《JavaScript 与 jQuery 网页前端开发与设计-第 2 版》教案
第 6 章 jQuery 入门

一、教学目标：

1. 了解 jQuery 的下载与使用；
2. 掌握 jQuery 的基础语法结构；
3. 掌握 jQuery 文档就绪函数的用法；
4. 掌握 jQuery 名称冲突的解决方案。

二、教学重点和难点：

重点：jQuery 的基础语法结构；

难点：jQuery 名称冲突的解决方案。

三、教学方法与手段：

采取互动式教学方法，理论教学使用多媒体投影教室。

四、课程简介：

本章主要介绍 jQuery 基础知识入门，包括 jQuery 的下载和使用、jQuery 基础语法结构、jQuery 文档就绪函数以及 jQuery 名称冲突的解决方案。

五、教学基本内容：

6.1 jQuery 下载和使用

6.1.1 jQuery 的下载

jQuery 是一种开源函数库，读者可以直接访问官网页面（http://jquery.com/download/）进行下载。目前常用的 jQuery 分为 1.x、2.x 和 3.x 版本，本书将选择官方推荐的 1.12.x 系列版本作为示例，因为该版本的浏览器兼容性相对较好。

以 1.12.3 版本为例，下载完整版点击“Download the uncompressed development jQuery 1.12.3”，下载压缩版点击“Download the compressed production jQuery 1.12.3”。完整版的文件后缀名为.js，常用于开发和调试；压缩版的文件后缀名为.min.js，里面保留了所有的 jQuery 函数并提升了产品性能，适用于正式发布。

注：由于官方不定期会更新可供下载的页面 jQuery 版本，可能实际访问的时候已经无法在官网的下载页面下载 1.x 版的 jQuery 文件了，虽然官方也另外提供了一个历年 jQuery 版本下载地址 https://github.com/DanielRuf/snyk-js-jquery-565129，但是由于服务器在海外有时打开非常慢。读者也可以直接使用本书配套提供的源码包，从第 6 章开始后续每章节例题源代码包中的 js 目录下均包含了 jquery-1.12.3.js（未压缩包，可查看源代码，适合开发学习过程）和 jquery-1.12.3.min.js（混淆压缩包，更加精简加载效率高，适合正式环境）供读者使用。

6.1.2 jQuery 的使用

和其他 JavaScript 文件的使用方式一样，可以通过<script>标签在 HTML 文档的首部标签<head>和</head>中添加 jQuery 的引用声明。语法如下：

```
<script src=" jQuery 文件 URL"></script>
```

上述代码中的 jQuery 文件 URL 需要替换为实际的 jQuery 文件引用地址。

需要注意的是，HTML4.01 版<script>元素首标签需要写成<script type="text/javascript"

src="jQuery 文件 URL">；而在 HTML5 中可以省略其中的 type="text/javascript"，直接写成<script src="jQuery 文件 URL">即可。

以 jquery1.12.3.js 为例，将该文件放置在和网页同一个文件夹下，则使用声明写法如下：

```
<script src="jquery1.12.3.js"></script>
```

上述代码声明完成后就可以在页面上添加 jQuery 相关语句了。

注：引用的 jQuery 文件名是可以下载后由开发者重新自定义的，例如上述代码中的文件名如若改成了 jquery.js，那么引用时也需要同步更新为<script src="jquery.js"></script>即可。

6.2 jQuery 语法

jQuery 的语法是专门为 HTML 元素的选取编制的，可以对元素执行操作。

6.2.1 基础语法结构

jQuery 的基础语法结构如下：

```
$(selector).action()
```

其中美元符号$表示 jQuery 语句，选择符 selector 用于查询 HTML 元素，action()需要替换为对元素某种具体操作的方法名。例如：

```
$("p").hide();
```

在 HTML 中<p>表示段落标签，hide()为 jQuery 中的新方法用于隐藏元素。因此上述代码表示隐藏所有段落。

6.2.2 文档就绪函数

为了避免文档在加载完成前就运行了 jQuery 代码导致潜在的错误，所有的 jQuery 函数都需要写在一个文档就绪（document ready）函数中。例如当前 HTML 页面还没有加载完，因此某 HTML 元素标签可能还无法查询获取。

文档就绪函数的写法如下：

```
$(document).ready(function(){
   jQuery 函数内容
});
```

6.2.3 jQuery 名称冲突

jQuery 通常使用美元符号$作为简介方式，在同时使用了多个 JavaScript 函数库的 HTML 文档中 jQuery 就有可能与其他同样使用$符号的函数（例如 Prototype）冲突。因此 jQuery 使用 noConflict()方法自定义其他名称来替换可能产生冲突的$符号表达方式。

【例 6-1】jQuery 自定义名称代替$符号

使用 noConflict()方法创建自定义名称 jq 代替$符号。

【代码说明】

本示例使用别名 jq 代替$符号，并在文档就绪函数中为按钮 button 绑定了一个 click 点击事件。如果点击按钮后 alert 语句仍然能被执行，说明 jQuery 的自定义别名有效。由图可见，当前 jQuery 的别名可以正常使用。

6.3 本章小结

本章主要是 jQuery 的基础知识入门，首先介绍了 jQuery 文件如何下载和使用，其次介绍了 jQuery 的常用语法，包括基础语法结构、文档就绪函数以及 jQuery 别名的使用。

六、课后习题

1.如何在文档中引用 jQuery？

答：以 jquery-1.12.3.min.js 为例，代码如下。

```
<script src="js/jquery-1.12.3.min.js"></script>
```

2.请使用文档就绪函数弹出 alert 提示框，并显示"Hello jQuery!"。

答：

```
$(document).ready(function() {
        alert ("Hello jQuery!");
});
```

3.如何使用别名 demo 代替可能产生冲突的$符号。

答：

```
var demo = jQuery.noConflict();
```

《JavaScript 与 jQuery 网页前端开发与设计-第 2 版》教案
第 7 章 jQuery 选择器与过滤器

一、教学目标：

5. 理解 jQuery 选择器和过滤器的作用；
6. 掌握 jQuery 选择器的常见用法；
7. 掌握 jQuery 过滤器的常见用法。

二、教学重点和难点：

重点：jQuery 选择器的用法；
难点：jQuery 过滤器的用法。

三、教学方法与手段：

采取互动式教学方法，理论教学使用多媒体投影教室。

四、课程简介：

本章主要介绍 jQuery 选择器与 jQuery 过滤器的相关知识。jQuery 选择器包括基础选择器、属性选择器、表单选择器、层次选择器以及 CSS 选择器；jQuery 过滤器包括基础过滤器、子元素过滤器、内容过滤器和可见性过滤器。

五、教学基本内容：

7.1 jQuery 选择器

jQuery 选择器可用于快速选定需要的 HTML 元素，并为其进行后续处理。jQuery 选择器的部分语法规则来自于 CSS 选择器，加上其他功能模块形成了 jQuery 特有的选择匹配元素工具，简化了用户使用 JavaScript 选择和操作元素的复杂度。

7.1.1 基础选择器(Basic Selector)

jQuery 基础选择器(Basic Selector)的语法规则基本和 CSS 选择器相同，可以通过指定 HTML 元素的标签名称、类名称或 ID 名称对元素进行筛选定位。
常见用法如表 7-1 所示。

表 7-1 jQuery 基础选择器常见用法示例

选择器	描述	用法示例	示例描述
*	用于选择所有元素	$("*")	选择文档中的所有元素
element	元素选择器，用于选择指定标签名称的元素。	$("p")	选择文档中所有的段落标签<p>元素
#id	ID 选择器，用于选择指定 id 的元素。	$("#test")	选择文档中 id="test"的元素
.class	类选择器，用于选择所有具有同一个指定 class 的元素。	$(".style01")	选择文档中所有 class="style01"的元素
selector1, selector2, ... selectorN	多重选择器，用于选择符合条件的所有结果。	$("p, h1, div")	选择文档中所有段落元素<p>、标题元素<h1>和块元素

1. 全局选择器

全局选择器用于选择文档中所有的元素。其语法结构如下：

```
$("*")
```

全局选择器会遍历文档中所有的元素标签，甚至包括首部标签<head>及其内部的<meta>、<script>等，运行速度较慢。

【例 7-1】jQuery 全局选择器的使用

【代码说明】

本示例使用了 jQuery 全局选择器将所有的 HTML 元素都设置为带有 5 像素宽的红色实线边框样式。由图可见，页面中的所有内容都显示了红色边框。由于全局选择器也包括了<body>、<head>等通常不在页面具体显示的元素，因此页面中的红色边框个数大于可见元素个数。

2. 元素选择器

元素选择器用于选择所有指定标签名称的元素。其语法结构如下：

```
$("element")
```

这里的 element 在使用时需要换成真正的元素标签名称。例如，$("h1")表示选中所有<h1>标题元素。使用元素选择器时，jQuery 会调用 JavaScript 中的原生方法 getElementsByTagName()来获取指定的元素。该方法化简了原先 JavaScript 的代码量。

【例 7-2】jQuery 元素选择器的使用

【代码说明】

本示例包含了区域元素 与段落元素<p>各两个，并在 CSS 内部样式表中为其设置统一样式：宽和高均为 100 像素，向左浮动，各边内外边距均为 10 像素，带有 1 像素宽的灰色实线边框。

使用 jQuery 元素选择器选择所有的段落元素<p>并将其设置为带有 5 像素宽的红色实线边框样式。由图可见，只有<p>元素的样式被更改， 元素没有受到任何影响。

3. ID 选择器

ID 选择器用于选择指定 ID 名称的单个元素。其语法结构如下：

```
$("#ID")
```

这里的 ID 在使用时需要换成元素真正的 ID 名称。例如，$("#test")表示选中 id="test"的元素。使用 ID 选择器时，jQuery 会调用 JavaScript 中的原生方法 getElementById()来获取指定 ID 名称的元素。

ID 选择器也可以和元素选择器配合使用，例如：

```
$("p#test01")
```

表示选择 id="test01"的段落元素<p>。

【例 7-3】jQuery ID 选择器的使用

4. 类选择器

类选择器用于筛选出具有同一个 class 属性值的所有元素。其语法结构如下：

```
$(".class")
```

这里的 class 在使用时需要换成真正的类名称。例如，$(".box")表示选择所有 class="box"的元素。如果一个元素包含了多个类，只要其中任意一个类符合条件即可被选中。使用类选择器时，jQuery 会调用 JavaScript 中的原生方法 getElementsByClassName()来获取指定的元素。

类选择器也可以和元素选择器配合使用，例如：

```
$("p.style01")
```

表示选择所有具有 class="style01"的段落元素<p>。

【例 7-4】jQuery 类选择器的使用

【代码说明】

本示例包含了两个区域元素 和一个段落元素<p>，为测试类选择器的效果，将其中一个 和<p>元素设置了同样的 class 属性即 class="style01"。另一个 元素设置为 class="style02"作为参照对比。在 CSS 内部样式表中为所有 和<p>元素设置统一样式：宽 180 像素、高 100 像素，向左浮动，各边内外边距均为 10 像素，带有 1 像素宽的灰色实线边框。

使用 jQuery 类选择器选择 class="style01"的所有元素，并将其边框样式修改为 5 像素宽的红色实线。由图可见，类选择器已经生效。如果将本示例中的类选择器修改为$("p.style01")则运行结果是只会修改段落元素<p>的边框样式。

5. 多重选择器

多重选择器适用于需要批量处理的多种元素，可以将不同的筛选条件用逗号隔开写入同一个选择器中。其语法结构如下：

```
$("selector1 [, selector2] ..... [, selectorN]")
```

其中 selector1-selectorN 需要全部换成具体的 jQuery 选择器，数量可自定义。这里的选择器可以是元素选择器、ID 选择器或类选择器的任意一种或组合使用，只要满足其中任意一个条件的元素即可被选中。例如：

```
$("p, div.style01, #news")
```

上述代码表示选中所有的段落元素<p>、class="style01"的 元素以及 id="news"的元素。

【例 7-5】jQuery 多重选择器的使用

【代码说明】

本示例中的多重选择器$("h3, p, div.style01")表示选择所有的<h3>标签、<p>标签以及 class="style01"的 标签。由图可见，页面中的标题<h3>、段落元素<p>以及 class="style01"的 元素被修改了边框样式，证明多重选择器已生效。

7.1.2 属性选择器(Attribute Selector)

属性选择器可以用于选择具有指定属性要求的元素。jQuery 使用路径表达式（XPath）在 HTML 文档中进行导航，从而选择指定属性的元素。jQuery 属性选择器的常见用法如表 7-2 所示。

表 7-2 jQuery 属性选择器常见用法示例

选择器	描述	用法示例	示例描述

[attribute]	带有指定属性的元素	$("[alt]")	所有带有 alt 属性的元素
[attribute=value]	属性等于指定值的元素	$("[href='#']")	所有 href 属性值等于"#"的元素
[attribute!=value]	属性不等于指定值的元素	$("[href!= '#']")	所有 href 属性值不等于"#"的元素
[attribute$=value]	属性以指定值结尾的元素	$("[src$='.png']")	所有 src 属性值以".png"结尾的元素

属性选择器也可以和其他选择器配合使用，能缩小匹配范围。例如：

```
$("img[src$='.png']")
```

上述代码表示找出页面中所有 src 属性值以".png"结尾的图像元素<img>。

【例 7-6】jQuery 属性选择器的使用

7.1.3 表单选择器(Form Selector)

jQuery 表单选择器(Form Selector)可用于选择指定类型或处于指定状态的表单元素。常见用法示例如表 7-3 所示。

表 7-3 jQuery 表单选择器常见用法示例

指定类型的表单元素		
选择器	**描述**	**用法示例**
:input	所有的<input>元素	$(“:input”)
:text	选择 type=”text”的<input>元素	$(“:text”)
:password	选择 type=”password”的<input>元素	$(“:password”)
:radio	选择 type=”radio”的<input>元素	$(“:radio”)
:checkbox	选择 type=”checkbox”的<input>元素	$(“:checkbox”)
:submit	选择 type=”submit”的<input>和<button>元素	$(“input:submit”)
:reset	选择 type=”reset”的<input>和<button>元素	$(“:reset”)
:button	选择 type=”button”的<input>和<button>元素	$(“:button”)
:image	选择 type=”image”的<input>元素	$(“:image”)
:file	选择 type=”file”的<input>元素	$(“:file”)
指定状态的表单元素		
选择器	**描述**	**用法示例**
:enabled	所有启用的<input>和<button>元素	$(“:enabled”)
:disabled	所有被禁用的<input>和<button>元素	$(“:disabled”)
:selected	下拉列表中处于选中状态的<option>元素	$(“:selected”)
:checked	所有被选中的单选按钮或者复选框	$(“:checked”)

【例 7-7】jQuery 表单选择器的使用

7.1.4 层次选择器(Hierarchy Selector)

1. 子元素选择器

子元素选择器（Child Selector）只能选择指定元素的第一层子元素。其语法结构如下：

$("parent>child")
其中参数 parent 可以是任何一个有效的 jQuery 选择器，参数 child 填入的选择器筛选的必须是 parent 的第一层子元素。例如：

```
<p>
这是一个<span><strong>测试</strong>段落</span>，用于测试子元素的层次。
</p>
```

在上述代码中段落元素<p>的第一层子元素为，而是的第一层子元素，只能算是<p>元素的后代。因此使用子元素选择器只能是$("p>span")或者是$("span>strong")的形式，不可以写成$("p>strong")的形式。

【例 7-8】jQuery 子元素选择器的使用
【代码说明】

本示例使用无序标签<ul>制作了一个简易章节目录，其中包含了 3 个列表选项元素<li>分别表示第一、二、三章，以及在第二章和第三章之间包含了另一个无序标签<ul>，并在其中包含了 2 个列表选项元素<li>分别表示 2.1 和 2.2 小节的目录。

在 jQuery 中使用了子元素选择器$("ul.all>li")，表示从 class="all"的<ul>元素中选择出所有的第一层<li>元素，并将其边框样式设置为 1 像素宽的红色实线。由图可见，用于表示第一、二、三章的 3 个列表选项元素<li>边框样式被重置，但是用于表示 2.1 和 2.2 小节的<li>元素不受影响。这是由于表示 2.1 和 2.2 小节的<li>元素包含在内部的<ul>元素中，是第二层后代元素，不符合子元素选择器的筛选规则。

2. 后代选择器

后代选择器（Descendant Selector）可以用于选择指定元素内包含的所有后代元素。它比子元素选择器的涵盖范围更加广泛。其语法结构如下：

```
$("ancestor descendant")
```

其中参数 ancestor 可以是任何一个有效的 jQuery 选择器，参数 descendant 填入的选择器筛选的必须是 parent 的后代元素，该后代元素可以是 parent 元素的第一层子元素，也可以是其中子元素的后代。例如：

```
<p>
这是一个<span><strong>测试</strong>段落</span>，用于测试子元素的层次。
</p>
```

在上述代码中段落元素<p>的第一层子元素为，而是的第一层子元素，属于是<p>元素的后代。因此使用后代选择器选择其中的标签可以是$("p strong")或者$("span strong")的形式均可。

【例 7-9】jQuery 后代选择器的使用
【代码说明】

本示例包含了 2 个区域元素 ，分别定义其 class 名称为 style01 和 style02 以示区别。在 CSS 内部样式表中为 元素设置统一样式：宽 100 像素，各边外边距为 10 像素，带有 1 像素宽的黑色实线边框。在这两个 元素放置相同的子元素：标题元素<h3>和段落元素<p>，并在段落元素<p>的内部将局部文字使用标签包围。因此元素是<p>元素的子元素，也是 元素的后代。

在 jQuery 中使用了后代选择器$("div.style01

span")，表示从 class="style01"的 元素中选择出其内部包含的所有元素，并将其边框样式设置为 2 像素宽的红色实线。由图可见，第一个 元素中的标签被标记了出来，而第二个 元素中的标签完全没有受到影响。

3. 后相邻选择器

后相邻选择器（Next Adjacent Selector）可以用于选择指定元素相邻的后一个元素。其语法结构如下：

```
$("prev+next")
```

其中参数 prev 可以是任何一个有效的 jQuery 选择器，参数 next 填入的选择器筛选的必须是与 prev 相邻的后一个元素。

当需要选择的元素没有 id 名称或 class 属性值可以进行选择时，可以考虑使用该方法先获取其相邻的前一个元素，然后再定位到需要的元素。例如：

```
<p class="test">这是第一个段落元素。</p>
<p>这是第二个段落元素。</p>
```

上述代码包含了两个段落元素<p>，其中第一个元素可以使用类选择器$("p.test")获取，第二个元素无 id 名称和 class 属性值，因此可以考虑使用相邻选择器$("p.test+p")获取。

【例 7-10】jQuery 后相邻选择器的使用

【代码说明】

本示例包含了 2 个区域元素 ，在 CSS 内部样式表中为 元素设置统一样式：宽 100 像素、高 50 像素，带有 1 像素宽的黑色实线边框，各边外边距为 10 像素，并且向左浮动。其中第一个 元素具有 id="test01"的属性值，而第二个 元素没有任何特征。

在 jQuery 中使用了后相邻选择器$("div#test01+div")，表示先找到 id="test01"的 元素，然后定位与其相邻的下一个 元素，并将其边框样式设置为 2 像素宽的红色实线。由图可见，第二个 元素的边框已经更新为红色样式，说明后相邻选择器生效。

4. 后兄弟选择器

后兄弟选择器（Next Siblings Selector）可以用于选择指定元素后面跟随的所有符合条件的兄弟元素。其语法结构如下：

```
$("prev~siblings")
```

其中参数 prev 可以是任何一个有效的 jQuery 选择器，参数 siblings 填入的选择器筛选的必须是位置在 prev 元素后面的兄弟元素。

该选择器与上一小节介绍的$("prev+next")不同之处在于：$("prev+next")只能筛选紧跟在指定元素后面的下一个相邻元素，而$("prev~siblings")可以筛选指定元素后面所有符合条件的兄弟元素，可以是多个元素。

当在同一个父元素中有多个元素需要选择，可以考虑使用该选择器先找到它们的前一个兄弟元素，然后在批量选中这些元素。例如：

```
<p class="test">这是第一个段落元素。</p>
<p>这是第二个段落元素。</p>
<p>这是第三个段落元素。</p>
```

上述代码包含了三个段落元素<p>，其中第一个段落元素可以使用类选择器$("p.test")获取，后两个段落元素无 id 名称和 class 属性值，因此可以考虑使用后兄弟选择器

$("p.test~p")获取。

【例 7-11】jQuery 后兄弟选择器的使用

【代码说明】

本示例包含了 3 个区域元素 和 1 个段落元素<p>，在 CSS 内部样式表中为 和<p>元素设置统一样式：宽 100 像素、高 100 像素，带有 1 像素宽的黑色实线边框，各边外边距为 10 像素，并且向左浮动。其中第一个 元素具有 id="test"的属性值，用于作为参照元素。紧跟其后的其他兄弟元素均没有任何特征。

在 jQuery 中使用了后兄弟选择器$("div#test~div")，表示先找到 id="test"的 元素，然后查找该元素后面的所有兄弟元素，如果有 元素就将其边框样式设置为 2 像素宽的红色实线。由图可见，第 2、4 个方块的边框已经更新为红色样式，说明后兄弟选择器生效。其中第 3 个方块是<p>元素用于作为对比，它不符合选择器条件因此未被更改边框样式。

7.1.5 jQuery CSS 选择器

jQuery CSS 选择器用于改变指定 HTML 元素的 CSS 属性。其语法结构如下：

```
$(selector).css(propertyName, value);
```

其中 selector 参数位置可以是任意有效的选择器，propertyName 参数位置为 CSS 属性名称，value 参数位置为需要设置的 CSS 属性值。

例如，将所有 h1 标签的背景颜色改为灰色，写法如下：

```
$("h1").css("background-color","gray");
```

【例 7-12】jQuery CSS 选择器的使用

使用 jQuery CSS 选择器将页面上所有的元素标记为红色字体。

【代码说明】

本示例包含了两个段落元素<p>，并在 CSS 内部样式表中为其设置统一样式：宽 100 像素、高 50 像素，带有 1 像素宽的实线边框，各边外边距为 10 像素，向左浮动。在这两个段落元素<p>中各包含了一个元素用于测试 CSS 选择器的效果。

在 jQuery 中使用了 CSS 选择器$("span").css("color", "red")，表示先找到所有的元素，然后将其中的字体颜色更新为红色。由图可见，两个段落中的被标签包含的文字已变为红色，说明 CSS 选择器生效。

jQuery CSS 选择器中的 css()方法还可以用于批量设置元素的样式属性。关于 css()方法的更多用法，请参考后续 10.1.5 节中的 jQuery css()部分。

7.2 jQuery 过滤器

jQuery 过滤器可单独使用，也可以与其他选择器配合使用。根据筛选条件可归纳为基础过滤器、子元素过滤器、内容过滤器和可见性过滤器。

7.2.1 基础过滤器(Basic Filter)

jQuery 基础过滤器(Basic Filter)包含了一些常用的过滤功能。

- :first用于选择第一个符合条件的元素。
- :last 用于选择最后一个符合条件的元素。
- :even　用于选择偶数的元素（元素从 0 开始计数）。
- :odd用于选择奇数的元素（元素从 0 开始计数）。

- :eq()用于选择指定序号的元素（元素从 0 开始计数）。
- :gt() 用于选择大于指定序号的元素（元素从 0 开始计数）。
- :lt() 用于选择小于指定序号的元素（元素从 0 开始计数）。
- :not()　用于选择所有不符合指定要求的元素。
- :header 用于选择所有的标题元素<h1>-<h6>

1. :first 和:last

:first 过滤器用于筛选第一个符合条件的元素，其语法结构如下：

```
$(":first")
```

:first 过滤器只能选择符合条件的第一个元素。例如：

```
$("div:first")
```

上述代码表示选择页面上的第一个 元素。

:last 过滤器用于筛选最后一个符合条件的元素，其语法结构如下：

```
$(":last")
```

:last 过滤器可单独使用，也可以与其他选择器配合使用。

【例 7-13】jQuery 基础过滤器:first 和:last 的使用

【代码说明】

本示例包含了三个段落元素<p>，并在 CSS 内部样式表中为其设置统一样式：宽 100 像素、高 50 像素，带有 1 像素宽的实线边框，各边外边距为 10 像素，向左浮动。

在 jQuery 中分别使用了基础过滤器:first 和:last 来选择元素。其中$("p:first")表示找到网页中的第一个<p>元素，css("border", "1px solid red")表示将其边框样式更新为 1 像素宽的红色实线边框；而$("p:last")表示找到网页中最后一个<p>元素，css("border", "1px dashed blue")表示将其边框样式更新为 1 像素宽的虚线蓝色边框。由图可见，两个指定段落的边框样式均已发生变化，说明 CSS 选择器生效。

2. :even 和:odd

:even 过滤器用于筛选符合条件的偶数个元素，序号从 0 开始计数。其语法结构如下：

```
$(":even")
```

例如，筛选表格中的偶数行写法如下：

```
$("tr:even")
```

由于:even 过滤器是基于 JavaScript 数组的原理，同样继承了从 0 开始计数的规则，因此上述代码表示筛选表格的第 1、3、5……以及更多行。

:odd 过滤器用于筛选符合条件的奇数个元素，序号从 0 开始计数。其语法结构如下：

```
$(":odd")
```

例如，筛选表格中的奇数行写法如下：

```
$("tr:odd")
```

需要注意的是，:odd 过滤器同样继承了从 0 开始计数的规则，因此上述代码表示筛选表格的第 2、4、6……以及更多行。

【例 7-14】jQuery 基础过滤器:even 和:odd 的使用

3. :eq()、:gt()和:lt()

:eq()过滤器用于选择指定序号为 n 的元素，序号从 0 开始计数。其中 eq 来源于英文单词 equal（等于）的前两个字母缩写。其语法结构如下：

```
$(":eq(index)")
```

参数 index 可替换为指定的序号。在 jQuery1.8 版以后，若 index 填入负数，表示倒数第 n 个元素。其中:eq(0)等同于:first 过滤器的效果。

:gt()过滤器用于选择所有大于序号为 n 的元素，序号从 0 开始计数。其中 gt 来源于英文单词 greater than（大于）的首字缩写。其语法结构如下：

```
$(":gt(index)")
```

其中参数 index 替换为指定的序号。在 jQuery1.8 版以后，若 index 填入负数，表示序号大于倒数第 n 个元素。

:lt()过滤器用于选择所有小于序号为 n 的元素，序号从 0 开始计数。其中 lt 来源于英文单词 less than（小于）的首字缩写。其语法结构如下：

```
$(":lt(index)")
```

参数 index 可替换为指定的序号。在 jQuery1.8 版以后，若 index 填入负数，表示序号小于倒数第 n 个元素。其中:lt(1)相当于:first 过滤器的效果。

【例 7-15】jQuery 基础过滤器:eq()、:gt()和:lt()的使用

【代码说明】

本示例包含了一个宽度为 100 像素的无序列表<ul>及其内部的 5 个列表选项元素<li>。在 jQuery 中使用了$("li:eq(2)")、$("li:gt(2)")和$("li:lt(2)")分别表示查找序号等于 2、大于 2 和小于 2 的<li>元素，并将其边框样式分别设置为 2 像素宽的红色、黄色和蓝色实线。

由图可见，$("li:eq(2)")影响的是第三行的列表选项元素<li>；$("li:gt(2)")影响的是第三行之后的所有列表选项；$("li:lt(2)")影响的是第三行之前的所有列表选项。

4. :not()

:not()过滤器用于筛选所有不符合条件的元素，其语法结构如下：

```
$(":not(selector)")
```

所有的选择器都可以与:not()配合使用来筛选相反的条件。

【例 7-16】jQuery 基础过滤器:not()的使用

【代码说明】

本示例包含了三个段落元素<p>，并在 CSS 内部样式表中为其设置了统一的样式：宽、高均为 100 像素，带有 1 像素宽的实线边框，向左浮动以及各边外边距为 10 像素。其中第二个段落元素设置了 id="test"以便使用过滤器:not()测试效果。

在 jQuery 中使用了选择器$(":not(p#test)")表示筛选所有除 id="test"的段落元素之外的所有 HTML 元素，并为它们添加 1 像素宽的红色实线边框。由图可见，中间的段落元素被排除在了选择范围外，因此过滤器:not()的效果实现。

5. :header

:header()过滤器用于筛选所有的标题元素，从<h1>到<h6>均在此选择范围内。

其语法结构如下：

```
$(":header")
```

【例 7-17】jQuery 基础过滤器:header()的使用

7.2.2 子元素过滤器(Child Filter)

jQuery 子元素过滤器(Child Filter)可筛选指定元素的子元素。

- :first-child 用于选择所有在父元素中的第一个子元素。
- :last-child 用于选择所有在父元素中的最后一个子元素。
- :nth-child() 用于选择所有在父元素中的第 n 个子元素。
- :nth-last-child() 用于选择所有在父元素中的倒数第 n 个子元素。
- :only-child 用于选择所有在父元素中唯一的子元素。

1. :first-child

:first-child 过滤器用于筛选页面上每个父元素中的第一个子元素，其语法结构如下：

```
$(":first-child")
```

与只能选择唯一元素的:first 过滤器不同，只要是页面上的父元素都可以同时使用:first-child 过滤器从中选出其第一个子元素，因此选择结果可能不止一个元素。

:first-child 过滤器可单独使用，也可以与其他选择器配合使用。例如：

```
$("p:first-child")
```

上述代码表示在页面上所有包含有段落元素<p>的父元素中筛选出每个父元素内部的第一个段落子元素<p>。需要注意的是，这里所筛选出来的段落子元素<p>有可能并不是其父元素的第一个子元素，例如以下这种情况：

```
<div>
  <span>我是第一个子元素。</span>
  <p>我是第二个子元素，但是也是第一个段落元素。我将被$("p:first-child")筛选出来。
</p>
<p>我是第三个子元素。</p>
</div>
```

【例 7-18】jQuery 基础过滤器:first-child ()的使用

2. :last-child

:last-child 过滤器用于筛选页面上每个父元素中的最后一个子元素，其语法结构如下：

```
$(":last-child")
```

与:first-child 过滤器类似，选择结果可能不止一个元素。

:last-child 过滤器可单独使用，同样也可以与其他选择器配合使用。例如：

```
$("p:last-child")
```

上述代码表示在页面上所有包含有段落元素<p>的父元素中筛选出每个父元素内部的最后一个段落子元素<p>。需要注意的是，这里所筛选出来的段落子元素<p>有可能并不是其父元素的最后一个子元素，例如以下这种情况：

```
<div>
<p>我是第一个子元素。</p>
  <p>我是第二个子元素，但是也是最后一个段落元素。我将被$("p:last-child")筛选出来。
</p>
  <span>我是最后一个子元素。</span>
</div>
```

【例 7-19】jQuery 基础过滤器:last-child ()的使用

3. :nth-child

:nth-child()过滤器用于筛选页面上每个父元素中的第 n 个子元素，序号从 1 开始计数。其语法结构如下：

```
$(":nth-child(index)")
```

其中 index 参数可以填入具体的数值，例如：

```
$(":nth-child(2)")
```

上述代码表示筛选父元素中的第 2 个子元素。

也可以在:nth-child()过滤器的 index 参数位置填入 even 或 odd 字样，分别表示偶数个或奇数个元素。例如：

```
$(":nth-child(odd)")
```

上述代码表示筛选父元素中的第 1、3、5、7...个子元素。

还可以在:nth-child()过滤器的 index 参数位置填入数字与字母 n 的算术组合，n 的取值从 0 开始，每次自增 1 直到筛选完全部符合条件的子元素为止。例如：

```
$(":nth-child(3n+1)")
```

上述代码表示筛选父元素中的第 3n+1 个元素，即第 1、4、7、10...个子元素。

:nth-child()过滤器可单独使用，也可以与其他选择器配合使用。:nth-child(1)表示筛选第一个子元素，等同于:first-child()。

【例 7-20】jQuery 基础过滤器:nth-child ()的使用

4. :only-child

:only-child 过滤器用于筛选所有在父元素中有且仅有一个的子元素。其语法结构如下：

```
$(":only-child")
```

:only-child()过滤器可单独使用，也可以与其他选择器配合使用。例如：

```
$("div span:only-child")
```

上述代码表示在所有只包含一个子元素的 父元素中，查找类型的子元素。

如果父元素中包含了其他子元素则不匹配，例如以下这种情况：

```
<div>
<span>这是 span 元素</span>
<button>这是 button 元素</button>
</div>
```

上述代码如果使用$("div span:only-child")进行筛选则匹配失败，因为父元素 中还包含了其他子元素<button>。

需要注意的是，即使其他子元素是
或<hr>等内容也会匹配失败。例如：

```
<div>
<span>这是 span 元素</span>
<hr>
</div>
```

上述代码如果使用$("div span:only-child")进行筛选也会匹配失败，因为<hr>也会被认为是父元素 的第二个子元素。

如果父元素中只包含其他文本内容不影响:only-child 过滤器的判断。例如：

```
<div>
<span>这是 span 元素</span>
这段文字不会影响 span 作为 div 的唯一子元素。
</div>
```

上述代码如果使用$("div span:only-child")进行筛选会匹配成功。

如果子元素内部还包含自身的子元素也不会影响匹配。例如:

```
<div>
<span>
这是 span 元素<br>
这里的 br 元素是 span 的子元素<br>
不影响 span 作为 div 的唯一子元素。
</span>
</div>
```

上述代码如果使用$("div span:only-child")进行筛选也会匹配成功。

【例 7-21】jQuery 基础过滤器:only-child ()的使用

7.2.3 内容过滤器(Content Filter)

jQuery 内容过滤器(Content Filter)可以根据元素所包含的子元素或文本内容进行过滤筛选。

- :contains()　用于选择所有处于隐藏状态的元素。
- :empty　用于选择未包含子节点（子元素和文本）的元素。
- :parent　用于选择拥有子节点（子元素和文本）的元素。
- :has()　用于选择包含指定选择器的元素

1. :contains()

:contains()过滤器用于筛选出所有包含指定文本内容的元素，其语法结构如下：

```
$(":contains(text)")
```

其中 text 替换成指定的字符串文本，由于过滤器外面已经存在一对双引号，因此该文本可以用单引号括住具体文字内容。例如：

```
$("p:contains('hi')")
```

上述代码表示选择所有文本内容包含"hi"字样的段落元素<p>。

:contains()过滤器的筛选文本是大小写敏感型，例如：

$("p:contains('hello')")

$("p:contains('HELLO')")

上述两个选择器的表示完全不同的筛选结果。

【例 7-22】jQuery 内容过滤器:contains()的使用

2. :empty

:empty 过滤器用于选择未包含子节点（子元素和文本）的元素，其语法结构如下：

$(":empty")

:empty 过滤器可以和其他有效选择器配合使用，例如：

$("td:empty")

上述代码表示选择所有无内容的表格单元格元素<td>。部分元素标签直接默认为不包含任何子节点，例如水平线标签<hr>、换行标签
、图像标签<img>、表单标签<input>等。

【例 7-23】jQuery 内容过滤器:empty()的使用

```
1.  <!DOCTYPE html>
2.  <html>
3.      <head>
4.          <meta charset="utf-8">
5.          <title>jQuery 内容过滤器:empty 示例</title>
6.          <script src="js/jquery-1.12.3.min.js"></script>
7.      </head>
8.      <body>
9.          <h3>jQuery 内容过滤器:empty 示例</h3>
10.         <hr>
11.         <table border="1">
12.             <tr><th>第一季度</th><th>第二季度</th><th>第三季度</th></tr>
13.             <tr><td>100</td><td>120</td><td>140</td></tr>
14.             <tr><td>200</td><td>220</td><td>240</td></tr>
15.             <tr><td>300</td><td>320</td><td></td></tr>
16.         </table>
17.         <script>
18.             $(document).ready(function() {
```

```
19.                         $("td:empty").css("background","lightblue");
20.                     });
21.             </script>
22.         </body>
23.  </html>
```

运行效果如图 7-25 所示。

图 7-23 jQuery 内容过滤器:empty()的使用效果

3. :parent

:parent 过滤器用于选择包含了子节点（子元素和文本）的元素，其语法结构如下：

```
$(":parent")
```

:parent 过滤器可以和其他有效选择器配合使用，例如：

```
$("td:parent")
```

上述代码表示选择所有包含内容的表格单元格元素<td>。需要注意的是，W3C 规定了段落元素<p>起码包含一个子节点，即使该元素中没有任何文本内容。

【例 7-24】jQuery 内容过滤器:parent()的使用

4. :has

:has()过滤器用于选择包含指定选择器的元素，其语法结构如下：

```
$(":has(selector)")
```

所有的选择器都可以与:has()配合使用作为包含的条件。例如：

```
$("div:has(table)")
```

上述代码表示选择所有包含表格的块元素 。

【例 7-25】jQuery 内容过滤器:has()的使用

7.2.4 可见性过滤器(Visibility Filter)

jQuery 可见性过滤器(Visibility Filter)根据元素当前状态是否可见进行过滤筛选。

- :hidden 用于选择所有处于隐藏状态的元素。
- :visible 用于选择所有处于可见状态的元素。

1. :hidden

:hidden 过滤器用于筛选出所有处于隐藏状态的元素，其语法结构如下：

```
$(":hidden")
```

:hidden 过滤器可单独使用，也可以与其他选择器配合使用对元素进一步过滤筛选。

例如：

```
$("p:hidden")
```

上述代码表示查找所有隐藏的段落元素<p>。

元素在网页中不占用任何位置空间就被认为是隐藏的，具体有以下几种情况：

- 元素的宽度和高度明确设置为 0；
- 元素的 CSS 属性中 display 的值为 none；
- 表单元素的 type 属性设置为 hidden；
- 元素的父元素处于隐藏状态，因此元素也一并无法显示出来；
- 下拉列表中的所有选项<option>元素也被认为是隐藏的，无论其是否为 selected 状态；

【例 7-26】jQuery 可见性过滤器:hidden()的使用

使用可见性过滤器:hidden()查找处于隐藏状态的 元素与<input>元素。

【代码说明】

本示例包含了两组测试元素：一对表单<input>元素，其 type 值分别为 text 和 hidden；一对 元素，其中一个设置为 display:none 的状态。页面在首次加载后两组元素都只能显示其中非隐藏状态的一个元素。

在 jQuery 中使用了 find(selector)方法查找隐藏的元素，该方法可以返回符合条件的元素对象数组。其中 $("body") 表示在 <body> 元素中查找， find("div:hidden") 和 find("input:hidden")分别表示查找所有处于隐藏状态的 元素和<input>元素。当前仅为 find()方法的简单使用，关于 find()方法的更多介绍可以查阅本书第七章 jQuery 遍历的相关内容。

2. :visible

:visible 过滤器用于筛选出所有处于可见状态的元素，其语法结构如下：

```
$(":visible")
```

:visible 过滤器可单独使用，也可以与其他选择器配合使用对元素进一步过滤筛选。:visible 过滤器与:hidden 过滤器的筛选条件完全相反，因此无法同时使用。

需要注意的是，元素处于以下几种特殊情况中也被认为是可见状态:

- 元素的透明度属性 opacity 为 0，此时元素仍然占据原来的位置；
- 元素的可见属性 visibility 值为 hidden，此时元素仍然占据原来的位置；
- 当元素处于逐渐被隐藏的动画效果中，到动画结束之前都被认为仍然是可见的；
- 当元素处于逐渐被显现的动画效果中，从动画一开始启动就被认为是可见的。

【例 7-27】jQuery 可见性过滤器:visible()的使用

使用可见性过滤器:visible()查找处于显示状态的元素。

【代码说明】

本示例包含了三个段落元素<p>：前两个为默认可见状态，第三个设置为 display:none 的状态。页面在首次加载后只能显示前两个处于可见状态的段落元素<p>。

在 jQuery 中使用了 find(selector)方法查找处于可见状态的元素，该方法可以返回符合条件的元素对象数组。其中 $("div#box") 表示在 id="box" 的 元素中查找，find("p:visible")表示查找所有处于可见状态的<p>元素。因此由上图可见，关于处于显示状态<p>元素的统计结果为 2 个。

7.3 阶段案例：网页一键换肤