目录

[jQuery 教程 3](#_Toc33798690)

[安装 3](#_Toc33798691)

[网页中添加 jQuery 3](#_Toc33798692)

[下载 jQuery 3](#_Toc33798693)

[替代方案 4](#_Toc33798694)

[jQuery 使用版本 5](#_Toc33798695)

[语法 5](#_Toc33798696)

[文档就绪事件 6](#_Toc33798697)

[选择器 7](#_Toc33798698)

[元素选择器 7](#_Toc33798699)

[#id 选择器 8](#_Toc33798700)

[.class 选择器 8](#_Toc33798701)

[更多实例 9](#_Toc33798702)

[独立文件中使用 jQuery 函数 10](#_Toc33798703)

[事件 11](#_Toc33798704)

[jQuery 事件方法语法 11](#_Toc33798705)

[常用的 jQuery 事件方法 12](#_Toc33798706)

[隐藏和显示 12](#_Toc33798707)

[jQuery hide() 和 show() 12](#_Toc33798708)

[jQuery toggle() 14](#_Toc33798709)

[淡入淡出 15](#_Toc33798710)

[Fading 方法 15](#_Toc33798711)

[fadeIn() 方法 15](#_Toc33798712)

[fadeOut() 方法 16](#_Toc33798713)

[fadeToggle() 方法 17](#_Toc33798714)

[fadeTo() 方法 18](#_Toc33798715)

[滑动 18](#_Toc33798716)

[slideDown() 方法 19](#_Toc33798717)

[slideUp() 方法 20](#_Toc33798718)

[slideToggle() 方法 20](#_Toc33798719)

[动画 21](#_Toc33798720)

[animate() 方法 21](#_Toc33798721)

[停止动画 25](#_Toc33798722)

[stop() 方法 25](#_Toc33798723)

[Callback 方法 28](#_Toc33798724)

[jQuery 动画的问题 28](#_Toc33798725)

[链(Chaining) 29](#_Toc33798726)

[jQuery 方法链接 30](#_Toc33798727)

[End 31](#_Toc33798728)

## jQuery 教程

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。

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。目前jQuery兼容于所有主流浏览器。

## 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>

注：**为什么没有在 <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。

Staticfile CDN、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。

* Staticfile CDN:

<head>

<script src="https://cdn.staticfile.org/jquery/1.10.2/jquery.min.js"></script>

</head>

* 百度 CDN:

<head>

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

</head>

* Google CDN:

<head>

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

</head>

* Microsoft CDN:

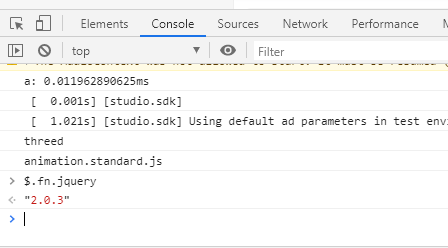
<head>

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

</head>

### jQuery 使用版本

我们可以在浏览器的 Console 窗口中使用 **$.fn.jquery** 命令查看当前 jQuery 使用的版本：



## 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

### 文档就绪事件

$(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...

});

与JavaScript入口函数对比：

JavaScript 入口函数:

window.onload = function () {

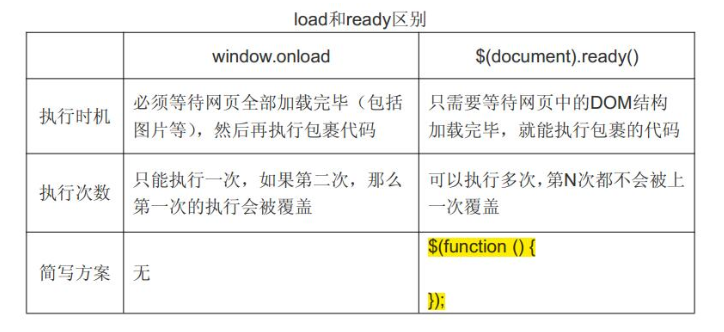
// 执行代码

}

jQuery 入口函数与 JavaScript 入口函数的区别：

 jQuery 的入口函数是在 html 所有标签(DOM)都加载之后，就会去执行。

 JavaScript 的 window.onload 事件是等到所有内容，包括外部图片之类的文件加载完后，才会执行。



## 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

<h2>这是一个标题</h2>

<p>这是一个段落。</p>

<p>这是另一个段落。</p>

<button>点我</button>

<script>

$(document).ready(function () {

$("button").click(function () {

$("p").hide();

});

});

</script>

### #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

<h2>这是一个标题</h2>

<p>这是一个段落</p>

<p id="test">这是另外一个段落</p>

<button>点我</button>

<script>

$(document).ready(function () {

$("button").click(function () {

$("#test").hide();

});

});

</script>

### .class 选择器

jQuery 类选择器可以通过指定的 class 查找元素。

语法： $(".test")

**实例：**用户点击按钮后所有带有 class="test" 属性的元素都隐藏

<h2 class="test">这是一个标题</h2>

<p class="test">这是一个段落。</p>

<p>这是另外一个段落。</p>

<button>点我</button>

<script>

$(document).ready(function () {

$("button").click(function () {

$(".test").hide();

});

});

</script>

### 更多实例

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

关于 **:** 和 **[]** 这两个符号的理解

**：**可以理解为种类的意思，如：**p:first**，**p** 的种类为第一个。

**[]** 很自然的可以理解为属性的意思，如：**[href]** 选取带有 **href** 属性的元素。

**$(":button")** 为 jQuery 中表单选择器（貌似与过滤选择器同级），旨在选择所有的按钮，所以会找到 **<input>**、**<button>** 元素；而 **$("button")** 则为基本选择器，旨在选择为 **<button>** 的标签。

**:** 即为 jQuery 的过滤选择器，语法类似于 css 中的伪类选择器；其过滤选择器大概可以分为基本过滤（p:first 之类）、内容过滤（:empty）、子元素过滤(:first-child)和属性过滤 **[href]** 选择器。

$("#id", ".class") //复合选择器

$(div p span) //层级选择器 //div下的p元素中的span元素

$(div > p) //父子选择器 //div下的所有p元素

$(div + p) //相邻元素选择器 //div后面的p元素(仅一个p)

$(div~p) //兄弟选择器 //div后面的所有p元素(同级别)

$(.p: last) //类选择器 加 过滤选择器 第一个和最后一个（first 或者 last）

$("#mytable td:odd") //层级选择 加 过滤选择器 奇偶（odd 或者 even）

$("div p:eq(2)") //索引选择器 div下的第三个p元素（索引是从0开始）

$("a[href='www.baidu.com']") //属性选择器

$("p:contains(test)") // 内容过滤选择器，包含text内容的p元素

$(":emtyp") //内容过滤选择器，所有空标签（不包含子标签和内容的标签）parent 相反

$(":hidden") //所有隐藏元素 visible

$("input:enabled") //选取所有启用的表单元素

$(":disabled") //所有不可用的元素

$("input:checked") //获取所有选中的复选框单选按钮等

$("select option:selected") //获取选中的选项元素

### 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）：

<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>

## 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。

常见 DOM 事件：

|  |  |  |  |
| --- | --- | --- | --- |
| **鼠标事件** | **键盘事件** | **表单事件** | **文档/窗口事件** |
| [click](https://www.runoob.com/jquery/event-click.html) | [keypress](https://www.runoob.com/jquery/event-keypress.html) | [submit](https://www.runoob.com/jquery/event-submit.html) | [load](https://www.runoob.com/jquery/event-load.html) |
| [dblclick](https://www.runoob.com/jquery/event-dblclick.html) | [keydown](https://www.runoob.com/jquery/event-keydown.html) | [change](https://www.runoob.com/jquery/event-change.html) | [resize](https://www.runoob.com/jquery/event-resize.html) |
| [mouseenter](https://www.runoob.com/jquery/event-mouseenter.html) | [keyup](https://www.runoob.com/jquery/event-keyup.html) | [focus](https://www.runoob.com/jquery/event-focus.html) | [scroll](https://www.runoob.com/jquery/event-scroll.html) |
| [mouseleave](https://www.runoob.com/jquery/event-mouseleave.html) |  | [blur](https://www.runoob.com/jquery/event-blur.html) | [unload](https://www.runoob.com/jquery/event-unload.html) |
| [hover](https://www.runoob.com/jquery/event-hover.html) |  |  |  |

### 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() 方法允许我们在文档完全加载完后执行函数。
* 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。
* dblclick() 当双击元素时，会发生 dblclick 事件。
* mouseenter() 当鼠标指针穿过元素时，会发生 mouseenter 事件。
* mouseleave() 当鼠标指针离开元素时，会发生 mouseleave 事件。
* 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。
* mouseup() 当在元素上松开鼠标按钮时，会发生 mouseup 事件。
* 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)。
* 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。
* blur() 当元素失去焦点时，发生 blur 事件。

## 隐藏和显示

### jQuery hide() 和 show()

#### 实例：使用 hide() 和 show() 方法来隐藏和显示 HTML 元素

<p>如果你点击“隐藏” 按钮，我将会消失。</p>

<button id="hide">隐藏</button>

<button id="show">显示</button>

<script>

$(document).ready(function () {

$("#hide").click(function () {

$("p").hide();

});

$("#show").click(function () {

$("p").show();

});

});

</script>

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。

#### 实例：使用 hide() 和 show() 方法+显示速度

<p>如果你点击“隐藏” 按钮，我将会消失。</p>

<button id="hide">隐藏</button>

<button id="show">显示</button>

<script>

$(document).ready(function () {

$("#hide").click(function () {

$("p").hide(1000);

});

$("#show").click(function () {

$("p").show(1000);

});

});

</script>

#### 实例：使用 hide() 和 show() 方法+显示速度

<p>如果你点击“隐藏” 按钮，我将会消失。</p>

<button id="hide">隐藏</button>

<button id="show">显示</button>

<script>

$(document).ready(function () {

$("#hide").click(function () {

$("p").hide(1000, "linear", function () {

alert("Hide() 方法已完成!");

});

});

$("#show").click(function () {

$("p").show(1000, "linear", function () {

alert("Show() 方法已完成!");

});

});

});

</script>

第二个参数是一个字符串，表示过渡使用哪种缓动函数。（译者注：jQuery自身提供"linear" 和 "swing"，其他可以使用相关的插件）。

### jQuery toggle()

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

#### 实例：显示被隐藏的元素，并隐藏已显示的元素

<button>隐藏/显示</button>

<p>这是一个文本段落。</p>

<p>这是另外一个文本段落。</p>

<script>

$(document).ready(function () {

$("button").click(function () {

$("p").toggle(1000);

});

});

</script>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。

对于可选的 callback 参数，有以下两点说明：

1.$(*selector*)选中的元素的个数为n个，则callback函数会执行n次；

2.callback函数名后加括号，会立刻执行函数体，而不是等到显示/隐藏完成后才执行；

3.callback既可以是函数名，也可以是匿名函数；

## 淡入淡出

### 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()

注：

* 注意大小写，fadeIn()   fadeOut()   fadeToggle()   fadeTo()  大小写不能变。
* fadeTo()  没有默认参数，必须加上  slow/fast/Time

### fadeIn() 方法

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。

#### 实例：淡入已隐藏的元素

<p>以下实例演示了 fadeIn() 使用了不同参数的效果。</p>

<button>点击淡入 div 元素。</button>

<br><br>

<div id="div1" style="width:80px;height:80px;display:none;background-color:red;"></div><br>

<div id="div2" style="width:80px;height:80px;display:none;background-color:green;"></div><br>

<div id="div3" style="width:80px;height:80px;display:none;background-color:blue;"></div>

<script>

$(document).ready(function () {

$("button").click(function () { //以下代码同时执行

$("#div1").fadeIn();

$("#div2").fadeIn("slow");

$("#div3").fadeIn(3000);

});

});

</script>

## fadeOut() 方法

 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。

#### 实例：淡出可见元素

<p>以下实例演示了 fadeOut() 使用了不同参数的效果。</p>

<button>点击淡出 div 元素。</button>

<br><br>

<div id="div1" style="width:80px;height:80px;background-color:red;"></div><br>

<div id="div2" style="width:80px;height:80px;background-color:green;"></div><br>

<div id="div3" style="width:80px;height:80px;background-color:blue;"></div>

<script>

$(document).ready(function () {

$("button").click(function () {

$("#div1").fadeOut();

$("#div2").fadeOut("slow");

$("#div3").fadeOut(3000);

});

});

</script>

### 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。

#### 实例：fadeToggle()

<p>实例演示了 fadeToggle() 使用了不同的 speed(速度) 参数。</p>

<button>点击淡入/淡出</button>

<br><br>

<div id="div1" style="width:80px;height:80px;background-color:red;"></div>

<br>

<div id="div2" style="width:80px;height:80px;background-color:green;"></div>

<br>

<div id="div3" style="width:80px;height:80px;background-color:blue;"></div>

<script>

$(document).ready(function () {

$("button").click(function () {

$("#div1").fadeToggle();

$("#div2").fadeToggle("slow");

$("#div3").fadeToggle(3000);

});

});

</script>

### 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。

#### 实例：允许渐变为给定的不透明度

<p>演示 fadeTo() 使用不同参数</p>

<button>点我让颜色变淡</button>

<br><br>

<div id="div1" style="width:80px;height:80px;background-color:red;"></div><br>

<div id="div2" style="width:80px;height:80px;background-color:green;"></div><br>

<div id="div3" style="width:80px;height:80px;background-color:blue;"></div>

<script>

$(document).ready(function () {

$("button").click(function () {

$("#div1").fadeTo("slow", 0.15);

$("#div2").fadeTo("slow", 0.4);

$("#div3").fadeTo("slow", 0.7);

});

});

</script>

## 滑动

jQuery 拥有以下滑动方法：

* slideDown()
* slideUp()
* slideToggle()

### 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。

<style type="text/css">

#panel, #flip {

padding: 5px;

text-align: center;

background-color: #e5eecc;

border: solid 1px #c3c3c3;

}

#panel {

padding: 50px;

display: none;

}

</style>

</head>

<body>

<div id="flip">点我滑下面板</div>

<div id="panel">Hello world!</div>

<script>

$(document).ready(function () {

$("#flip").click(function () {

$("#panel").slideDown("slow");

});

});

</script>

### 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。

<style type="text/css">

#panel, #flip {

padding: 5px;

text-align: center;

background-color: #e5eecc;

border: solid 1px #c3c3c3;

}

#panel {

padding: 50px;

}

</style>

</head>

<body>

<div id="flip">点我拉起面板</div>

<div id="panel">Hello world!</div>

<script>

$(document).ready(function () {

$("#flip").click(function () {

$("#panel").slideUp("slow");

});

});

</script>

### 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。

<style type="text/css">

#panel, #flip {

padding: 5px;

text-align: center;

background-color: #e5eecc;

border: solid 1px #c3c3c3;

}

#panel {

padding: 50px;

display: none;

}

</style>

</head>

<body>

<div id="flip">点我，显示或隐藏面板。</div>

<div id="panel">Hello world!</div>

<script>

$(document).ready(function () {

$("#flip").click(function () {

$("#panel").slideToggle("slow");

});

});

</script>

## 动画

### animate() 方法

用于创建自定义动画。

$(*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。

#### 实例：动画

<button>开始动画</button>

<p>默认情况下，所有的 HTML 元素有一个静态的位置，且是不可移动的。

如果需要改变为，我们需要将元素的 position 属性设置为 relative, fixed, 或 absolute!</p>

<div style="background:#98bf21;height:100px;width:100px;position:absolute;">

</div>

<script>

$(document).ready(function () {

$("button").click(function () {

$("div").animate({ left: '250px' });

});

});

</script>

注：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！

#### 实例：animate() - 操作多个属性

<button>开始动画</button>

<p>默认情况下，所有的 HTML 元素有一个静态的位置，且是不可移动的。

如果需要改变，我们需要将元素的 position 属性设置为 relative, fixed, 或 absolute!</p>

<div style="background:#98bf21;height:100px;width:100px;position:absolute;">

</div>

<script>

$(document).ready(function () {

$("button").click(function () {

$("div").animate({

left: '250px',

opacity: '0.5',

height: '150px',

width: '150px'

});

});

});

</script>

注：**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。

#### 实例：animate() - 使用相对值

<button>开始动画</button>

<p>默认情况下，所有的 HTML 元素有一个静态的位置，且是不可移动的。

如果需要改变为，我们需要将元素的 position 属性设置为 relative, fixed, 或 absolute!</p>

<div style="background:#98bf21;height:100px;width:100px;position:absolute;">

</div>

<script>

$(document).ready(function () {

$("button").click(function () {

$("div").animate({

left: '250px',

height: '+=150px',

width: '+=150px'

});

});

});

</script>

解析：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：animate() - 使用预定义的值

如可以把属性的动画值设置为 "show"、"hide" 或 "toggle"

<button>开始动画</button>

<p>默认情况下，所有的 HTML 元素有一个静态的位置，且是不可移动的。

如果需要改变为，我们需要将元素的 position 属性设置为 relative, fixed, 或 absolute!</p>

<div style="background:#98bf21;height:100px;width:100px;position:absolute;">

</div>

<script>

$(document).ready(function () {

$("button").click(function () {

$("div").animate({

height: 'toggle'

});

});

});

</script>

#### 实例：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。

例子1：

<button>开始动画</button>

<p>默认情况下，所有的 HTML 元素有一个静态的位置，且是不可移动的。

如果需要改变为，我们需要将元素的 position 属性设置为 relative, fixed, 或 absolute!</p>

<div style="background:#98bf21;height:100px;width:100px;position:absolute;">

</div>

<script>

$(document).ready(function () {

$("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");

});

});

</script>

例子2：

<button>开始动画</button>

<p>默认情况下，所有的 HTML 元素有一个静态的位置，且是不可移动的。

如果需要改变为，我们需要将元素的 position 属性设置为 relative, fixed, 或 absolute!</p>

<div style="background:#98bf21;height:100px;width:200px;position:absolute;">HELLO</div>

<script>

$(document).ready(function () {

$("button").click(function () {

var div = $("div");

div.animate({ left: '100px' }, "slow");

div.animate({ fontSize: '3em' }, "slow");

});

});

</script>

## 停止动画

### 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。

$(*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。

#### 实例：停止动画

<style type="text/css">

#panel, #flip {

padding: 5px;

text-align: center;

background-color: #e5eecc;

border: solid 1px #c3c3c3;

}

#panel {

padding: 50px;

display: none;

}

</style>

</head>

<body>

<button id="stop">停止滑动</button>

<div id="flip">点我向下滑动面板</div>

<div id="panel">Hello world!</div>

<script>

$(document).ready(function () {

$("#flip").click(function () {

$("#panel").slideDown(5000);

});

$("#stop").click(function () {

$("#panel").stop();

});

});

</script>

#### 实例：只停止当前正在进行的动画

<style type="text/css">

#panel, #flip {

padding: 5px;

text-align: center;

background-color: #e5eecc;

border: solid 1px #c3c3c3;

}

#panel {

padding: 50px;

display: none;

}

</style>

<button id="stop">停止滑动</button>

<div id="flip">点我向下滑动面板</div>

<div id="panel">Hello world!</div>

<script>

$(document).ready(function () {

$("#flip").click(function () {

//以下有两个动画，当stop()作用时，仅仅停止当前动画，未完成的动画也会继续进行

$("#panel").slideDown(5000);

$("#panel").slideUp(5000);

});

$("#stop").click(function () {

$("#panel").stop();

});

});

</script>

进阶：可以在 stop() 中设置 stopAll 的参数为 true，这样就可以停止所有动画效果而不是只停止当前动画

<style type="text/css">

#panel, #flip {

padding: 5px;

text-align: center;

background-color: #e5eecc;

border: solid 1px #c3c3c3;

}

#panel {

padding: 50px;

display: none;

}

</style>

</head>

<body>

<button id="stop">停止滑动</button>

<div id="flip">点我向下滑动面板</div>

<div id="panel">Hello world!</div>

<script>

$(document).ready(function () {

$("#flip").click(function () {

$("#panel").slideDown(5000);

$("#panel").slideUp(5000);

});

$("#stop").click(function () {

$("#panel").stop(true); //当参数为true时，将停止指定元素的所有动画

});

});

</script>

## Callback 方法

### 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。

Callback 函数

在当前动画 100% 完成之后执行。

#### 实例：Callback函数

<button>隐藏</button>

<p>我们段落内容，点击“隐藏”按钮我就会消失</p>

<script>

$(document).ready(function () {

$("button").click(function () {

$("p").hide("slow", function () {

alert("段落现在被隐藏了"); //警告框将在隐藏动画后执行

});

});

});

</script>

对比：以下实例没有回调函数，警告框会在隐藏效果完成前弹出：

$(document).ready(function () {

$("button").click(function () {

$("p").hide(1000);

alert("段落现在被隐藏了"); //警告框将于隐藏代码同时执行

});

});

注：被立即停止的动画不会触发回调，被立即完成的动画会触发回调。

<button>隐藏</button>

<span id="happy" style="color:red;">点我停止动画</span>

<p>我们段落内容，点击“隐藏”按钮我就会消失</p>

<script>

$(document).ready(function () {

$("button").click(function () {

$("p").hide(3000, function () {

alert("段落现在被隐藏了");

});

});

$("#happy").click(function () {

$("p").stop(false, true); //第二参数为true时，表示立即完成动画，此时立即执行callback代码

});

});

</script>

## 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。

#### 实例：方法链接

<p id="p1">菜鸟教程!!</p>

<button>点我</button>

<script>

$(document).ready(function () {

$("button").click(function () {

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

//"p1" 元素首先会变为红色，然后向上滑动，再然后向下滑动：

});

});

</script>

**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：

$("#p1").css("color", "red")

.slideUp(2000)

.slideDown(2000);

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。

## 获取内容和属性

### DOM 操作

jQuery 中非常重要的部分，就是操作 DOM 的能力。DOM = Document Object Model（文档对象模型）

### 获得内容 - text()、html() 以及 val()

三个简单实用的用于 DOM 操作的 jQuery 方法：

* 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

#### 实例：通过 jQuery text() 和 html() 方法来获得内容

<p id="test">这是段落中的 <b>粗体</b> 文本。</p>

<button id="btn1">显示文本</button>

<button id="btn2">显示 HTML</button>

<script>

$(document).ready(function () {

$("#btn1").click(function () {

alert("Text: " + $("#test").text()); //通过.test()方法获取所选元素的文本内容

});

$("#btn2").click(function () {

alert("HTML: " + $("#test").html()); //通过.html()方法获取所选元素的内容（包括 HTML 标记）

});

});

</script>

#### 实例：通过 jQuery val() 方法获得输入字段的值

<p>名称: <input type="text" id="test" value="菜鸟教程"></p>

<button>显示值</button>

<script>

$(document).ready(function () {

$("button").click(function () {

alert("值为: " + $("#test").val()); //通过.val()获取表单的内容

});

});

</script>

### 获取属性 - attr()

attr() 方法用于获取属性值。

#### 实例：获取href属性的值

<p><a href="//www.runoob.com" id="runoob">菜鸟教程</a></p>

<button>显示 href 属性的值</button>

<script>

$(document).ready(function () {

$("button").click(function () {

alert($("#runoob").attr("href")); //通过.attr()获取href的值

});

});

</script>

**attr** 和 **prop** 的区别介绍：

对于 HTML 元素本身就带有的固有属性，在处理时，使用 **prop** 方法。

对于 HTML 元素我们自己自定义的 DOM 属性，在处理时，使用 **attr** 方法。

具有 true 和 false 两个属性的属性，如 checked, selected 或者 disabled 使用prop()

**prop()函数的结果:**

      1.如果有相应的属性，返回指定属性值。

      2.如果没有相应的属性，返回值是空字符串。

**attr()函数的结果:**

      1.如果有相应的属性，返回指定属性值。

      2.如果没有相应的属性，返回值是 undefined。

例子： <a href="https://www.runoob.com" target="\_self" class="btn">菜鸟教程</a>

这个例子里 **<a>** 元素的 DOM 属性有: **href、target** 和 **class**，这些属性就是 **<a>** 元素本身就带有的属性，也是 W3C 标准里就包含有这几个属性，或者说在 IDE 里能够智能提示出的属性，这些就叫做固有属性。处理这些属性时，建议使用 **prop** 方法。

例子： <a href="#" id="link1" action="delete" rel="nofollow">删除</a>

这个例子里 **<a>** 元素的 DOM 属性有: **href、id** 和 **action**，很明显，前两个是固有属性，而后面一个 **action** 属性是我们自己自定义上去的，**<a>** 元素本身是没有这个属性的。这种就是自定义的 DOM 属性。处理这些属性时，建议使用 **attr** 方法。

## 设置内容和属性

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

#### 实例：通过 text()、html() 以及 val() 方法来设置内容

<p id="test1">这是一个段落。</p>

<p id="test2">这是另外一个段落。</p>

<p>输入框: <input type="text" id="test3" value="菜鸟教程"></p>

<button id="btn1">设置文本</button>

<button id="btn2">设置 HTML</button>

<button id="btn3">设置值</button>

<script>

$(document).ready(function () {

$("#btn1").click(function () {

$("#test1").text("Hello world!");

});

$("#btn2").click(function () {

$("#test2").html("<b>Hello world!</b>");

});

$("#btn3").click(function () {

$("#test3").val("RUNOOB");

});

});

</script>

### 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。

#### 实例：带有回调函数的 text() 和 html()

<p id="test1">这是一个有 <b>粗体</b> 字的段落。</p>

<p id="test2">这是另外一个有 <b>粗体</b> 字的段落。</p>

<button id="btn1">显示 新/旧 文本</button>

<button id="btn2">显示 新/旧 HTML</button>

<script>

$(document).ready(function () {

$("#btn1").click(function () {

//使用.text()方法设置元素的内容；第一个参数为当前元素的下标，第二个参数为原始值

$("#test1").text(function (i, origText) {

//返回值为设置的新的内容值

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 + ")";

});

});

});

</script>

### 设置属性 - attr()

attr() 方法也用于设置/改变属性值。

$("button").click(function () {

$("#runoob").attr("href", "http://www.runoob.com/jquery");

});

attr() 方法也允许您同时设置多个属性。

$("button").click(function () {

$("#runoob").attr({

"href": "http://www.runoob.com/jquery",

"title": "jQuery 教程"

});

});

### attr() 的回调函数

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。

<p><a href="//www.runoob.com" id="runoob">菜鸟教程</a></p>

<button>修改 href 值</button>

<p>点击按钮修改后，可以点击链接查看 href 属性是否变化。</p>

<script>

$(document).ready(function () {

$("button").click(function () {

//回调函数第一个参数为当前元素下标，第二个参数为原始的值

$("#runoob").attr("href", function (i, origValue) {

return origValue + "/jquery";

});

});

});

</script>

## 添加元素

通过 jQuery，可以很容易地添加新元素/内容。

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

### append() 方法

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：在被选元素结尾插入内容（append()）

<p>这是一个段落。</p>

<p>这是另外一个段落。</p>

<ol>

<li>List item 1</li>

<li>List item 2</li>

<li>List item 3</li>

</ol>

<button id="btn1">添加文本</button>

<button id="btn2">添加列表项</button>

<script>

$(document).ready(function () {

$("#btn1").click(function () {

$("p").append(" <b>追加文本</b>。");

});

$("#btn2").click(function () {

$("ol").append("<li>追加列表项</li>");

});

});

</script>

### prepend() 方法

prepend() 方法在被选元素的开头插入内容。

#### 实例：在被选元素的开头插入内容（prepend()）

<p>这是一个段落。</p>

<p>这是另外一个段落。</p>

<ol>

<li>列表 1</li>

<li>列表 2</li>

<li>列表 3</li>

</ol>

<button id="btn1">添加文本</button>

<button id="btn2">添加列表项</button>

<script>

$(document).ready(function () {

$("#btn1").click(function () {

$("p").prepend("<b>在开头追加文本</b>。 ");

});

$("#btn2").click(function () {

$("ol").prepend("<li>在开头添加列表项</li>");

});

});

</script>

### 通过 append() 和 prepend() 方法添加若干新元素

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。

#### 实例：在被选元素末尾添加若干新元素（append()）

<p>这是一个段落。</p>

<button onclick="appendText()">追加文本</button>

<script>

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

}

</script>

### after() 和 before() 方法

after() 方法在被选元素之后插入内容。before() 方法在被选元素之前插入内容。

#### 实例：在被选元素前后插入内容

<img src="/images/logo.png" >

<br><br>

<button id="btn1">之前插入</button>

<button id="btn2">之后插入</button>

<script>

$(document).ready(function () {

$("#btn1").click(function () {

$("img").before("<b>之前</b>");

});

$("#btn2").click(function () {

$("img").after("<i>之后</i>");

});

});

</script>

### 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。

<img src="/images/logo2.png" >

<br><br>

<button onclick="afterText()">之后插入</button>

<script>

function afterText() {

var txt1 = "<b>I </b>"; // 使用 HTML 创建元素

var txt2 = $("<i></i>").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

}

</script>

参数可以是个 list:

function afterText() {

var txt1 = "<b>I </b>"; // 使用 HTML 创建元素

var txt2 = $("<i></i>").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

}

### 区别：append/prepend与after/before

append/prepend 是在选择元素内部嵌入。

<p>

<span class="s1">s1</span>

</p>

<script>

$("p").append('<span class="s2">s2</span>');

</script>

显示的结果：

<p>

<span class="s1">s1</span>

<span class="s2">s2</span>

</p>

after/before 是在元素外面追加。

<p>

<span class="s1">s1</span>

</p>

<script>

$("p").after('<span class="s2">s2</span>');

</script>

显示结果：

<p>

<span class="s1">s1</span>

</p>

<span class="s2">s2</span>

## 删除元素

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

### remove() 方法

 remove() 方法删除被选元素及其子元素。

<div id="div1" style="height:100px;width:300px;border:1px solid black;background-color:yellow;">

这是 div 中的一些文本。

<p>这是在 div 中的一个段落。</p>

<p>这是在 div 中的另外一个段落。</p>

</div>

<br>

<button>移除div元素</button>

<script>

$(document).ready(function () {

$("button").click(function () {

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

});

});

</script>

### empty() 方法

empty() 方法删除被选元素的子元素。

<div id="div1" style="height:100px;width:300px;border:1px solid black;background-color:yellow;">

这是 div 中的一些文本。

<p>这是在 div 中的一个段落。</p>

<p>这是在 div 中的另外一个段落。</p>

</div>

<br>

<button>清空div元素</button>

<script>

$(document).ready(function () {

$("button").click(function () {

$("#div1").empty(); //只删除子元素，不会删除被选元素本身

});

});

</script>

### 过滤被删除的元素

 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。

#### 实例：删除指定样式的所有 <p> 元素

<p>这是一个段落。</p>

<p class="italic"><i>这是另外一个段落。</i></p>

<p class="italic"><i>这是另外一个段落。</i></p>

<button>移除所有 class="italic" 的 p 元素。</button>

<script>

$(document).ready(function () {

$("button").click(function () {

$("p").remove(".italic"); //移除所有 class="italic" 的 p 元素

});

});

</script>

在使用 remove() 的过滤器删除时，不能删除带有过滤器的子元素。

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

**$(selector)** 语法的返回结果是一个元素的列表，即：将 **$("#div1")** 看作一个列表，**remove()** 中的筛选条件实际上是对这个列表中的元素进行筛选删除，而不会去删除这个列表中不存在的元素（子元素不在这个列表中）。

## 获取并设置 CSS 类

jQuery 拥有若干进行 CSS 操作的方法：

* 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

以下实例样式表：

<style>

.important {

font-weight: bold;

font-size: xx-large;

}

.blue {

color: blue;

}

</style>

### addClass() 方法

#### 实例：向不同的元素添加 class 属性。

<h1>标题 1</h1>

<h2>标题 2</h2>

<p>这是一个段落。</p>

<p>这是另外一个段落。</p>

<div>这是一些重要的文本!</div>

<br>

<button>为元素添加 class</button>

<script>

$(document).ready(function () {

$("button").click(function () {

$("h1,h2,p").addClass("blue"); //将指定样式附加给指定元素

$("div").addClass("important");

});

});

</script>

可以在 addClass() 方法中规定多个类：

$("button").click(function () {

$("body div:first").addClass("important blue");

});

### removeClass() 方法

#### 实例：在不同的元素中删除指定的 class 属性

<h1 class="blue">标题 1</h1>

<h2 class="blue">标题 2</h2>

<p class="blue">这是一个段落。</p>

<p class="important">这是另外一个段落。</p>

<br>

<button>从元素中移除 class</button>

<script>

$(document).ready(function () {

$("button").click(function () {

$("h1,h2,p").removeClass("blue"); //对指定元素移除指定的样式

});

});

</script>

### toggleClass() 方法

#### 实例：对被选元素进行添加/删除类的切换操作

<h1 class="blue">标题 1</h1>

<h2 class="blue">标题 2</h2>

<p class="blue">这是一个段落。</p>

<p>这是另外一个段落。</p>

<br>

<button>切换 class</button>

<script>

$(document).ready(function () {

$("button").click(function () {

$("h1,h2,p").toggleClass("blue"); //自动对指定元素使用或移除指定的class

});

});

</script>

其他：

addClass('c1 c2 ...' | function(i, c)) //添加一个或多个类。

removerClass('c1 c2 ...' | function(i, c)) //删除一个或多个类。

toggleClass('c1 c2 ...' | function(i, c), switch?) //切换一个或多个类的添加和删除。

第一个参数表示要添加或删除的类，既可以用类列表，也可以用函数返回值指定（i 是选择器选中的所有元素中当前对象的索引值，c 是当前对象的类名）。

**switch**: 布尔值，true 表示只添加，false 表示只删除。

## 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

语法：css("*propertyname*");

<h2>这是一个标题</h2>

<p style="background-color:#ff0000">这是一个段落。</p>

<p style="background-color:#00ff00">这是一个段落。</p>

<p style="background-color:#0000ff">这是一个段落。</p>

<button>返回第一个 p 元素的 background-color </button>

<script>

$(document).ready(function () {

$("button").click(function () {

alert("背景颜色 = " + $("p").css("background-color"));

//返回第一个 p 元素的 background-color

});

});

</script>

上例中只获取到了第一个 p 标签的 background-color，那么怎么才能获取到第二或第三个 p 的背景颜色呢，加入选择器 **:nth-child()**：

<h2>这是一个标题</h2>

<p style="background-color:#ff0000">这是一个段落。</p>

<p style="background-color:#00ff00">这是一个段落。</p>

<p style="background-color:#0000ff">这是一个段落。</p>

<button>返回第一个 p 元素的 background-color </button>

<script>

$("button").click(function () {

alert("p1背景颜色 = " + $("p:nth-child(2)").css("background-color"));

alert("p2背景颜色 = " + $("p:nth-child(3)").css("background-color"));

alert("p3背景颜色 = " + $("p:nth-child(4)").css("background-color"));

});

</script>

上例中用的 **nth-child** 太麻烦。

当页面的元素特别多，父子关系特别繁杂的时候，不可能一个个去数要的元素是 body 的第几个元素。

另一方面，页面可以动态的，body 下的索引容易变化，用这个方法指定某个元素非常不靠谱。

要选取第几个 p 的方法，正确的应该是这样：

$("button").click(function () {

alert("p1背景颜色 = " + $("p").eq(0).css("background-color"));

alert("p1背景颜色 = " + $("p").eq(1).css("background-color"));

alert("p1背景颜色 = " + $("p").eq(2).css("background-color"));

});

### 设置 CSS 属性

语法：css("*propertyname*","*value*");

<h2>这是一个标题</h2>

<p style="background-color:#ff0000">这是一个段落。</p>

<p style="background-color:#00ff00">这是一个段落。</p>

<p style="background-color:#0000ff">这是一个段落。</p>

<p>这是一个段落。</p>

<button>设置 p 元素的 background-color </button>

<script>

$(document).ready(function () {

$("button").click(function () {

$("p").css("background-color", "yellow"); //设置<p>元素背景颜色为黄色

});

});

</script>

### 设置多个 CSS 属性

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

<h2>这是一个标题</h2>

<p style="background-color:#ff0000">这是一个段落。</p>

<p style="background-color:#00ff00">这是一个段落。</p>

<p style="background-color:#0000ff">这是一个段落。</p>

<p>这是一个段落。</p>

<button>为 p 元素设置多个样式</button>

<script>

$(document).ready(function () {

$("button").click(function () {

$("p").css({ "background-color": "yellow", "font-size": "200%" }); //设置背景颜色和字体大小

});

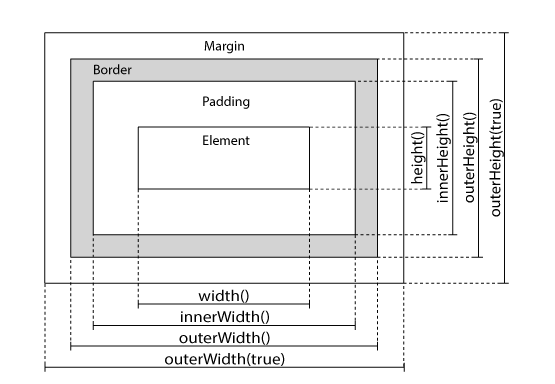
});

</script>

## 尺寸

jQuery 提供多个处理尺寸的重要方法：

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



### 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）。

#### 实例：获取元素宽度和高度

<div id="div1" style="height:100px;width:300px;padding:10px;margin:3px;border:1px solid blue;background-color:lightblue;"></div>

<br>

<button>显示 div 元素的尺寸</button>

<p>width() - 返回元素的宽度</p>

<p>height() - 返回元素的高度</p>

<script>

$(document).ready(function () {

$("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);

});

});

</script>

### 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）

<div id="div1" style="height:100px;width:300px;padding:10px;margin:3px;border:1px solid blue;background-color:lightblue;"></div>

<br>

<button>显示 div 元素的尺寸</button>

<p>innerWidth() - 返回元素的宽度 (包含内边距)。</p>

<p>innerHeight() - 返回元素的高度 (包含内边距)。</p>

<script>

$(document).ready(function () {

$("button").click(function () {

var txt = "";

txt += "div 宽度: " + $("#div1").width() + "</br>";

txt += "div 高度: " + $("#div1").height() + "</br>";

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);

});

});

</script>

### 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）。

<div id="div1" style="height:100px;width:300px;padding:10px;margin:3px;border:1px solid blue;background-color:lightblue;"></div>

<br>

<button>显示 div 元素的尺寸</button>

<p>outerWidth() - 返回元素的宽度 (包含内边距和边框)。</p>

<p>outerHeight() - 返回元素的高度 (包含内边距和边框)。</p>

<script>

$(document).ready(function () {

$("button").click(function () {

var txt = "";

txt += "div 宽度: " + $("#div1").width() + "</br>";

txt += "div 高度: " + $("#div1").height() + "</br>";

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);

});

});

</script>

注意：

设置了 box-sizing 后，width() 获取的是 css 设置的 width 减去 padding 和 border 的值。

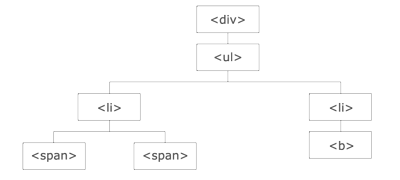
.test{width:100px;height:100px;padding:10px;border:10px;box-sizing:border-box;}

* width() 获取为: 60
* innerWidth() 获取值为: 80
* outWidth() 获取值为: 100

## 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：

* <div> 元素是 <ul> 的父元素，同时是其中所有内容的祖先。
* <ul> 元素是 <li> 元素的父元素，同时是 <div> 的子元素
* 左边的 <li> 元素是 <span> 的父元素，<ul> 的子元素，同时是 <div> 的后代。
* <span> 元素是 <li> 的子元素，同时是 <ul> 和 <div> 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，<ul> 的子元素，同时是 <div> 的后代。
* <b> 元素是右边的 <li> 的子元素，同时是 <ul> 和 <div> 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）。

## 祖先

通过 jQuery，您能够向上遍历 DOM 树，以查找元素的祖先。

这些 jQuery 方法很有用，它们用于向上遍历 DOM 树：

* parent()
* parents()
* parentsUntil()

### 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。

#### 实例：返回每个 <span> 元素的直接父元素

<style>

.ancestors \* {

display: block;

border: 2px solid lightgrey;

color: lightgrey;

padding: 5px;

margin: 15px;

}

</style>

<div class="ancestors">

<div style=" width: 500px;">div (曾祖父元素)

<ul>ul (祖父元素)

<li>li (父元素)

<span>span</span>

</li>

</ul>

</div>

​

<div style=" width: 500px;">div (祖父元素)

<p>p (父元素)

<span>span</span>

</p>

</div>

</div>

<script>

$(document).ready(function () {

$("span").parent().css({ "color": "red", "border": "2px solid red" });

});

</script>

### 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>)。

#### 实例：返回被选元素的所有祖先元素

<style>

.ancestors \* {

display: block;

border: 2px solid lightgrey;

color: lightgrey;

padding: 5px;

margin: 15px;

}

</style>

<script>

$(document).ready(function () {

$("span").parents().css({ "color": "red", "border": "2px solid red" });

});

</script>

</head>

<body class="ancestors">body (曾曾祖父元素)

<div style="width:500px;">div (曾祖父元素)

<ul>ul (祖父元素)

<li>li (父元素)

<span>span</span>

</li>

</ul>

</div>

#### 实例：使用可选参数来过滤对祖先元素的搜索。

<style>

.ancestors \* {

display: block;

border: 2px solid lightgrey;

color: lightgrey;

padding: 5px;

margin: 15px;

}

</style>

<script>

$(document).ready(function () {

$("span").parents("ul").css({ "color": "red", "border": "2px solid red" }); //只遍历ul元素

});

</script>

</head>

<body class="ancestors">body (great-great-grandparent)

<div style="width:500px;">div (great-grandparent)

<ul>ul (grandparent)

<li>li (direct parent)

<span>span</span>

</li>

</ul>

</div>

</body>

### parentsUntil() 方法

返回介于两个给定元素之间的所有祖先元素。

#### 实例：返回介于 <span> 与 <div> 元素之间的所有祖先元素

<style>

.ancestors \* {

display: block;

border: 2px solid lightgrey;

color: lightgrey;

padding: 5px;

margin: 15px;

}

</style>

<script>

$(document).ready(function () {

$("span").parentsUntil("div").css({ "color": "red", "border": "2px solid red" });

//不包含div标签本身

});

</script>

</head>

<body class="ancestors"> body (曾曾祖父元素)

<div style="width:500px;">div (曾祖父元素)

<ul>ul (祖父元素)

<li>li (父元素)

<span>span</span>

</li>

</ul>

</div>

</body>

## 后代

下面是两个用于向下遍历 DOM 树的 jQuery 方法：

* children()
* find()

### 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。

#### 实例：返回每个 <div> 元素的所有直接子元素

<style>

.descendants \* {

display: block;

border: 2px solid lightgrey;

color: lightgrey;

padding: 5px;

margin: 15px;

}

</style>

<script>

$(document).ready(function () {

$("div").children().css({ "color": "red", "border": "2px solid red" });

});

</script>

</head>

<body>

<div class="descendants" style="width:500px;">div (当前元素)

<p>p (儿子元素)

<span>span (孙子元素)</span>

</p>

<p>p (儿子元素)

<span>span (孙子元素)</span>

</p>

</div>

可以使用可选参数来过滤对子元素的搜索

#### 实例：返回类名为 "1" 的所有 <p> 元素，并且它们是 <div> 的直接子元素

<style>

.descendants \* {

display: block;

border: 2px solid lightgrey;

color: lightgrey;

padding: 5px;

margin: 15px;

}

</style>

<script>

$(document).ready(function () {

$("div").children("p.1").css({ "color": "red", "border": "2px solid red" });

});

</script>

<div class="descendants" style="width:500px;">div (当前元素)

<p class="1">p (儿子元素)

<span>span (孙子元素)</span>

</p>

<p class="2">p (儿子元素)

<span>span (孙子元素)</span>

</p>

</div>

### 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。

#### 实例：返回属于 <div> 后代的所有 <span> 元素

<style>

.descendants \* {

display: block;

border: 2px solid lightgrey;

color: lightgrey;

padding: 5px;

margin: 15px;

}

</style>

<script>

$(document).ready(function () {

$("div").find("span").css({ "color": "red", "border": "2px solid red" });

});

</script>

<div class="descendants" style="width:500px;">div (当前元素)

<p>p (儿子元素)

<span>span (孙子元素)</span>

</p>

<p>p (儿子元素)

<span>span (孙子元素)</span>

</p>

</div>

#### 实例：返回 <div> 的所有后代

<style>

.descendants \* {

display: block;

border: 2px solid lightgrey;

color: lightgrey;

padding: 5px;

margin: 15px;

}

</style>

<script>

$(document).ready(function () {

$("div").find("\*").css({ "color": "red", "border": "2px solid red" });

});

</script>

<div class="descendants" style="width:500px;">div (当前元素)

<p>p (儿子元素)

<span>span (孙子元素)</span>

</p>

<p>p (儿子元素)

<span>span (孙子元素)</span>

</p>

</div>

## 同胞(siblings)

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()

### siblings() 方法

siblings() 方法返回被选元素的所有同胞元素。

#### 实例：返回 <h2> 的所有同胞元素

<style>

.siblings \* {

display: block;

border: 2px solid lightgrey;

color: lightgrey;

padding: 5px;

margin: 15px;

}

</style>

<script>

$(document).ready(function () {

$("h2").siblings().css({ "color": "red", "border": "2px solid red" });

//不包括h2自身

});

</script>

</head>

<body class="siblings">

<div>div (父元素)

<p>p</p>

<span>span</span>

<h2>h2</h2>

<h3>h3</h3>

<p>p</p>

</div>

</body>

可以使用可选参数来过滤对同胞元素的搜索。

#### 实例：返回属于 <h2> 的同胞元素的所有 <p> 元素

<style>

.siblings \* {

display: block;

border: 2px solid lightgrey;

color: lightgrey;

padding: 5px;

margin: 15px;

}

</style>

<script>

$(document).ready(function () {

$("h2").siblings("p").css({ "color": "red", "border": "2px solid red" });

});

</script>

</head>

<body class="siblings">

<div>div (父元素)

<p>p</p>

<span>span</span>

<h2>h2</h2>

<h3>h3</h3>

<p>p</p>

</div>

### 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。

#### 实例：例子返回 <h2> 的下一个同胞元素

<style>

.siblings \* {

display: block;

border: 2px solid lightgrey;

color: lightgrey;

padding: 5px;

margin: 15px;

}

</style>

<script>

$(document).ready(function () {

$("h2").next().css({ "color": "red", "border": "2px solid red" });

});

</script>

</head>

<body class="siblings">

<div>div (父元素)

<p>p</p>

<span>span</span>

<h2>h2</h2>

<h3>h3</h3>

<p>p</p>

</div>

### nextAll() 方法

nextAll() 方法返回被选元素的所有跟随的同胞元素。

#### 实例：返回 <h2> 的所有跟随的同胞元素

<style>

.siblings \* {

display: block;

border: 2px solid lightgrey;

color: lightgrey;

padding: 5px;

margin: 15px;

}

</style>

<script>

$(document).ready(function () {

$("h2").nextAll().css({ "color": "red", "border": "2px solid red" });

});

</script>

</head>

<body class="siblings">

<div>div (父元素)

<p>p</p>

<span>span</span>

<h2>h2</h2>

<h3>h3</h3>

<p>p</p>

</div>

</body>

### 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。

#### 实例：返回介于 <h2> 与 <h6> 元素之间的所有同胞元素

<style>

.siblings \* {

display: block;

border: 2px solid lightgrey;

color: lightgrey;

padding: 5px;

margin: 15px;

}

</style>

<script>

$(document).ready(function () {

$("h2").nextUntil("h6").css({ "color": "red", "border": "2px solid red" });

});

</script>

</head>

<body class="siblings">

<div>div (父元素)

<p>p</p>

<span>span</span>

<h2>h2</h2>

<h3>h3</h3>

<h4>h4</h4>

<h5>h5</h5>

<h6>h6</h6>

<p>p</p>

</div>

</body>

### 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）。

## 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。

### first() 方法

first() 方法返回被选元素的首个元素。

$(document).ready(function () {

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

});

等价于：

$("p:first").ss("background-color","yellow");

### last() 方法

last() 方法返回被选元素的最后一个元素。

$(document).ready(function () {

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

});

等价于：

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

### 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。

$(document).ready(function () {

$("p").eq(1).css("background-color", "yellow");

});

### 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。

$(document).ready(function () {

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

//含class是url的元素

});

注意：

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

等价于

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

### 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。

$(document).ready(function () {

$("p").not(".url").css("background-color", "yellow");

//不含class是url的元素

});

等价于：

$("p:not(.url)");

not 和 eq 可以实现反选的效果：

选取索引值不为 1 的 p 元素，并把背景颜色设置为黄色：

$("p").not(":eq(1)").css("background-color","yellow");

## 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。

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。

## AJAX load() 方法

### load() 方法

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。

#### 实例：AJAX读取txt文件

<div id="div1"><h2>使用 jQuery AJAX 修改文本内容</h2></div>

<button>获取外部内容</button>

<script>

$(document).ready(function () {

$("button").click(function () {

$("#div1").load("{% static '/app/txt/demo\_test.txt' %}");

});

});

</script>

可以把 jQuery 选择器添加到 URL 参数。

#### 实例：AJAX读取TXT文件中指定内容

<script>

$(document).ready(function () {

$("button").click(function () {

$("#div1").load("{% static 'app/txt/demo\_test.txt' %} #p1");

//读取文件中 id="p1" 的元素的内容，加载到指定的 <div> 元素中

});

});

</script>

</head>

<body>

<div id="div1"><h2>使用 jQuery AJAX 修改文本</h2></div>

<button>获取外部文本</button>

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

#### 实例：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：

<script>

$(document).ready(function () {

$("button").click(function () {

$("#div1").load("{% static '/app/txt/demo\_test.txt'%}", function (responseTxt, statusTxt, xhr) {

if (statusTxt == "success") //判断调用是否成功

alert("外部内容加载成功!");

if (statusTxt == "error")

alert("Error: " + xhr.status + ": " + xhr.statusText); //显示错误代码和错误信息

});

});

});

</script>

<div id="div1"><h2>使用 jQuery AJAX 修改该文本</h2></div>

<button>获取外部内容</button>

为了避免多页面情形下的代码重复，可以利用 load() 方法，将重复的部分（例如导航栏）放入单独的文件，使用下列方法进行导入：

//1.当前文件中要插入的地方使用此结构：

<div class="include" file="\*\*\*.html"></div>

//2.\*\*\*.html中放入内容，用html格式仅仅因为会有编辑器的书写辅助。。

//3.代码：

$(".include").each(function() {

if (!!$(this).attr("file")) {

var $includeObj = $(this);

$(this).load($(this).attr("file"), function(html) {

$includeObj.after(html).remove(); //加载的文件内容写入到当前标签后面并移除当前标签

})

}

});

或者在index文件里只写重复部分，剩下的一股脑放各自单独文件 load() 进来~

## AJAX get() 和 post() 方法

### 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。

### $.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。

#### 实例：使用 $.get() 方法从服务器上的一个文件中取回数据

<script>

$(document).ready(function () {

$("button").click(function () {

//获取指定网页的内容

$.get("{% url 'home' %}", function (data, status) {

alert("数据: " + data + "\n状态: " + status);

});

});

});

</script>

<button>发送一个 HTTP GET 请求并获取返回结果</button>

解析：$.get() 的第一个参数是我们希望请求的 URL。

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。

### $.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。

#### 实例：使用 $.post() 连同请求一起发送数据

<script>

$(document).ready(function () {

$("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);

});

});

});

</script>

<button>发送一个 HTTP POST 请求页面并获取返回内容</button>

解析：

$.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。

### 区别：

**GET 和 POST 方法的区别**：

**1、发送的数据数量**

在 GET 中，只能发送有限数量的数据，因为数据是在 URL 中发送的。

在 POST 中，可以发送大量的数据，因为数据是在正文主体中发送的。

**2、安全性**

GET 方法发送的数据不受保护，因为数据在 URL 栏中公开，这增加了漏洞和黑客攻击的风险。

POST 方法发送的数据是安全的，因为数据未在 URL 栏中公开，还可以在其中使用多种编码技术，这使其具有弹性。

**3、加入书签中**

GET 查询的结果可以加入书签中，因为它以 URL 的形式存在；而 POST 查询的结果无法加入书签中。

**4、编码**

在表单中使用 GET 方法时，数据类型中只接受 ASCII 字符。

在表单提交时，POST 方法不绑定表单数据类型，并允许二进制和 ASCII 字符。

**5、可变大小**

GET 方法中的可变大小约为 2000 个字符。

POST 方法最多允许 8 Mb 的可变大小。

**6、缓存**

GET 方法的数据是可缓存的，而 POST 方法的数据是无法缓存的。

**7、主要作用**

GET 方法主要用于获取信息。而 POST 方法主要用于更新数据。

## noConflict() 方法

解决在页面上同时使用 jQuery 和其他框架。

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。

也可以通过全名替代简写的方式来使用 jQuery。

<p>这是一个段落。</p>

<button>点我</button>

<script>

$.noConflict();

jQuery(document).ready(function () {

jQuery("button").click(function () {

jQuery("p").text("jQuery 仍然在工作!");

});

});

</script>

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。

<p>这是一个段落。</p>

<button>点我</button>

<script>

var jq = $.noConflict();

jq(document).ready(function () {

jq("button").click(function () {

jq("p").text("jQuery 仍然在工作!");

});

});

</script>

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"：

<p>这是一个段落。</p>

<button>点我</button>

<script>

$.noConflict();

jQuery(document).ready(function ($) {

$("button").click(function () {

$("p").text("jQuery 仍然在工作!");

});

});

</script>

## jQuery 选择器

|  |  |  |
| --- | --- | --- |
| **选择器** | **实例** | **选取** |
| [\*](https://www.runoob.com/jquery/jq-sel-all.html) | $("\*") | 所有元素 |
| [#id](https://www.runoob.com/jquery/jq-sel-id.html) | $("#lastname") | id="lastname" 的元素 |
| [.class](https://www.runoob.com/jquery/jq-sel-class.html) | $(".intro") | class="intro" 的所有元素 |
| [.class,.class](https://www.runoob.com/jquery/sel-multiple-classes.html) | $(".intro,.demo") | class 为 "intro" 或 "demo" 的所有元素 |
| [element](https://www.runoob.com/jquery/jq-sel-element.html) | $("p") | 所有 <p> 元素 |
| [el1,el2,el3](https://www.runoob.com/jquery/sel-multiple-elements.html) | $("h1,div,p") | 所有 <h1>、<div> 和 <p> 元素 |
|  |  |  |
| [:first](https://www.runoob.com/jquery/sel-first.html) | $("p:first") | 第一个 <p> 元素 |
| [:last](https://www.runoob.com/jquery/sel-last.html) | $("p:last") | 最后一个 <p> 元素 |
| [:even](https://www.runoob.com/jquery/sel-even.html) | $("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](https://www.runoob.com/jquery/sel-odd.html) | $("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](https://www.runoob.com/jquery/jq-sel-firstchild.html) | $("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](https://www.runoob.com/jquery/sel-firstoftype.html) | $("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](https://www.runoob.com/jquery/sel-lastchild.html) | $("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](https://www.runoob.com/jquery/sel-lastoftype.html) | $("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)](https://www.runoob.com/jquery/sel-nthchild.html) | $("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)](https://www.runoob.com/jquery/sel-nthlastchild.html) | $("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)](https://www.runoob.com/jquery/sel-nthoftype.html) | $("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)](https://www.runoob.com/jquery/sel-nthlastoftype.html) | $("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](https://www.runoob.com/jquery/sel-onlychild.html) | $("p:only-child") | 属于其父元素的唯一子元素的所有 <p> 元素 |
| [:only-of-type](https://www.runoob.com/jquery/sel-onlyoftype.html) | $("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](https://www.runoob.com/jquery/sel-parent-child.html) | $("div > p") | <div> 元素的直接子元素的所有 <p> 元素 |
| [parent descendant](https://www.runoob.com/jquery/sel-parent-descendant.html) | $("div p") | <div> 元素的后代的所有 <p> 元素 |
| [element + next](https://www.runoob.com/jquery/sel-previous-next.html) | $("div + p") | 每个 <div> 元素相邻的下一个 <p> 元素 |
| [element ~ siblings](https://www.runoob.com/jquery/sel-previous-siblings.html) | $("div ~ p") | <div> 元素同级的所有 <p> 元素 |
|  |  |  |
| [:eq(index)](https://www.runoob.com/jquery/sel-eq.html) | $("ul li:eq(3)") | 列表中的第四个元素（index 值从 0 开始） |
| [:gt(no)](https://www.runoob.com/jquery/sel-gt.html) | $("ul li:gt(3)") | 列举 index 大于 3 的元素 |
| [:lt(no)](https://www.runoob.com/jquery/sel-lt.html) | $("ul li:lt(3)") | 列举 index 小于 3 的元素 |
| [:not(selector)](https://www.runoob.com/jquery/jq-sel-not.html) | $("input:not(:empty)") | 所有不为空的输入元素 |
|  |  |  |
| [:header](https://www.runoob.com/jquery/sel-header.html) | $(":header") | 所有标题元素 <h1>, <h2> ... |
| [:animated](https://www.runoob.com/jquery/sel-animated.html) | $(":animated") | 所有动画元素 |
| [:focus](https://www.runoob.com/jquery/jq-sel-focus.html) | $(":focus") | 当前具有焦点的元素 |
| [:contains(text)](https://www.runoob.com/jquery/sel-contains.html) | $(":contains('Hello')") | 所有包含文本 "Hello" 的元素 |
| [:has(selector)](https://www.runoob.com/jquery/sel-has.html) | $("div:has(p)") | 所有包含有 <p> 元素在其内的 <div> 元素 |
| [:empty](https://www.runoob.com/jquery/jq-sel-empty.html) | $(":empty") | 所有空元素 |
| [:parent](https://www.runoob.com/jquery/sel-parent.html) | $(":parent") | 匹配所有含有子元素或者文本的父元素。 |
| [:hidden](https://www.runoob.com/jquery/sel-hidden.html) | $("p:hidden") | 所有隐藏的 <p> 元素 |
| [:visible](https://www.runoob.com/jquery/sel-visible.html) | $("table:visible") | 所有可见的表格 |
| [:root](https://www.runoob.com/jquery/jq-sel-root.html) | $(":root") | 文档的根元素 |
| [:lang(language)](https://www.runoob.com/jquery/jq-sel-lang.html) | $("p:lang(de)") | 所有 lang 属性值为 "de" 的 <p> 元素 |
|  |  |  |
| [[attribute]](https://www.runoob.com/jquery/jq-sel-attribute.html) | $("[href]") | 所有带有 href 属性的元素 |
| [[attribute=value]](https://www.runoob.com/jquery/sel-attribute-equal-value.html) | $("[href='default.htm']") | 所有带有 href 属性且值等于 "default.htm" 的元素 |
| [[attribute!=value]](https://www.runoob.com/jquery/sel-attribute-notequal-value.html) | $("[href!='default.htm']") | 所有带有 href 属性且值不等于 "default.htm" 的元素 |
| [[attribute$=value]](https://www.runoob.com/jquery/sel-attribute-end-value.html) | $("[href$='.jpg']") | 所有带有 href 属性且值以 ".jpg" 结尾的元素 |
| [[*attribute*|=*value*]](https://www.runoob.com/jquery/sel-attribute-prefix-value.html) | $("[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*]](https://www.runoob.com/jquery/sel-attribute-beginning-value.html) | $("[title^='Tom']") | 所有带有 title 属性且值以 "Tom" 开头的元素 |
| [[*attribute*~=*value*]](https://www.runoob.com/jquery/sel-attribute-contains-value.html) | $("[title~='hello']") | 所有带有 title 属性且值包含单词 "hello" 的元素 |
| [[*attribute\**=*value*]](https://www.runoob.com/jquery/sel-attribute-contains-string-value.html) | $("[title\*='hello']") | 所有带有 title 属性且值包含字符串 "hello" 的元素 |
| [[*name*=*value*][*name2*=*value2*]](https://www.runoob.com/jquery/sel-multipleattribute-equal-value.html) | $( "input[id][name$='man']" ) | 带有 id 属性，并且 name 属性以 man 结尾的输入框 |
|  |  |  |
| [:input](https://www.runoob.com/jquery/sel-input.html) | $(":input") | 所有 input 元素 |
| [:text](https://www.runoob.com/jquery/sel-input-text.html) | $(":text") | 所有带有 type="text" 的 input 元素 |
| [:password](https://www.runoob.com/jquery/sel-input-password.html) | $(":password") | 所有带有 type="password" 的 input 元素 |
| [:radio](https://www.runoob.com/jquery/sel-input-radio.html) | $(":radio") | 所有带有 type="radio" 的 input 元素 |
| [:checkbox](https://www.runoob.com/jquery/sel-input-checkbox.html) | $(":checkbox") | 所有带有 type="checkbox" 的 input 元素 |
| [:submit](https://www.runoob.com/jquery/sel-input-submit.html) | $(":submit") | 所有带有 type="submit" 的 input 元素 |
| [:reset](https://www.runoob.com/jquery/sel-input-reset.html) | $(":reset") | 所有带有 type="reset" 的 input 元素 |
| [:button](https://www.runoob.com/jquery/sel-input-button.html) | $(":button") | 所有带有 type="button" 的 input 元素 |
| [:image](https://www.runoob.com/jquery/sel-input-image.html) | $(":image") | 所有带有 type="image" 的 input 元素 |
| [:file](https://www.runoob.com/jquery/sel-input-file.html) | $(":file") | 所有带有 type="file" 的 input 元素 |
| [:enabled](https://www.runoob.com/jquery/sel-input-enabled.html) | $(":enabled") | 所有启用的元素 |
| [:disabled](https://www.runoob.com/jquery/sel-input-disabled.html) | $(":disabled") | 所有禁用的元素 |
| [:selected](https://www.runoob.com/jquery/sel-input-selected.html) | $(":selected") | 所有选定的下拉列表元素 |
| [:checked](https://www.runoob.com/jquery/sel-input-checked.html) | $(":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](https://www.runoob.com/jquery/jq-sel-target.html) | $( "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 |

### 1.基本选择器

$("#id") //ID选择器

$("div") //元素选择器

$(".classname") //类选择器

$(".classname,.classname1,#id1") //组合选择器

### 2.层次选择器

$("#id>.classname ") //子元素选择器

$("#id .classname ") //后代元素选择器

$("#id + .classname ") //紧邻下一个元素选择器

$("#id ~ .classname ") //兄弟元素选择器

### 3.过滤选择器(重点)

$("li:first") //第一个li

$("li:last") //最后一个li

$("li:even") //挑选下标为偶数的li

$("li:odd") //挑选下标为奇数的li

$("li:eq(4)") //下标等于 4 的li(第五个 li 元素)

$("li:gt(2)") //下标大于 2 的li

$("li:lt(2)") //下标小于 2 的li

$("li:not(#runoob)") //挑选除 id="runoob" 以外的所有li

**3.2内容过滤选择器**

$("div:contains('Runob')") // 包含 Runob文本的元素

$("td:empty") //不包含子元素或者文本的空元素

$("div:has(selector)") //含有选择器所匹配的元素

$("td:parent") //含有子元素或者文本的元素

**3.3可见性过滤选择器**

$("li:hidden") //匹配所有不可见元素，或type为hidden的元素

$("li:visible") //匹配所有可见元素

**3.4属性过滤选择器**

$("div[id]") //所有含有 id 属性的 div 元素

$("div[id='123']") // id属性值为123的div 元素

$("div[id!='123']") // id属性值不等于123的div 元素

$("div[id^='qq']") // id属性值以qq开头的div 元素

$("div[id$='zz']") // id属性值以zz结尾的div 元素

$("div[id\*='bb']") // id属性值包含bb的div 元素

$("input[id][name$='man']") //多属性选过滤，同时满足两个属性的条件的元素

**3.5状态过滤选择器**

$("input:enabled") // 匹配可用的 input

$("input:disabled") // 匹配不可用的 input

$("input:checked") // 匹配选中的 input

$("option:selected") // 匹配选中的 option

### 4.表单选择器

$(":input") //匹配所有 input, textarea, select 和 button 元素

$(":text") //所有的单行文本框，$(":text") 等价于$("[type=text]")，推荐使用$("input:text")效率更高，下同

$(":password") //所有密码框

$(":radio") //所有单选按钮

$(":checkbox") //所有复选框

$(":submit") //所有提交按钮

$(":reset") //所有重置按钮

$(":button") //所有button按钮

$(":file") //所有文件域

## 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。

|  |  |
| --- | --- |
| **方法** | **描述** |
| [bind()](https://www.runoob.com/jquery/event-bind.html) | 向元素添加事件处理程序 |
| [blur()](https://www.runoob.com/jquery/event-blur.html) | 添加/触发失去焦点事件 |
| [change()](https://www.runoob.com/jquery/event-change.html) | 添加/触发 change 事件 |
| [click()](https://www.runoob.com/jquery/event-click.html) | 添加/触发 click 事件 |
| [dblclick()](https://www.runoob.com/jquery/event-dblclick.html) | 添加/触发 double click 事件 |
| [delegate()](https://www.runoob.com/jquery/event-delegate.html) | 向匹配元素的当前或未来的子元素添加处理程序 |
| [die()](https://www.runoob.com/jquery/event-die.html) | 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()](https://www.runoob.com/jquery/event-error.html) | 在版本 1.8 中被废弃。添加/触发 error 事件 |
| [event.currentTarget](https://www.runoob.com/jquery/jq-event-currenttarget.html) | 在事件冒泡阶段内的当前 DOM 元素 |
| [event.data](https://www.runoob.com/jquery/event-data.html) | 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](https://www.runoob.com/jquery/event-delegatetarget.html) | 返回当前调用的 jQuery 事件处理程序所添加的元素 |
| [event.isDefaultPrevented()](https://www.runoob.com/jquery/event-isdefaultprevented.html) | 返回指定的 event 对象上是否调用了 event.preventDefault() |
| [event.isImmediatePropagationStopped()](https://www.runoob.com/jquery/event-isimmediatepropagationstopped.html) | 返回指定的 event 对象上是否调用了 event.stopImmediatePropagation() |
| [event.isPropagationStopped()](https://www.runoob.com/jquery/event-ispropagationstopped.html) | 返回指定的 event 对象上是否调用了 event.stopPropagation() |
| [event.namespace](https://www.runoob.com/jquery/event-namespace.html) | 返回当事件被触发时指定的命名空间 |
| [event.pageX](https://www.runoob.com/jquery/event-pagex.html) | 返回相对于文档左边缘的鼠标位置 |
| [event.pageY](https://www.runoob.com/jquery/event-pagey.html) | 返回相对于文档上边缘的鼠标位置 |
| [event.preventDefault()](https://www.runoob.com/jquery/event-preventdefault.html) | 阻止事件的默认行为 |
| [event.relatedTarget](https://www.runoob.com/jquery/jq-event-relatedtarget.html) | 返回当鼠标移动时哪个元素进入或退出 |
| [event.result](https://www.runoob.com/jquery/event-result.html) | 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()](https://www.runoob.com/jquery/event-stopimmediatepropagation.html) | 阻止其他事件处理程序被调用 |
| [event.stopPropagation()](https://www.runoob.com/jquery/event-stoppropagation.html) | 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](https://www.runoob.com/jquery/jq-event-target.html) | 返回哪个 DOM 元素触发事件 |
| [event.timeStamp](https://www.runoob.com/jquery/jq-event-timestamp.html) | 返回从 1970 年 1 月 1 日到事件被触发时的毫秒数 |
| [event.type](https://www.runoob.com/jquery/jq-event-type.html) | 返回哪种事件类型被触发 |
| [event.which](https://www.runoob.com/jquery/event-which.html) | 返回指定事件上哪个键盘键或鼠标按钮被按下 |
| [event.metaKey](https://www.runoob.com/jquery/event_metakey.html) | 事件触发时 META 键是否被按下 |
| [focus()](https://www.runoob.com/jquery/event-focus.html) | 添加/触发 focus 事件 |
| [focusin()](https://www.runoob.com/jquery/event-focusin.html) | 添加事件处理程序到 focusin 事件 |
| [focusout()](https://www.runoob.com/jquery/event-focusout.html) | 添加事件处理程序到 focusout 事件 |
| [hover()](https://www.runoob.com/jquery/event-hover.html) | 添加两个事件处理程序到 hover 事件 |
| [keydown()](https://www.runoob.com/jquery/event-keydown.html) | 添加/触发 keydown 事件 |
| [keypress()](https://www.runoob.com/jquery/event-keypress.html) | 添加/触发 keypress 事件 |
| [keyup()](https://www.runoob.com/jquery/event-keyup.html) | 添加/触发 keyup 事件 |
| [live()](https://www.runoob.com/jquery/event-live.html) | 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()](https://www.runoob.com/jquery/event-load.html) | 在版本 1.8 中被废弃。添加一个事件处理程序到 load 事件 |
| [mousedown()](https://www.runoob.com/jquery/event-mousedown.html) | 添加/触发 mousedown 事件 |
| [mouseenter()](https://www.runoob.com/jquery/event-mouseenter.html) | 添加/触发 mouseenter 事件 |
| [mouseleave()](https://www.runoob.com/jquery/event-mouseleave.html) | 添加/触发 mouseleave 事件 |
| [mousemove()](https://www.runoob.com/jquery/event-mousemove.html) | 添加/触发 mousemove 事件 |
| [mouseout()](https://www.runoob.com/jquery/event-mouseout.html) | 添加/触发 mouseout 事件 |
| [mouseover()](https://www.runoob.com/jquery/event-mouseover.html) | 添加/触发 mouseover 事件 |
| [mouseup()](https://www.runoob.com/jquery/event-mouseup.html) | 添加/触发 mouseup 事件 |
| [off()](https://www.runoob.com/jquery/event-off.html) | 移除通过 on() 方法添加的事件处理程序 |
| [on()](https://www.runoob.com/jquery/event-on.html) | 向元素添加事件处理程序 |
| [one()](https://www.runoob.com/jquery/event-one.html) | 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 |
| [$.proxy()](https://www.runoob.com/jquery/event-proxy.html) | 接受一个已有的函数，并返回一个带特定上下文的新的函数 |
| [ready()](https://www.runoob.com/jquery/event-ready.html) | 规定当 DOM 完全加载时要执行的函数 |
| [resize()](https://www.runoob.com/jquery/event-resize.html) | 添加/触发 resize 事件 |
| [scroll()](https://www.runoob.com/jquery/event-scroll.html) | 添加/触发 scroll 事件 |
| [select()](https://www.runoob.com/jquery/event-select.html) | 添加/触发 select 事件 |
| [submit()](https://www.runoob.com/jquery/event-submit.html) | 添加/触发 submit 事件 |
| [toggle()](https://www.runoob.com/jquery/event-toggle.html) | 在版本 1.9 中被移除。添加 click 事件之间要切换的两个或多个函数 |
| [trigger()](https://www.runoob.com/jquery/event-trigger.html) | 触发绑定到被选元素的所有事件 |
| [triggerHandler()](https://www.runoob.com/jquery/event-triggerhandler.html) | 触发绑定到被选元素的指定事件上的所有函数 |
| [unbind()](https://www.runoob.com/jquery/event-unbind.html) | 从被选元素上移除添加的事件处理程序 |
| [undelegate()](https://www.runoob.com/jquery/event-undelegate.html) | 从现在或未来的被选元素上移除事件处理程序 |
| [unload()](https://www.runoob.com/jquery/event-unload.html) | 在版本 1.8 中被废弃。添加事件处理程序到 unload 事件 |
| [contextmenu()](https://www.runoob.com/jquery/event-contextmenu.html) | 添加事件处理程序到 contextmenu 事件 |
| [$.holdReady()](https://www.runoob.com/jquery/event-holdready.html) | 用于暂停或恢复.ready() 事件的执行 |

## 动画效果的 jQuery 方法

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

## jQuery HTML / CSS 方法

|  |  |
| --- | --- |
| **方法** | **描述** |
| [addClass()](https://www.runoob.com/jquery/html-addclass.html) | 向被选元素添加一个或多个类名 |
| [after()](https://www.runoob.com/jquery/html-after.html) | 在被选元素后插入内容 |
| [append()](https://www.runoob.com/jquery/html-append.html) | 在被选元素的结尾插入内容 |
| [appendTo()](https://www.runoob.com/jquery/html-appendto.html) | 在被选元素的结尾插入 HTML 元素 |
| [attr()](https://www.runoob.com/jquery/html-attr.html) | 设置或返回被选元素的属性/值 |
| [before()](https://www.runoob.com/jquery/html-before.html) | 在被选元素前插入内容 |
| [clone()](https://www.runoob.com/jquery/html-clone.html) | 生成被选元素的副本 |
| [css()](https://www.runoob.com/jquery/css-css.html) | 为被选元素设置或返回一个或多个样式属性 |
| [detach()](https://www.runoob.com/jquery/html-detach.html) | 移除被选元素（保留数据和事件） |
| [empty()](https://www.runoob.com/jquery/html-empty.html) | 从被选元素移除所有子节点和内容 |
| [hasClass()](https://www.runoob.com/jquery/html-hasclass.html) | 检查被选元素是否包含指定的 class 名称 |
| [height()](https://www.runoob.com/jquery/css-height.html) | 设置或返回被选元素的高度 |
| [html()](https://www.runoob.com/jquery/html-html.html) | 设置或返回被选元素的内容 |
| [innerHeight()](https://www.runoob.com/jquery/html-innerheight.html) | 返回元素的高度（包含 padding，不包含 border） |
| [innerWidth()](https://www.runoob.com/jquery/html-innerwidth.html) | 返回元素的宽度（包含 padding，不包含 border） |
| [insertAfter()](https://www.runoob.com/jquery/html-insertafter.html) | 在被选元素后插入 HTML 元素 |
| [insertBefore()](https://www.runoob.com/jquery/html-insertbefore.html) | 在被选元素前插入 HTML 元素 |
| [offset()](https://www.runoob.com/jquery/css-offset.html) | 设置或返回被选元素的偏移坐标（相对于文档） |
| [offsetParent()](https://www.runoob.com/jquery/css-offsetparent.html) | 返回第一个定位的祖先元素 |
| [outerHeight()](https://www.runoob.com/jquery/html-outerheight.html) | 返回元素的高度（包含 padding 和 border） |
| [outerWidth()](https://www.runoob.com/jquery/html-outerwidth.html) | 返回元素的宽度（包含 padding 和 border） |
| [position()](https://www.runoob.com/jquery/css-position.html) | 返回元素的位置（相对于父元素） |
| [prepend()](https://www.runoob.com/jquery/html-prepend.html) | 在被选元素的开头插入内容 |
| [prependTo()](https://www.runoob.com/jquery/html-prependto.html) | 在被选元素的开头插入 HTML 元素 |
| [prop()](https://www.runoob.com/jquery/html-prop.html) | 设置或返回被选元素的属性/值 |
| [remove()](https://www.runoob.com/jquery/html-remove.html) | 移除被选元素（包含数据和事件） |
| [removeAttr()](https://www.runoob.com/jquery/html-removeattr.html) | 从被选元素移除一个或多个属性 |
| [removeClass()](https://www.runoob.com/jquery/html-removeclass.html) | 从被选元素移除一个或多个类 |
| [removeProp()](https://www.runoob.com/jquery/html-removeprop.html) | 移除通过 prop() 方法设置的属性 |
| [replaceAll()](https://www.runoob.com/jquery/html-replaceall.html) | 把被选元素替换为新的 HTML 元素 |
| [replaceWith()](https://www.runoob.com/jquery/html-replacewith.html) | 把被选元素替换为新的内容 |
| [scrollLeft()](https://www.runoob.com/jquery/css-scrollleft.html) | 设置或返回被选元素的水平滚动条位置 |
| [scrollTop()](https://www.runoob.com/jquery/css-scrolltop.html) | 设置或返回被选元素的垂直滚动条位置 |
| [text()](https://www.runoob.com/jquery/html-text.html) | 设置或返回被选元素的文本内容 |
| [toggleClass()](https://www.runoob.com/jquery/html-toggleclass.html) | 在被选元素中添加/移除一个或多个类之间切换 |
| [unwrap()](https://www.runoob.com/jquery/html-unwrap.html) | 移除被选元素的父元素 |
| [val()](https://www.runoob.com/jquery/html-val.html) | 设置或返回被选元素的属性值（针对表单元素） |
| [width()](https://www.runoob.com/jquery/css-width.html) | 设置或返回被选元素的宽度 |
| [wrap()](https://www.runoob.com/jquery/html-wrap.html) | 在每个被选元素的周围用 HTML 元素包裹起来 |
| [wrapAll()](https://www.runoob.com/jquery/html-wrapall.html) | 在所有被选元素的周围用 HTML 元素包裹起来 |
| [wrapInner()](https://www.runoob.com/jquery/html-wrapinner.html) | 在每个被选元素的内容周围用 HTML 元素包裹起来 |
| [$.escapeSelector()](https://www.runoob.com/jquery/html-escapeSelector.html) | 转义CSS选择器中有特殊意义的字符或字符串 |
| [$.cssHooks](https://www.runoob.com/jquery/html-csshooks.html) | 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

|  |  |
| --- | --- |
| **方法** | **描述** |
| [add()](https://www.runoob.com/jquery/traversing-add.html) | 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()](https://www.runoob.com/jquery/traversing-children.html) | 返回被选元素的所有直接子元素 |
| [closest()](https://www.runoob.com/jquery/traversing-closest.html) | 返回被选元素的第一个祖先元素 |
| [contents()](https://www.runoob.com/jquery/traversing-contents.html) | 返回被选元素的所有直接子元素（包含文本和注释节点） |
| [each()](https://www.runoob.com/jquery/traversing-each.html) | 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()](https://www.runoob.com/jquery/traversing-eq.html) | 返回带有被选元素的指定索引号的元素 |
| [filter()](https://www.runoob.com/jquery/traversing-filter.html) | 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()](https://www.runoob.com/jquery/traversing-find.html) | 返回被选元素的后代元素 |
| [first()](https://www.runoob.com/jquery/traversing-first.html) | 返回被选元素的第一个元素 |
| [has()](https://www.runoob.com/jquery/traversing-has.html) | 返回拥有一个或多个元素在其内的所有元素 |
| [is()](https://www.runoob.com/jquery/traversing-is.html) | 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()](https://www.runoob.com/jquery/traversing-last.html) | 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()](https://www.runoob.com/jquery/traversing-next.html) | 返回被选元素的后一个同级元素 |
| [nextAll()](https://www.runoob.com/jquery/traversing-nextall.html) | 返回被选元素之后的所有同级元素 |
| [nextUntil()](https://www.runoob.com/jquery/traversing-nextuntil.html) | 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()](https://www.runoob.com/jquery/traversing-not.html) | 从匹配元素集合中移除元素 |
| [offsetParent()](https://www.runoob.com/jquery/traversing-offsetparent.html) | 返回第一个定位的父元素 |
| [parent()](https://www.runoob.com/jquery/traversing-parent.html) | 返回被选元素的直接父元素 |
| [parents()](https://www.runoob.com/jquery/traversing-parents.html) | 返回被选元素的所有祖先元素 |
| [parentsUntil()](https://www.runoob.com/jquery/traversing-parentsuntil.html) | 返回介于两个给定参数之间的所有祖先元素 |
| [prev()](https://www.runoob.com/jquery/traversing-prev.html) | 返回被选元素的前一个同级元素 |
| [prevAll()](https://www.runoob.com/jquery/traversing-prevall.html) | 返回被选元素之前的所有同级元素 |
| [prevUntil()](https://www.runoob.com/jquery/traversing-prevuntil.html) | 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()](https://www.runoob.com/jquery/traversing-siblings.html) | 返回被选元素的所有同级元素 |
| [slice()](https://www.runoob.com/jquery/traversing-slice.html) | 把匹配元素集合缩减为指定范围的子集 |

## jQuery AJAX 方法

|  |  |
| --- | --- |
| **方法** | **描述** |
| [$.ajax()](https://www.runoob.com/jquery/ajax-ajax.html) | 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()](https://www.runoob.com/jquery/ajax-ajaxsetup.html) | 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()](https://www.runoob.com/jquery/ajax-get.html) | 使用 AJAX 的 HTTP GET 请求从服务器加载数据 |
| [$.getJSON()](https://www.runoob.com/jquery/ajax-getjson.html) | 使用 HTTP GET 请求从服务器加载 JSON 编码的数据 |
| [$.getScript()](https://www.runoob.com/jquery/ajax-getscript.html) | 使用 AJAX 的 HTTP GET 请求从服务器加载并执行 JavaScript |
| [$.param()](https://www.runoob.com/jquery/ajax-param.html) | 创建数组或对象的序列化表示形式（可用于 AJAX 请求的 URL 查询字符串） |
| [$.post()](https://www.runoob.com/jquery/ajax-post.html) | 使用 AJAX 的 HTTP POST 请求从服务器加载数据 |
| [ajaxComplete()](https://www.runoob.com/jquery/ajax-ajaxcomplete.html) | 规定 AJAX 请求完成时运行的函数 |
| [ajaxError()](https://www.runoob.com/jquery/ajax-ajaxerror.html) | 规定 AJAX 请求失败时运行的函数 |
| [ajaxSend()](https://www.runoob.com/jquery/ajax-ajaxsend.html) | 规定 AJAX 请求发送之前运行的函数 |
| [ajaxStart()](https://www.runoob.com/jquery/ajax-ajaxstart.html) | 规定第一个 AJAX 请求开始时运行的函数 |
| [ajaxStop()](https://www.runoob.com/jquery/ajax-ajaxstop.html) | 规定所有的 AJAX 请求完成时运行的函数 |
| [ajaxSuccess()](https://www.runoob.com/jquery/ajax-ajaxsuccess.html) | 规定 AJAX 请求成功完成时运行的函数 |
| [load()](https://www.runoob.com/jquery/ajax-load.html) | 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()](https://www.runoob.com/jquery/ajax-serialize.html) | 编码表单元素集为字符串以便提交 |
| [serializeArray()](https://www.runoob.com/jquery/ajax-serializearray.html) | 编码表单元素集为 names 和 values 的数组 |

## jQuery 杂项方法

|  |  |
| --- | --- |
| **方法** | **描述** |
| [data()](https://www.runoob.com/jquery/misc-data.html) | 向被选元素附加数据，或者从被选元素获取数据 |
| [each()](https://www.runoob.com/jquery/misc-each.html) | 为每个匹配元素执行函数 |
| [get()](https://www.runoob.com/jquery/misc-get.html) | 获取由选择器指定的 DOM 元素 |
| [index()](https://www.runoob.com/jquery/misc-index.html) | 从匹配元素中搜索给定元素 |
| [$.noConflict()](https://www.runoob.com/jquery/misc-noconflict.html) | 释放变量 $ 的 jQuery 控制权 |
| [$.param()](https://www.runoob.com/jquery/misc-param.html) | 创建数组或对象的序列化表示形式（可在生成 AJAX 请求时用于 URL 查询字符串中） |
| [removeData()](https://www.runoob.com/jquery/misc-removedata.html) | 移除之前存放的数据 |
| [size()](https://www.runoob.com/jquery/misc-size.html) | 在版本 1.8 中被废弃。返回被 jQuery 选择器匹配的 DOM 元素的数量 |
| [toArray()](https://www.runoob.com/jquery/misc-toarray.html) | 以数组的形式检索所有包含在 jQuery 集合中的所有 DOM 元素 |
| [pushStack()](https://www.runoob.com/jquery/misc-pushstack.html) | 将一个DOM元素集合加入到jQuery栈 |
| [$.when()](https://www.runoob.com/jquery/misc-when.html) | 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** |
| $.boxModel | 在版本 1.8 中被废弃。检测浏览器是否使用W3C的CSS盒模型渲染当前页面 |
| [$.browser](https://www.runoob.com/jquery/misc-browser.html) | 在版本 1.9 中被废弃。返回用户当前使用的浏览器的相关信息 |
| [$.contains()](https://www.runoob.com/jquery/misc-contains.html) | 判断另一个DOM元素是否是指定DOM元素的后代 |
| [$.each()](https://www.runoob.com/jquery/misc_each.html) | 遍历指定的对象和数组 |
| [$.extend()](https://www.runoob.com/jquery/misc-extend.html) | 将一个或多个对象的内容合并到目标对象 |
| [$.fn.extend()](https://www.runoob.com/jquery/misc-fn-extend.html) | 为jQuery扩展一个或多个实例属性和方法 |
| [$.globalEval()](https://www.runoob.com/jquery/misc-globaleval.html) | 全局性地执行一段JavaScript代码 |
| [$.grep()](https://www.runoob.com/jquery/misc-grep.html) | 过滤并返回满足指定函数的数组元素 |
| [$.inArray()](https://www.runoob.com/jquery/misc-inarray.html) | 在数组中查找指定值并返回它的索引值（如果没有找到，则返回-1） |
| [$.isArray()](https://www.runoob.com/jquery/misc-isarray.html) | 判断指定参数是否是一个数组 |
| [$.isEmptyObject()](https://www.runoob.com/jquery/misc-isemptyobject.html) | 检查对象是否为空（不包含任何属性） |
| [$.isFunction()](https://www.runoob.com/jquery/misc-isfunction.html) | 判断指定参数是否是一个函数 |
| [$.isNumeric()](https://www.runoob.com/jquery/misc-isnumeric.html) | 判断指定参数是否是一个数字值 |
| [$.isPlainObject()](https://www.runoob.com/jquery/misc-isplainobject.html) | 判断指定参数是否是一个纯粹的对象 |
| [$.isWindow()](https://www.runoob.com/jquery/misc-iswindow.html) | 判断指定参数是否是一个窗口 |
| [$.isXMLDoc()](https://www.runoob.com/jquery/misc-isxmldoc.html) | 判断一个DOM节点是否位于XML文档中，或者其本身就是XML文档 |
| [$.makeArray()](https://www.runoob.com/jquery/misc-makearray.html) | 将一个类似数组的对象转换为真正的数组对象 |
| [$.map()](https://www.runoob.com/jquery/misc-map.html) | 指定函数处理数组中的每个元素(或对象的每个属性)，并将处理结果封装为新的数组返回 |
| [$.merge()](https://www.runoob.com/jquery/misc-merge.html) | 合并两个数组内容到第一个数组 |
| [$.noop()](https://www.runoob.com/jquery/misc-noop.html) | 一个空函数 |
| [$.now()](https://www.runoob.com/jquery/misc-now.html) | 返回当前时间 |
| [$.parseHTML()](https://www.runoob.com/jquery/misc-parsehtml.html) | 将HTML字符串解析为对应的DOM节点数组 |
| [$.parseJSON()](https://www.runoob.com/jquery/misc-parsejson.html) | 将符合标准格式的JSON字符串转为与之对应的JavaScript对象 |
| [$.parseXML()](https://www.runoob.com/jquery/misc-parsexml.html) | 将字符串解析为对应的XML文档 |
| [$.trim()](https://www.runoob.com/jquery/misc-trim.html) | 去除字符串两端的空白字符 |
| [$.type()](https://www.runoob.com/jquery/misc-type.html) | 确定JavaScript内置对象的类型 |
| [$.unique()](https://www.runoob.com/jquery/misc-unique.html) | 在jQuery 3.0中被弃用。对DOM元素数组进行排序，并移除重复的元素 |
| [$.uniqueSort()](https://www.runoob.com/jquery/misc-uniquesort.html) | 对DOM元素数组进行排序，并移除重复的元素 |
| [$.data()](https://www.runoob.com/jquery/misc_data.html) | 在指定的元素上存取数据，并返回设置值 |
| [$.hasData()](https://www.runoob.com/jquery/misc-hasdata.html) | 确定一个元素是否有相关的jQuery数据 |
| [$.sub()](https://www.runoob.com/jquery/misc-jquery-sub.html) | 创建一个新的jQuery副本，其属性和方法可以修改，而不会影响原来的jQuery对象 |
| [$.speed](https://www.runoob.com/jquery/misc-jquery-speed.html) | 创建一个包含一组属性的对象用来定义自定义动画 |
| [$.htmlPrefilter()](https://www.runoob.com/jquery/misc-jquery-htmlprefilter.html) | 通过jQuery操作方法修改和过滤HTML字符串 |
| [$.readyException()](https://www.runoob.com/jquery/misc-jquery-readyexception.html) | 处理包裹在jQuery()中函数同步抛出的错误 |

## jQuery 回调对象

jQuery 1.7 版本中新增的 jQuery.Callbacks() 函数,返回一个多功能对象，此对象提供了一种强大的方法来管理回调列表。它能够增加、删除、触发、禁用回调函数。

|  |  |
| --- | --- |
| **方法** | **描述** |
| [$.Callbacks()](https://www.runoob.com/jquery/misc-callbacks.html) | 一个多用途的回调列表对象，用来管理回调函数列表 |
| [callbacks.add()](https://www.runoob.com/jquery/misc-callbacks-add.html) | 在回调列表中添加一个回调或回调的集合 |
| [callbacks.disable()](https://www.runoob.com/jquery/misc-callbacks-disable.html) | 禁用回调列表中的回调函数 |
| [callbacks.disabled()](https://www.runoob.com/jquery/misc-callbacks-disabled.html) | 确定回调列表是否已被禁用 |
| [callbacks.empty()](https://www.runoob.com/jquery/misc-callbacks-empty.html) | 从列表中清空所有的回调 |
| [callbacks.fire()](https://www.runoob.com/jquery/misc-callbacks-fire.html) | 传入指定的参数调用所有的回调 |
| [callbacks.fired()](https://www.runoob.com/jquery/misc-callbacks-fired.html) | 确定回调是否至少已经调用一次 |
| [callbacks.firewith()](https://www.runoob.com/jquery/misc-callbacks-firewith.html) | 给定的上下文和参数访问列表中的所有回调 |
| [callbacks.has()](https://www.runoob.com/jquery/misc-callbacks-has.html) | 判断回调列表中是否添加过某回调函数 |
| [callbacks.lock()](https://www.runoob.com/jquery/misc-callbacks-lock.html) | 锁定当前状态的回调列表 |
| [callbacks.locked()](https://www.runoob.com/jquery/misc-callbacks-locked.html) | 判断回调列表是否被锁定 |
| [callbacks.remove()](https://www.runoob.com/jquery/misc-callbacks-remove.html) | 从回调列表中的删除一个回调或回调集合 |

## jQuery 延迟对象

在jQuery 1.5中介绍了 Deferred 延迟对象，它是通过调用 jQuery.Deferred() 方法来创建的可链接的实用对象。它可注册多个回调函数到回调列表，调用回调列表并且传递异步或同步功能的成功或失败的状态。  
延迟对象是可链接的，类似于一个 jQuery 对象可链接的方式，区别于它有自己的方法。在创建一个 Deferred 对象之后，您可以使用以下任何方法，直接链接到通过调用一个或多个的方法创建或保存的对象。

|  |  |
| --- | --- |
| **方法** | **描述** |
| [$.Deferred()](https://www.runoob.com/jquery/misc-jquery-deferred.html) | 返回一个链式实用对象方法来注册多个回调 |
| [deferred.always()](https://www.runoob.com/jquery/misc-deferred-always.html) | 当Deferred（延迟）对象被受理或被拒绝时，调用添加的处理程序 |
| [deferred.done()](https://www.runoob.com/jquery/misc-deferred-done.html) | 当Deferred（延迟）对象被受理时，调用添加的处理程序 |
| [deferred.fail()](https://www.runoob.com/jquery/misc-deferred-fail.html) | 当Deferred（延迟）对象被拒绝时，调用添加的处理程序 |
| [deferred.isRejected()](https://www.runoob.com/jquery/misc-deferred-isrejected.html) | 从jQuery1.7开始已经过时，确定 Deferred 对象是否已被拒绝 |
| [deferred.isResolved()](https://www.runoob.com/jquery/misc-deferred-isresolved.html) | 从jQuery1.7开始已经过时，确定 Deferred 对象是否已被解决 |
| [deferred.notify()](https://www.runoob.com/jquery/misc-deferred-notify.html) | 给定一个参数，调用正在延迟对象上进行的回调函数( progressCallbacks ) |
| [deferred.notifyWith()](https://www.runoob.com/jquery/misc-deferred-notifywith.html) | 给定上下文和参数，调用正在延迟对象上进行的回调函数( progressCallbacks ) |
| [deferred.pipe()](https://www.runoob.com/jquery/misc-deferred-pipe.html) | 过滤 and/or 链式延迟对象的工具方法 |
| [deferred.progress()](https://www.runoob.com/jquery/misc-deferred-progress.html) | 当Deferred（延迟）对象生成进度通知时，调用添加处理程序 |
| [deferred.promise()](https://www.runoob.com/jquery/misc-deferred-promise.html) | 返回 Deferred(延迟)的 Promise 对象 |
| [deferred.reject()](https://www.runoob.com/jquery/misc-deferred-reject.html) | 拒绝 Deferred（延迟）对象，并根据给定的参数调用任何 failCallbacks 回调函数 |
| [deferred.rejectWith()](https://www.runoob.com/jquery/misc-deferred-rejectWith.html) | 拒绝 Deferred（延迟）对象，并根据给定的 context 和 args 参数调用任何 failCallbacks 回调函数 |
| [deferred.resolve()](https://www.runoob.com/jquery/misc-deferred-resolve.html) | 解决Deferred（延迟）对象，并根据给定的参数调用任何 doneCallbacks 回调函数 |
| [deferred.resolveWith()](https://www.runoob.com/jquery/misc-deferred-resolveWith.html) | 解决Deferred（延迟）对象，并根据给定的context 和 args 参数调用任何 doneCallbacks 回调函数 |
| [deferred.state()](https://www.runoob.com/jquery/misc-deferred-state.html) | 确定一个Deferred（延迟）对象的当前状态 |
| [deferred.then()](https://www.runoob.com/jquery/misc-deferred-then.html) | 当Deferred（延迟）对象解决，拒绝或仍在进行中时，调用添加处理程序 |
| [.promise()](https://www.runoob.com/jquery/misc-promise.html) | 返回一个 Promise 对象，观察某种类型被绑定到集合的所有行动，是否已被加入到队列中 |

## jQuery 属性

|  |  |
| --- | --- |
| **方法** | **描述** |
| [context](https://www.runoob.com/jquery/prop-context.html) | 在版本 1.10 中被废弃。包含被传递到 jQuery 的原始上下文 |
| [jquery](https://www.runoob.com/jquery/prop-jquery.html) | 包含 jQuery 的版本号 |
| [jQuery.fx.interval](https://www.runoob.com/jquery/prop-jquery-fx-interval.html) | 改变以毫秒计的动画运行速率 |
| [jQuery.fx.off](https://www.runoob.com/jquery/prop-jquery-fx-off.html) | 对所有动画进行全局禁用或启用 |
| [jQuery.support](https://www.runoob.com/jquery/prop-jquery-support.html) | 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](https://www.runoob.com/jquery/prop-length.html) | 包含 jQuery 对象中元素的数目 |
| [jQuery.cssNumber](https://www.runoob.com/jquery/prop-cssnumber.html) | 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。

### 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/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}." )

}

### 中文提示包

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>

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")

});

}));

#### 实例：将校验规则写到控件中

<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>

</head>

<body>

<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="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>

<script>

$.validator.setDefaults({ //默认设置

submitHandler: function () { //当提交后的操作

alert("提交事件!");

}

});

$().ready(function () {

$("#commentForm").validate(); //对commentForm控件进行验证

});

</script>

#### 实例：将校验规则写到 js 代码中

* head引用：

<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>

<style>

.error {

color: red;

}

</style>

* body代码：

<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>

<script>

$().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 处，如果某个控件没有 message，将调用默认的信息

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"

}

})

});

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。

#### 实例：将错误信息放在 label 元素后并使用 span 元素包裹

* head：

<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>

<style>

.error {

color: red;

}

</style>

* body：

form method="get" class="cmxform" id="form1" action="">

<fieldset>

<legend>登录框</legend>

<p>

<label for="user">用户名</label>

<input id="user" name="user" required minlength="3">

</p>

<p>

<label for="password">密码</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>

<script>

$.validator.setDefaults({

submitHandler: function () {

alert("提交事件!");

}

});

$().ready(function () {

// 提交时验证表单

var validator = $("#form1").validate({

errorPlacement: function (error, element) {

// Append error within linked label

$(element)

.closest("form")

.find("label[for='" + element.attr("id") + "']")

.append(error);

},

errorElement: "span",

messages: {

user: {

required: " (必需字段)",

minlength: " (不能少于 3 个字母)"

},

password: {

required: " (必需字段)",

minlength: " (字母不能少于 5 个且不能大于 12 个)",

maxlength: " (字母不能少于 5 个且不能大于 12 个)"

}

}

});

$(".cancel").click(function () {

validator.resetForm();

});

});

</script>

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 &nbsp; as text for IE

label.html("&nbsp;").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。

### 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。

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

<form id="myform">

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

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

</form>

<script>

$("#myform").validate({

meta: "validate",

submitHandler: function () {

alert("Submitted!")

}

})

</script>

## End