析

---

jQuery 遍历 - 祖先

jQuery parent()
演示 jQuery parent() 方法。

jQuery parents()
演示 jQuery parents() 方法。

jQuery parentsUntil()
演示 jQuery parentsUntil() 方法。

实例解析

---

jQuery 遍历 - 后代

jQuery children()
演示 jQuery children() 方法。

jQuery find()
演示 jQuery find() 方法。

实例解析

---

jQuery 遍历 - 同胞(siblings)

jQuery siblings()
演示 jQuery siblings() 方法。

jQuery next()
演示 jQuery next() 方法。

jQuery nextAll()
演示 jQuery nextAll() 方法。

jQuery nextUntil()
演示 jQuery nextUntil() 方法。

实例解析

---

jQuery AJAX load() 方法

jQuery load()
异步载入文件内容并插入到 元素中。

jQuery load()
异步载入文件内容中指定的元素内容并插入到 元素.

jQuery load() - 使用回调函数(callback)
使用 jQuery load() 方法的回调函数。

实例解析

---

jQuery AJAX get() 和 post() 方法

jQuery get()
使用 $.get() 方法从服务端异步获取数据

jQuery post()
使用 $.post() 方法从服务端异步获取数据

实例解析

其他实例

jQuery 动态粒子效果

jQuery 选择器

jQuery 选择器

请使用我们的 jQuery 选择器检测器 来演示不同的选择器。

选择器	实例	选取
*	$("*")	所有元素
#id	$("#lastname")	id="lastname" 的元素
.class	$(".intro")	class="intro" 的所有元素
.class,.class	$(".intro,.demo")	class 为 "intro" 或 "demo" 的所有元素
element	$("p")	所有 <p> 元素
el1,el2,el3	$("h1,div,p")	所有 <h1>、 和 <p> 元素
:first	$("p:first")	第一个 <p> 元素
:last	$("p:last")	最后一个 <p> 元素
:even	$("tr:even")	所有偶数 <tr> 元素，索引值从 0 开始，第一个元素是偶数 (0)，第二个元素是奇数 (1)，以此类推。
:odd	$("tr:odd")	所有奇数 <tr> 元素，索引值从 0 开始，第一个元素是偶数 (0)，第二个元素是奇数 (1)，以此类推。
:first-child	$("p:first-child")	属于其父元素的第一个子元素的所有 <p> 元素
:first-of-type	$("p:first-of-type")	属于其父元素的第一个 <p> 元素的所有 <p> 元素
:last-child	$("p:last-child")	属于其父元素的最后一个子元素的所有 <p> 元素
:last-of-type	$("p:last-of-type")	属于其父元素的最后一个 <p> 元素的所有 <p> 元素
:nth-child(n)	$("p:nth-child(2)")	属于其父元素的第二个子元素的所有 <p> 元素
:nth-last-child(n)	$("p:nth-last-child(2)")	属于其父元素的第二个子元素的所有 <p> 元素，从最后一个子元素开始计数
:nth-of-type(n)	$("p:nth-of-type(2)")	属于其父元素的第二个 <p> 元素的所有 <p> 元素
:nth-last-of-type(n)	$("p:nth-last-of-type(2)")	属于其父元素的第二个 <p> 元素的所有 <p> 元素，从最后一个子元素开始计数
:only-child	$("p:only-child")	属于其父元素的唯一子元素的所有 <p> 元素
:only-of-type	$("p:only-of-type")	属于其父元素的特定类型的唯一子元素的所有 <p> 元素
parent > child	$("div > p")	元素的直接子元素的所有 <p> 元素
parent descendant	$("div p")	元素的后代的所有 <p> 元素
element + next	$("div + p")	每个 元素相邻的下一个 <p> 元素
element ~ siblings	$("div ~ p")	元素同级的所有 <p> 元素
:eq(index)	$("ul li:eq(3)")	列表中的第四个元素（index 值从 0 开始）
:gt(no)	$("ul li:gt(3)")	列举 index 大于 3 的元素
:lt(no)	$("ul li:lt(3)")	列举 index 小于 3 的元素
:not(selector)	$("input:not(:empty)")	所有不为空的输入元素
:header	$(":header")	所有标题元素 <h1>, <h2> ...
:animated	$(":animated")	所有动画元素
:focus	$(":focus")	当前具有焦点的元素
:contains(text)	$(":contains('Hello')")	所有包含文本 "Hello" 的元素
:has(selector)	$("div:has(p)")	所有包含有 <p> 元素在其内的 元素
:empty	$(":empty")	所有空元素

:parent	$(":parent")	匹配含有子元素或者文本的元素。
:hidden	$("p:hidden")	所有隐藏的 <p> 元素
:visible	$("table:visible")	所有可见的表格
:root	$(":root")	文档的根元素
:lang(*language*)	$("p:lang(de)")	所有 lang 属性值为 "de" 的 <p> 元素
[*attribute*]	$("[href]")	所有带有 href 属性的元素
[*attribute=value*]	$("[href='default.htm']")	所有带有 href 属性且值等于 "default.htm" 的元素
[*attribute!=value*]	$("[href!='default.htm']")	所有带有 href 属性且值不等于 "default.htm" 的元素
[*attribute$=value*]	$("[href$='.jpg']")	所有带有 href 属性且值以 ".jpg" 结尾的元素
[*attribute|=value*]	$("[title|='Tomorrow']")	所有带有 title 属性且值等于 'Tomorrow' 或者以 'Tomorrow' 后跟连接符作为开头的字符串
[*attribute^=value*]	$("[title^='Tom']")	所有带有 title 属性且值以 "Tom" 开头的元素
[*attribute~=value*]	$("[title~='hello']")	所有带有 title 属性且值包含单词 "hello" 的元素
[*attribute**=*value*]	$("[title*='hello']")	所有带有 title 属性且值包含字符串 "hello" 的元素
[*name=value*] [*name2=value2*]	$("input[id][name$='man']")	带有 id 属性，并且 name 属性以 man 结尾的输入框
:input	$(":input")	所有 input 元素
:text	$(":text")	所有带有 type="text" 的 input 元素
:password	$(":password")	所有带有 type="password" 的 input 元素
:radio	$(":radio")	所有带有 type="radio" 的 input 元素
:checkbox	$(":checkbox")	所有带有 type="checkbox" 的 input 元素
:submit	$(":submit")	所有带有 type="submit" 的 input 元素
:reset	$(":reset")	所有带有 type="reset" 的 input 元素
:button	$(":button")	所有带有 type="button" 的 input 元素
:image	$(":image")	所有带有 type="image" 的 input 元素
:file	$(":file")	所有带有 type="file" 的 input 元素
:enabled	$(":enabled")	所有启用的元素
:disabled	$(":disabled")	所有禁用的元素
:selected	$(":selected")	所有选定的下拉列表元素
:checked	$(":checked")	所有选中的复选框选项
.selector	$(selector).selector	在jQuery 1.7中已经不被赞成使用。返回传给jQuery()的原始选择器
:target	$("p:target")	选择器将选中ID和URI中一个格式化的标识符相匹配的<p>元素

jQuery 事件 方法

jQuery 事件方法

事件方法触发器或添加一个函数到被选元素的事件处理程序。

下面的表格列出了所有用于处理事件的 jQuery 方法。

方法	描述
bind()	向元素添加事件处理程序
blur()	添加/触发失去焦点事件
change()	添加/触发 change 事件
click()	添加/触发 click 事件
dblclick()	添加/触发 double click 事件
delegate()	向匹配元素的当前或未来的子元素添加处理程序
die()	在版本 1.9 中被移除。移除所有通过 live() 方法添加的事件处理程序
error()	在版本 1.8 中被废弃。添加/触发 error 事件
event.currentTarget	在事件冒泡阶段内的当前 DOM 元素
event.data	包含当前执行的处理程序被绑定时传递到事件方法的可选数据
event.delegateTarget	返回当前调用的 jQuery 事件处理程序所添加的元素
event.isDefaultPrevented()	返回指定的 event 对象上是否调用了 event.preventDefault()
event.isImmediatePropagationStopped()	返回指定的 event 对象上是否调用了 event.stopImmediatePropagation()
event.isPropagationStopped()	返回指定的 event 对象上是否调用了 event.stopPropagation()
event.namespace	返回当事件被触发时指定的命名空间
event.pageX	返回相对于文档左边缘的鼠标位置
event.pageY	返回相对于文档上边缘的鼠标位置
event.preventDefault()	阻止事件的默认行为
event.relatedTarget	返回当鼠标移动时哪个元素进入或退出
event.result	包含由被指定事件触发的事件处理程序返回的最后一个值
event.stopImmediatePropagation()	阻止其他事件处理程序被调用
event.stopPropagation()	阻止事件向上冒泡到 DOM 树，阻止任何父处理程序被事件通知
event.target	返回哪个 DOM 元素触发事件
event.timeStamp	返回从 1970 年 1 月 1 日到事件被触发时的毫秒数
event.type	返回哪种事件类型被触发
event.which	返回指定事件上哪个键盘键或鼠标按钮被按下
event.metaKey	事件触发时 META 键是否被按下
focus()	添加/触发 focus 事件
focusin()	添加事件处理程序到 focusin 事件
focusout()	添加事件处理程序到 focusout 事件
hover()	添加两个事件处理程序到 hover 事件
keydown()	添加/触发 keydown 事件
keypress()	添加/触发 keypress 事件
keyup()	添加/触发 keyup 事件
live()	在版本 1.9 中被移除。添加一个或多个事件处理程序到当前或未来的被选元素
load()	在版本 1.8 中被废弃。添加一个事件处理程序到 load 事件
mousedown()	添加/触发 mousedown 事件

mouseenter()	添加/触发 mouseenter 事件
mouseleave()	添加/触发 mouseleave 事件
mousemove()	添加/触发 mousemove 事件
mouseout()	添加/触发 mouseout 事件
mouseover()	添加/触发 mouseover 事件
mouseup()	添加/触发 mouseup 事件
off()	移除通过 on() 方法添加的事件处理程序
on()	向元素添加事件处理程序
one()	向被选元素添加一个或多个事件处理程序。该处理程序只能被每个元素触发一次
$.proxy()	接受一个已有的函数，并返回一个带特定上下文的新的函数
ready()	规定当 DOM 完全加载时要执行的函数
resize()	添加/触发 resize 事件
scroll()	添加/触发 scroll 事件
select()	添加/触发 select 事件
submit()	添加/触发 submit 事件
toggle()	在版本 1.9 中被移除。添加 click 事件之间要切换的两个或多个函数
trigger()	触发绑定到被选元素的所有事件
triggerHandler()	触发绑定到被选元素的指定事件上的所有函数
unbind()	从被选元素上移除添加的事件处理程序
undelegate()	从现在或未来的被选元素上移除事件处理程序
unload()	在版本 1.8 中被废弃。添加事件处理程序到 unload 事件
contextmenu()	添加事件处理程序到 contextmenu 事件
$.holdReady()	用于暂停或恢复.ready() 事件的执行

jQuery 效果 方法

---

jQuery 效果方法

下面的表格列出了所有用于创建动画效果的 jQuery 方法。

方法	描述
animate()	对被选元素应用"自定义"的动画
clearQueue()	对被选元素移除所有排队函数（仍未运行的）
delay()	对被选元素的所有排队函数（仍未运行）设置延迟
dequeue()	移除下一个排队函数，然后执行函数
fadeIn()	逐渐改变被选元素的不透明度，从隐藏到可见
fadeOut()	逐渐改变被选元素的不透明度，从可见到隐藏
fadeTo()	把被选元素逐渐改变至给定的不透明度
fadeToggle()	在 fadeIn() 和 fadeOut() 方法之间进行切换
finish()	对被选元素停止、移除并完成所有排队动画
hide()	隐藏被选元素
queue()	显示被选元素的排队函数
show()	显示被选元素
slideDown()	通过调整高度来滑动显示被选元素
slideToggle()	slideUp() 和 slideDown() 方法之间的切换
slideUp()	通过调整高度来滑动隐藏被选元素
stop()	停止被选元素上当前正在运行的动画
toggle()	hide() 和 show() 方法之间的切换

jQuery HTML / CSS 方法

jQuery HTML / CSS 方法

下面的表格列出了所有用于处理 HTML 和 CSS 的 jQuery 方法。

下面的方法适用于 HTML 和 XML 文档。除了：html() 方法。

方法	描述
addClass()	向被选元素添加一个或多个类名
after()	在被选元素后插入内容
append()	在被选元素的结尾插入内容
appendTo()	在被选元素的结尾插入 HTML 元素
attr()	设置或返回被选元素的属性/值
before()	在被选元素前插入内容
clone()	生成被选元素的副本
css()	为被选元素设置或返回一个或多个样式属性
detach()	移除被选元素（保留数据和事件）
empty()	从被选元素移除所有子节点和内容
hasClass()	检查被选元素是否包含指定的 class 名称
height()	设置或返回被选元素的高度
html()	设置或返回被选元素的内容
innerHeight()	返回元素的高度（包含 padding，不包含 border）
innerWidth()	返回元素的宽度（包含 padding，不包含 border）
insertAfter()	在被选元素后插入 HTML 元素
insertBefore()	在被选元素前插入 HTML 元素
offset()	设置或返回被选元素的偏移坐标（相对于文档）
offsetParent()	返回第一个定位的祖先元素
outerHeight()	返回元素的高度（包含 padding 和 border）
outerWidth()	返回元素的宽度（包含 padding 和 border）
position()	返回元素的位置（相对于父元素）
prepend()	在被选元素的开头插入内容
prependTo()	在被选元素的开头插入 HTML 元素
prop()	设置或返回被选元素的属性/值
remove()	移除被选元素（包含数据和事件）
removeAttr()	从被选元素移除一个或多个属性
removeClass()	从被选元素移除一个或多个类
removeProp()	移除通过 prop() 方法设置的属性
replaceAll()	把被选元素替换为新的 HTML 元素
replaceWith()	把被选元素替换为新的内容
scrollLeft()	设置或返回被选元素的水平滚动条位置
scrollTop()	设置或返回被选元素的垂直滚动条位置
text()	设置或返回被选元素的文本内容
toggleClass()	在被选元素中添加/移除一个或多个类之间切换
unwrap()	移除被选元素的父元素
val()	设置或返回被选元素的属性值（针对表单元素）

width()	设置或返回被选元素的宽度
wrap()	在每个被选元素的周围用 HTML 元素包裹起来
wrapAll()	在所有被选元素的周围用 HTML 元素包裹起来
wrapInner()	在每个被选元素的内容周围用 HTML 元素包裹起来
$.escapeSelector()	转义CSS选择器中有特殊意义的字符或字符串
$.cssHooks	提供了一种方法通过定义函数来获取和设置特定的CSS值

jQuery 遍历 方法

jQuery 遍历方法

方法	描述
add()	把元素添加到匹配元素的集合中
addBack()	把之前的元素集添加到当前集合中
andSelf()	在版本 1.8 中被废弃。addBack() 的别名
children()	返回被选元素的所有直接子元素
closest()	返回被选元素的第一个祖先元素
contents()	返回被选元素的所有直接子元素（包含文本和注释节点）
each()	为每个匹配元素执行函数
end()	结束当前链中最近的一次筛选操作，并把匹配元素集合返回到前一次的状态
eq()	返回带有被选元素的指定索引号的元素
filter()	把匹配元素集合缩减为匹配选择器或匹配函数返回值的新元素
find()	返回被选元素的后代元素
first()	返回被选元素的第一个元素
has()	返回拥有一个或多个元素在其内的所有元素
is()	根据选择器/元素/jQuery 对象检查匹配元素集合，如果存在至少一个匹配元素，则返回 true
last()	返回被选元素的最后一个元素
map()	把当前匹配集合中的每个元素传递给函数，产生包含返回值的新 jQuery 对象
next()	返回被选元素的后一个同级元素
nextAll()	返回被选元素之后的所有同级元素
nextUntil()	返回介于两个给定参数之间的每个元素之后的所有同级元素
not()	从匹配元素集合中移除元素
offsetParent()	返回第一个定位的父元素
parent()	返回被选元素的直接父元素
parents()	返回被选元素的所有祖先元素
parentsUntil()	返回介于两个给定参数之间的所有祖先元素
prev()	返回被选元素的前一个同级元素
prevAll()	返回被选元素之前的所有同级元素
prevUntil()	返回介于两个给定参数之间的每个元素之前的所有同级元素
siblings()	返回被选元素的所有同级元素
slice()	把匹配元素集合缩减为指定范围的子集

jQuery AJAX 方法

jQuery AJAX 方法

AJAX 是一种与服务器交换数据的技术，可以在不重新载入整个页面的情况下更新网页的一部分。

下面的表格列出了所有的 jQuery AJAX 方法：

方法	描述
$.ajax()	执行异步 AJAX 请求
$.ajaxPrefilter()	在每个请求发送之前且被 $.ajax() 处理之前，处理自定义 Ajax 选项或修改已存在选项
$.ajaxSetup()	为将来的 AJAX 请求设置默认值
$.ajaxTransport()	创建处理 Ajax 数据实际传送的对象
$.get()	使用 AJAX 的 HTTP GET 请求从服务器加载数据
$.getJSON()	使用 HTTP GET 请求从服务器加载 JSON 编码的数据
$.getScript()	使用 AJAX 的 HTTP GET 请求从服务器加载并执行 JavaScript
$.param()	创建数组或对象的序列化表示形式（可用于 AJAX 请求的 URL 查询字符串）
$.post()	使用 AJAX 的 HTTP POST 请求从服务器加载数据
ajaxComplete()	规定 AJAX 请求完成时运行的函数
ajaxError()	规定 AJAX 请求失败时运行的函数
ajaxSend()	规定 AJAX 请求发送之前运行的函数
ajaxStart()	规定第一个 AJAX 请求开始时运行的函数
ajaxStop()	规定所有的 AJAX 请求完成时运行的函数
ajaxSuccess()	规定 AJAX 请求成功完成时运行的函数
load()	从服务器加载数据，并把返回的数据放置到指定的元素中
serialize()	编码表单元素集为字符串以便提交
serializeArray()	编码表单元素集为 names 和 values 的数组

jQuery 杂项方法

jQuery 杂项方法

方法	描述
data()	向被选元素附加数据，或者从被选元素获取数据
each()	为每个匹配元素执行函数
get()	获取由选择器指定的 DOM 元素
index()	从匹配元素中搜索给定元素
$.noConflict()	释放变量 $ 的 jQuery 控制权
$.param()	创建数组或对象的序列化表示形式（可在生成 AJAX 请求时用于 URL 查询字符串中）
removeData()	移除之前存放的数据
size()	在版本 1.8 中被废弃。返回被 jQuery 选择器匹配的 DOM 元素的数量
toArray()	以数组的形式检索所有包含在 jQuery 集合中的所有 DOM 元素
pushStack()	将一个DOM元素集合加入到jQuery栈
$.when()	提供一种方法来执行一个或多个对象的回调函数

jQuery 实用工具

方法	描述
$.boxModel	

jQuery 属性

jQuery 属性

方法	描述
context	在版本 1.10 中被废弃。包含被传递到 jQuery 的原始上下文
jquery	包含 jQuery 的版本号
jQuery.fx.interval	改变以毫秒计的动画运行速率
jQuery.fx.off	对所有动画进行全局禁用或启用
jQuery.support	包含表示不同浏览器特性或漏洞的属性集（主要用于 jQuery 的内部使用）
length	包含 jQuery 对象中元素的数目
jQuery.cssNumber	包含所有可以不使用单位的CSS属性的对象

jQuery Validate

jQuery Validate 插件为表单提供了强大的验证功能，让客户端表单验证变得更简单，同时提供了大量的定制选项，满足应用程序各种需求。该插件捆绑了一套有用的验证方法，包括 URL 和电子邮件验证，同时提供了一个用来编写用户自定义方法的 API。所有的捆绑方法默认使用英语作为错误信息，且已翻译成其他 37 种语言。

该插件是由 Jörn Zaefferer 编写和维护的，他是 jQuery 团队的一名成员，是 jQuery UI 团队的主要开发人员，是 QUnit 的维护人员。该插件在 2006 年 jQuery 早期的时候就已经开始出现，并一直更新至今。目前版本是 **1.14.0**。

访问 jQuery Validate 官网，下载最新版的 jQuery Validate 插件。

菜鸟教程提供的 1.14.0 版本下载地址：http://static.runoob.com/download/jquery-validation-1.14.0.zip

导入 js 库（使用菜鸟教程提供的CDN）

```
<script src="http://static.runoob.com/assets/jquery-validation-1.14.0/lib/jquery.js"></script>
<script src="http://static.runoob.com/assets/jquery-validation-1.14.0/dist/jquery.validate.min.js"></script>
```

默认校验规则

序号	规则	描述
1	required:true	必须输入的字段。
2	remote:"check.php"	使用 ajax 方法调用 check.php 验证输入值。
3	email:true	必须输入正确格式的电子邮件。
4	url:true	必须输入正确格式的网址。
5	date:true	必须输入正确格式的日期。日期校验 ie6 出错，慎用。
6	dateISO:true	必须输入正确格式的日期（ISO），例如：2009-06-23，1998/01/22。只验证格式，不验证有效性。
7	number:true	必须输入合法的数字（负数，小数）。
8	digits:true	必须输入整数。
9	creditcard:	必须输入合法的信用卡号。
10	equalTo:"#field"	输入值必须和 #field 相同。
11	accept:	输入拥有合法后缀名的字符串（上传文件的后缀）。
12	maxlength:5	输入长度最多是 5 的字符串（汉字算一个字符）。
13	minlength:10	输入长度最小是 10 的字符串（汉字算一个字符）。
14	rangelength:[5,10]	输入长度必须介于 5 和 10 之间的字符串（汉字算一个字符）。
15	range:[5,10]	输入值必须介于 5 和 10 之间。
16	max:5	输入值不能大于 5。
17	min:10	输入值不能小于 10。

默认提示

```
messages: {
    required: "This field is required.",
    remote: "Please fix this field.",
    email: "Please enter a valid email address.",
    url: "Please enter a valid URL.",
    date: "Please enter a valid date.",
    dateISO: "Please enter a valid date ( ISO ).",
    number: "Please enter a valid number.",
    digits: "Please enter only digits.",
    creditcard: "Please enter a valid credit card number.",
    equalTo: "Please enter the same value again.",
    maxlength: $.validator.format( "Please enter no more than {0} characters." ),
    minlength: $.validator.format( "Please enter at least {0} characters." ),
    rangelength: $.validator.format( "Please enter a value between {0} and {1} characters long." ),
    range: $.validator.format( "Please enter a value between {0} and {1}." ),
    max: $.validator.format( "Please enter a value less than or equal to {0}." ),
    min: $.validator.format( "Please enter a value greater than or equal to {0}." )
}
```

jQuery Validate提供了中文信息提示包，位于下载包的 dist/localization/messages_zh.js，内容如下：

```
(function( factory ) {
    if ( typeof define === "function" && define.amd ) {
        define( ["jquery", "../jquery.validate"], factory );
    } else {
        factory( jQuery );
    }
}(function( $ ) {

/*
 * Translated default messages for the jQuery validation plugin.
 * Locale: ZH (Chinese, 中文 (Zhōngwén), 汉语, 漢語)
 */
$.extend($.validator.messages, {
    required: "这是必填字段",
    remote: "请修正此字段",
    email: "请输入有效的电子邮件地址",
    url: "请输入有效的网址",
    date: "请输入有效的日期",
    dateISO: "请输入有效的日期 (YYYY-MM-DD)",
    number: "请输入有效的数字",
    digits: "只能输入数字",
    creditcard: "请输入有效的信用卡号码",
    equalTo: "你的输入不相同",
    extension: "请输入有效的后缀",
    maxlength: $.validator.format("最多可以输入 {0} 个字符"),
    minlength: $.validator.format("最少要输入 {0} 个字符"),
    rangelength: $.validator.format("请输入长度在 {0} 到 {1} 之间的字符串"),
    range: $.validator.format("请输入范围在 {0} 到 {1} 之间的数值"),
    max: $.validator.format("请输入不大于 {0} 的数值"),
    min: $.validator.format("请输入不小于 {0} 的数值")
});

}));
```

你可以将该本地化信息文件 dist/localization/messages_zh.js 引入到页面：

```
<script src="http://static.runoob.com/assets/jquery-validation-1.14.0/dist/localization/messages_zh.js"></script>
```

使用方式

1、将校验规则写到控件中

```
<script src="http://static.runoob.com/assets/jquery-validation-1.14.0/lib/jquery.js"></script>
<script src="http://static.runoob.com/assets/jquery-validation-1.14.0/dist/jquery.validate.min.js"></script>
<script src="http://static.runoob.com/assets/jquery-validation-1.14.0/dist/localization/messages_zh.js"></script>
<script>
$.validator.setDefaults({
    submitHandler: function() {
      alert("提交事件!");
    }
});
$().ready(function() {
    $("#commentForm").validate();
});
</script>

<form class="cmxform" id="commentForm" method="get" action="">
  <fieldset>
    <legend>输入您的名字，邮箱，URL，备注。</legend>
    <p>
      <label for="cname">Name (必需, 最小两个字母)</label>
      <input id="cname" name="name" minlength="2" type="text" required>
    </p>
    <p>
      <label for="cemail">E-Mail (必需)</label>
      <input id="cemail" type="email" name="email" required>
    </p>
    <p>
      <label for="curl">URL (可选)</label>
      <input id="curl" type="url" name="url">
    </p>
    <p>
      <label for="ccomment">备注 (必需)</label>
```

```
      <textarea id="ccomment" name="comment" required></textarea>
    </p>
    <p>
      <input class="submit" type="submit" value="Submit">
    </p>
  </fieldset>
</form>
```

[尝试一下 »]

2、将校验规则写到 js 代码中

```
$().ready(function() {
// 在键盘按下并释放及提交后验证提交表单
  $("#signupForm").validate({
    rules: {
      firstname: "required",
      lastname: "required",
      username: {
        required: true,
        minlength: 2
      },
      password: {
        required: true,
        minlength: 5
      },
      confirm_password: {
        required: true,
        minlength: 5,
        equalTo: "#password"
      },
      email: {
        required: true,
        email: true
      },
      topic: {
        required: "#newsletter:checked",
        minlength: 2
      },
      agree: "required"
    },
    messages: {
      firstname: "请输入您的名字",
      lastname: "请输入您的姓氏",
      username: {
        required: "请输入用户名",
        minlength: "用户名必需由两个字母组成"
      },
      password: {
        required: "请输入密码",
        minlength: "密码长度不能小于 5 个字母"
      },
      confirm_password: {
        required: "请输入密码",
        minlength: "密码长度不能小于 5 个字母",
        equalTo: "两次密码输入不一致"
      },
      email: "请输入一个正确的邮箱",
      agree: "请接受我们的声明",
      topic: "请选择两个主题"
     }
    })
});
```

messages 处，如果某个控件没有 message，将调用默认的信息

```
<form class="cmxform" id="signupForm" method="get" action="">
  <fieldset>
    <legend>验证完整的表单</legend>
    <p>
      <label for="firstname">名字</label>
      <input id="firstname" name="firstname" type="text">
    </p>
    <p>
      <label for="lastname">姓氏</label>
      <input id="lastname" name="lastname" type="text">
```

```
    </p>
    <p>
      <label for="username">用户名</label>
      <input id="username" name="username" type="text">
    </p>
    <p>
      <label for="password">密码</label>
      <input id="password" name="password" type="password">
    </p>
    <p>
      <label for="confirm_password">验证密码</label>
      <input id="confirm_password" name="confirm_password" type="password">
    </p>
    <p>
      <label for="email">Email</label>
      <input id="email" name="email" type="email">
    </p>
    <p>
      <label for="agree">请同意我们的声明</label>
      <input type="checkbox" class="checkbox" id="agree" name="agree">
    </p>
    <p>
      <label for="newsletter">我乐意接收新信息</label>
      <input type="checkbox" class="checkbox" id="newsletter" name="newsletter">
    </p>
    <fieldset id="newsletter_topics">
      <legend>主题 （至少选择两个） - 注意：如果没有勾选"我乐意接收新信息"以下选项会隐藏，但我们这里作为演示让它可见</legend>
      <label for="topic_marketflash">
        <input type="checkbox" id="topic_marketflash" value="marketflash" name="topic">Marketflash
      </label>
      <label for="topic_fuzz">
        <input type="checkbox" id="topic_fuzz" value="fuzz" name="topic">Latest fuzz
      </label>
      <label for="topic_digester">
        <input type="checkbox" id="topic_digester" value="digester" name="topic">Mailing list digester
      </label>
      <label for="topic" class="error">Please select at least two topics you'd like to receive.</label>
    </fieldset>
    <p>
      <input class="submit" type="submit" value="提交">
    </p>
  </fieldset>
</form>
```

尝试一下 »

required: true 值是必须的。
required: "#aa:checked" 表达式的值为真，则需要验证。
required: function(){} 返回为真，表示需要验证。

后边两种常用于，表单中需要同时填或不填的元素。

常用方法及注意问题

1、用其他方式替代默认的 SUBMIT

```
$().ready(function() {
 $("#signupForm").validate({
        submitHandler:function(form){
            alert("提交事件!");
            form.submit();
        }
    });
});
```

使用 ajax 方式

```
 $(".selector").validate({
 submitHandler: function(form)
   {
      $(form).ajaxSubmit();
   }
```

```
 })
```

可以设置 validate 的默认值，写法如下：

```
$.validator.setDefaults({
  submitHandler: function(form) { alert("提交事件!");form.submit(); }
});
```

如果想提交表单, 需要使用 form.submit()，而不要使用 $(form).submit()。

2、debug：只验证不提交表单

如果这个参数为true，那么表单不会提交，只进行检查，调试时十分方便。

```
$().ready(function() {
 $("#signupForm").validate({
        debug:true
    });
});
```

如果一个页面中有多个表单都想设置成为 debug，则使用：

```
$.validator.setDefaults({
   debug: true
})
```

3、ignore：忽略某些元素不验证

```
ignore: ".ignore"
```

4、更改错误信息显示的位置

```
errorPlacement: Callback
```

指明错误放置的位置，默认情况是：error.appendTo(element.parent());即把错误信息放在验证的元素后面。

```
errorPlacement: function(error, element) {
    error.appendTo(element.parent());
}
```

实例

```
<p>将错误信息放在 label 元素后并使用 span 元素包裹它</p>

<form method="get" class="cmxform" id="form1" action="">
  <fieldset>
    <legend>Login Form</legend>
    <p>
      <label for="user">Username</label>
      <input id="user" name="user" required minlength="3">
    </p>
    <p>
      <label for="password">Password</label>
      <input id="password" type="password" maxlength="12" name="password" required minlength="5">
    </p>
    <p>
      <input class="submit" type="submit" value="Login">
    </p>
  </fieldset>
</form>
```

尝试一下 »

代码的作用是：一般情况下把错误信息显示在 <td class="status"></td> 中，如果是 radio 则显示在 <td></td> 中，如果是 checkbox 则显示在内容的后面。

参数	类型	描述	默认值
errorClass	String	指定错误提示的 css 类名，可以自定义错误提示的样式。	"error"
errorElement	String	用什么标签标记错误，默认是 label，可以改成 em。	"label"

errorContainer	Selector	显示或者隐藏验证信息，可以自动实现有错误信息出现时把容器属性变为显示，无错误时隐藏，用处不大。 errorContainer: "#messageBox1, #messageBox2"
errorLabelContainer	Selector	把错误信息统一放在一个容器里面。
wrapper	String	用什么标签再把上边的 errorELement 包起来。

一般这三个属性同时使用，实现在一个容器内显示所有错误提示的功能，并且没有信息时自动隐藏。

```
errorContainer: "div.error",
errorLabelContainer: $("#signupForm div.error"),
wrapper: "li"
```

5、更改错误信息显示的样式

设置错误提示的样式，可以增加图标显示，在该系统中已经建立了一个 validation.css，专门用于维护校验文件的样式。

```
input.error { border: 1px solid red; }
label.error {
  background:url("./demo/images/unchecked.gif") no-repeat 0px 0px;

  padding-left: 16px;

  padding-bottom: 2px;

  font-weight: bold;

  color: #EA5200;
}
label.checked {
  background:url("./demo/images/checked.gif") no-repeat 0px 0px;
}
```

6、每个字段验证通过执行函数

```
success：String,Callback
```

要验证的元素通过验证后的动作，如果跟一个字符串，会当作一个 css 类，也可跟一个函数。

```
success: function(label) {
    // set   as text for IE
    label.html(" ").addClass("checked");
    //label.addClass("valid").text("Ok!")
}
```

添加 "valid" 到验证元素，在 CSS 中定义的样式 <style>label.valid {}</style>。

```
success: "valid"
```

7、验证的触发方式修改

下面的虽然是 boolean 型的，但建议除非要改为 false，否则别乱添加。

触发方式	类型	描述	默认值
onsubmit	Boolean	提交时验证。设置为 false 就用其他方法去验证。	true
onfocusout	Boolean	失去焦点时验证（不包括复选框/单选按钮）。	true
onkeyup	Boolean	在 keyup 时验证。	true
onclick	Boolean	在点击复选框和单选按钮时验证。	true
focusInvalid	Boolean	提交表单后，未通过验证的表单（第一个或提交之前获得焦点的未通过验证的表单）会获得焦点。	true
focusCleanup	Boolean	如果是 true 那么当未通过验证的元素获得焦点时，移除错误提示。避免和 focusInvalid 一起用。	false

```
// 重置表单
$().ready(function() {
 var validator = $("#signupForm").validate({
        submitHandler:function(form){
            alert("submitted");
```

```
            form.submit();
        }
    });
    $("#reset").click(function() {
        validator.resetForm();
    });

});
```

8、异步验证

```
remote：URL
```

使用 ajax 方式进行验证，默认会提交当前验证的值到远程地址，如果需要提交其他的值，可以使用 data 选项。

```
remote: "check-email.php"
```

```
remote: {
    url: "check-email.php",     //后台处理程序
    type: "post",               //数据发送方式
    dataType: "json",           //接受数据格式
    data: {                     //要传递的数据
        username: function() {
            return $("#username").val();
        }
    }
}
```

远程地址只能输出 "true" 或 "false"，不能有其他输出。

9、添加自定义校验

```
addMethod：name, method, message
```

自定义验证方法

```
// 中文字两个字节
jQuery.validator.addMethod("byteRangeLength", function(value, element, param) {
    var length = value.length;
    for(var i = 0; i < value.length; i++){
        if(value.charCodeAt(i) > 127){
            length++;
        }
    }
  return this.optional(element) || ( length >= param[0] && length <= param[1] );
}, $.validator.format("请确保输入的值在{0}-{1}个字节之间(一个中文字算2个字节)"));

// 邮政编码验证
jQuery.validator.addMethod("isZipCode", function(value, element) {
    var tel = /^[0-9]{6}$/;
    return this.optional(element) || (tel.test(value));
}, "请正确填写您的邮政编码");
```

注意：要在 additional-methods.js 文件中添加或者在 jquery.validate.js 文件中添加。建议一般写在 additional-methods.js 文件中。

注意：在 messages_cn.js 文件中添加：isZipCode: "只能包括中文字、英文字母、数字和下划线"。调用前要添加对 additional-methods.js 文件的引用。

10、radio 和 checkbox、select 的验证

radio 的 required 表示必须选中一个。

```
<input  type="radio" id="gender_male" value="m" name="gender" required />
<input  type="radio" id="gender_female" value="f" name="gender"/>
```

checkbox 的 required 表示必须选中。

```
<input type="checkbox" class="checkbox" id="agree" name="agree" required />
```

checkbox 的 minlength 表示必须选中的最小个数，maxlength 表示最大的选中个数，rangelength:[2,3] 表示选中个数区间。

```
<input type="checkbox" class="checkbox" id="spam_email" value="email" name="spam[]" required minlength="2" />
<input type="checkbox" class="checkbox" id="spam_phone" value="phone" name="spam[]" />
```

```
<input type="checkbox" class="checkbox" id="spam_mail" value="mail" name="spam[]" />
```

select 的 required 表示选中的 value 不能为空。

```
<select id="jungle" name="jungle" title="Please select something!" required>
    <option value=""></option>
    <option value="1">Buga</option>
    <option value="2">Baga</option>
    <option value="3">Oi</option>
</select>
```

select 的 minlength 表示选中的最小个数（可多选的 select），maxlength 表示最大的选中个数，rangelength:[2,3] 表示选中个数区间。

```
<select id="fruit" name="fruit" title="Please select at least two fruits" class="{required:true, minlength:2}"
multiple="multiple">
    <option value="b">Banana</option>
    <option value="a">Apple</option>
    <option value="p">Peach</option>
    <option value="t">Turtle</option>
</select>
```

jQuery.validate 中文 API

名称	返回类型	描述
validate(options)	Validator	验证所选的 FORM。
valid()	Boolean	检查是否验证通过。
rules()	Options	返回元素的验证规则。
rules("add",rules)	Options	增加验证规则。
rules("remove",rules)	Options	删除验证规则。
removeAttrs(attributes)	Options	删除特殊属性并且返回它们。
自定义选择器		
:blank	Validator	没有值的筛选器。
:filled	Array <Element>	有值的筛选器。
:unchecked	Array <Element>	没选择的元素的筛选器。
实用工具		
jQuery.format(template,argument,argumentN...)	String	用参数代替模板中的 {n}。

Validator

validate 方法返回一个 Validator 对象。Validator 对象有很多方法可以用来引发校验程序或者改变 form 的内容，下面列出几个常用的方法。

名称	返回类型	描述
form()	Boolean	验证 form 返回成功还是失败。
element(element)	Boolean	验证单个元素是成功还是失败。
resetForm()	undefined	把前面验证的 FORM 恢复到验证前原来的状态。
showErrors(errors)	undefined	显示特定的错误信息。
Validator 函数		
setDefaults(defaults)	undefined	改变默认的设置。
addMethod(name,method,message)	undefined	添加一个新的验证方法。必须包括一个独一无二的名字，一个 JAVASCRIPT 的方法和一个默认的信息。
addClassRules(name,rules)	undefined	增加组合验证类型，在一个类里面用多种验证方法时比较有用。
addClassRules(rules)	undefined	增加组合验证类型，在一个类里面用多种验证方法时比较有用。这个是同时加多个验证方法。

内置验证方式

名称	返回类型	描述
required()	Boolean	必填验证元素。

required(dependency-expression)	Boolean	必填元素依赖于表达式的结果。
required(dependency-callback)	Boolean	必填元素依赖于回调函数的结果。
remote(url)	Boolean	请求远程校验。url 通常是一个远程调用方法。
minlength(length)	Boolean	设置最小长度。
maxlength(length)	Boolean	设置最大长度。
rangelength(range)	Boolean	设置一个长度范围 [min,max]。
min(value)	Boolean	设置最小值。
max(value)	Boolean	设置最大值。
email()	Boolean	验证电子邮箱格式。
range(range)	Boolean	设置值的范围。
url()	Boolean	验证 URL 格式。
date()	Boolean	验证日期格式（类似 30/30/2008 的格式，不验证日期准确性只验证格式）。
dateISO()	Boolean	验证 ISO 类型的日期格式。
dateDE()	Boolean	验证德式的日期格式（29.04.1994 或 1.1.2006）。
number()	Boolean	验证十进制数字（包括小数的）。
digits()	Boolean	验证整数。
creditcard()	Boolean	验证信用卡号。
accept(extension)	Boolean	验证相同后缀名的字符串。
equalTo(other)	Boolean	验证两个输入框的内容是否相同。
phoneUS()	Boolean	验证美式的电话号码。

validate ()的可选项

描述	代码
debug：进行调试模式（表单不提交）。	`$(".selector").validate({ debug:true })`
把调试设置为默认。	`$.validator.setDefaults({ debug:true })`
submitHandler：通过验证后运行的函数，里面要加上表单提交的函数，否则表单不会提交。	`$(".selector").validate({ submitHandler:function(form) { $(form).ajaxSubmit(); } })`
ignore：对某些元素不进行验证。	`$("#myform").validate({ ignore:".ignore" })`
rules：自定义规则，key:value 的形式，key 是要验证的元素，value 可以是字符串或对象。	`$(".selector").validate({ rules:{ name:"required", email:{ required:true, email:true } } })`

messages：自定义的提示信息，key:value 的形式，key 是要验证的元素，value 可以是字符串或函数。

```
$(".selector").validate({
    rules:{
        name:"required",
        email:{
            required:true,
            email:true
        }
    },
    messages:{
        name:"Name不能为空",
        email:{
            required:"E-mail不能为空",
            email:"E-mail地址不正确"
        }
    }
})
```

groups：对一组元素的验证，用一个错误提示，用 errorPlacement 控制把出错信息放在哪里。

```
$("#myform").validate({
    groups:{
        username:"fname
        lname"
    },
    errorPlacement:function(error,element) {
        if (element.attr("name") == "fname" || element.attr("name") ==
"lname")
            error.insertAfter("#lastname");
        else
            error.insertAfter(element);
    },
   debug:true
})
```

OnSubmit：类型 Boolean，默认 true，指定是否提交时验证。

```
$(".selector").validate({
    onsubmit:false
})
```

onfocusout：类型 Boolean，默认 true，指定是否在获取焦点时验证。

```
$(".selector").validate({
    onfocusout:false
})
```

onkeyup：类型 Boolean，默认 true，指定是否在敲击键盘时验证。

```
$(".selector").validate({
   onkeyup:false
})
```

onclick：类型 Boolean，默认 true，指定是否在鼠标点击时验证（一般验证 checkbox、radiobox）。

```
$(".selector").validate({
   onclick:false
})
```

focusInvalid：类型 Boolean，默认 true。提交表单后，未通过验证的表单（第一个或提交之前获得焦点的未通过验证的表单）会获得焦点。

```
$(".selector").validate({
   focusInvalid:false
})
```

focusCleanup：类型 Boolean，默认 false。当未通过验证的元素获得焦点时，移除错误提示（避免和 focusInvalid 一起使用）。

```
$(".selector").validate({
   focusCleanup:true
})
```

errorClass：类型 String，默认 "error"。指定错误提示的 css 类名，可以自定义错误提示的样式。

```
$(".selector").validate({
    errorClass:"invalid"
})
```

errorElement：类型 String，默认 "label"。指定使用什么标签标记错误。

```
$(".selector").validate
   errorElement:"em"
})
```

wrapper：类型 String，指定使用什么标签再把上边的 errorELement 包起来。

```
$(".selector").validate({
   wrapper:"li"
})
```

errorLabelContainer：类型 Selector，把错误信息统一放在一个容器里面。

```
$("#myform").validate({
    errorLabelContainer:"#messageBox",
    wrapper:"li",
    submitHandler:function() {
        alert("Submitted!")
    }
})
```

showErrors：跟一个函数，可以显示总共有多少个未通过验证的元素。

```
$(".selector").validate({
    showErrors:function(errorMap,errorList) {
        $("#summary").html("Your form contains " + this.numberOfInvalids() +
" errors,see details below.");
        this.defaultShowErrors();
    }
})
```

errorPlacement：跟一个函数，可以自定义错误放到哪里。

```
$("#myform").validate({
    errorPlacement:function(error,element) {
        error.appendTo(element.parent("td").next("td"));
   },
   debug:true
})
```

success：要验证的元素通过验证后的动作，如果跟一个字符串，会当作一个 css 类，也可跟一个函数。

```
$("#myform").validate({
    success:"valid",
        submitHandler:function() {
            alert("Submitted!")
        }
})
```

highlight：可以给未通过验证的元素加效果、闪烁等。

addMethod(name,method,message)方法

参数 name 是添加的方法的名字。

参数 method 是一个函数，接收三个参数 (value,element,param) 。
value 是元素的值，element 是元素本身，param 是参数。

我们可以用 addMethod 来添加除内置的 Validation 方法之外的验证方法。比如有一个字段，只能输一个字母，范围是 a-f，写法如下：

```
$.validator.addMethod("af",function(value,element,params){
    if(value.length>1){
        return false;
    }
    if(value>=params[0] && value<=params[1]){
        return true;
    }else{
        return false;
    }
},"必须是一个字母,且a-f");
```

如果有个表单字段的 name="username"，则在 rules 中写：

```
username:{
   af:["a","f"]
}
```

addMethod 的第一个参数，是添加的验证方法的名字，这时是 af。
addMethod 的第三个参数，是自定义的错误提示，这里的提示为:"必须是一个字母,且a-f"。
addMethod 的第二个参数，是一个函数，这个比较重要，决定了用这个验证方法时的写法。

如果只有一个参数，直接写，比如 af"a"，那么 a 就是这个唯一的参数，如果多个参数，则写在 [] 里，用逗号分开。

meta String 方式

```
$("#myform").validate({

   meta:"validate",

   submitHandler:function() {
alert("Submitted!") }

})

<script type="text/javascript"
src="js/jquery.metadata.js"></script>

<script type="text/javascript"
src="js/jquery.validate.js"></script>

<form id="myform">

  <input type="text"
name="email" class="{validate:{ required:true,email:true }}" />

  <input type="submit"
value="Submit" />

</form>
```

实例演示

虚构的实例

- 错误消息容器
- 自定义消息作为元素数据
- radio（单选按钮）、checkbox（复选按钮）和 select（下拉框）
- 与表单（Form）插件的交互（AJAX 提交）
- 自定义方法和消息显示
- 动态表单
- 使用 jQuery UI Themeroller 定义表单样式
- TinyMCE - 一个轻量级的基于浏览器的所见即所得编辑器
- 文件输入框
- jQuery Mobile 表单验证

现实世界的实例

- Milk 注册表单
- Marketo 注册表单
- 房屋买卖折叠面板表单
- 远程 CAPTCHA（验证码）验证

实例下载

点击下载 官方实例

jQuery Accordion

jQuery Accordion 插件用于创建折叠菜单。它通常与嵌套的列表、定义列表或嵌套的 div 一起使用。选项用于指定结构、激活的元素和定制的动画。

该插件现在是 jQuery UI 的一部分，独立的版本不会再更新了。目前版本是 1.6。

jQuery Accordion 官网，jQuery Accordion 插件下载：http://www.runoob.com/try/download/jquery-accordion.zip。

如需了解更多有关 Accordion 的细节，请查看 API 文档 折叠面板部件（Accordion Widget）。

标准

标准代码如下:

```
jQuery('#list1a').accordion();
jQuery('#list1b').accordion({
    autoheight: false
});
```

导航

带有锚和嵌套列表的无序列表

```
jQuery('#navigation').accordion({
    active: false,
    header: '.head',
    navigation: true,
    event: 'mouseover',
    fillSpace: true,
    animated: 'easeslide'
});
```

带选项

容器是一个定义列表，标题是 dt，内容是 dd。

```
jQuery('#list2').accordion({
    event: 'mouseover',
    active: '.selected',
    selectedClass: 'active',
    animated: "bounceslide",
    header: "dt"
}).bind("change.ui-accordion", function(event, ui) {
    jQuery('<div>' + ui.oldHeader.text() + ' hidden, ' + ui.newHeader.text() + ' shown</div>').appendTo('#log');
});
```

查看演示 源码下载

jQuery Autocomplete

jQuery Autocomplete 插件根据用户输入值进行搜索和过滤，让用户快速找到并从预设值列表中选择。通过给 Autocomplete 字段焦点或者在其中输入字符，插件开始搜索匹配的条目并显示供选择的值的列表。通过输入更多的字符，用户可以过滤列表以获得更好的匹配。

该插件现在是 jQuery UI 的一部分，独立的版本不会再更新了。目前版本是 1.6。

访问 jQuery Autocomplete 官网，下载 jQuery Autocomplete 插件。

如需了解更多有关 Autocomplete 的细节，请查看 API 文档 自动完成部件（Autocomplete Widget）。

实例演示

jQuery Autocomplete 插件演示。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI Autocomplete - Default functionality</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.11.4/themes/smoothness/jquery-ui.css">
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.11.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="/resources/demos/style.css">
  <script>
  $(function() {
    var availableTags = [
      "ActionScript",
      "AppleScript",
      "Asp",
      "BASIC",
      "C",
      "C++",
      "Clojure",
      "COBOL",
      "ColdFusion",
      "Erlang",
      "Fortran",
      "Groovy",
      "Haskell",
      "Java",
      "JavaScript",
      "Lisp",
      "Perl",
      "PHP",
      "Python",
      "Ruby",
      "Scala",
      "Scheme"
    ];
    $( "#tags" ).autocomplete({
      source: availableTags
    });
  });
  </script>
</head>
<body>

<div class="ui-widget">
  <label for="tags">Tags: </label>
  <input id="tags">
</div>


</body>
```

```
</html>
```

查看演示

jQuery Growl 插件(消息提醒)

jQuery Growl 插件(消息提醒) 允许您很容易地在一个覆盖层显示反馈消息。消息会在一段时间后自动消失，不需要单击"确定"按钮等。用户也可以通过移动鼠标或点击关闭按钮加快隐藏信息。

该插件目前版本是 1.0.0。

访问 jQuery Growl 官网，下载 jQuery Growl 插件。

效果如下：

jQuery Password Validation（密码验证）

jQuery Password Validation（密码验证）插件扩展了 jQuery Validate 插件，提供了两种组件：
一种评价密码的相关因素的功能：比如大小写字母的混合情况、字符（数字、特殊字符）的混合情况、长度、与用户名的相似度（可选的）。
一种使用评价功能显示密码强度的验证插件自定义方法。显示的文本可以被本地化。

您可以简单地自定义强度显示的外观、本地化消息显示，并集成到已有的表单中。

该插件目前版本是 1.0.0。

使用方式

如需使用 Password Validation（密码验证）插件，请添加一个 class "password" 到 input，同时添加显示强度的基本标记在表单的需要显示的地方：

```
<form id="register">
    <label for="password">Password:</label>
    <input class="password" name="password" id="password" />
    <div class="password-meter">
        <div class="password-meter-message"> </div>
        <div class="password-meter-bg">
            <div class="password-meter-bar"></div>
        </div>
    </div>
</form>
```

对表单应用 Validate 插件：

```
$(document).ready(function() {
  $("#register").validate();
});
```

您可以重载 $.validator.passwordRating 实现不同的评价方法。或者重载 $.validator.passwordRating.messages 来提供其他消息，比如本地化。

实例演示

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Makes "field" required to be the same as #other</title>
<link rel="stylesheet" href="http://jqueryvalidation.org/files/demo/site-demos.css">

</head>
<body>
<form id="myform">
<label for="password">Password</label>
<input id="password" name="password" />
<br/>
<label for="password_again">Again</label>
<input class="left" id="password_again" name="password_again" />
<br>
<input type="submit" value="Validate!">
</form>
<script src="http://code.jquery.com/jquery-1.11.1.min.js"></script>
<script src="http://jqueryvalidation.org/files/dist/jquery.validate.min.js"></script>
<script src="http://jqueryvalidation.org/files/dist/additional-methods.min.js"></script>
<script>
// just for the demos, avoids form submit
jQuery.validator.setDefaults({
  debug: true,
```

```
  success: "valid"
});
$( "#myform" ).validate({
  rules: {
    password: "required",
    password_again: {
      equalTo: "#password"
    }
  }
});
</script>
</body>
</html>
```

查看演示

jQuery Prettydate

jQuery Prettydate 插件为表单提供了强大的验证功能，让客户端表单验证变得更简单，同时提供了大量的定制选项，满足应用程序各种需求。该插件捆绑了一套有用的验证方法，包括 URL 和电子邮件验证，同时提供了一个用来编写用户自定义方法的 API。所有的捆绑方法默认使用英语作为错误信息，且已翻译成其他 37 种语言。

该插件目前版本是 1.1.0。

下载 jQuery Prettydate Validation（密码验证）插件。

使用方式

如需使用 Prettydate 插件，您需要在 title 中带有 ISO8601 日期：

```
<a title="2008-01-28T20:24:17Z">January 28th, 2008</a>
<a title="2008-01-27T22:24:17Z">January 27th, 2008</a>
<a title="2008-01-26T22:24:17Z">January 26th, 2008</a>
```

然后对它们应用 prettyDate 方法：

```
$(function() { $("a").prettyDate(); });
```

如需本地化该插件，请在 $.prettyDate.messages 中重写属性。在这里，以德国本地化为例：

```
$.prettyDate.messages = { now: "gerade eben", minute: "vor einer Minute", minutes:
$.prettyDate.template("vor {0} Minuten"), hour: "vor einer Stunde", hours: $.prettyDate.template("vor
{0} Stunden"), yesterday: "Gestern", days: $.prettyDate.template("vor {0} Tagen"), weeks:
$.prettyDate.template("vor {0} Wochen") }
```

该插件每隔 10 秒中更新一次每个被选中的元素。这样子 "just now" 会变为 "1 minute ago" 再变为 "x minutes ago" 再变为 "1 hour ago" 等等。

您可以通过指定 interval 选项为 "false" 来禁用间隔更新：

```
$(function() { $("a").prettyDate({ interval: false }); });
```

或者设置一个不同的时间间隔，例如：interval: 1000，每隔一秒更新一次每个被选中的元素：

```
$(function() { $("a").prettyDate({ interval: 1000 }); });
```

value 选项默认读取 title 属性中的 ISO8601 日期字符串。重载该选项来使用其他属性，例如，一个自定义的 "isodate" 属性：

```
$(function() {
    $("a").prettyDate({
        function() { // "this" 是 DOM 元素
            return $(this).attr("isodate");
        }
    });
});
```

jQuery Tooltip

jQuery Tooltip 插件取代了原生的工具提示框，让它们可自定义，您只需要调整它们的内容、位置和外观即可。

该插件目前版本是 1.3，已经很长时间没有更新，推荐使用 jQuery UI 工具提示框（Tooltip）。

如需了解更多有关 jQuery UI 工具提示框（Tooltip）的细节，请查看 API 文档 jQuery UI 工具提示框部件（Tooltip Widget）。

实例演示

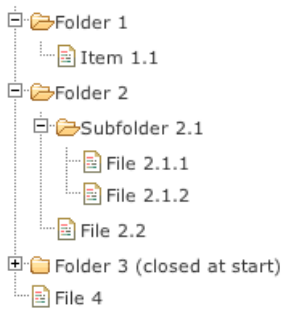
jQuery Tooltip 插件演示。

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <title>jQuery UI Tooltip - Default functionality</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.11.4/themes/smoothness/jquery-ui.css">
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.11.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="/resources/demos/style.css">
  <script>
  $(function() {
    $( document ).tooltip();
  });
  </script>
  <style>
  label {
    display: inline-block;
    width: 5em;
  }
  </style>
</head>
<body>

<p><a href="#" title="That&apos;s what this widget is">Tooltips</a> can be attached to any element. When you hover
the element with your mouse, the title attribute is displayed in a little box next to the element, just like a native tooltip.</p>
<p>But as it's not a native tooltip, it can be styled. Any themes built with
<a href="http://jqueryui.com/themeroller/" title="ThemeRoller: jQuery UI&apos;s theme builder application">ThemeRoller</a>
will also style tooltips accordingly.</p>
<p>Tooltips are also useful for form elements, to show some additional information in the context of each field.</p>
<p><label for="age">Your age:</label><input id="age" title="We ask for your age only for statistical purposes."></p>
<p>Hover the field to see the tooltip.</p>


</body>
</html>
```

查看演示

jQuery 树型菜单插件(Treeview)

jQuery Treeview 提供了一个无序灵活的可折叠的树形菜单。适用于一些菜单的导航，支持基于 cookie 的持久性菜单。




```
<ul id="browser" class="filetree treeview-famfamfam">
    <li><span class="folder">Folder 1</span>
        <ul>
            <li><span class="folder">Item 1.1</span>
                <ul>
                    <li><span class="file">Item 1.1.1</span></li>
                </ul>
            </li>
            <li><span class="folder">Folder 2</span>
                <ul>
                    <li><span class="folder">Subfolder 2.1</span>
                        <ul id="folder21">
                            <li><span class="file">File 2.1.1</span></li>
                            <li><span class="file">File 2.1.2</span></li>
                        </ul>
                    </li>
                    <li><span class="folder">Subfolder 2.2</span>
                        <ul>
                            <li><span class="file">File 2.2.1</span></li>
                            <li><span class="file">File 2.2.2</span></li>
                        </ul>
                    </li>
                </ul>
            </li>
            <li class="closed"><span class="folder">Folder 3 (closed at start)</span>
                <ul>
                    <li><span class="file">File 3.1</span></li>
                </ul>
            </li>
            <li><span class="file">File 4</span></li>
        </ul>
    </li>
</ul>
```

尝试一下 »

相关链接

jQuery Treeview 完整实例下载：http://static.runoob.com/download/jquery-treeview.zip。

jQuery Treeview Github 地址：https://github.com/jzaefferer/jquery-treeview。