jQuery 教程.

jQuery 是一个 JavaScript 库。

jQuery 库可以通过一行简单的标记被添加到网页中。

jQuery 库包含以下特性：

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

向您的页面添加 jQuery 库

jQuery 库位于一个 JavaScript 文件中，其中包含了所有的 jQuery 函数。

可以通过下面的标记把 jQuery 添加到网页中：

<head>

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

</head>

请注意，<script> 标签应该位于页面的 <head> 部分。

下载 jQuery

共有两个版本的 jQuery 可供下载：一份是精简过的，另一份是未压缩的（供调试或阅读）。

这两个版本都可从 http://jquery.com/download/#Download\_jQuery 下载。

库的替代

Google 和 Microsoft 对 jQuery 的支持都很好。

如果您不愿意在自己的计算机上存放 jQuery 库，那么可以从 Google 或 Microsoft 加载 CDN jQuery 核心文件。

使用 Google 的 CDN

<head>

<script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs

/jquery/1.4.0/jquery.min.js"></script>

</head>

使用 Microsoft 的 CDN

<head>

<script type="text/javascript" src="http://ajax.microsoft.com/ajax/jquery

/jquery-1.4.min.js"></script>

</head>

把 jQuery 添加到您的网页

如需使用 jQuery，您需要下载 jQuery 库（会在下面为您讲解），然后把它包含在希望使用的网页中。

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.js"></script>

</head>

请注意，<script> 标签应该位于页面的 <head> 部分。

提示：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！

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）

这两个版本都可以从 jQuery.com 下载。

提示：您可以把下载文件放到与页面相同的目录中，这样更方便使用。

jQuery 语法

jQuery 语法实例

$(this).hide()

演示 jQuery hide() 函数，隐藏当前的 HTML 元素。

$("#test").hide()

演示 jQuery hide() 函数，隐藏 id="test" 的元素。

$("p").hide()

演示 jQuery hide() 函数，隐藏所有 <p> 元素。

$(".test").hide()

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

jQuery 语法

jQuery 语法是为 HTML 元素的选取编制的，可以对元素执行某些操作。

基础语法是：$(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

文档就绪函数

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(){

--- jQuery functions go here ----

});

这是为了防止文档在完全加载（就绪）之前运行 jQuery 代码。

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

jQuery 选择器

jQuery 元素选择器

jQuery 使用 CSS 选择器来选取 HTML 元素。

$("p") 选取 <p> 元素。

$("p.intro") 选取所有 class="intro" 的 <p> 元素。

$("p#demo") 选取所有 id="demo" 的 <p> 元素。

jQuery 属性选择器

jQuery 使用 XPath 表达式来选择带有给定属性的元素。

$("[href]") 选取所有带有 href 属性的元素。

$("[href='#']") 选取所有带有 href 值等于 "#" 的元素。

$("[href!='#']") 选取所有带有 href 值不等于 "#" 的元素。

$("[href$='.jpg']") 选取所有 href 值以 ".jpg" 结尾的元素。

jQuery CSS 选择器

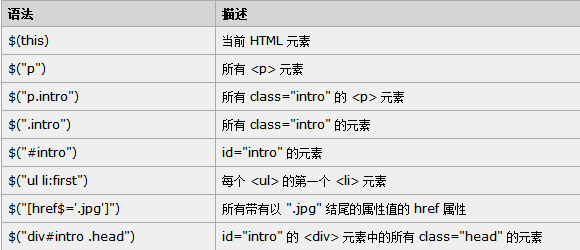
jQuery CSS 选择器可用于改变 HTML 元素的 CSS 属性。

下面的例子把所有 p 元素的背景颜色更改为红色：

实例

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

更多的选择器实例



jQuery 事件

jQuery 事件函数

jQuery 事件处理方法是 jQuery 中的核心函数。

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。术语由事件“触发”（或“激发”）经常会被使用。

通常会把 jQuery 代码放到 <head>部分的事件处理方法中：

<head>

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

<script type="text/javascript">

$(document).ready(function(){

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

$("p").hide();

});

});

</script>

</head>

在上面的例子中，当按钮的点击事件被触发时会调用一个函数：

$("button").click(function() {..some code... } )

该方法隐藏所有 <p> 元素：

$("p").hide();

单独文件中的函数

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

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

实例

<head>

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

<script type="text/javascript" src="my\_jquery\_functions.js"></script>

</head>

jQuery 名称冲突

jQuery 使用 $ 符号作为 jQuery 的简介方式。

某些其他 JavaScript 库中的函数（比如 Prototype）同样使用 $ 符号。

jQuery 使用名为 noConflict() 的方法来解决该问题。

var jq=jQuery.noConflict()，帮助您使用自己的名称（比如 jq）来代替 $ 符号。

结论

由于 jQuery 是为处理 HTML 事件而特别设计的，那么当您遵循以下原则时，您的代码会更恰当且更易维护：

把所有 jQuery 代码置于事件处理函数中

把所有事件处理函数置于文档就绪事件处理器中

把 jQuery 代码置于单独的 .js 文件中

如果存在名称冲突，则重命名 jQuery 库

jQuery 事件

下面是 jQuery 中事件方法的一些例子：



jQuery 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：

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

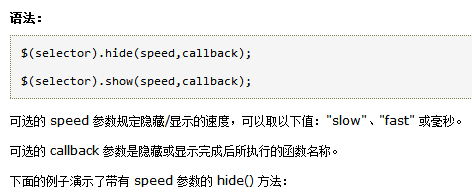
$("p").hide();

});

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

$("p").show();

});



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：

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

jQuery 效果 - 淡入淡出

jQuery Fading 方法

通过 jQuery，您可以实现元素的淡入淡出效果。

jQuery 拥有下面四种 fade 方法：

fadeIn()

fadeOut()

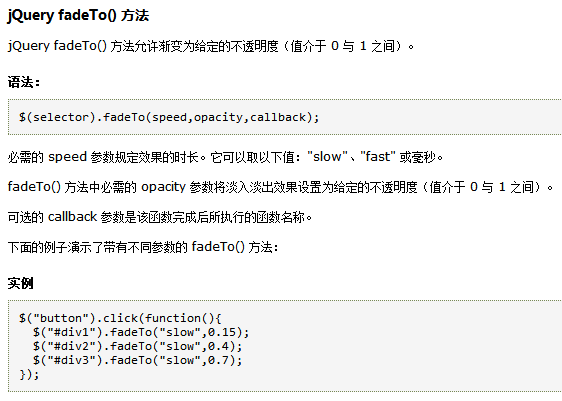
fadeToggle()

fadeTo()









jQuery 效果 - 滑动

jQuery 滑动方法

通过 jQuery，您可以在元素上创建滑动效果。

jQuery 拥有以下滑动方法：

slideDown()

slideUp()

slideToggle()



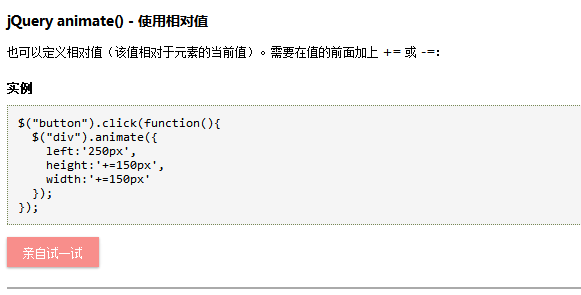
jQuery 效果 - 动画



提示：默认地，所有 HTML 元素都有一个静态位置，且无法移动。

如需对位置进行操作，要记得首先把元素的 CSS position 属性设置为 relative、fixed 或 absolute！





jQuery animate() - 使用预定义的值

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

实例

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

$("div").animate({

height:'toggle'

});

});

jQuery 停止动画

jQuery stop() 方法

jQuery stop() 方法用于停止动画或效果，在它们完成之前。

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

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

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

可选的 goToEnd 参数规定是否立即完成当前动画。默认是 false。

因此，默认地，stop() 会清除在被选元素上指定的当前动画。

下面的例子演示 stop() 方法，不带参数：

jQuery Callback 函数

jQuery – Chaining

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

例子 1

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

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

jQuery - 获得内容和属性

jQuery DOM 操作

jQuery 中非常重要的部分，就是操作 DOM 的能力。

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

提示：DOM = Document Object Model（文档对象模型）

DOM 定义访问 HTML 和 XML 文档的标准：

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

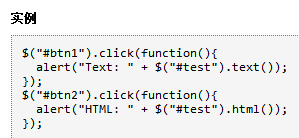
获得内容 - text()、html() 以及 val()

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



下面的例子演示如何通过 jQuery val() 方法获得输入字段的值：

实例

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

alert("Value: " + $("#test").val());

});

获取属性 - attr()

jQuery attr() 方法用于获取属性值。

下面的例子演示如何获得链接中 href 属性的值：

实例

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

alert($("#w3s").attr("href"));

});

jQuery - 设置内容和属性

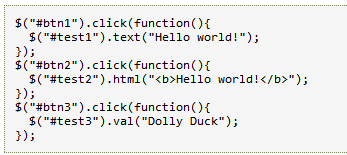
设置内容 - text()、html() 以及 val()

我们将使用前一章中的三个相同的方法来设置内容：

text() - 设置或返回所选元素的文本内容

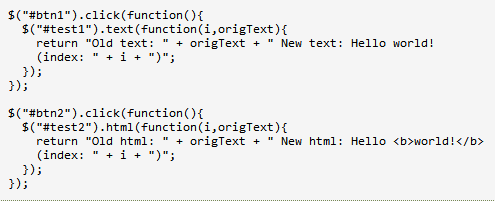
html() - 设置或返回所选元素的内容（包括 HTML 标记）

val() - 设置或返回表单字段的值



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。回调函数由两个参数：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。



设置属性 - attr()

jQuery attr() 方法也用于设置/改变属性值。

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

$("#w3s").attr("href","http://www.w3school.com.cn/jquery");

});

attr() 方法也允许您同时设置多个属性。

下面的例子演示如何同时设置 href 和 title 属性：

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

$("#w3s").attr({

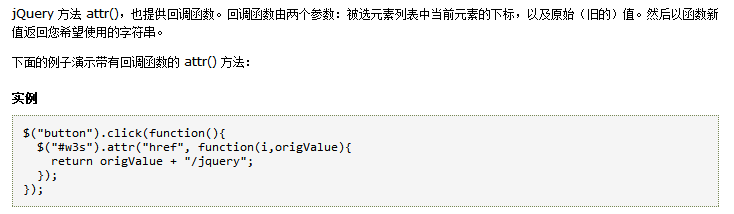
"href" : "http://www.w3school.com.cn/jquery",

"title" : "W3School jQuery Tutorial"

});

});

attr() 的回调函数



jQuery - 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 在元素内部

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 在元素之后插入一个新元素

before() - 在被选元素之前插入内容

$("p").append("Some appended text.");

$("p").prepend("Some prepended text.");

通过 append() 和 prepend() 方法添加若干新元素

function appendText()

{

var txt1="<p>Text.</p>"; // 以 HTML 创建新元素

var txt2=$("<p></p>").text("Text."); // 以 jQuery 创建新元素

var txt3=document.createElement("p"); // 以 DOM 创建新元素

txt3.innerHTML="Text.";

$("p").append(txt1,txt2,txt3); // 追加新元素

}

jQuery - 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

jQuery remove() 方法删除被选元素及其子元素。

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

jQuery empty() 方法删除被选元素的子元素。

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

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

该参数可以是任何 jQuery 选择器的语法。

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

jQuery - 获取并设置 CSS 类

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

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

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

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

});

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

$("#div1").addClass("important blue");

});

jQuery - css() 方法

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");

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

设置 CSS 属性

css("propertyname","value");

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

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

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

jQuery - 尺寸

jQuery 提供多个处理尺寸的重要方法：

width()

height()

innerWidth()

innerHeight()

outerWidth()

outerHeight()

jQuery width() 和 height() 方法

width() 方法设置或返回元素的宽度（不包括内边距、边框或外边距）。

height() 方法设置或返回元素的高度（不包括内边距、边框或外边距）。

下面的例子返回指定的 <div> 元素的宽度和高度：

实例

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

var txt="";

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

txt+="Height: " + $("#div1").height();

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

});

亲自试一试

jQuery innerWidth() 和 innerHeight() 方法

innerWidth() 方法返回元素的宽度（包括内边距）。

innerHeight() 方法返回元素的高度（包括内边距）。

下面的例子返回指定的 <div> 元素的 inner-width/height：

实例

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

var txt="";

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

txt+="Inner height: " + $("#div1").innerHeight();

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

});

亲自试一试

jQuery outerWidth() 和 outerHeight() 方法

outerWidth() 方法返回元素的宽度（包括内边距和边框）。

outerHeight() 方法返回元素的高度（包括内边距和边框）。

下面的例子返回指定的 <div> 元素的 outer-width/height：

实例

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

var txt="";

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

txt+="Outer height: " + $("#div1").outerHeight();

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

});

亲自试一试

outerWidth(true) 方法返回元素的宽度（包括内边距、边框和外边距）。

outerHeight(true) 方法返回元素的高度（包括内边距、边框和外边距）。

实例

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

var txt="";

txt+="Outer width (+margin): " + $("#div1").outerWidth(true) + "</br>";

txt+="Outer height (+margin): " + $("#div1").outerHeight(true);

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

});

亲自试一试

jQuery - 更多的 width() 和 height()

下面的例子返回文档（HTML 文档）和窗口（浏览器视口）的宽度和高度：

实例

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

var txt="";

txt+="Document width/height: " + $(document).width();

txt+="x" + $(document).height() + "\n";

txt+="Window width/height: " + $(window).width();

txt+="x" + $(window).height();

alert(txt);

});

亲自试一试

下面的例子设置指定的 <div> 元素的宽度和高度：

实例

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

$("#div1").width(500).height(500);

});

jQuery 遍历

jQuery 遍历 - 祖先

这些 jQuery 方法很有用，它们用于向上遍历 DOM 树：

parent()

parents()

parentsUntil()

jQuery 遍历 - 后代

children()

find()

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：

实例

$(document).ready(function(){

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

});

下面的例子返回 <div> 的所有后代：

实例

$(document).ready(function(){

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

});

jQuery 遍历 - 同胞

有许多有用的方法让我们在 DOM 树进行水平遍历：

siblings()

next()

nextAll()

nextUntil()

prev()

prevAll()

prevUntil()

jQuery siblings() 方法

siblings() 方法返回被选元素的所有同胞元素。

下面的例子返回 <h2> 的所有同胞元素：

实例

$(document).ready(function(){

$("h2").siblings();

});

亲自试一试

您也可以使用可选参数来过滤对同胞元素的搜索。

下面的例子返回属于 <h2> 的同胞元素的所有 <p> 元素：

实例

$(document).ready(function(){

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

});

亲自试一试

jQuery next() 方法

next() 方法返回被选元素的下一个同胞元素。

该方法只返回一个元素。

下面的例子返回 <h2> 的下一个同胞元素：

实例

$(document).ready(function(){

$("h2").next();

});

亲自试一试

jQuery nextAll() 方法

nextAll() 方法返回被选元素的所有跟随的同胞元素。

下面的例子返回 <h2> 的所有跟随的同胞元素：

实例

$(document).ready(function(){

$("h2").nextAll();

});

亲自试一试

jQuery nextUntil() 方法

nextUntil() 方法返回介于两个给定参数之间的所有跟随的同胞元素。

下面的例子返回介于 <h2> 与 <h6> 元素之间的所有同胞元素：

实例

$(document).ready(function(){

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

});

亲自试一试

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

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

jQuery 遍历 - 过滤

缩写搜索元素的范围

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

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

jQuery first() 方法

first() 方法返回被选元素的首个元素。

下面的例子选取首个 <div> 元素内部的第一个 <p> 元素：

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：

实例

$(document).ready(function(){

$("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：

实例

$(document).ready(function(){

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

});

jQuery eq() 方法

eq() 方法返回被选元素中带有指定索引号的元素。

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

实例

$(document).ready(function(){

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

});

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 "intro" 的所有 <p> 元素：

实例

$(document).ready(function(){

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

});

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 "intro" 的所有 <p> 元素：

实例

$(document).ready(function(){

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

});

jQuery - AJAX 简介

AJAX 是与服务器交换数据的艺术，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。

使用 AJAX 的应用程序案例：谷歌地图、腾讯微博、优酷视频、人人网等等

jQuery 提供多个与 AJAX 有关的方法。

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

提示：如果没有 jQuery，AJAX 编程还是有些难度的。

编写常规的 AJAX 代码并不容易，因为不同的浏览器对 AJAX 的实现并不相同。这意味着您必须编写额外的代码对浏览器进行测试。不过，jQuery 团队为我们解决了这个难题，我们只需要一行简单的代码，就可以实现 AJAX 功能。

jQuery - AJAX 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 and AJAX is FUN!!!</h2>

<p id="p1">This is some text in a paragraph.</p>

下面的例子会把文件 "demo\_test.txt" 的内容加载到指定的 <div> 元素中：

示例

$("#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：

实例

$("#div1").load("demo\_test.txt #p1");

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

responseTxt - 包含调用成功时的结果内容

statusTXT - 包含调用的状态

xhr - 包含 XMLHttpRequest 对象

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

实例

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

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

if(statusTxt=="success")

alert("外部内容加载成功！");

if(statusTxt=="error")

alert("Error: "+xhr.status+": "+xhr.statusText);

});

});

jQuery - AJAX get() 和 post() 方法

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

HTTP 请求：GET vs. POST

两种在客户端和服务器端进行请求-响应的常用方法是：GET 和 POST。

GET - 从指定的资源请求数据

POST - 向指定的资源提交要处理的数据

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

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

jQuery $.get() 方法

$.get() 方法通过 HTTP GET 请求从服务器上请求数据。

语法：

$.get(URL,callback);

必需的 URL 参数规定您希望请求的 URL。

可选的 callback 参数是请求成功后所执行的函数名。

下面的例子使用 $.get() 方法从服务器上的一个文件中取回数据：

实例

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

$.get("demo\_test.asp",function(data,status){

alert("Data: " + data + "\nStatus: " + status);

});

});

这个 ASP 文件 ("demo\_test.asp") 类似这样：

<%

response.write("This is some text from an external ASP file.")

%>

jQuery $.post() 方法

$.post() 方法通过 HTTP POST 请求从服务器上请求数据。

语法：

$.post(URL,data,callback);

必需的 URL 参数规定您希望请求的 URL。

可选的 data 参数规定连同请求发送的数据。

可选的 callback 参数是请求成功后所执行的函数名。

下面的例子使用 $.post() 连同请求一起发送数据：

实例

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

$.post("demo\_test\_post.asp",

{

name:"Donald Duck",

city:"Duckburg"

},

function(data,status){

alert("Data: " + data + "\nStatus: " + status);

});

});

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

然后我们连同请求（name 和 city）一起发送数据。

"demo\_test\_post.asp" 中的 ASP 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 ASP 文件 ("demo\_test\_post.asp") 类似这样：

<%

dim fname,city

fname=Request.Form("name")

city=Request.Form("city")

Response.Write("Dear " & fname & ". ")

Response.Write("Hope you live well in " & city & ".")

%>

jQuery - noConflict() 方法

jQuery 和其他 JavaScript 框架

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

如果其他 JavaScript 框架也使用 $ 符号作为简写怎么办？

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

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

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

jQuery noConflict() 方法

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

实例

当然，您仍然可以通过全名替代简写的方式来使用 jQuery：

$.noConflict();

jQuery(document).ready(function(){

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

jQuery("p").text("jQuery 仍在运行！");

});

});

亲自试一试

实例

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

var jq = $.noConflict();

jq(document).ready(function(){

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

jq("p").text("jQuery 仍在运行！");

});

});

亲自试一试

实例

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"：

$.noConflict();

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

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

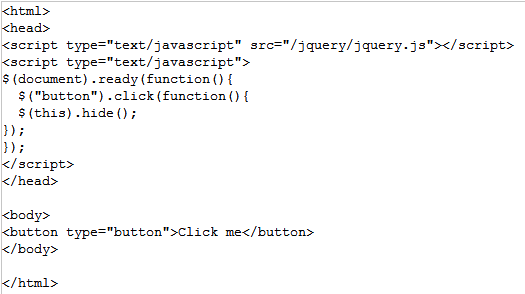
$("p").text("jQuery 仍在运行！");

});

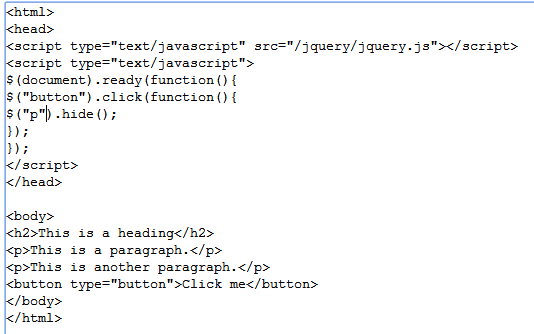
});

jQuery 实例

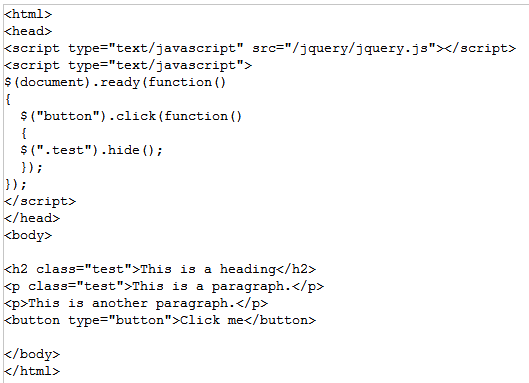
演示 jQuery 的 hide() 函数，隐藏当前的 HTML 元素。



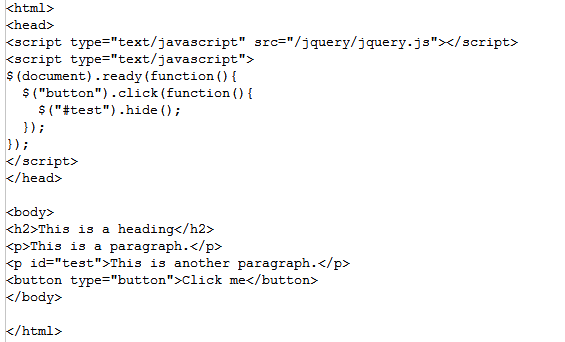
演示 jQuery 的 hide() 函数，隐藏所有 <p> 元素。



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



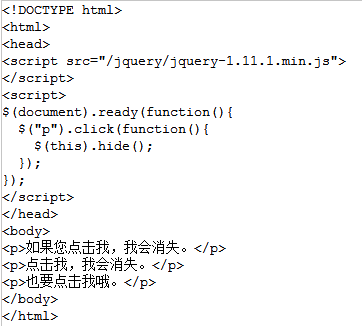
演示 jQuery 的 hide() 函数，隐藏 id="test" 的元素。



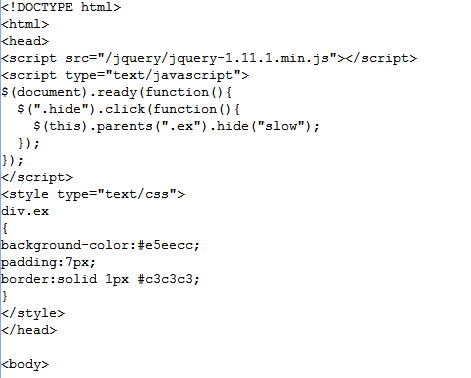
演示简单的 jQuery fadeout() 函数。

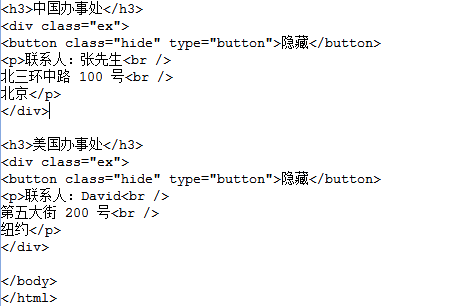


演示简单的 jQuery hide() 函数。

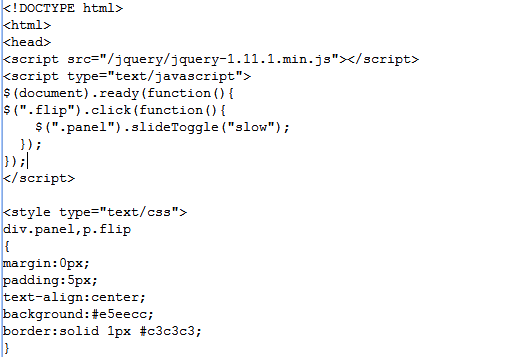


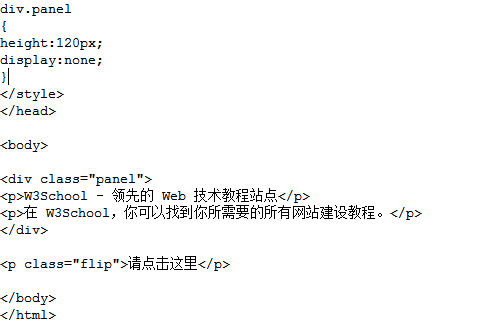
演示如何隐藏部分文本。





演示简单的 Slide Panel 效果。



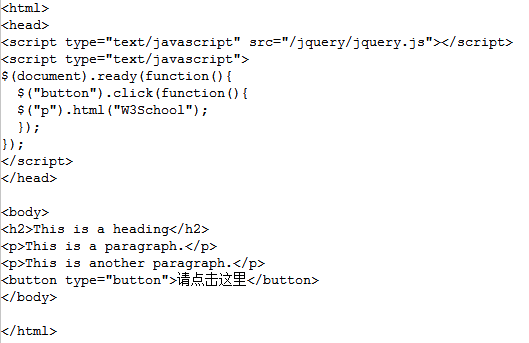


演示简单的 jQuery animate() 函数。

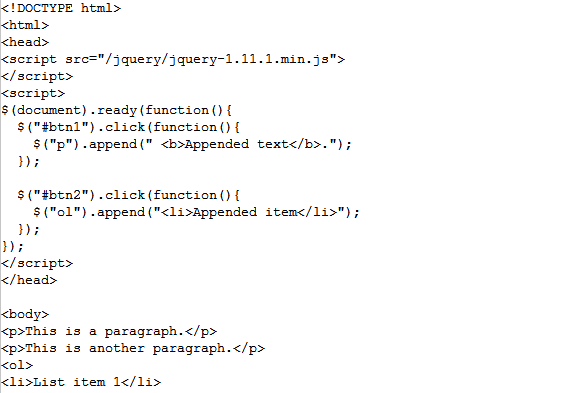


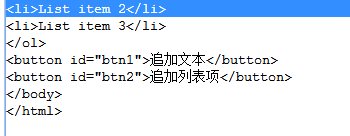


改变HTML里面的内容



向HTML里面追加内容

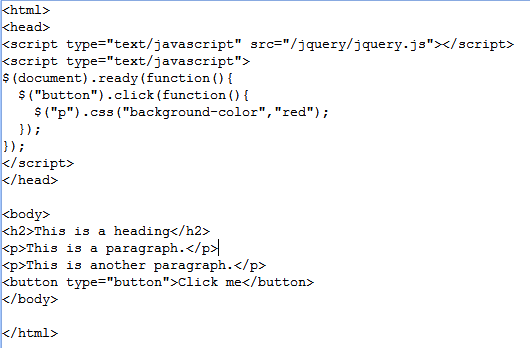




在HTML元素之后追加内容



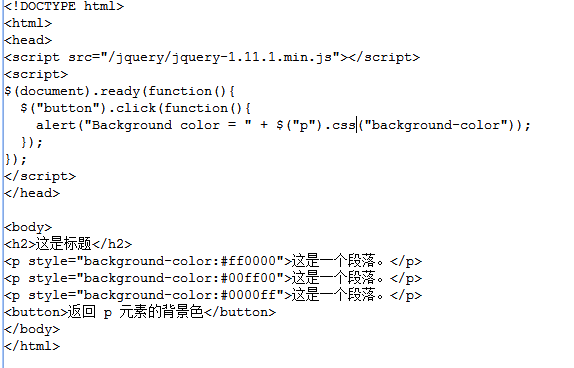
改变HTML元素的CSS属性



改变多个CSS属性



获得元素的CSS属性



使用 $(selector).load(url) 来改变 HTML 内容。



使用 $.ajax(options) 来改变 HTML 内容。

