# 第八章 jquery基础-选择器和样式操作

# 本节所讲内容：

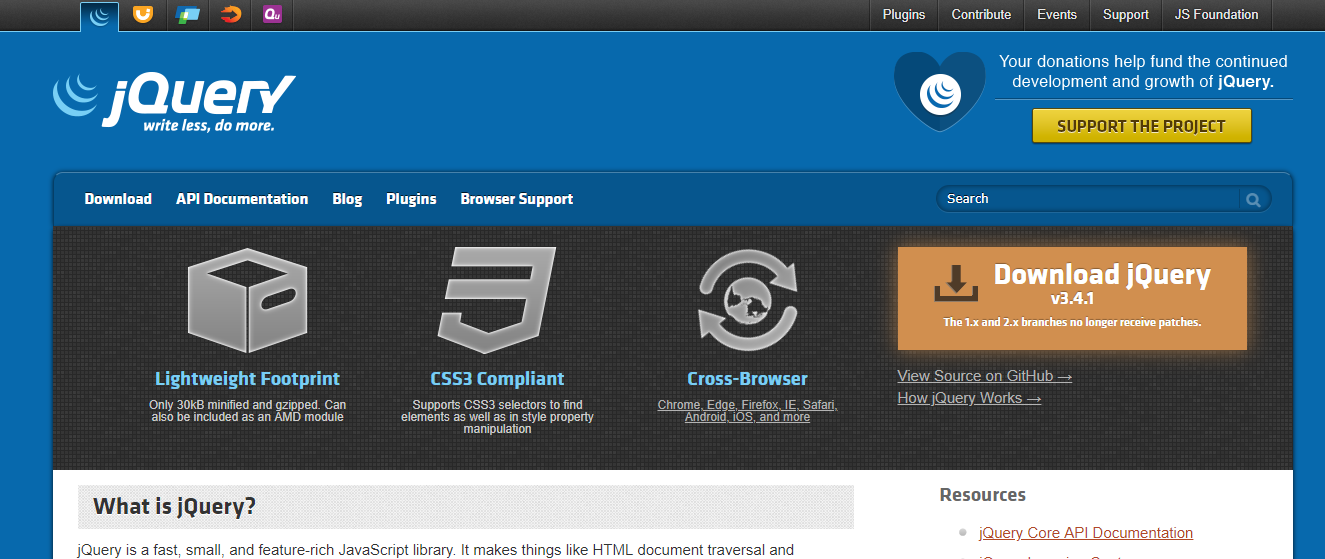
8.1 jQuery概述

8.2 jQuery选择器

8.3 jQuery设置元素及内容

## 8.1 jQuery概述

jQuery是一个快速、简洁的JavaScript框架，是继Prototype之后又一个优秀的JavaScript代码库（或JavaScript框架）。jQuery设计的宗旨是“write Less，Do More”，即倡导写更少的代码，做更多的事情。它封装JavaScript常用的功能代码，提供一种简便的JavaScript设计模式，优化HTML文档操作、事件处理、动画设计和Ajax交互。



jQuery的核心特性可以总结为：具有独特的链式语法和短小清晰的多功能接口；具有高效灵活的css选择器，并且可对CSS选择器进行扩展；拥有便捷的插件扩展机制和丰富的插件。jQuery兼容各种主流浏览器，如IE 6.0+、FF 1.5+、Safari 2.0+、Opera 9.0+等。

jQuery 作为 JavaScript 封装的库，他的目的就是为了简化开发者使用 JavaScript。主要

功能有以下几点：

1.像 CSS 那样访问和操作 DOM

2.修改 CSS 控制页面外观

3.简化 JavaScript 代码操作

4.事件处理更加容易

5.各种动画效果使用方便

6.让 Ajax 技术更加完美

7.基于 jQuery 大量插件

8.自行扩展功能插件

jQuery 最大的优势，就是特别的方便。比如模仿 CSS 获取 DOM，比原生的 JavaScript要方便太多。并且在多个 CSS 设置上的集中处理非常舒服，而最常用的 CSS 功能又封装到单独的方法，感觉非常有心。最重要的是 jQuery 的代码兼容性非常好，你不需要总是头疼着考虑不同浏览器的兼容问题。

目前除了 jQuery，还有 5 个库较为流行，他们分别是 YUI、Prototype、Mootools、Dojo

和 ExtJS。

YUI，是雅虎公司开发的一套完备的、扩展性良好的富交互网页工具集。

Prototype，是最早成型的 JavaScript 库之一，对 JavaScript 内置对象做了大量的扩展。

Dojo，Dojo 强大之处在于提供了其他库没有的功能。离线存储、图标组件等等。

Mootools，轻量、简洁、模块化和面向对象的 JavaScript 框架。

ExtJS，简称 Ext，原本是对 YUI 的一个扩展，主要创建前端用户界面。(付费的)

//单击按钮弹窗

<script type="text/javascript" src="https://code.jquery.com/jquery-3.4.1.min.js"></script>  
……  
<div>点击我</div>  
<script>  
 $(function () {  
 $('div').click(function () {  
 *alert*('这是第一个Jquery程序')  
 });  
  
 })  
</script>

## 8.2 jQuery选择器

在使用 jQuery 选择器时，我们首先必须使用“$()”函数来包装我们的 CSS 规则。而CSS 规则作为参数传递到 jQuery 对象内部后，再返回包含页面中对应元素的 jQuery 对象。



随后，我们就可以对这个获取到的 DOM 节点进行行为操作了。

#box { //使用 ID 选择器的 CSS 规则

color:red; //将 ID 为 box 的元素字体颜色变红

}

在 jQuery 选择器里，我们使用如下的方式获取同样的结果：

**<div id="box">点击我</div>  
<script>  
 $('#box').css('color','red');  
</script>**

//获取 DOM 节点对象，并添加行为

那么除了 ID 选择器之外，还有两种基本的选择器，分别为：元素标签名和类(class)：

|  |  |  |  |
| --- | --- | --- | --- |
| 选择器 | CSS 模式 | jQuery 模式 | 描述 |
| 元素名 | div {} | $('div') | 获取所有 div 元素的 DOM 对象 |
| ID | #box {} | $('#box') | 获取一个 ID 为 box 元素的 DOM 对象 |
| 类 | . box{} | $('.box') | 获取所有class为box的所有DOM对象 |

$('div').css('color', 'red'); //元素选择器，返回多个元素

$('#box').css('color', 'red'); //ID 选择器，返回单个元素

$('.box').css('color', 'red'); //类(class)选择器，返回多个元素

为了证明 ID 返回的是单个元素，而元素标签名和类(class)返回的是多个，我们可以采用 jQuery 核心自带的一个属性 length 或 size()方法来查看返回的元素个数。

alert($('div').size()); //3 个

alert($('#box').size()); //1 个，后面两个失明了

alert($('.box').size()); //3 个

同理，你也可以直接使用 jQuery 核心属性来操作：

alert($('#box').length); //1 个，后面失明了

警告：有个问题特别要注意，ID 在页面只允许出现一次，我们一般都是要求开发者要遵守和保持这个规则。但如果你在页面中出现三次，并且在 CSS 使用样式，那么这三个元素还会执行效果。但如果，你想在 jQuery 这么去做，那么就会遇到失明的问题。所以，开发者必须养成良好的遵守习惯，在一个页面仅使用一个 ID。

$('#box').css('color', 'red'); //只有第一个 ID 变红，后面两个失明

jQuery 选择器的写法与 CSS 选择器十分类似，只不过他们的功能不同。CSS 找到元素后添加的是单一的样式，而 jQuery 则添加的是动作行为。最重要的一点是：CSS 在添加样式的时候，高级选择器会对部分浏览器不兼容，而 jQuery 选择器在添加 CSS 样式的时候却不必为此烦恼。

**<style>  
 #box > p {  
 color: red;  
 }  
</style>**

**$('#box p').css('color','black'); //jQuery 子选择器，兼容了 IE6**

jQuery 选择器支持 CSS1、CSS2 的全部规则，支持 CSS3 部分实用的规则，同时它还有少量独有的规则。所以，对于已经掌握 CSS 的开发人员，学习 jQuery 选择器几乎是零成本。而jQuery 选择器在获取节点对象的时候不但简单，还内置了容错功能，这样避免像 JavaScript那样每次对节点的获取需要进行有效判断。

$('#pox').css('color', 'red'); //不存在 ID 为 pox 的元素，也不报错

document.getElementById('pox').style.color = 'red'; //报错了

因为 jQuery 内部进行了判断，而原生的 DOM 节点获取方法并没有进行判断，所以导致了一个错误，原生方法可以这么判断解决这个问题：

if (document.getElementById('pox')) { //先判断是否存在这个对象

document.getElementById('pox').style.color = 'red';

}

那么对于缺失不存在的元素，我们使用 jQuery 调用的话，怎么去判断是否存在呢？因为本身返回的是 jQuery 对象，可能会导致不存在元素存在与否，都会返回 true。

if ($('#pox').length > 0) { //判断元素包含数量即可

$('#pox').css('color', 'red');

}

除了这种方式之外，还可以用转换为 DOM 对象的方式来判断，例如：

if ($('#pox').get(0)) {} 或 if ($('#pox')[0]) {} //通过数组下标也可以获取 DOM 对象

我们在之前的代码一直在使用$(function () {});这段代码进行首尾包裹，那么为什么必须要包裹这段代码呢？

原因是我们jQuery库文件是在body元素之前加载的，我们必须等待所有的DOM元素加载后，延迟支持DOM操作，否则就无法获取到。在延迟等待加载，JavaScript提供了一个事件为load，方法如下：

window.onload = function () {}; //JavaScript等待加载

$(document).ready(function () {}); //jQuery等待加载

|  |  |  |
| --- | --- | --- |
| load和ready区别 | window.onload | $(document).ready() |
| 执行时机 | 必须等待网页全部加载完毕（包括图片等），然后再执行包裹代码 | 只需要等待网页中的DOM结构  加载完毕，就能执行包裹的代码 |
| 执行次数 | 只能执行一次，如果第二次，那么第一次的执行会被覆盖 | 可以执行多次，第N次都不会被上一次覆盖 |
| 简写方案 | 无 | $(function () {}); |

在实际应用中，我们都很少直接去使用window.onload，因为他需要等待图片之类的大型元素加载完毕后才能执行JS代码。所以，最头疼的就是网速较慢的情况下，页面已经全面展开，图片还在缓慢加载，这时页面上任何的JS交互功能全部处在假死状态。并且只能执行单次在多次开发和团队开发中会带来困难。

## 8.3 jQuery设置元素及内容

在 jQuery 中，已经将最常用的 DOM 操作方法进行了有效封装，并且不需要考虑浏览器的兼容性。



1.D 表示的是页面文档 Document、O 表示对象，即一组含有独立特性的数据集合、M表示模型，即页面上的元素节点和文本节点。

2.DOM 有三种形式，标准 DOM、HTML DOM、CSS DOM，大部分都进行了一系列的封装，在 jQuery 中并不需要深刻理解它。

3.树形结构用来表示 DOM，就非常的贴切，大部分操作都是元素节点操作，还有少部分是文本节点操作。

我们通过前面所学习的各种选择器、过滤器来得到我们想要操作的元素。这个时候，我们就可以对这些元素进行 DOM 的操作。那么，最常用的操作就是对元素内容的获取和修改。

|  |  |
| --- | --- |
| 方法名 | 描述 |
| html() | 获取元素中 HTML 内容 |
| html(value) | 设置元素中 HTML 内容 |
| text() | 获取元素中文本内容 |
| text(value) | 设置原生中文本内容 |
| val() | 获取表单中的文本内容 |
| val(value) | 设置表单中的文本内容 |

在常规的 DOM 元素中，我们可以使用 html()和 text()方法获取内部的数据。html()方法可以获取或设置 html 内容，text()可以获取或设置文本内容。

$('#box').html(); //获取 html 内容

$('#box').text(); //获取文本内容，会自动清理 html 标签

$('#box').html('<em>www.li.cc</em>'); //设置 html 内容

$('#box').text('<em>www.li.cc</em>'); //设置文本内容，会自动转义 html 标签

注意：当我们使用 html()或 text()设置元素里的内容时，会清空原来的数据。而我们期望能够追加数据的话，需要先获取原本的数据。

$('#box').html($('#box').html() + '<em>www.li.cc</em>'); //追加数据如果元素是表单的话，jQuery 提供了 val()方法进行获取或设置内部的文本数据。

$('input').val(); //获取表单内容

$('input').val('www.li.cc'); //设置表单内容

如果想设置多个选项的选定状态，比如下拉列表、单选复选框等等，可以通过数组传递操作。

$("input").val(["check1","check2", "radio1" ]); //value 值是这些的将被选定除了对元素内容进行设置和获取，通过 jQuery 也可以对元素本身的属性进行操作，包括获取属性的属性值、设置属性的属性值，并且可以删除掉属性。

attr()和 removeAttr()

|  |  |
| --- | --- |
| 方法名 | 描述 |
| attr(key) | 获取某个元素 key 属性的属性值 |
| attr(key, value) | 设置某个元素 key 属性的属性值 |
| attr({key1:value2, key2:value2...}) | 设置某个元素多个 key 属性的属性值 |
| attr(key, function (index, value) {}) | 设置某个元素 key 通过 fn 来设置 |

$('div').attr('title'); //获取属性的属性值

$('div').attr('title', '我是域名'); //设置属性及属性值

$('div').attr('title', function () { //通过匿名函数返回属性值

return '我是域名';

});

**var box = $('div').attr('title',function (index,value) {  
 return value + (index +1)+',我是域名'  
});  
*console*.log(box);**

index-- 表示调用attr对象集合中的索引值，value--该对象原来的值。删除指定的属性，这个方法就不可以使用匿名函数，传递 index 和 value 均无效。

$('div').removeAttr('title'); //删除指定的属性

元素样式操作包括了直接设置 CSS 样式、增加 CSS 类别、类别切换、删除类别这几种操作方法。而在整个 jQuery 使用频率上来看，CSS 样式的操作也是极高的，所以需要重点掌握。

|  |  |
| --- | --- |
| 方法名 | 描述 |
| css(name) | 获取某个元素行内的 CSS 样式 |
| css([name1, name2, name3]) | 获取某个元素行内多个 CSS 样式 |
| css(name, value) | 设置某个元素行内的 CSS 样式 |
| css(name, function (index, value) ) | 设置某个元素行内的 CSS 样式 |
| css({name1 : value1, name2 : value2}) | 设置某个元素行内多个 CSS 样式 |
| addClass(class) | 给某个元素添加一个 CSS 类 |
| addClass(class1 class2 class3...) | 给某个元素添加多个 CSS 类 |
| removeClass(class) | 删除某个元素的一个 CSS 类 |
| removeClass(class1 class2 class3...) | 删除某个元素的多个 CSS 类 |
| toggleClass(class) | 来回切换默认样式和指定样式 |
| toggleClass(class1 class2 class3...) | 同上 |
| toggleClass(class, switch) | 来回切换样式的时候设置切换频率 |
| toggleClass(function () {}) | 通过匿名函数设置切换的规则 |
| toggleClass(function () {}, switch) | 在匿名函数设置时也可以设置频率 |
| toggleClass(function (i, c, s) {}, switch) | 在匿名函数设置时传递三个参数 |

$('div').css('color'); //获取元素行内 CSS 样式的颜色

$('div').css('color', 'red'); //设置元素行内 CSS 样式颜色为红色

在 CSS 获取上，我们也可以获取多个 CSS 样式，而获取到的是一个对象数组，如果用

传统方式进行解析需要使用 for in 遍历。

**var box = $('div').css(['color', 'height', 'width']); *//得到多个 CSS 样式的数组对象*for (var i in box) { *//逐个遍历出来  
 alert*(i + ':' + box[i]);  
}**

jQuery 提供了一个遍历工具专门来处理这种对象数组，$.each()方法，这个方法可以轻松的遍历对象数组。

**$.each(box, function (attr, value) { *//遍历 JavaScript 原生态的对象数组  
 alert*(attr + ':' + value);  
});**

使用$.each()可以遍历原生的 JavaScript 对象数组，如果是 jQuery 对象的数组怎么使用.each()方法呢？

**$('div').each(function (index, element) { *//index 为索引，element 为元素 DOM  
 alert*(index + ':' + element);  
});**

在需要设置多个样式的时候，我们可以传递多个 CSS 样式的键值对即可。

$('div').css({

'background-color' : '#ccc',

'color' : 'red',

'font-size' : '20px'

});

如果想设置某个元素的 CSS 样式的值，但这个值需要计算我们可以传递一个匿名函数。

**$('div').css('width', function (index, value) {  
 return (*parseInt*(value) - 500) + 'px';  
});**

除了行内 CSS 设置，我们也可以直接给元素添加 CSS 类，可以添加单个或多个，并且也可以删除它。

**$('div').addClass('red'); *//添加一个 CSS 类* $('div').addClass('red bg'); *//添加多个 CSS 类* $('div').removeClass('bg'); *//删除一个 CSS 类* $('div').removeClass('red bg'); *//删除多个 CSS 类  
 // 我们还可以结合事件来实现 CSS 类的样式切换功能。* $('div').click(function () { *//当点击后触发* $(this).toggleClass('red size'); *//单个样式多个样式均可*});**

.toggleClass()方法的第二个参数可以传入一个布尔值，true表示执行切换到 class类，false表示执行回默认 class 类(默认的是空 class)，运用这个特性，我们可以设置切换的频率。

**var count = 0;  
$('div').click(function () { *//每点击两次切换一次 red* $(this).toggleClass('red', count++ % 3 == 0);  
});**

默认的 CSS 类切换只能是无样式和指定样式之间的切换，如果想实现样式 1 和样式 2之间的切换还必须自己写一些逻辑。

**$('div').click(function () {  
 $(this).toggleClass('red size'); *//一开始切换到样式 2* if ($(this).hasClass('red')) { *//判断样式 2 存在后* $(this).removeClass('blue'); *//删除样式 1* } else {  
 $(this).toggleClass('blue'); *//添加样式 1，这里也可以 addClass* }  
});**

上面的方法较为繁琐，.toggleClass()方法提供了传递匿名函数的方式，来设置你所需要切换的规则。

**$('div').click(function () {  
 $(this).toggleClass(function () { *//传递匿名函数，返回要切换的样式* return $(this).hasClass('red') ? 'blue' : 'red size';  
 });  
});**

注意：上面虽然一句话实现了这个功能，但还是有一些小缺陷，因为原来的 class 类没有被删除，只不过被替代了而已。所以，需要改写一下。

**$('div').click(function () {  
 $(this).toggleClass(function () {  
 if ($(this).hasClass('red')) {  
 $(this).removeClass('red');  
 return 'green';  
 } else {  
 $(this).removeClass('green');  
 return 'red';  
 }  
 });  
});**

我们也可以在传递匿名函数的模式下，增加第二个频率参数。

**var count = 0;  
$('div').click(function () {  
 $(this).toggleClass(function () {  
 return $(this).hasClass('red') ? 'blue' : 'red size';  
 },count++ % 3 == 0); *//增加了频率*});**

对于.toggleClass()传入匿名函数的方法，还可以传递 index 索引、class 类两个参数以及频率布尔值，可以得到当前的索引、class 类名和频率布尔值。

**var count = 0;  
$('div').click(function () {  
 $(this).toggleClass(function (index, className, switchBool) {  
 *alert*(index + ':' + className + ':' + switchBool); *//得到当前值* return $(this).hasClass('red') ? 'blue' : 'red size';  
 },count++ % 3 == 0);  
});**

总结：

8.1 jQuery概述

8.2 jQuery选择器

8.3 jQuery设置元素及内容