第 3 章 常规选择器

学习要点：

1.简单选择器

2.进阶选择器

3.高级选择器

主讲教师：李炎恢
官方网站：http://www.ycku.com
合作网站：http://www.ibeifeng.com

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。备注：课程必须有(X)html+CSS 基础。

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 {                                  //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：

```
$('#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	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(class)	.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。

```
#box > p {                                 //CSS 子选择器，IE6 不支持
    color:red;
}

$('#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
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
```

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	ul li a {}	$('ul li a')	获取追溯到的多个 DOM 对象
通配选择器	* {}	$('*')	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
    color:red;
}
$('span, em, .box').css('color', 'red');        //群组选择器 jQuery 方式

//后代选择器
ul li a {                               //层层追溯到的元素添加红色字体
    color:red;
}
$('ul li a').css('color', 'red');               //群组选择器 jQuery 方式

//通配选择器
* {                                     //页面所有元素都添加红色字体
    color:red;
}
$('*').css('color', 'red');                     //通配选择器
```

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
$('p#box div.side');                    //同上
```

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 {                              //双 class 选择器，IE6 出现异常
    color:red;
```

```
}
$('.box.pox').css('color', 'red');                    //兼容 IE6，解决了异常
```

多 class 选择器是必须一个 DOM 节点同时有多个 class，用这多个 class 进行精确限定。而群组 class 选择器，只不过是多个 class 进行选择而已。

```
$('.box, .pox').css('color', 'red');                  //加了逗号，体会区别
```

警告：在构造选择器时，有一个通用的优化原则：只追求必要的确定性。当选择器筛选越复杂，jQuery 内部的选择器引擎处理字符串的时间就越长。比如：

```
$('div#box ul li a#link');                            //让 jQuery 内部处理了不必要的字符串
$('#link');                                           //ID 是唯一性的，准确度不变，性能提升
```

三．高级选择器

在前面我们学习六种最常规的选择器，一般来说通过这六种选择器基本上可以解决所有 DOM 节点对象选择的问题。但在很多特殊的元素上，比如父子关系的元素，兄弟关系的元素，特殊属性的元素等等。在早期 CSS 的使用上，由于 IE6 等低版本浏览器不支持，所以这些高级选择器的使用也不具备普遍性，但随着 jQuery 兼容，这些选择器的使用频率也越来越高。

层次选择器

选择器	CSS 模式	jQuery 模式	描述
后代选择器	ul li a {}	$('ul li a')	获取追溯到的多个 DOM 对象
子选择器	div > p {}	$('div p')	只获取子类节点的多个 DOM 对象
next 选择器	div + p {}	$('div + p')	只获取某节点后一个同级 DOM 对象
nextAll 选择器	div ~ p {}	$('div ~ p')	获取某节点后面所有同级 DOM 对象

在层次选择器中，除了后代选择器之外，其他三种高级选择器是不支持 IE6 的，而 jQuery 却是兼容 IE6 的。

```
//后代选择器
$('#box p').css('color', 'red');                      //全兼容
```

jQuery 为后代选择器提供了一个等价 find()方法

```
$('#box').find('p').css('color', 'red');              //和后代选择器等价

//子选择器，孙子后失明
#box > p {                                            //IE6 不支持
    color:red;
}
$('#box > p').css('color', 'red');                    //兼容 IE6
```

jQuery 为子选择器提供了一个等价 children()方法：

```
$('#box').children('p').css('color', 'red');          //和子选择器等价
```

```
//next 选择器(下一个同级节点)
#box + p {                                  //IE6 不支持
    color:red;
}
$('#box+p').css('color', 'red');            //兼容 IE6
```

jQuery 为 next 选择器提供了一个等价的方法 next()：

```
$('#box').next('p').css('color', 'red');    //和 next 选择器等价
```

```
//nextAll 选择器(后面所有同级节点)
#box ~ p {                                  //IE6 不支持
    color:red;
}
$('#box ~ p').css('color', 'red');          //兼容 IE6
```

jQuery 为 nextAll 选择器提供了一个等价的方法 nextAll()：

```
$('#box').nextAll('p').css('color', 'red');  //和 nextAll 选择器等价
```

层次选择器对节点的层次都是有要求的，比如子选择器，只有子节点才可以被选择到，孙子节点和重孙子节点都无法选择到。next 和 nextAll 选择器，必须是同一个层次的后一个和后 N 个，不在同一个层次就无法选取到了。

在 find()、next()、nextAll()和 children()这四个方法中，如果不传递参数，就相当于传递了“*”，即任何节点，我们不建议这么做，不但影响性能，而且由于精准度不佳可能在复杂的 HTML 结构时产生怪异的结果。

```
$('#box').next();                           //相当于$('#box').next('*');
```

为了补充高级选择器的这三种模式，jQuery 还提供了更加丰富的方法来选择元素：

```
$('#box').prev('p').css('color', 'red');    //同级上一个元素
$('#box').prevAll('p').css('color', 'red');  //同级所有上面的元素
```

nextUntil()和 prevUnitl()方法是选定同级的下面或上面的所有节点，选定非指定的所有元素，一旦遇到指定的元素就停止选定。

```
$('#box').prevUntil('p').css('color', 'red');  //同级上非指定元素选定，遇到则停止
$('#box').nextUntil('p').css('color', 'red');  //同级下非指定元素选定，遇到则停止
```

siblings()方法正好集成了 prevAll()和 nextAll()两个功能的效果，及上下相邻的所有元素进行选定：

```
$('#box').siblings('p').css('color', 'red');  //同级上下所有元素选定
//等价于下面：
$('#box').prevAll('p').css('color', 'red');   //同级上所有元素选定
$('#box').nextAll('p').css('color', 'red');   //同级下所有元素选定
```

警告：切不可写成“$('#box').prevAll('p').nextAll('p').css('color', 'red');”这种形式，因为prevAll('p')返回的是已经上方所有指定元素，然后再 nextAll('p')选定下方所有指定元素，这样必然出现错误。

理论上来讲，jQuery 提供的方法 find()、next()、nextAll()和 children()运行速度要快于使用高级选择器。因为他们实现的算法有所不同，高级选择器是通过解析字符串来获取节点对象，而 jQuery 提供的方法一般都是单个选择器，是可以直接获取的。但这种快慢的差异，对于客户端脚本来说没有太大的实用性，并且速度的差异还要取决了浏览器和选择的元素内容。比如，在 IE6/7 不支持 querySelectorAll()方法，则会使用“Sizzle”引擎，速度就会慢，而其他浏览器则会很快。有兴趣的可以了解这个方法和这个引擎。

选择器快慢分析：

```
//这条最快，会使用原生的 getElementById、ByName、ByTagName 和 querySelectorAll()
$('#box').find('p');

//jQuery 会自动把这条语句转成$('#box').find('p')，这会导致一定的性能损失。它比最快的形式慢了 5%-10%
$('p', '#box');

//这条语句在 jQuery 内部，会使用$.sibling()和 javascript 的 nextSibling()方法，一个个遍历节点。它比最快的形式大约慢 50%
$('#box').children('p');

//jQuery 内部使用 Sizzle 引擎，处理各种选择器。Sizzle 引擎的选择顺序是从右到左，所以这条语句是先选 p，然后再一个个过滤出父元素#box，这导致它比最快的形式大约慢 70%
$('#box > p');

//这条语句与上一条是同样的情况。但是，上一条只选择直接的子元素，这一条可以于选择多级子元素，所以它的速度更慢，大概比最快的形式慢了 77%。
$('#box p');

//jQuery 内部会将这条语句转成$('#box').find('p')，比最快的形式慢了 23%。
$('p', $('#parent'));
```

综上所属，最快的是 find()方法，最慢的是$('#box p')这种高级选择器。如果一开始将$('#box')进行赋值，那么 jQuery 就对其变量进行缓存，那么速度会进一步提高。

```
var box = $('#box');
var p = box.find('p');
```

注意：我们应该推荐使用哪种方案呢？其实，使用哪种都差不多。这里，我们推荐使用 jQuery 提供的方法。因为不但方法的速度比高级选择器运行的更快，并且它的灵活性和扩展性要高于高级选择器。使用“+”或“~”从字面上没有 next 和 nextAll 更加语义化，更加清晰，jQuery 的方法更加丰富，提供了相对的 prev 和 prevAll。毕竟 jQuery 是编程语言，需要

能够灵活的拆分和组合选择器，而使用 CSS 模式过于死板。所以，如果 jQuery 提供了独立的方法来代替某些选择器的功能，我们还是推荐优先使用独立的方法。

属性选择器

CSS 模式	jQuery 模式	描述
a[title]	$('a[title]')	获取具有这个属性的 DOM 对象
a[title=num1]	$('a[title=num1]')	获取具有这个属性=这个属性值的 DOM 对象
a[title^=num]	$('a[title^=num]')	获取具有这个属性且开头属性值匹配的 DOM 对象
a[title|=num]	$('a[title|=num]')	获取具有这个属性且等于属性值或开头属性值匹配后面跟一个“-”号的 DOM 对象
a[title$=num]	$('a[title$=num]')	获取具有这个属性且结尾属性值匹配的 DOM 对象
a[title!=num]	$('a[title!=num]')	获取具有这个属性且不等于属性值的 DOM 对象
a[title~=num]	$('a[title~=num]')	获取具有这个属性且属性值是以一个空格分割的列表，其中包含属性值的 DOM 对象
a[title*=num]	$('a[title*=num]')	获取具有这个属性且属性值含有一个指定字串的 DOM 对象
a[bbb][title=num1]	$('a[bbb][title=num1]')	获取具有这个属性且属性值匹配的 DOM 对象

属性选择器也不支持 IE6，所以在 CSS 界如果要兼容低版本，那么也是非主流。但 jQuery 却不必考虑这个问题。

```
//选定这个属性的
a[title] {                                    //IE6 不支持
    color:red;
}
$('a[title]').css('color', 'red');            //兼容 IE6 了

//选定具有这个属性=这个属性值的
a[title=num1] {                               //IE6 不支持
    color:red;
}
$('a[title=num1]').css('color', 'red');       //兼容 IE6 了

//选定具有这个属性且开头属性值匹配的
a[title^=num] {                               //IE6 不支持
    color:red;
}
$('a[title=^num]').css('color', 'red');       //兼容 IE6 了
```

```
//选定具有这个属性且等于属性值或开头属性值匹配后面跟一个“-”号
a[title|=num] {                                    //IE6 不支持
      color:red;
}
$('a[title|="num"]').css('color', 'red');          //兼容 IE6 了

//选定具有这个属性且结尾属性值匹配的
a[title$=num] {                                    //IE6 不支持
      color:red;
}
$('a[title$=num]').css('color','red');             //兼容 IE6 了

//选定具有这个属性且属性值不想等的
a[title!=num1] {                                   //不支持此 CSS 选择器
      color:red;
}
$('a[title!=num1]').css('color','red');            //jQuery 支持这种写法

//选定具有这个属性且属性值是以一个空格分割的列表，其中包含属性值的
a[title~=num] {                                    //IE6 不支持
      color:red;
}
$('a[title~=num1]').css('color','red');            //兼容 IE6

//选定具有这个属性且属性值含有一个指定字串的
a[title*=num] {                                    //IE6 不支持
      color:red;
}
$('a[title*=num]').css('color','red');             //兼容 IE6

//选定具有多个属性且属性值匹配成功的
a[bbb][title=num1] {                               //IE6 不支持
      color:red;
}
$('a[bbb][title=num1]').css('color','red');        //兼容 IE6
```

北风网 IBEIFENG.COM

Ycku.com 瓢城Web俱乐部，做最酷的教学

感谢收看本次教程！

本课程是由北风网(ibeifeng.com)

瓢城 Web 俱乐部(ycku.com)联合提供：

本次主讲老师：李炎恢

谢谢大家，再见！