# Jquery

# 教程

## 简介

### 什么是 jQuery ？

jQuery 是一个 JavaScript 函数库。

jQuery 是一个轻量级的"写的少，做的多"的 JavaScript 库。

jQuery 库包含以下功能：

HTML 元素选取

HTML 元素操作

CSS 操作

HTML 事件函数

JavaScript 特效和动画

HTML DOM 遍历和修改

AJAX

Utilities

提示： 除此之外，jQuery 还提供了大量的插件。

## 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](http://jquery.com/download/" \t "https://www.runoob.com/jquery/_blank) 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](http://jquery.com/download/" \t "https://www.runoob.com/jquery/_blank) 中下载。

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

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

提示： 将下载的文件放在网页的同一目录下，就可以使用jQuery。

您是否很疑惑为什么我们没有在 <script> 标签中使用 type="text/javascript" ？  
  
在 HTML5 中，不必那样做了。JavaScript 是 HTML5 以及所有现代浏览器中的默认脚本语言！

### cdn载入jQuery

百度 CDN:

<head>

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

</script> </head>

## 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

### 文档就绪事件

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

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

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

如果在文档没有完全加载之前就运行函数，操作可能失败。下面是两个具体的例子：

试图隐藏一个不存在的元素

获得未完全加载的图像的大小

提示：简洁写法（与以上写法效果相同）:

$(function(){ // 开始写 jQuery 代码... });

## 选择器

### jQuery 选择器

jQuery 选择器允许您对 HTML 元素组或单个元素进行操作。

jQuery 选择器基于元素的 id、类、类型、属性、属性值等"查找"（或选择）HTML 元素。 它基于已经存在的 [CSS 选择器](https://www.runoob.com/cssref/css-selectors.html)，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：$()。

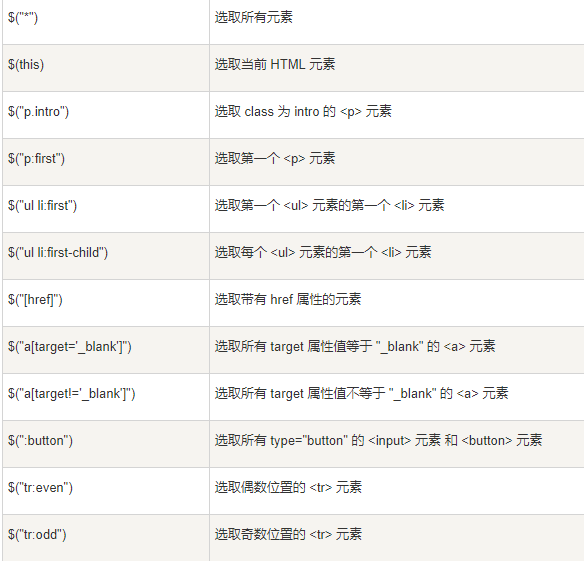
$("button") $("#test") $("p")

$(document).ready(function(){

$("button").click(function(){ //变量.onclick=function(){}

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

更多实例



添加对div的描述，想要将偶数列背景变为灰色，偶数列需要用到:odd

所以需要输入

$("div:odd").css("background-color","gray")

<div id=”.div”></div>

var div=document.getElementById(“div”);

div.style.background-color=”gray”

### css选择器

前面通常是一个标签或是其他类型的名字；

后面冒号:隔开,然后n表示第n个，child 表示子元素；

child 后面跟着一对括号( )，这个括号里表示究竟是第几个元素；

里面可以是一个数字或者是odd(奇数个元素)或者是even(偶数个元素)

括号后面是大括号 {  } ，可以在里面放上背景颜色样式；

odd是单行 tr:nth-children(odd){background-color:#ffc5ce;}

eve是双行 tr:nth-children(even){background-color:#acbfea;}

tr:first-child tr:first:last-child

## 事件

常见 DOM 事件：



mousedown mouseup hover

Js里面的DOM事件都在前面加一个on

jQuery 事件方法语法

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

$("p").click(function(){ $(this).hide(); });触发隐藏效果

# 效果

## 隐藏和显示

jQuery hide() 和 show()

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

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

需要加# 而document.getElementByClassName(show)不需要加#

原生js块状元素

function showContainer2() {

var container1 = document.getElementById('container1');

var container2 = document.getElementById('container2');

container1.style.display = "none";

container2.style.display = "block";

}

两个按钮 一个控制隐藏一个控制展示



原生js实现灯泡的亮和暗 用if-else可以实现少一个按钮

$(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。

$("div").hide(1000,"linear",function(){ alert("Hide() 方法已完成!"); });//1秒钟消失 linear匀速 消失后弹出对话框

一个按钮用来控制隐藏和展示

jQuery toggle()

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

显示被隐藏的元素，并隐藏已显示的元素：

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

一个按钮 原生js来实现jquery的toggle

function showdiv(){

if(showdiv\_display = document.getElementById('show').style.display=='none'){//如果show是隐藏的

document.getElementById('show').style.display='block';//show的display属性设置为block(显示)

}else{//如果show是显示的

document.getElementById('show').style.display='none';//show的display属性设置为none(隐藏)

}

}

## 淡入淡出

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()** 淡入按顺序3秒依次显示

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

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

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

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

**fadeOut()** 淡入按顺序3秒依次消失

**fadeToggle()**

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。

**fadeTo() 透明度变大 颜色变暗**

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

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

$("#div1").fadeTo("slow",0.15); $("#div2").fadeTo("slow",0.4); $("#div3").fadeTo("slow",0.7);

## 滑动

jQuery 拥有以下滑动方法：

slideDown()

slideUp()

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

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

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

});

## 动画

jQuery animate() 方法用于创建自定义动画。

语法：

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

必需的 params 参数定义形成动画的 **CSS 属性**。

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

可选的 callback 参数是动画完成后所执行的函数名称。

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

$("div").animate({

left:'250px', height:'+=150px', width:'+=150px' }); });

Heigth和width可以一直变大

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

$("div").animate({ height:'toggle' });

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");

## 停止动画

jQuery stop() 方法用于停止动画或效果，在它们完成之前。

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

语法:

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

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

可选的 goToEnd 参数规定是否立即完成当前动画。默认是 false。

因此，默认地，stop() 会清除在被选元素上指定的当前动画。

动画队列停止动画测试，只停止当前正在进行的动画，停止当前动画后，队列中的下一个动画开始进行：

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

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

    $("#panel").slideUp(5000);

});

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

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

});

只能停止当前活动的动画，后面的的队列动画向后进行

## jQuery Callback

Callback 函数在当前动画 100% 完成之后执行。

jQuery 动画的问题

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

例子：$("p").hide("slow")

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

使用 callback 实例

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

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

alert("段落现在被隐藏了"); }); });

## 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。

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

# HTML

## DOM 操作

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

attr() 方法用于获取属性值。

$("#test").text() 这是段落中的 粗体 文本。

$("#test").html() 这是段落中的 <b>粗体</b> 文本。

$("#test").val() 输出 菜鸟教程

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

$("#test").attr("href")输出 //www.runoob.com

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

## 设置

### 设置内容

text("Hello world!")

html("<b>Hello world!</b>")

val("RUNOOB")



### text()、html() 以及 val() 的回调函数

$("#test2").html(function(i,origText){

return "旧 html: " + origText + " 新 html: Hello <b>world!</b> (index: " + i + ")";

});

旧 html: 这是另外一个有 **粗体** 字的段落。 新 html: Hello **world!** (index: 0)

origText是 这是另外一个有 **粗体** 字的段落 i是0

### 设置属性

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

### attr() 的回调函数

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

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

return origValue + "/jquery";

});

[https://www.runoob.com](https://www.runoob.com/)

点击之后

<https://www.runoob.com/jquery/jquery-tutorial.html>

origValue是[https://www.runoob.com](https://www.runoob.com/)

## 添加元素

### 添加新的 HTML 内容

我们将学习用于添加新内容的四个 jQuery 方法：

append() - 在被选元素的结尾插入内容

prepend() - 在被选元素的开头插入内容

after() - 在被选元素之后插入内容

before() - 在被选元素之前插入内容

$("ol").append("<li>追加列表项</li>");

List item 1

List item 2

List item 3

追加列表项

### 通过 append() 和 prepend() 方法添加若干新元素

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

var txt1="<p>文本-1。</p>"; // 使用 HTML 标签创建文本 var txt2=$("<p></p>").text("文本-2。"); // 使用 jQuery 创建文本 var txt3=document.createElement("p");

txt3.innerHTML="文本-3。"; // 使用 DOM 创建文本 text with DOM $("body").append(txt1,txt2,txt3); // 追加新元素

文本-1。

文本-2。

文本-3。

## 删除元素

### 删除元素/内容

如需删除元素和内容，一般可使用以下两个 jQuery 方法：

remove() - 删除被选元素（及其子元素）

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

empty() - 从被选元素中删除子元素保留了被选元素

### 过滤被删除的元素

jQuery remove() 方法也可接受一个参数，允许您对被删元素进行过滤。  
$("p").remove(".italic");将.italic的元素进行删除

## Css类

### jQuery 操作 CSS

jQuery 拥有若干进行 CSS 操作的方法。我们将学习下面这些：

addClass() - 向被选元素添加一个或多个类

removeClass() - 从被选元素删除一个或多个类

toggleClass() - 对被选元素进行添加/删除类的切换操作

css() - 设置或返回样式属性

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

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

$("h1,h2,p").addClass("blue"); $("div").addClass("important"); });

### toggleClass() 方法

$("h1,h2,p").toggleClass("blue");

<p class="blue">这是一个段落。</p>

<p>这是另外一个段落。</p>

## Css方法

### jQuery css() 方法

css() 方法设置或返回被选元素的一个或多个样式属性。

### 返回 CSS 属性

如需返回指定的 CSS 属性的值，请使用如下语法：

css("propertyname");

### 设置 CSS 属性

如需设置指定的 CSS 属性，请使用如下语法：

css("propertyname","value");

### 设置多个 CSS 属性

如需设置多个 CSS 属性，请使用如下语法：

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

### 尺寸

## jQuery 尺寸方法

jQuery 提供多个处理尺寸的重要方法：

width()

height()

innerWidth()

innerHeight()

outerWidth()

outerHeight()



<div id="div1"

style="height:100px;width:300px;padding:10px;margin:3px;border:1px

solid blue;background-color:lightblue;"></div>

var txt="";

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

txt+="div 的高度是: " + $("#div1").height();

$("#div1").html(txt);

# 遍历

## 向上遍历 DOM 树

这些 jQuery 方法很有用，它们用于向上遍历 DOM 树：

parent()

parents()

parentsUntil()

### jQuery parent() 方法

parent() 方法返回被选元素的直接父元素。

该方法只会向上一级对 DOM 树进行遍历。

### jQuery parents() 方法

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

$("span").parents();

您也可以使用可选参数来过滤对祖先元素的搜索。

下面的例子返回所有 <span> 元素的所有祖先，并且它是 <ul> 元素：只会渲染ul元素

$("span").parents("ul");

### jQuery parentsUntil() 方法

parentsUntil() 方法返回介于两个给定元素之间的所有祖先元素。

下面的例子返回介于 <span> 与 <div> 元素之间的所有祖先元素：

$("span").parentsUntil("div");不包括div元素

## 向下遍历 DOM 树

下面是两个用于向下遍历 DOM 树的 jQuery 方法：

children()

find()

### jQuery children() 方法

children() 方法返回被选元素的所有直接子元素。

该方法只会向下一级对 DOM 树进行遍历。

您也可以使用可选参数来过滤对子元素的搜索。

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

$("div").children("p.1");

### jQuery find() 方法

find() 方法返回被选元素的后代元素，一路向下直到最后一个后代。

下面的例子返回属于 <div> 后代的所有 <span> 元素：

$("div").find("span");

下面的例子返回 <div> 的所有后代：

$("div").find("\*");

## 同胞

### 在 DOM 树中水平遍历

有许多有用的方法让我们在 DOM 树进行水平遍历：

siblings()

next()

nextAll()

nextUntil()

prev()

prevAll()

prevUntil()

### jQuery siblings() 方法

siblings() 方法返回被选元素的所有同胞元素。

下面的例子返回 <h2> 的所有同胞元素：

$("h2").siblings();包括自己

您也可以使用可选参数来过滤对同胞元素的搜索。

下面的例子返回属于 <h2> 的同胞元素的所有 <p> 元素：

$("h2").siblings("p");

### jQuery next() 方法

next() 方法返回被选元素的下一个同胞元素。

该方法只返回一个元素。

下面的例子返回 <h2> 的下一个同胞元素：

### jQuery nextAll() 方法

nextAll() 方法返回被选元素的所有跟随的同胞元素。

下面的例子返回 <h2> 的所有跟随的同胞元素：

### jQuery nextUntil() 方法

nextUntil() 方法返回介于两个给定参数之间的所有跟随的同胞元素。

下面的例子返回介于 <h2> 与 <h6> 元素之间的所有同胞元素：

$("h2").nextUntil("h6");

## 过滤

### jQuery first() 方法

first() 方法返回被选元素的首个元素。

下面的例子选取首个 <div> 元素内部的第一个 <p> 元素：

$("div p").first();

### jQuery last() 方法

last() 方法返回被选元素的最后一个元素。

下面的例子选择最后一个 <div> 元素中的最后一个 <p> 元素：

### jQuery eq() 方法

eq() 方法返回被选元素中带有指定索引号的元素。

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

$("p").eq(1);

### jQuery filter() 方法

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

下面的例子返回带有类名 "url" 的所有 <p> 元素：

$("p").filter(".url");

### jQuery not() 方法

not() 方法返回不匹配标准的所有元素。

提示：not() 方法与 filter() 相反。

下面的例子返回不带有类名 "url" 的所有 <p> 元素：

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

## AJAX

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

### jQuery load() 方法

jQuery load() 方法是简单但强大的 AJAX 方法。

load() 方法从服务器加载数据，并把返回的数据放入被选元素中。

语法:

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

必需的 URL 参数规定您希望加载的 URL。

可选的 data 参数规定与请求一同发送的查询字符串键/值对集合。

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

这是示例文件（"demo\_test.txt"）的内容：

<h2>jQuery AJAX 是个非常棒的功能！</h2>

<p id="p1">这是段落的一些文本。</p>

$("#div1").load("demo\_test.txt");

也可以把 jQuery 选择器添加到 URL 参数。

下面的例子把 "demo\_test.txt" 文件中 id="p1" 的元素的内容，加载到指定的 <div> 元素中：

$("#div1").load("demo\_test.txt #p1");只会输出p标签去的类容

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

$("#div1").load("demo\_test.txt",function(responseTxt,statusTxt,xhr){

if(statusTxt=="success") alert("外部内容加载成功!");

if(statusTxt=="error") alert("Error: "+xhr.status+": "+xhr.statusText); });

### jQuery get() 和 post() 方法

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

HTTP 请求：GET vs. POST

两种在客户端和服务器端进行请求-响应的常用方法是：GET 和 POST。

GET - 从指定的资源请求数据

POST - 向指定的资源提交要处理的数据

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

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

### jQuery $.get() 方法

$.get() 方法通过 HTTP GET 请求从服务器上请求数据。

语法：

$.get(URL,callback);

必需的 URL 参数规定您希望请求的 URL。

可选的 callback 参数是请求成功后所执行的函数名。

$.get("demo\_test.php",function(data,status){ alert("数据: " + data + "\n状态: " + status); });

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

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

提示： 这个 PHP 文件 ("demo\_test.php") 类似这样：

demo\_test.php 文件代码:

<?php echo '这是个从PHP文件中读取的数据。'; ?>



### jQuery $.post() 方法

$.post() 方法通过 HTTP POST 请求向服务器提交数据。

语法:

$.post(URL,data,callback);

必需的 URL 参数规定您希望请求的 URL。

可选的 data 参数规定连同请求发送的数据。

可选的 callback 参数是请求成功后所执行的函数名。

$.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); });

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

然后我们连同请求（name 和 url）一起发送数据。

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

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

提示： 这个 PHP 文件 ("demo\_test\_post.php") 类似这样：

demo\_test\_post.php 文件代码:

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

