* jQuery 简介

jQuery 极大地简化了 JavaScript 编程。

jQuery 库可以通过一行简单的标记被添加到网页中。

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 语法

通过 jQuery，您可以选取（查询，query） HTML 元素，并对它们执行“操作”（actions）。

**1.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。

**2.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

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。

* jQuery 选择器

**选择器允许您对元素组或单个元素进行操作。**

**1.jQuery 选择器**

在前面的章节中，我们展示了一些有关如何选取 HTML 元素的实例。

jQuery 元素选择器和属性选择器允许您通过标签名、属性名或内容对 HTML 元素进行选择。

选择器允许您对 HTML 元素组或单个元素进行操作。

在 HTML DOM 术语中：

选择器允许您对 DOM 元素组或单个 DOM 节点进行操作。

**2.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。

**3.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。

**4.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：

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

**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 |

如需完整的参考手册，请访问 [jQuery 选择器参考手册](http://doc.tedu.cn/w3/jquery/jquery_ref_selectors.html)。

* jQuery 事件

**jQuery 是为事件处理特别设计的。**

**1.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：

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

$("p").hide();

});

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();**

**2.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。

<head>

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

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

</head>

**3.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

**4.jQuery 事件**

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 |

如需完整的参考手册，请访问[jQuery 事件参考手册](http://doc.tedu.cn/w3/jquery/jquery_ref_events.html)。

* jQuery - 获得内容和属性

jQuery 拥有可操作 HTML 元素和属性的强大方法。

jQuery DOM 操作

jQuery 中非常重要的部分，就是操作 DOM 的能力。

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：

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：

$("#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：

$("#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：

$("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

下面的例子演示如何通过 text()、html() 以及 val() 方法来设置内容：

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

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

});

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

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

});

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

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

});

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。

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 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。

* jQuery - 添加元素

通过 jQuery，可以很容易地添加新元素/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

**jQuery append() 方法**

jQuery append() 方法在被选元素的结尾插入内容。

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

**jQuery prepend() 方法**

jQuery prepend() 方法在被选元素的开头插入内容。

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

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。

在下面的例子中，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）：

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 after() 和 before() 方法**

jQuery after() 方法在被选元素之后插入内容。

jQuery before() 方法在被选元素之前插入内容。

$("img").after("Some text after");

$("img").before("Some text before");

**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。

在下面的例子中，我们创建若干新元素。这些元素可以通过 text/HTML、jQuery 或者 JavaScript/DOM 来创建。然后我们通过 after() 方法把这些新元素插到文本中（对 before() 同样有效）：

function afterText()

var txt1="<b>I </b>"; // 以 HTML 创建新元素

var txt2=$("<i></i>").text("love "); // 通过 jQuery 创建新元素

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

txt3.innerHTML="jQuery!";

$("img").after(txt1,txt2,txt3); // 在 img 之后插入新元素

* jQuery - 删除元素

通过 jQuery，可以很容易地删除已有的 HTML 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

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

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

**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。

该参数可以是任何 jQuery 选择器的语法。

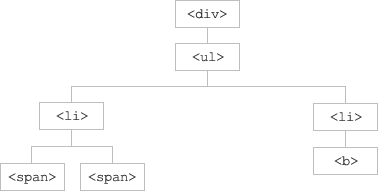
下面的例子删除 class="italic" 的所有 <p> 元素：

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

* jQuery 遍历

什么是遍历？

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。



遍历 DOM 树 •<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**

遍历方法中最大的种类是树遍历（tree-traversal）。

**1.jQuery 遍历 - 祖先**

祖先是父、祖父或曾祖父等等。

通过 jQuery，您能够向上遍历 DOM 树，以查找元素的祖先。

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。

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：

$(document).ready(function(){

$("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：

$(document).ready(function(){

$("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：

$(document).ready(function(){

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

});

**2.jQuery 遍历 - 后代**

后代是子、孙、曾孙等等。

通过 jQuery，您能够向下遍历 DOM 树，以查找元素的后代。

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：

$(document).ready(function(){

$("div").children();

});

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：

$(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：

$(document).ready(function(){

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

});

下面的例子返回 <div> 的所有后代：

$(document).ready(function(){

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

});

**3.jQuery 遍历 - 同胞**

同胞拥有相同的父元素。

通过 jQuery，您能够在 DOM 树中遍历元素的同胞元素。

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：

$(document).ready(function(){

$("h2").siblings();

});

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：

$(document).ready(function(){

$("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：

$(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：

$(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：

$(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）。

**4.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：

$(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：

$(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）：

$(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：

$(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：

$(document).ready(function(){

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

});

* **演示1:**

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Insert title here</title>

<script src="../js/jquery-1.11.1.js"></script>

<script>

function bigger() {

//获取段落原来的字号(16px)

var size = $("p").css("font-size");

//去掉单位以便于计算

size = size.replace("px","");

//字号+1再设置给所有段落

$("p").css("font-size",++size+"px");

}

</script>

</head>

<body>

<input type="button" value="+"

onclick="bigger();"/>

<p>1.jquery是一个轻量级的框架</p>

<p>2.它提供了简洁而统一的API</p>

<p>3.它极大的提高了js编程效率</p>

</body>

**演示2:动画**

</html>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Insert title here</title>

<style>

div {

width: 200px;

height: 200px;

background-color: red;

/\*自定义动画基于定位\*/

position: relative;

}

</style>

<script src="../js/jquery-1.11.1.js"></script>

<script>

function f1() {

$("div").show(3000);

}

function f2() {

//参数2是匿名函数,在动画完成时,

//由jquery自动调用.

//某件事完成时被自动调用的函数,

//称之为回调函数.

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

console.log("OVER");

});

//动画的实现原理:

//通过定时器连续不断的修改元素的样式.

//定时器和主线程(f2)并发执行.

console.log("OK");

}

$(function(){

//给div绑定悬停事件

$("div").hover(

function(){

$(this).animate({"left":"20px"},600);

},

function(){

$(this).animate({"left":"0"},600);

}

);

});

</script>

</head>

<body>

<p>

<input type="button" value="显示"

onclick="f1();"/>

<input type="button" value="隐藏"

onclick="f2();"/>

</p>

<div></div>

</body>

* **演示:3**

</html>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Insert title here</title>

<script src="../js/jquery-1.11.1.js"></script>

<script>

$(function(){

//1.读写节点的内容

//1) html() == innerHTML

console.log($("p:first").html());

$("p:first").html("1.jQuery支持<u>读写</u>节点");

//2) text() == innerText

//2.读写节点的值

//val() == value

console.log($(":button:first").val());

$(":button:first").val("BBB");

//3.读写节点的属性

//attr() == setAttribute()+getAttribute()

console.log($("img:first").attr("src"));

$("img:first").attr("src","../images/02.jpg");

//根据层次查询节点

var gz = $("#gz");

console.log(gz.parent());

console.log(gz.prev());

console.log(gz.siblings());

var ul = $("ul");

console.log(ul.find("li[id]"));

});

function f1() {

var li = $("<li>天津</li>");

console.log(li);

$("ul").append(li);

}

function f2() {

var li = $("<li>澳门</li>");

$("#gz").after(li);

}

function f3() {

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

}

</script>

</head>

<body>

<p>1.jQuery支持<b>读写</b>节点</p>

<p>2.jQuery支持<b>增删</b>节点</p>

<p>3.jQuery支持<b>查询</b>节点</p>

<p>

<input type="button" value="AAA"/>

</p>

<p>

<img src="../images/01.jpg"/>

</p>

<ul>

<li>北京</li>

<li>上海</li>

<li id="gz">广州</li>

<li>深圳</li>

<li>杭州</li>

</ul>

<p>

<input type="button" value="追加"

onclick="f1();"/>

<input type="button" value="插入"

onclick="f2();"/>

<input type="button" value="删除"

onclick="f3();"/>

</p>

</body>

</html>