* [html](#_html:)
* [css](#_css)
* [javascript](#_javascript)
* [html dom](#_html_dom)
* [jQuery](#_jQuery)
* [AJAX](#_AJAX)

# html:

HTML 标签对大小写不敏感

注释不太一样<!—ahha-->

<!Doctype html>

<html>

<head>

<title>Title of the document</title>

<link rel="stylesheet" type="text/css" href="mystyle.css">//外部样式表

<style type="text/css"> //内部样式表

body {background-color: red}

p {margin-left: 20px}

</style>

</head>

<body>

<div> //div分块span在行里分小部分

<h1>heading</h1>

<p>para <b>This text is bold</b><span>哈哈哈哈哈哈哈哈 </span></p>

</div>

<a href=””>abc</a> //属性的值要加引号 abc是图像的话就是图像链接 像<><><>这样； target属性

<img src=””width=””> //中间没逗号， img直接闭合

<br/> //换行

<hr/> //水平分割线

<p style="font-family:arial;color:red;font-size:20px;">A paragraph.</p>

//内联样式 注意： ；

<p>WWF 的目标是：<q>构建世界。</q></p>//引用

<p>以下内容引用自 WWF 的网站：</p>//长引用

<blockquote cite="http://www.worldwildlife.org/who/index.html">

五十年来，WWF 一直致力于保护自然界的未来。

世界领先的环保组织，WWF 工作于 100 个国家，

并得到美国一百二十万会员及全球近五百万会员的支持。

</blockquote>

<p><abbr title="World Health Organization">WHO</abbr> 成立于 1948 年。</p> //缩写

<table border=”1”> //表格

<tr>

<th>name</th>

<th>id</th>

</tr>

<tr>

<td>maoyutao</td>

<td>249</td>

</tr>

</table>

<ul> //无序列表

<li>Coffee</li>

<li>Milk</li>

</ul>

<ol>//有序列表

<li>Coffee</li>

<li>Milk</li>

</ol>

<form onsubmit=””id=””> //表单 input(文本 单选 提交) select 下拉列表

<input type=”text”name=”firstname”value=”dufaultname”reaonly>//自己闭合 readonly表示只读 disabled表示禁用 （一样 没有等于）

<br/>

<input type="password" name="psw">

<br/>

<input type="checkbox" name="vehicle" value="Bike">I have a bike//多选框

<br/>

<input type="checkbox" name="vehicle" value="Car">I have a car

<br/>

<input name="" type="radio" id="checkbox" value="3" onclick="ChangBgColor(3);a1();a2()" /> //一次可以调用很多函数 用分号

<input type=”radio”name=”sex”value=”male”>male

<input type="radio" name="sex" value="female">Female

<br/>

<select name="cars">

<option value="volvo"selected>Volvo</option>//通过添加 selected 属性来定义预定义选项。

<option value="saab">Saab</option>

<option value="fiat">Fiat</option>

<option value="audi">Audi</option>

</select>

<br/>

<textarea name="message" rows="10" cols="30">

The cat was playing in the garden.

</textarea>

<br/>

<button type="button" onclick="alert('Hello World!')">Click Me!</button>

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

/\*Action 属性

\*action 属性定义在提交表单时执行的动作。

\*向服务器提交表单的通常做法是使用提交按钮。

\*通常，表单会被提交到 web 服务器上的网页。

\*在上面的例子中，指定了某个服务器脚本来处理被提交表单：

\*<form action="action\_page.php">

\*如果省略 action 属性，则 action 会被设置为当前页面。

\*如果要正确地被提交，每个输入字段必须设置一个 name 属性。否则不会被提交

\*/

</form>

</body>

</html>

锚

<p>锚 通过使用 name 属性 - 创建文档内的书签 实现文档内的跳转 <br/>

命名锚的语法：

<a name="label">锚（显示在页面上的文本）</a></p>//定义也要a

<a href="#label">查看锚</a>

<p>也可以在其他页面中创建指向该锚的链接,我们将 # 符号和锚名称添加到 URL 的末端，就可以直接链接到 tips 这个命名锚了。</p>

框架：

<html>

<frameset cols="25%,50%,25%">

<frame src="/example/html/frame\_a.html#name">//跳到指定的节

<frame src="/example/html/frame\_b.html">

<frameset rows="50%,50%">

<frame src="/example/html/frame\_c.html">

<frame src="/example/html/frame\_d.html">

<noframes>

<body>您的浏览器无法处理框架！</body>

</noframes>

</frameset>

</html>

内联框架

<iframe src="demo\_iframe.htm" width="200" height="200"></iframe>

实体：

&nbsp; 空格

html5语义元素:

|  |  |
| --- | --- |
| <article> | 定义文章。 |
| <aside> | 定义页面内容以外的内容。 |
| <details> | 定义用户能够查看或隐藏的额外细节。 |
| <figcaption> | 定义 <figure> 元素的标题。 |
| <figure> | 规定自包含内容，比如图示、图表、照片、代码清单等。 |
| <footer> | 定义文档或节的页脚。 |
| <header> | 规定文档或节的页眉。 |
| <main> | 规定文档的主内容。 |
| <mark> | 定义重要的或强调的文本。 |
| <nav> | 定义导航链接。 |
| <section> | 定义文档中的节。 |
| <summary> | 定义 <details> 元素的可见标题。 |
| <time> | 定义日期/时间。 |

# css

浏览器缺省设置

外部样式表//注意后缀名

<head>

<link rel="stylesheet" type="text/css" href="mystyle.css" />

</head>

内部样式表（位于 <head> 标签内部）

<head>

<style type="text/css">

hr {color: sienna;}

p {margin-left: 20px;}

body {background-image: url("images/back40.gif");}

</style>

</head>

内联样式（在 HTML 元素内部）

<p style="color: sienna; margin-left: 20px"></p>

h1,h2,h3,h4,h5,h6 {

color: green;

}

body {

font-family: Verdana, sans-serif;

} //子元素会继承 若不想要继承 就单独写出子元素的样式

li strong {

font-style: italic;

font-weight: normal;

} //派生选择器

h1 > strong {color:red;} //子元素选择器

h1 + p {margin-top:50px;}//相邻兄弟选择器 p在后面

p:first-child {font-weight: bold;}//伪类

p:first-line

{

color:#ff0000;

font-variant:small-caps;

} //伪元素

":after" 伪元素可以在元素的内容之后插入新内容。（before）

下面的例子在每个 <h1> 元素后面插入一幅图片：

h1:after

{

content:url(logo.gif);

}

#sidebar {

font-style: italic;

text-align: right;

margin-top: 0.5em;

}//id选择器

与类不同，在一个 HTML 文档中，ID 选择器会使用一次，而且仅一次。

不同于类选择器，ID 选择器不能结合使用，因为 ID 属性不允许有以空格分隔的词列表。

p#sidebar {

border: 1px dotted #000;

padding: 10px;

}// p里找

.fancy td {

color: #f60;

background: #666;

}//类选择器 <h1 class="fancy mike"></h1> 可以有多类 空格隔开

通过把两个类选择器链接在一起，仅可以选择同时包含这些类名的元素（类名的顺序不限）。

.important.urgent {background:silver;}

td.fancy {

color: #f60;

background: #666;

}//td（单元格）里找

[title]

{

color:red;

}//属性选择器

[title~=hello] { color:red; }

input[type="text"]

{

width:150px;

display:block;

margin-bottom:10px;

background-color:yellow;

font-family: Verdana, Arial;

}//前面也能加东西

|  |  |
| --- | --- |
| [[*attribute*]](http://www.w3school.com.cn/cssref/selector_attribute.asp) | 用于选取带有指定属性的元素。 |
| [[*attribute*=*value*]](http://www.w3school.com.cn/cssref/selector_attribute_value.asp) | 用于选取带有指定属性和值的元素。 |
| [[*attribute*~=*value*]](http://www.w3school.com.cn/cssref/selector_attribute_value_contain.asp) | 用于选取属性值中包含指定词汇的元素。 |
| [[*attribute*|=*value*]](http://www.w3school.com.cn/cssref/selector_attribute_value_start.asp) | 用于选取带有以指定值开头的属性值的元素，该值必须是整个单词。 |
| [[*attribute*^=*value*]](http://www.w3school.com.cn/cssref/selector_attr_begin.asp) | 匹配属性值以指定值开头的每个元素。 |
| [[*attribute*$=*value*]](http://www.w3school.com.cn/cssref/selector_attr_end.asp) | 匹配属性值以指定值结尾的每个元素。 |
| [[*attribute*\*=*value*]](http://www.w3school.com.cn/cssref/selector_attr_contain.asp) | 匹配属性值中包含指定值的每个元素。 |

链接：

a:link - 普通的、未被访问的链接

a:visited - 用户已访问的链接

a:hover - 鼠标指针位于链接的上方

a:active - 链接被点击的时刻

定位：

CSS 有三种基本的定位机制：普通流、浮动和绝对定位。

position 属性值的含义：

static

元素框正常生成。块级元素生成一个矩形框，作为文档流的一部分，行内元素则会创建一个或多个行框，置于其父元素中。

relative

元素框偏移某个距离。元素仍保持其未定位前的形状，它原本所占的空间仍保留。

#box\_relative {

position: relative;

left: 30px;

top: 20px;

}

absolute

元素框从文档流完全删除，并相对于其包含块定位。包含块可能是文档中的另一个元素或者是初始包含块。元素原先在正常文档流中所占的空间会关闭，就好像元素原来不存在一样。元素定位后生成一个块级框，而不论原来它在正常流中生成何种类型的框。

#box\_relative {

position: absolute;

left: 30px;

top: 20px;

}

fixed

元素框的表现类似于将 position 设置为 absolute，不过其包含块是视窗本身。

提示：相对定位实际上被看作普通流定位模型的一部分，因为元素的位置相对于它在普通流中的位置。

浮动：

.news p {

float: right;

}

.clear {

clear: both;

}//用来禁止浮动 left表示左边禁止

# javascript

JavaScript 对大小写是敏感的。

注释正常写

您可以在 HTML 文档中放入不限数量的脚本。

脚本可位于 HTML 的 <body> 或 <head> 部分中，或者同时存在于两个部分中。

通常的做法是把函数放入 <head> 部分中，或者放在页面底部。这样就可以把它们安置到同一处位置，不会干扰页面的内容。

也可引用外部文档 提示：外部脚本不能包含 <script> 标签。

<!DOCTYPE html>

<html>

<body>

<script src="myScript.js"></script>

</body>

</html> //注意后缀名

<script>

document.write("<p>My First JavaScript</p>");

</script>

//请使用 document.write() 仅仅向文档输出写内容。

如果在文档已完成加载后执行 document.write，整个 HTML 页面将被覆盖

对代码行进行折行

您可以在文本字符串中使用反斜杠对代码行进行换行。下面的例子会正确地显示：

document.write("Hello \

World!");

不过，您不能像这样折行：

document.write \

("Hello World!");

变量

var carname;

carname="Volvo"; //引号！！

var name="Gates", age=56, job="CEO";

var cars=new Array();//数组

cars[0]="Audi";

cars[1]="BMW";

cars[2]="Volvo";

var cars=new Array("Audi","BMW","Volvo");

JavaScript 对象：

对象由花括号分隔。在括号内部，对象的属性以名称和值对的形式 (name : value) 来定义。空格和折行无关紧要。声明可横跨多行，属性由逗号分隔：（对比记忆 ：类似style 但和元素里的属性组合不同）

var person={firstname:"Bill", lastname:"Gates", id:5566};

对象寻址：

name=person.lastname;

name=person["lastname"];

类型：

var carname=new String;

var x= new Number;

var y= new Boolean;

var cars= new Array;

var person= new Object;//对象

**字符串、数字、布尔、数组、对象、Null、Undefined**

数字和布尔不加引号

可以直接这样：

person=new Object();

person.firstname="Bill";

person.lastname="Gates";

person.age=56;

person.eyecolor="blue";

方法：

var x=message.toUpperCase(); //变大写

函数：

<script>

function myFunction()

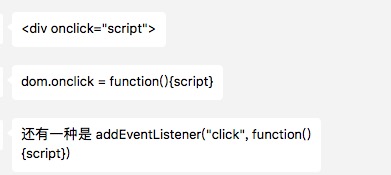
{

document.getElementById("demo").innerHTML="My First JavaScript Function";

}

</script>

<button type="button" onclick="myFunction()">Try it</button>



还有一种是$(“#button1”).click(function(){})

## 带有返回值的函数

function myFunction()

{

var x=5;

return x;

}

var myVar=myFunction();

向未声明的 JavaScript 变量来分配值

如果您把值赋给尚未声明的变量，该变量将被自动作为全局变量声明。

这条语句：

carname="Volvo";

将声明一个全局变量 carname，即使它在函数内执行。

运算符：

用于字符串的 + 运算符

+ 运算符用于把文本值或字符串变量加起来（连接起来）。

如需把两个或多个字符串变量连接起来，请使用 + 运算符。

txt1="What a very";

txt2="nice day";

txt3=txt1+txt2;

**如果把数字与字符串相加，结果将成为字符串。**

其他+-\*／（直接除 不是整除） % += == != < <= ？：都一样

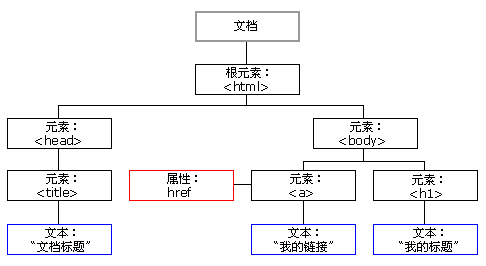
(==是等于 ===是值和类型都等于)

if else switch for while break continue try catch throw

补：通过标签引用，break 语句可用于跳出任何 JavaScript 代码块

正则表达式

# html dom



|  |  |
| --- | --- |
| getElementById() | 返回带有指定 ID 的元素。 |
| getElementsByTagName() | 返回包含带有指定标签名称的所有元素的节点列表（集合/节点数组）。 |
| getElementsByClassName() | 返回包含带有指定类名的所有元素的节点列表。 |
| appendChild() | 把新的子节点添加到指定节点。 |
| removeChild() | 删除子节点。 |
| replaceChild() | 替换子节点。 |
| insertBefore() | 在指定的子节点前面插入新的子节点。 |
| createAttribute() | 创建属性节点。 |
| createElement() | 创建元素节点。 |
| createTextNode() | 创建文本节点。 |
| getAttribute() | 返回指定的属性值。 |
| setAttribute() | 把指定属性设置或修改为指定的值。 |

## nodeName 属性

nodeName 属性规定节点的名称。

* nodeName 是只读的
* 元素节点的 nodeName 与标签名相同
* 属性节点的 nodeName 与属性名相同
* 文本节点的 nodeName 始终是 #text
* 文档节点的 nodeName 始终是 #document

## nodeValue 属性

nodeValue 属性规定节点的值。

* 元素节点的 nodeValue 是 undefined 或 null
* 文本节点的 nodeValue 是文本本身
* 属性节点的 nodeValue 是属性值

var txt=document.getElementById("intro").childNodes[0].nodeValue;

## nodeType 属性

nodeType 属性返回节点的类型。nodeType 是只读的。

比较重要的节点类型有：

|  |  |
| --- | --- |
| 元素 | 1 |
| 属性 | 2 |
| 文本 | 3 |
| 注释 | 8 |
| 文档 | 9 |

查找元素：

1.id

var x=document.getElementById("intro");

2.标签

var x=document.getElementById("main");

var y=x.getElementsByTagName("p"); //document. 换成x.

3.类

document.getElementsByClassName("intro");

注：节点列表：

getElementsByTagName() 方法返回节点列表。节点列表是一个节点数组。

下面的代码选取文档中的所有 <p> 节点：

实例

var x=document.getElementsByTagName("p");

可以通过下标号（0开始）访问这些节点。如需访问第二个 <p>，您可以这么写：

y=x[1];

节点长度：x.length

这里有两个特殊的属性，可以访问全部文档：

document.documentElement - 全部文档

document.body - 文档的主体

（方法有（） 属性没有）

修改内容：

document.getElementById("p1").innerHTML="New text!";

<h1 onclick="this.innerHTML='谢谢!'">请点击该文本</h1>//this的用法在script外面

修改属性：

document.getElementById("image").src="landscape.jpg";

修改样式：

document.getElementById("p2").style.color="blue";

<button type="button" onclick="document.getElementById('id1').style.color='red'">//三个引号在script外面

使用 HTML DOM 来分配事件：

document.getElementById("myBtn").onclick=function(){displayDate()};//没引号噢！ 还有addEventListener("click", function(){script})

还有直接的button里的onclick=”displaydate()”

<body onload="checkCookies()">

<input type="text" id="fname" onchange="upperCase()">

onmouseover 和 onmouseout 事件

onmouseover 和 onmouseout 事件可用于在用户的鼠标移至 HTML 元素上方或移出元素时触发函数。

onmousedown、onmouseup 以及 onclick 事件

onmousedown, onmouseup 以及 onclick 构成了鼠标点击事件的所有部分。首先当点击鼠标按钮时，会触发 onmousedown 事件，当释放鼠标按钮时，会触发 onmouseup 事件，最后，当完成鼠标点击时，会触发 onclick 事件。

onfocus当输入字段获得焦点

增删节点：

方法向节点添加最后一个子节点。

提示：如果您需要创建包含文本的新段落，请记得添加到段落的文本的文本节点，然后向文档添加该段落。

<script>

var para=document.createElement("p");

var node=document.createTextNode("这是新段落。");

para.appendChild(node);

var element=document.getElementById("div1");

element.appendChild(para);

</script>

<script>

var parent=document.getElementById("div1");

var child=document.getElementById("p1");

parent.removeChild(child);

</script>

或：

var child=document.getElementById("p1");

child.parentNode.removeChild(child);

//还有parentNode、firstChild 以及 lastChild

替换：

var para=document.createElement("p");

var node=document.createTextNode("This is new.");

para.appendChild(node);

var parent=document.getElementById("div1");

var child=document.getElementById("p1");

parent.replaceChild(para,child);

# jQuery

<head>

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

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

</head>

* 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

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。

<head>

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

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

</head>

所有 jQuery 函数位于一个 document ready 函数中：

$(document).ready(function(){

--- jQuery functions go here ----

}); //document没引号，下面的很多都有

基础语法： $(selector).action()

$(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

和css一样 有元素选择器 属性选择器 都要引号

例：$("ul li:first") $("[href$='.jpg']") $("div#intro .head")

css选择器

所有 p 元素的背景颜色更改为红色：

$("p").css("background-color","red");//注意标点

事件：

$(document).ready(function(){

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

$("p").hide();

});

});

当点击元素时，会发生 click 事件。click() 方法触发 click 事件，或规定当发生 click 事件时运行的函数。

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。

双击：$(selector).dblclick(function)

高级效果：

隐藏显示：

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

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

$(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。

Toggle 切换

淡入淡出：

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

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

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

$(selector).fadeTo(speed,opacity,callback); //透明度

滑动：

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

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

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

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。

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！

注意冒号和引号：

$("div").animate({

left:'250px',

opacity:'0.5',

height:'150px',

width:'150px'

});

$("div").animate({

left:'250px',

height:'+=150px',

width:'+=150px'

});

队列功能：

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：

$(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。

Callback：

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

alert("The paragraph is now hidden");

});

Callback 函数在当前动画 100% 完成之后执行

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

加折行：

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

.slideUp(2000)

.slideDown(2000);

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() 方法用于设置或获取属性值。

css() 设置或返回样式

返回的实例：

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

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

});

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

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

});

返回首个匹配元素的 background-color 值：

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

设置的实例：

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

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

});

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

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

});

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

$("#test3").val("Dolly Duck");

});

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

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

});

为所有匹配元素设置样式：

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

$("p").css({"background-color":"yellow","font-size":"200%"});//多个样式的时候符号就要变拉！

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。

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 + ")";

});

});

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

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

return origValue + "/jquery";

});

});

同时设置多个（跟css选择器那里还是不一样）：

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

$("#w3s").attr({

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

"title" : "W3School jQuery Tutorial"

});

});

添加：

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

例：

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

创建新元素：使用jquery创建新元素的方法为：$(html标签)，例如 $("<p></p>")创建了一个段落。然后再添加。

$(function(){

    $("input[type='button']").click(function() {

        new\_obj = $("<input type='text'>");

        $(this).before(new\_obj);

    });

});

比较：

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

}

删除：

Query remove() 方法删除被选元素及其子元素。

实例：

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

jQuery empty() 方法删除被选元素的子元素。

实例

$("#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。

下面的例子删除 class="italic" 的所有 <p> 元素：

实例：

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

jQuery 操作 CSS

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

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

$("h1,h2,p").addClass("blue important");//可以是多个类

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

});

遍历：

parent()

parents()

parentsUntil()

children() 所有直接子元素 参数过滤

find()被选元素的后代元素，一路向下直到最后一个后代

siblings()被选元素的所有同胞元素

next()下一个同胞元素

nextAll()被选元素的所有跟随的同胞元素 后面的

nextUntil()

prev()

prevAll()

prevUntil()

筛选：

first() 方法返回被选元素的首个元素。

last() 方法返回被选元素的最后一个元素。

eq() 方法返回被选元素中带有指定索引号的元素。 eq(4) 从0开始

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。 $("p").filter(".intro");

not() 方法与 filter() 相反。

例：

$("span").parents("ul"); 参数用来筛选

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

AJAX load() 方法从服务器加载数据，并把返回的数据放入被选元素

$(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。

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

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);

});

});//xhr=XMLHttpRequest

$.get()

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。

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

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

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

});

});

$.post(URL,data,callback);//三个参数！

# AJAX

var xmlhttp;

if (window.XMLHttpRequest)

{// code for IE7+, Firefox, Chrome, Opera, Safari

xmlhttp=new XMLHttpRequest();

}

else

{// code for IE6, IE5

xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");

}

|  |  |
| --- | --- |
| open(*method*,*url*,*async*) | 规定请求的类型、URL 以及是否异步处理请求。   * *method*：请求的类型；GET 或 POST * *url*：文件在服务器上的位置 * *async*：true（异步）或 false（同步） |
| send(*string*) | 将请求发送到服务器。   * *string*：仅用于 POST 请求 |

为了避免打开缓存，请向 URL 添加一个唯一的 ID：

xmlhttp.open("GET","demo\_get.asp?t=" + Math.random(),true);

xmlhttp.send();

如果您希望通过 GET 方法发送信息，请向 URL 添加信息：

xmlhttp.open("GET","demo\_get2.asp?fname=Bill&lname=Gates",true);

xmlhttp.send();

如果需要像 HTML 表单那样 POST 数据，请使用 setRequestHeader() 来添加 HTTP 头。然后在 send() 方法中规定您希望发送的数据：

xmlhttp.open("POST","ajax\_test.asp",true);

xmlhttp.setRequestHeader("Content-type","application/x-www-form-urlencoded");

xmlhttp.send("fname=Bill&lname=Gates");

响应：

document.getElementById("myDiv").innerHTML=xmlhttp.responseText;

当请求被发送到服务器时，我们需要执行一些基于响应的任务。

xmlhttp.onreadystatechange=function()

{

if (xmlhttp.readyState==4 && xmlhttp.status==200)

{

document.getElementById("myDiv").innerHTML=xmlhttp.responseText;

}

}

完整例子

<html>

<head>

<script type="text/javascript">

function loadXMLDoc()

{

var xmlhttp;

if (window.XMLHttpRequest)

{// code for IE7+, Firefox, Chrome, Opera, Safari

xmlhttp=new XMLHttpRequest();

}

else

{// code for IE6, IE5

xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");

}

xmlhttp.onreadystatechange=function()

{

if (xmlhttp.readyState==4 && xmlhttp.status==200)

{

document.getElementById("myDiv").innerHTML=xmlhttp.responseText;

}

}

xmlhttp.open("GET","/ajax/test1.txt",true);

xmlhttp.send();

}

</script>

</head>

<body>

<div id="myDiv"><h2>Let AJAX change this text</h2></div>

<button type="button" onclick="loadXMLDoc()">通过 AJAX 改变内容</button>

</body>

</html>