HTML 中的脚本必须位于 <script> 与 </script> 标签之间。

脚本可被放置在 HTML 页面的 <body> 和 <head> 部分中。

**<head> 中的 JavaScript 函数**

在本例中，我们把一个 JavaScript 函数放置到 HTML 页面的 <head> 部分。

该函数会在点击按钮时被调用：

**实例**

<!DOCTYPE html>  
<html>

<head>  
<script>  
function myFunction()  
{  
document.getElementById("demo").innerHTML="我的第一个 JavaScript 函数";  
}  
</script>  
</head>

<body>

<h1>我的 Web 页面</h1>

<p id="demo">一个段落</p>

<button type="button" onclick="myFunction()">尝试一下</button>

</body>  
</html>

**<body> 中的 JavaScript 函数**

在本例中，我们把一个 JavaScript 函数放置到 HTML 页面的 <body> 部分。

该函数会在点击按钮时被调用：

**实例**

<!DOCTYPE html>  
<html>  
<body>

<h1>我的 Web 页面</h1>

<p id="demo">一个段落</p>

<button type="button" onclick="myFunction()">尝试一下</button>

<script>  
function myFunction()  
{  
document.getElementById("demo").innerHTML="我的第一个 JavaScript 函数";  
}  
</script>

</body>  
</html>

也可以把脚本保存到外部文件中。外部文件通常包含被多个网页使用的代码。

外部 JavaScript 文件的文件扩展名是 .js。

如需使用外部文件，请在 <script> 标签的 "src" 属性中设置该 .js 文件：

**实例**

<!DOCTYPE html>  
<html>  
<body>  
<script src="myScript.js"></script>  
</body>  
</html>

JavaScript 语句通常于**关键词**为开头。 **var** 关键词告诉浏览器创建一个新的变量：

var x = 5 + 6;  
var y = x \* 10;

不是所有的 JavaScript 语句都是"命令"。双斜杠 **//** 后的内容将会被浏览器忽略：

JavaScript 有多种数据类型：数字，字符串，数组，对象等等：

var length = 16;                                  // Number 通过数字字面量赋值   
var points = x \* 10;                              // Number 通过表达式字面量赋值  
var lastName = "Johnson";                         // String 通过字符串字面量赋值  
var cars = ["Saab", "Volvo", "BMW"];              // Array  通过数组字面量赋值  
var person = {firstName:"John", lastName:"Doe"};  // Object 通过对象字面量赋值

JavaScript 对大小写是敏感的, 基本if else和脚本语言类似

var person = {  
    firstName:"John",  
    lastName:"Doe",  
    age:50,  
    eyeColor:"blue"  
};

你可以通过两种方式访问对象属性:

person.lastName;

person["lastName"];

你可以使用以下语法创建对象方法：

*methodName : function() { code lines }*

你可以使用以下语法访问对象方法：

*objectName.methodName()*

Javascript 函数

<button onclick="myFunction('Harry Potter','Wizard')">Try it</button>  
  
<script>  
function myFunction(name,job)  
{  
alert("Welcome " + name + ", the " + job);  
}  
</script>

function myFunction()  
{  
var x=5;  
return x;  
}

**HTML 事件**

HTML 事件可以是浏览器行为，也可以是用户行为。

以下是 HTML 事件的实例：

* HTML 页面完成加载
* HTML input 字段改变时
* HTML 按钮被点击

通常，当事件发生时，你可以做些事情。

在事件触发时 JavaScript 可以执行一些代码。

HTML 元素中可以添加事件属性，使用 JavaScript 代码来添加 HTML 元素。

单引号:

<*some-HTML-element* *some-event*=**'*some JavaScript*'**>

双引号:

<*some-HTML-element* *some-event*=**"*some JavaScript*"**>

在以下实例中，按钮元素中添加了 onclick 属性 (并加上代码):

**实例**

<button onclick='getElementById("demo").innerHTML=Date()'>The time is?</button>

**常见的HTML事件**

下面是一些常见的HTML事件的列表:

|  |  |
| --- | --- |
| **事件** | **描述** |
| onchange | HTML 元素改变 |
| onclick | 用户点击 HTML 元素 |
| onmouseover | 用户在一个HTML元素上移动鼠标 |
| onmouseout | 用户从一个HTML元素上移开鼠标 |
| onkeydown | 用户按下键盘按键 |
| onload | 浏览器已完成页面的加载 |

JQuery

jQuery 是一个 JavaScript 函数库。

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

• 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。

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

示例

$(this).hide() - 隐藏当前元素

$("p").hide() - 隐藏所有段落

$(".test").hide() - 隐藏所有 class="test" 的所有元素

$("#test").hide() - 隐藏所有 id="test" 的元素

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 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");

亲自试一试

更多的选择器实例

语法 描述

$(this) 当前 HTML 元素

$("p") 所有 <p> 元素

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

$(".intro") 所有 class="intro" 的元素

$("#intro") id="intro" 的元素

$("ul li:first") 每个 <ul> 的第一个 <li> 元素

$("[href$='.jpg']") 所有带有以 ".jpg" 结尾的属性值的 href 属性

$("div#intro .head") id="intro" 的 <div> 元素中的所有 class="head" 的元素

jQuery 事件函数

jQuery 事件处理方法是 jQuery 中的核心函数。

<html>

<head>

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

<script type="text/javascript">

$(document).ready(function(){

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

$("p").hide();

});

});

</script>

</head>

<body>

<h2>This is a heading</h2>

<p>This is a paragraph.</p>

<p>This is another paragraph.</p>

<button>Click me</button>

</body>

</html>

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。

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：

Event 函数 绑定函数至

$(document).ready(function) 将函数绑定到文档的就绪事件（当文档完成加载时）

$(selector).click(function) 触发或将函数绑定到被选元素的点击事件

$(selector).dblclick(function) 触发或将函数绑定到被选元素的双击事件

$(selector).focus(function) 触发或将函数绑定到被选元素的获得焦点事件

$(selector).mouseover(function) 触发或将函数绑定到被选元素的鼠标悬停事件

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：

实例

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

$("p").toggle();

});

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 toggle() 方法完成后所执行的函数名称。

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()

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

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()

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

jQuery animate() 方法用于创建自定义动画。

语法：

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

下面的例子把 <div> 元素移动到右边，然后增加文本的字号：

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

var div=$("div");

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

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

});

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

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

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

});

callback

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

alert("The paragraph is now hidden");

});

通过 jQuery，您可以把动作/方法链接起来。

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）。

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

这样写也可以运行：

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

.slideUp(2000)

.slideDown(2000);

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 text() 和 html() 方法来获得内容：

实例

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

alert("Text: " + $("#test").text());

});

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

alert("HTML: " + $("#test").html());

});

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"));

});

下面的例子演示带有回调函数的 text() 和 html()：

实例

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

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

return "Old text: " + origText + " New text: Hello world!

(index: " + i + ")";

});

});

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

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

return "Old html: " + origText + " New html: Hello <b>world!</b>

(index: " + i + ")";

});

});

attr() 方法也允许您同时设置多个属性。

下面的例子演示如何同时设置 href 和 title 属性：

实例

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

$("#w3s").attr({

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

"title" : "W3School jQuery Tutorial"

});

});

attr() 的回调函数

jQuery 方法 attr()，也提供回调函数。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。

下面的例子演示带有回调函数的 attr() 方法：

实例

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

$("#w3s").attr("href", function(i,origValue){

return origValue + "/jquery";

});

});

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

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

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

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

}

after() 和 before() 类似

如需删除元素和内容，一般可使用以下两个 jQuery 方法：

remove() - 删除被选元素（及其子元素）

empty() - 从被选元素中删除子元素

jQuery remove() 方法

jQuery remove() 方法删除被选元素及其子元素。

实例

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

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

下面的例子删除 class="italic" 的所有 <p> 元素：

实例

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

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

下面的样式表将用于本页的所有例子：

.important

{

font-weight:bold;

font-size:xx-large;

}

.blue

{

color:blue;

}

jQuery addClass() 方法

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

实例

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

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

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

});

您也可以在 addClass() 方法中规定多个类：

实例

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

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

});

下面的例子演示如何不同的元素中删除指定的 class 属性：

实例

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

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

});

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

实例

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

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

});

返回 CSS 属性

如需返回指定的 CSS 属性的值，请使用如下语法：

css("propertyname");

下面的例子将返回首个匹配元素的 background-color 值：

实例

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

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");

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

如需设置多个 CSS 属性，请使用如下语法：

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

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

实例

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

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);

});

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);

});

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);

});

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>";

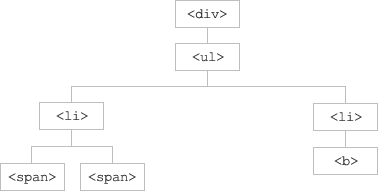
txt+="Outer height (+margin): " + $("#div1").outerHeight(true);

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

});

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。



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()

parent() 方法返回被选元素的直接父元素。

该方法只会向上一级对 DOM 树进行遍历。

下面的例子返回每个 <span> 元素的的直接父元素：

实例

$(document).ready(function(){

$("span").parent();

});

jQuery parents() 方法

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

下面的例子返回所有 <span> 元素的所有祖先：

实例

$(document).ready(function(){

$("span").parents();

});

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

实例

$(document).ready(function(){

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

});

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：

实例

$(document).ready(function(){

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

});

向下遍历 DOM 树

下面是两个用于向下遍历 DOM 树的 jQuery 方法：

children()

find()

jQuery children() 方法

children() 方法返回被选元素的所有直接子元素。

该方法只会向下一级对 DOM 树进行遍历。

下面的例子返回每个 <div> 元素的所有直接子元素：

实例

$(document).ready(function(){

$("div").children();

});

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：

实例

$(document).ready(function(){

$("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：

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("\*");

});

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();

});

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");

});

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();

});

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();

});

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");

});

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）。

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：

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);

});

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 "intro" 的所有 <p> 元素：

实例

$(document).ready(function(){

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

});

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 "intro" 的所有 <p> 元素：

实例

$(document).ready(function(){

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

});

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。

关于 jQuery 与 AJAX

jQuery 提供多个与 AJAX 有关的方法。

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

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

HTTP 请求：GET vs. POST

两种在客户端和服务器端进行请求-响应的常用方法是：GET 和 POST。

GET - 从指定的资源请求数据

POST - 向指定的资源提交要处理的数据

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

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

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

jQuery $.get() 方法

$.get() 方法通过 HTTP GET 请求从服务器上请求数据。

语法：

$.get(URL,callback);

必需的 URL 参数规定您希望请求的 URL。

可选的 callback 参数是请求成功后所执行的函数名。

下面的例子使用 $.get() 方法从服务器上的一个文件中取回数据：

实例

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

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

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

});

});

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

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 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 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！");

});

});

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！");

});

});

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！");

});

});