第 **3** 章 常规选择器

学习要点：

1

2

3

.简单选择器

.进阶选择器

.高级选择器

jQuery 最核心的组成部分就是：选择器引擎。它继承了 CSS 的语法，可以对 DOM 元

素的标签名、属性名、状态等进行快速准确的选择，并且不必担心浏览器的兼容性。jQuery

选择器实现了 CSS1~CSS3 的大部分规则之外，还实现了一些自定义的选择器，用于各种

特殊状态的选择。

一．简单选择器

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 {

color:red;

//使用 ID 选择器的 CSS 规则

//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：

$('#box').css('color', 'red'); //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

元素名

ID

CSS 模式

div {}

jQuery 模式

$('div')

描述

获取所有 div 元素的 DOM 对象

获取一个 ID 为 box 元素的 DOM 对象

#box {}

$('#box')

类(class)

.box{}

$('.box')

获取所有 class 为 box 的所有 DOM 对象

$

$

$

('div').css('color', 'red');

('#box').css('color', 'red');

('.box').css('color', 'red');

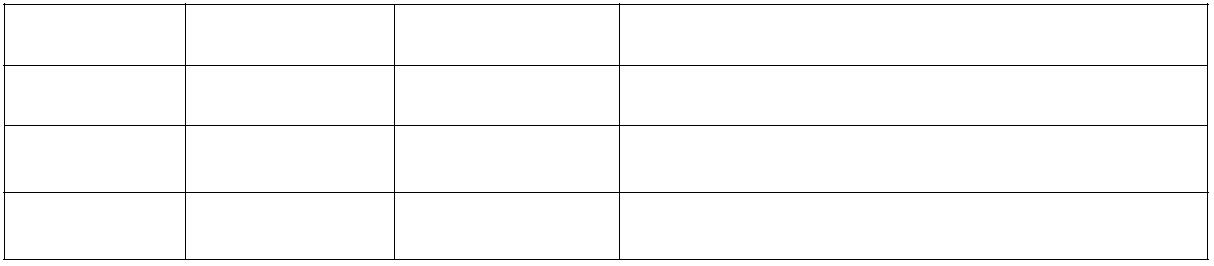
//元素选择器，返回多个元素

//ID 选择器，返回单个元素

//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());

alert($('#box').size());

alert($('.box').size());

//3 个

//1 个，后面两个失明了

//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。

#

}

$

box > p {

color:red;

//CSS 子选择器，IE6 不支持

('#box > p').css('color','red');

//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

//报错了

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

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

二．进阶选择器

在简单选择器中，我们了解了最基本的三种选择器：元素标签名、ID 和类(class)。那么

在基础选择器外，还有一些进阶和高级的选择器方便我们更精准的选择元素。

选择器 CSS 模式 jQuery 模式 描述

群组选择器 span,em,.box {} $('span,em,.box') 获取多个选择器的 DOM 对象

后代选择器

通配选择器

ul li a {}

\* {}

$('ul li a')

$('\*')

获取追溯到的多个 DOM 对象

获取所有元素标签的 DOM 对象

//群组选择器

span, em, .box {

//多种选择器添加红色字体

//群组选择器 jQuery 方式

//层层追溯到的元素添加红色字体

//群组选择器 jQuery 方式

//页面所有元素都添加红色字体

//通配选择器

color:red;

}

$

('span, em, .box').css('color', 'red');

/

/后代选择器

ul li a {

color:red;

}

$

('ul li a').css('color', 'red');

//通配选择器

\*

{

color:red;

}

$

('\*').css('color', 'red');

目前介绍的六种选择器，在实际应用中，我们可以灵活的搭配，使得选择器更加的精准

和快速：

('#box p, ul li \*').css('color', 'red');

$

//组合了多种选择器

警告：在实际使用上，通配选择器一般用的并不多，尤其是在大通配上，比如：$('\*')，

这种使用方法效率很低，影响性能，建议竟可能少用。

还有一种选择器，可以在 ID 和类(class)中指明元素前缀，比如：

$

$

('div.box');

//限定必须是.box 元素获取必须是 div

//同上

('p#box div.side');

类(class)有一个特殊的模式，就是同一个 DOM 节点可以声明多个类(class)。那么对于这

种格式，我们有多 class 选择器可以使用，但要注意和 class 群组选择器的区别。

.

box.pox {

color:red;

//双 class 选择器，IE6 出现异常

