2

选择、修饰和增强

“在第二阶段，我们希望利用社会力量，支持 Web 2.0 基于社区的、众性质的、Ajax 及互操作”，我们的新客户团队中的一位成员如是说。“但是现在，我们只需要对网站做一些基本的改变。”

这位客户正在启动一个称为 StarTrackr!的创业项目，它使用 GPS 和 RFID 技术追踪受欢迎的名人准确的物理位置，然后向粉丝出售这些信息。他们本来一直在努力经营一个朋友的本地小店，但现在准备在网上试试运气。

“你能做到么？这是周五晚上生意结束之前需要实现的需求列表。”

你调查了这个列表，巧合的是，注意到所有请求都能用 jQuery 实现。你看了看自己的日历，现在是周五早晨，那就让我们开始吧！

列表中的第一个任务是在现有网站加载时添加一个简单的 JavaScript 提示。显然这只是个临时性措施，只是为了让访问者知道 StarTrackr! 目前没有遭到侵犯隐私的控告（最近一家本地报纸上对此事有所暗示）。

当然，使用以前普通的 JavaScript 也能完成上述功能，但我们现在知道使用 jQuery 可以让生活更轻松，而且，可以一边用一边学习一门新技术。在第 1 章已经看过 jQuery 语句的解剖，现在来看看让 jQuery 实际发挥作用需要的步骤：等待页面加载完成，选择我们的目标，然后改变它。

添加提示看起来可能没什么意思，你可能会想，jQuery 的功能应该比这更复杂，它应该能够做什么什么……而现在就这么简单！即使高级的特效也依赖于这个基本公式（虽然最后两步的多重迭代可能需要有一点 JavaScript 知识）。现在，我们先从方便好用的操作学起。

2.1 确保页面已经就绪

在与页面上的 HTML 元素交互之前，这些元素需要已经加载完成：只有在它们加载之后才能改

变它们。在过去的 JavaScript 时代，唯一可靠的方式就是等待整个页面（包括图片）都加载完成后才运行脚本。

幸运的是，jQuery 有一个非常酷的内置事件，它可以尽可能快地执行我们的操作。因此，对最终用户来说，加载页面和应用程序的速度似乎要快得多：

chapter_02/01_document_ready/script.js

```
$(document).ready(function() {
  alert('Welcome to StarTrackr! Now no longer under police
        investigation!);
});
```

这里的重要部分（加粗显示）的功能是："在文档就绪时，运行函数"。日后你会看到，这是 jQuery 最常见的代码片断。做这样一个简单的提示测试通常是一个好主意，它可以确保你已经正确地包含了 jQuery 库，不会出现什么意外。

将会看到大量的$(document).ready()！

用 jQuery 做的几乎每件事情都需要在文档已经就绪之后才能进行，所以我们会大量地使用这个操作。从现在起，将其称为"文档就绪"事件。本书后续的每个示例，除非另有说明，都需要在文档就绪事件内运行。但在每个页面上应该只需要声明这个事件一次。

"文档就绪"这个用语实际上有点高级，但它如此普遍，以致于 jQuery 包含了一个快捷版本，移除了(document)和 ready：

```
$(function() {
  alert('Ready to do your bidding!');
});
```

如果你喜欢使用这个快捷方法，那就用吧！有人认为，扩展版本是自文档代码一个较好的示例，扩展版本更容易准确地看出代码的功能，尤其是当代码淹灭在页面中其他开发人员的 JavaScript 中的时候！

乍一看，文档就绪事件不符合前面在 jQuery 剖析中介绍的结构，但如果细看，就会看到，必要的部分都在：选择器是 document，操作是 ready，参数是个运行某些代码（上述为 alert）的函数。

2.2 选择：jQuery 的核心

时光飞逝，最后期限的来临不会等待任何人。客户团队已经注意到，人们在网站上引用了错误的名人 ID。这是因为名人的名字全放在一个大表内，因此用户很难逐行找到名人正确的引用 ID。客户团队的一位成员告诉我们，他希望每隔一行就用淡灰色显示，以便用户能够方便地找到自己喜欢的名人。

现在 jQuery 已经就绪，可以执行我们的命令了；只需要为它选择一个目标。jQuery 真正的艺术

是在页面上选择要修改的元素。菜鸟和忍者最大的不同之一，就是为了挑出要操作的元素而花费的时间的多少！

你可能记得，在 1.4.2 节 jQuery 语句剖析中讲解的内容，所有选择器都封装在 jQuery 函数内：

```
jQuery(<selectors go here>)
```

或者在别名内：

```
$(<selectors go here>)
```

本书后续将使用快捷别名，因为这样更方便。前面提到过，除存在与其他库的冲突问题（请参阅第 9 章“避免冲突”一节）外，没有需要使用完整 jQuery 名称的理由。

2.2.1 简单选择

我们的任务是选择名人表中间隔的行。如何才能做到呢？在用 jQuery 进行选择的时候，你的目标要同要求一样具体：需要找到最简洁的选择器，准确地返回你要修改的内容。我们先来看一下名人表的标签，如图 2.1 所示。

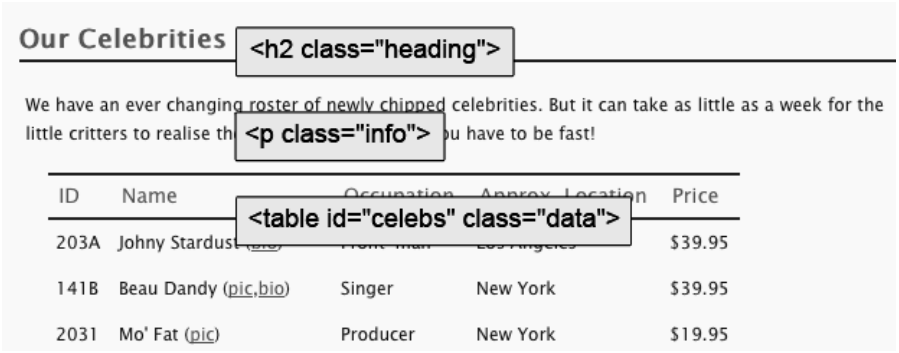

图 2.1 HTML 页面中的 class 和 id 属性

我们先从选择整个页面中的每个表格行元素开始。要按元素类型选择，只需要将元素的 HTML 名称作为字符串参数传递给$函数即可。要选择表格的全部行元素（使用<tr>标签表示），只需要这样写：

```
$('tr')
```

什么也没发生！

如果运行这条命令，页面上什么也不会发生。这在预料之内，毕竟，我们仅仅选择了元素。但不需要担心，很快我们就会用各种奇妙而精彩的方式修改选中的项目。

类似的，如果我们想选择页面上的每个段落、 元素、<h1>标题或者<input>框，可以分别使用下面的这些选择器：

```
$('p')
$('div')
$('h1')
$('input')
```

但是，我们并不想修改名人页面中的每一个表格行；只想修改拥有名人数据的表格行。我们需要做得更具体一点，首先选择容纳名人列表的包含元素。如果查看以下的 HTML 和图 2.1，可以看到包含名人数据的表格的 id 是 celebs，class 是 data。我们可以使用其中任何一个来选择这个表格。

chapter_02/02_selecting/index.html (*excerpt*)

```
<table class="data" id="celebs">
  <thead>
    <tr>
      <th>ID</th>
      <th>Name</th>
      <th>Occupation</th>
      <th>Approx. Location</th>
      <th>Price</th>
    </tr>
  </thead>
```

jQuery 借用 CSS 的规范来引用 id 和 class 名。要按 id 进行选择，在#号后加元素的 id，并将其作为字符串传递给 jQuery 函数：


```
$('#celebs')
```

你应该注意到，我们传递给 jQuery 函数的字符串格式，与 CSS 的 id 选择器的格式完全相同。因为 id 应该是唯一的，所以我们认为这个选择只会返回一个元素。jQuery 现在持有对这个元素的一个引用。

类似的，我们也可以使用一个 CSS 的 class 选择器按类（class）进行选择。我们将以句点（.）开始、后接元素的 class 名称的字符串传递给 jQuery 函数：

```
$('.data')
```

使用上述两条语句都可以选择表格，但就像介绍 DOM 时讲过的，一个 class 可以由多个元素共享，而 jQuery 会非常乐于选择我们让它选择的尽可能多的元素。如果有多个表格（或其他元素）也有 class 数据，那么这些元素都会被选中。因此，对于这个元素，我们坚持使用 id！

能不能再具体点？

就像 CSS 一样，我们可以选择$('.data')，也可选择更具体的$('table.data')。通过在 class 之上再指定元素类型，选择器就会只返回带有 class 数据的 table 元素，而不是带有 class 数据的全部元素。另外，与 CSS 一样，可以添加父容器选择器来进一步缩小选择范围。

2.2.2 缩小选择范围

我们已经成功地选择了表格，但我们的兴趣并不在选择表格本身，而在于选择表格中相隔的每

行。我们已经选择了包含元素，从这个包含元素出发，选择它的全部后代，即表格的行：我们想指定包含 table 内部的全部表格行。为了做到这点，我们在祖先和后代之间空一个空格：

```
$('#celebs tr')
```

可以用这个结构逐级深入到要寻找的元素，但是为了清晰，请保持选择器尽可能的简洁。

我们将这个做法再延伸一步。假设我们要选择<p>元素内的全部元素，<p>元素都在 元素内，但是只针对那些名为 fancy 的 class 的那些 ，可以使用下面的选择器：

```
$('div.fancy p span')
```

如果你能看得懂上述内容，那现在就可以选择一切了！

2.2.3 测试选择

回到手上的任务，我们感觉到正在接近目标，但迄今为止，还只是盲目地选择，并没有方法知道事实上已经选择了什么元素。如今，有许多非常棒可在手边使用的调试工具（我们很快就会看到它们），但目前，有一个简单的技巧，就是查看选择的 length 属性。length 会返回当前与选择器匹配的元素的个数。我们可以将这个属性与好用而信得过 alert 语句组合起来，确保已经选中了元素：

chapter_02/02_selecting/script.js

```
$(document).ready(function() {
  alert($('#celebs tr').length + ' elements!');
});
```

这时会在提示中显示选择的长度：名人<table>的 7 个元素。结果可能与你期望的不同，因为表格中只有 6 个名人！如果你查看 HTML，就会注意到问题所在：<table>的标题也是<tr>，所以合计有 7 行。对这个问题的快速解决办法包括：缩小选择器范围，只寻找位于<tbody>元素内的行：

chapter_02/03_narrowing_selection/script.js

```
$(document).ready(function() {
  alert($('#celebs tbody tr').length + ' elements!');
});
```

这时会在提示中显示正确的长度，即 6 个元素。jQuery 对象现在包含名人表格的 6 个行元素。

如果提示显示为 0，你就知道选择器有错误。处理这类问题的一个好办法是尽可能地将选择器精简到最小、最简单。

在本示例中，我们可以先简单地使用$('#celebs')，则只会选择 table 元素，并提示长度为 1。从这开始，可以将选择器逐步具体化，一边具体，一边检查是否选择了正确的元素个数。

2.2.4 筛选器

成功选择表格的所有行之后，将选择范围缩小到每隔一行就简单了，因为 jQuery 有专门做这件事的**筛选器**。筛选器会删除特定项目，只保留我们需要的项目。通过后面更多的一些示例，你对哪些东西可以筛选会有所感觉。现在，直接跳到斑马条纹效果需要的筛选器：

chapter_02/04_filters/script.js

```
$(document).ready(function() {
  alert($('#celebs tbody  tr:even').length + ' elements!');
});
```

筛选器加到要筛选的项目上（在本示例中，为表格的行）时，使用冒号（:）进行定义，其后接筛选器的名称。这里使用的:even 筛选器保留选择中索引号为偶数的元素，删除其余元素，这正是我们需要的效果。如果现在用提示显示选择的长度，可以看到结果为 3，符合预期。所有奇数行都从选择中筛选掉。jQuery 选择器上可用的筛选器范围很广，包括:odd（奇数）、:first（第一个）、:last（最后一个）、:eq()（指定选择范围，例如第三个元素），以及其他更多。本书后面将在需要的时候介绍每个筛选器的细节。

2.2.5 使用多个选择器

基本选择的最后一个技巧是：能够在一条语句中执行多个查询。这个能力非常有用，因为我们经常需要在页面不相关部分的多个元素上应用同一个操作。将多个选择器字符串用逗号分隔就可以使用多个选择器。例如，如果我们想选择页面上的每个段落、每个 元素、每个<h1>标题及每个<input>框，可以使用下面的选择器：

```
$('p,div,h1,input')
```

学习使用所有这些不同的选择器对所需元素进行精确访问，是掌握 jQuery 功能的一大部分，也是最让人满意的一部分，因为你可以将一些相当复杂的选择逻辑打包成一行简短的代码！

2.2.6 用好选择器

选择操作看起来似乎相当容易，在某种程度上，也确实如此。但目前为止介绍的选择方面的知识只触及到选择的皮毛。在多数情况下，掌握这些基础知识就足够了：如果只是想定位到一个元素或者一组相关的元素，则使用元素的名称、id、class 是最有效、最方便的选择方式。

当指定元素开始在 DOM 中移动时，事情就变得有些复杂。jQuery 提供了多种选择器和操作用来遍历 DOM。**遍历**意味着通过父元素和子元素在页面的层次结构中上下移动。在移动的过程中，可以添加和删除元素，在每个步骤中应用不同的操作，仅用一条 jQuery 语句就能执行一些令人难以想象的复杂操作。

如果你精通 CSS，可能已经熟悉了许多语句，这些语句多数是直接从 CSS 规范直接借用的。但可能还有一些你不熟悉的地方，尤其在没有花太多时间学习 CSS3 选择器的情况下。当然，我们将在后面实现示例和演示的过程中介绍和学习更加高级的选择技术。因此，在你希望对可用的全部 jQuery 选择器有更多了解的时候，可以随时访问在线文档细细浏览。[①]

① http://api.jquery.com/category/selectors/

2.3 修饰：用 jQuery 处理 CSS

在 jQuery 中选择元素是最困难的部分。其余的事情既简单又有趣。选择目标后，我们就能操纵它们，构建特效或界面。本节，我们将介绍一系列与 CSS 有关的 jQuery 操作：添加和删除样式、类及其他。我们执行的操作将逐一应用到我们选择的每个元素，从而让页面服从我们的意愿。

2.3.1 读取 CSS 属性

在修改 CSS 属性之前，先来看看如何访问它们。jQuery 允许我们使用 css 函数访问 CSS 属性，请尝试以下代码：

chapter_02/05_reading_css_properties/script.js

```
$(document).ready(function() {
  var fontSize = $('#celebs tbody  tr:first').css('font-size');
  alert(fontSize);
});
```

这个代码会提示与选择器匹配的第一个元素的字体大小（你可能已经猜到，:first 筛选器返回与选项器匹配的第一个元素）。

多个元素的 CSS 属性

你可能会在选择多个元素之后请求一个 CSS 属性，但这种想法通常不好：一个函数只能返回一个结果，所以只会得到第一个匹配元素的属性。

用这种方法获取 CSS 属性的优点是 jQuery 提供了元素实际计算出来的样式：得到的值是用户浏览器中实际渲染使用的值，而不是 CSS 定义中输入的值。因此，如果你在 CSS 文件中指定某个 的高度为 200 像素，但它其中的内容将它的高度撑高超过 200 像素，这时 jQuery 返回的会是元素实际的高度，而不是 CSS 文件中规定的 200 像素。

稍后在实现一些精彩技巧时，我们将会看到这一点真正重要的原因。

2.3.2 设置 CSS 属性

迄今为止，还没有看到 jQuery 实际做任何事情，现在该补救这个不足了。我们知道页面已经就绪（因为弹出了一条提示），也相当确定已经选择了感兴趣的元素。那就来检查一下我们真正有些什么吧：

chapter_02/06_zebra_striping/script.js

```
$(document).ready(function() {
  $('#celebs tbody  tr:even').css('background-color','#dddddd');
});
```

你可能很快就会看到这些！这与我们用来读取 CSS 属性使用的 css 函数相同，但现在给函数传递了额外的参数：希望给这个属性设置的值。我们用这个操作将背景颜色（background color）值设为#dddddd（浅灰色）。请用浏览器从代码库中打开对应的文件，测试它是否工作正确。你可以看到如图 2.2 所示的效果。

ID	Name	Occupation	Approx. Location	Price
203A	Johny Stardust (bio)	Front-man	Los Angeles	$39.95
141B	Beau Dandy (pic,bio)	Singer	New York	$39.95
2031	Mo' Fat (pic)	Producer	New York	$19.95
007F	Kellie Kelly (bio,press)	Singer	Omaha	$11.95
8A05	Darth Fader (pic)	DJ	London	$19.95
6636	Glendatronix (bio,press)	Keytarist	London	$39.95

图 2.2　用 jQuery 实现的斑马条纹效果

你准备好了么？

前面提到过，命令必须在$(document).ready 函数内部发出。如果我们在 DOM 就绪之前运行命令，选择器会去寻找#celebs 元素，但不会找到匹配的内容。这样它就会放弃，不会继续寻找 tr 元素，更不用说修改背景样式了。

对后续的全部示例来说，都存在这种情况，所以请记住，要将代码封装在$(document).ready 函数内。

看起来效果相当好！但可能还应该再加点东西——毕竟，越多越好！用一个更浅的字体颜色来定义条纹如何？添加第二个 CSS 属性有几种方式。最简单的方式是用新的值重复整条 jQuery 语句：

chapter_02/07_multiple_properties_1/script.js *(excerpt)*

```
$('#celebs tbody  tr:even').css('background-color','#dddddd');
$('#celebs tbody  tr:even').css('color', '#666666');
```

这些代码逐行执行。虽然最后的结果是正确的，但如果我们必须修改一大堆属性，这样做就太麻烦且低效。值得庆幸的是，jQuery 提供了一种很好的方式——**对象字面量（object literal）**来同时设置多个属性。对象字面量是个 JavaScript 概念，超出了本书范围，对于我们的用途来说，只需要了解它们提供一种方便的方式将键/值对组合起来。对于 CSS 来说，用对象字面量可以将 CSS 属性（键）和匹配的 CSS 值（值）在一个整洁的包里匹配起来：

chapter_02/08_multiple_properties_2/script.js *(excerpt)*

```
$('#celebs tbody  tr:even').css(
  {'background-color': '#dddddd', 'color': '#666666'}
);
```

对象字面量封装在尖括号内，里面的每个键和值之间用冒号分隔，每对键/值之间用逗号分隔。这个整体作为一个参数传递给 css 函数。使用这个方法，可以根据需要指定尽可能多的键/值对，只要用逗号分隔即可。用具有可读性的方式排列键/值对是个好习惯，这样日后回来看代码的时候就能方便地看出它们的功能。在需要设置大量属性的时候，这种做法尤其有帮助。例如：

chapter_02/09_multiple_properties_3/script.js *(excerpt)*

```
$('#celebs tbody  tr:even').css({
  'background-color': '#dddddd',
  'color': '#666666',
  'font-size': '11pt',
  'line-height': '2.5em'
});
```

加引号或者不加引号

一般来说，在操作 JavaScript 对象时，不必将键用引用括起来。但是，为了让 jQuery 正常工作，包含连字符的键（如前面示例中的 background-color 和 font-size）必须放在引号内，或者以驼峰大小写方式书写（例如 backgroundColor）。

另外，如果使用的键是 JavaScript 语言的关键字（如 float 和 class），也必须写在引号内。

记住哪些键需要加引号哪些不需要，会让人发晕，所以简单的解决方案就是永远将对象键都放在引号内。

2.3.3 类

真棒！我们已经完成了客户列表的两项任务，也看到了一些精彩的 jQuery 效果。但是如果现在停下来，看看我们最终的解决方案，可能注意到有些事情值得怀疑。如果用 Firebug 之类的开发工具查看以斑马条纹显示的行，可能会看到在表的行上，内联地添加了 CSS 属性，如图 2.3 所示。

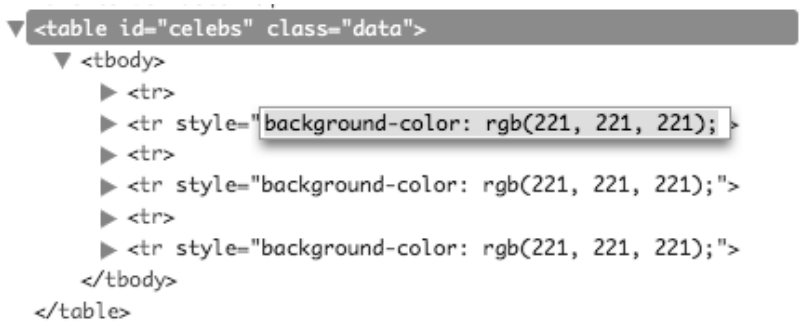

图 2.3　用 Firebug 查看的内联样式

Firebug

Firebug（或同类工具，如 Google Chrome 的 Developer Tools[①]）支持在浏览器中查看页面的 DOM，还能监视和编辑 CSS、HTML、JavaScript（包括 jQuery）。它就像是 Web 的瑞士军刀，可以帮你查看浏览器的确切做法，从而节省大量时间。Firebug 可以作为 Mozilla Firefox[②]的扩展使用，如果用其他浏览器开发，也可以作为独立的 JavaScript 文件[③]包含在你的项目内。

内联样式是 HTML/CSS 最佳做法的大忌，对吧？绝对是的，在 jQuery 中这个禁忌同样适用：要保持代码整洁、易维护，将所有样式信息放在 CSS 文件的同一位置更合理。然后（很快就会看到），通过在 HTML 标签上附加或删除 class 属性，就能方便地切换这些样式。

很多时候，最好按前面见过的方式使用 jQuery 的 css 方法。内联 CSS 最常见的用途是在快速调试代码的时候：如果只想用红色突出某个元素，以确保选中了它，那么切换回 CSS 文件添加一条新规则就有点浪费时间了。

添加和删除类

如果我们需要删除内联样式规则的 CSS，应该将它们放在哪呢？当然是独立的样式表内！我们将样式放在一个 CSS 的 class 内，用 jQuery 在 HTML 的目标元素上添加和删除这个 class。jQuery 自然也提供了一些方便的方法来操纵 DOM 元素的 class 属性。让我们使用其中最常用的方法 addClass，将斑马条纹样式转移到其所属的 CSS 文件中。

addClass 函数接受的字符串参数包含一个 class 名称。也可以同时添加多个 class，class 名称之间用空格分隔，就像写 HTML 时一样：

```
$('div').addClass('class_name');
$('div').addClass('class_name1 class_name2 class_name3');
```

但是我们只需要添加一个 class 名称，我们称之为 zebra。首先，我们要在 HTML 内添加一个到 CSS 文件的链接：

chapter_02/10_adding_classes/index.html *(excerpt)*

```
<head>
  ⋮
  <title>StarTrackr!</title>
  <link  rel="stylesheet" href="zebra.css" type="text/css" />
⋮
</head>
```

然后，我们将新规则添加到新的 CSS 文件：

① https://chrome.google.com/webstore/category/ext/11-web-development
② https://addons.mozilla.org/en-US/firefox/addon/firebug/?src=search
③ http://getfirebug.com/firebuglite

chapter_02/10_adding_classes/zebra.css

```
.zebra  {
  background-color: #dddddd;
  color:  #666666;
}
```

然后，回到 JavaScript 文件，我们修改选择器，用 jQuery 的 addClass 方法代替 css：

chapter_02/10_adding_classes/script.js *(excerpt)*

```
$('#celebs tbody  tr:even').addClass('zebra');
```

结果完全相同，但在这里再次使用 Firebug 查看表格，就会看到内联样式消失了，被新的类定义代替，如图 2.4 所示。

```
▼ <table class="data">
    ▶ <thead>
    ▼ <tbody>
        ▶ <tr class="zebra">
        ▶ <tr>
        ▶ <tr class="zebra">
        ▶ <tr>
        ▶ <tr class="zebra">
        ▶ <tr>
      </tbody>
  </table>
```

图 2.4　在表的行上添加 class

这就好多了。现在，如果我们想修改斑马条纹的外观，只要修改 CSS 文件就可以了，不必在 jQuery 代码（可能分散在多个位置）中寻找并修改相应的值。

有些时候，我们想从元素上删除类名（下面将看到一个示例）。删除类的操作称为 removeClass。这个函数的用法与 addClass 相同，只要将（不）需要的类名作为参数传递给它：

```
$('#celebs tr.zebra').removeClass('zebra');
```

添加和删除类是非常强大的功能，因为这些操作是许多视觉特效的构造块。

2.4　增强：用 jQuery 添加特效

现在你已经达到一个重要的里程碑。你已经学习了 jQuery 语句的组成部分：选择器、操作、参数。你已经学习了使用语句的步骤：确保文档就绪、选择元素和修改元素。

下一节我们将运用这些知识实现一些又酷又有用的特效，如果运气好的话，可以同时强化您对 jQuery 基础知识的理解。

2.4.1　隐藏和显示元素

客户团队不喜欢网站的免责声明——他们觉得这对产品的形象不好——但律师坚持说这是必需

的。所以客户团队要求你添加一个按钮，在用户读完免责声明之后，删除文本：

chapter_02/11_hiding/index.html *(excerpt)*

```
<input type="button" id="hideButton" value="hide" />
```

我们在页面上添加了 id 为 hideButton 的 HTML 按钮。当用户单击这个按钮时，我们希望隐藏 id 为 disclaimer 的免责声明元素：

chapter_02/11_hiding/script.js *(excerpt)*

```
$('#hideButton').click(function() {
  $('#disclaimer').hide();
});
```

运行这段代码，确保在单击隐藏按钮时，免责声明元素消失。

示例中让元素消失的部分是 hide 操作。你可能会问，这行的其他代码是什么？这是所谓的事件处理器，对它的理解，对于成为一个 jQuery 忍者至关重要。我们有许多事件处理器可以使用（这里已经使用了 click 事件处理器），在后续的学习过程中，还将使用许多事件处理器。

事件处理器

事件处理器，顾名思义，它们用于处理事件。事件是在网页发生的操作（由用户或浏览器本身执行）。事件发生时，我们说触发了事件。当我们编写代码处理事件（封装在常规的 JavaScript 函数内），就说我们捕获了这个事件。

网页上可以触发的事件成千上万：用户移动鼠标时、单击按钮时、调整浏览器窗口大小时、移动滚动条时。我们可以捕捉这些事件，并对事件进行操作。

本书介绍的第一个事件就是文档就绪事件。是的，这是一个事件处理器：当文档说“我已经准备就绪”时，就触发一个事件，被我们的 jQuery 语句捕获。

我们使用 click 事件处理器告诉 jQuery，在单击按钮时隐藏免责声明：

```
$('#hideButton').click(function() {
  $('#disclaimer').hide();
});
```

this

事件触发时，我们通常要引用触发它的元素。例如，我们可能想修改用户刚刚单击的按钮。在事件处理器代码中，这个引用可以通过 JavaScript 关键字 this 得到。为了将 JavaScript 对象转变成 jQuery 对象，我们将其封装在 jQuery 选择器内：

chapter_02/12_this/script.js *(excerpt)*

```
$('#hideButton').click(function() {
  $(this).hide(); // a curious disappearing button.
});
```

不必重新选择元素，$(this)为与触发事件的元素对话提供了一个更好的方式。

操作在哪？

刚开始的时候，这可能让人有些困惑，因为 jQuery 语句的“操作”部分貌似有多个用途：我们已经看到它被用来运行动画、取值，现在，又用来处理事件！确实如此——它无处不在！

操作的名字通常是其用途的良好线索，但是如果你感到困惑，最好是参考索引。不久之后，你就可以在实用工具中将事件处理器与动画区分开来。

显示隐藏的元素

现在回到我们的任务！客户团队还要求，如果客户错误地关闭了免责声明，需要能够重新取回免责声明。所以我们在 HTML 上再添加一个按钮，这次按钮的 id 是 showButton：

chapter_02/13_revealing/index.html *(excerpt)*

```
<input type="button" id="showButton" value="show" />
```

我们还要在脚本文件中添加另一条 jQuery 语句，在单击 show 按钮时显示免责声明：

chapter_02/13_revealing/script.js *(excerpt)*

```
$('#showButton').click(function() {
  $('#disclaimer').show();
});
```

切换元素

用独立的按钮隐藏和显示免责声明，对宝贵的屏幕空间似乎是种浪费。如果用一个按钮执行两个任务应该更好：免责声明可见时负责隐藏它，免责声明隐藏时则负责显示它。

一种做法是，检查元素处于可见状态还是非可见，然后隐藏或显示。我们将删除以前的按钮，添加下面这个更好的新按钮：

chapter_02/14_toggle_1/index.html *(excerpt)*

```
<input type="button" id="toggleButton" value="toggle" />
```

在单击这个按钮时，我们检查免责声明的状态，看应该显示还是隐藏：

chapter_02/14_toggle_1/script.js *(excerpt)*

```
$('#toggleButton').click(function() {
  if ($('#disclaimer').is(':visible')) {
    $('#disclaimer').hide();
  } else {
    $('#disclaimer').show();
  }
});
```

这次引入了 is 操作。is 接受我们传递给 jQuery 函数的任何选择器，检查它们与调用它的元素是否匹配。在这个示例中，我们检查选中的#disclaimer 是否也被伪选择器:visible 选中。也可以用它来检查其他属性，例如，某个选择是 form 还是 div，选择是否启用等。

if 语句

如果你在编程方面完全是新手（即以前只使用过 HTML 和 CSS），那么面对大块的代码可能让你非常头晕。但不要担心，实际非常简单：

```
if (condition) {
  // this part happens if the condition is true
} else {
  // this part happens if the condition is false
}
```

条件可以是任何事情，只要 JavaScript 能够计算出真（true）或假（false）就可以。这类结构在任何类型的编程中都极为常见，且在本书后续部分将大量使用。如果对此感觉不适应，那么最佳的学习方式就是多做练习：尝试使用 jQuery 的 is 操作编写不同的 if/else 块，就像上面那个一样。不用多少时间，你就会游刃有余！

根据元素与选择器的匹配情况，is 会返回 true 或 false。对于上面的示例，我们会在元素处于隐藏状态时显示它，在处于可视状态时隐藏它。这类逻辑——在两个状态间翻转——称为**切换（toggle）**，是一个非常有用的构造。

元素两个状态之间的切换操作如此普遍，所以许多 jQuery 函数都有支持切换的版本。show/hide 的切换版本就称为 toggle，用法如下所示：

chapter_02/15_toggle_2/script.js *(excerpt)*

```
$('#toggleButton').click(function() {
  $('#disclaimer').toggle();
});
```

每次单击按钮时，元素就会在可视和隐藏两个状态之间切换。

清理旧代码

我们现在仅添加了另外一个事件处理器来切换按钮，因此需要保证删除以前的事件处理器代码（包含 if/else 语句）；否则两个代码都会运行！

对用户来说 toggle 按钮标签可能标识不清，如果能更换成一个更有用的单词，可能会更好。如果想让按钮文本也随之切换，该怎么办？这在使用 jQuery 时是常有的情况，所以有几种方式可以解决这个问题。下面是其中一种：

chapter_02/16_toggle_text/script.js *(excerpt)*

```
$('#toggleButton').click(function() {
  $('#disclaimer').toggle();
  if ($('#disclaimer').is(':visible')) {
    $(this).val('Hide');
  } else {
  $(this).val('Show');
  }
});
```

这个代码中有很多新东西。多数细节稍后再介绍，现在只大致看一下，看你自己是否能弄明白。（提示：请记住选择器$(this)引用的是触发该事件的元素；在这个示例中是按钮。）

2.4.2 渐进增强

我们的免责声明功能工作得很完美，无疑会打动我们的客户。但是，我们的解决方案中还有一个细微的方面应该注意：如果用户使用不支持 JavaScript 的浏览器访问我们的网站，他们只会在页面上看到一个按钮，当他单击按钮时，什么作用也没有。这会让客户倍感困惑，甚至会放弃我们的网站。

“不支持 JavaScript？”你可能对此嗤之以鼻，“哪个浏览器不能运行 JavaScript 啊？！”

不使用 JavaScript 的浏览网页的人数可能超出你的想象：使用非常老式的计算机或者功能有限的设备（如一些手机）的用户、从受限的企业内网上网的用户，以及那些认为 JavaScript 是不必要的安全风险而选择禁用的人。

根据网站的人口统计情况，最多会有 3%的用户没有 JavaScript 能力[①]，如果你有一百万名用户，就想当于有 3 万名客户被你拒之门外！对此问题的解决方案是向这些用户提供一个基本可以接受的体验，而对其他人提供更丰富的体验。这种做法称为**渐进增强。**

对于我们的免责声明功能来说，可以做出如下妥协：希望所有客户都能看到免责声明，所以将它放在 HTML 内。然后，我们为支持 JavaScript 的用户添加隐藏免责声明的能力。这就是说，对于不能使用 JavaScript 的用户，我们选择不显示“显示/隐藏”按钮。

一种实现方法是用 CSS 隐藏按钮，只通过 jQuery 的 css 语句显示它。另一种方法是通过 jQuery 将按钮添加到页面；这样，只有支持 JavaScript 的用户才能看到按钮。非常完美！

2.4.3 添加新元素

迄今为止，我们已经见识过用于选择的 jQuery 函数功能，除此之外还有另外一个函数同样重要，这就是：创建新元素。实际上，放在 jQuery 函数内的任何有效的 HTML 字符串都会被创建，并准备就绪，可以加到页面上。下面创建一个简单的段落元素：

```
$('<p>A new paragraph!</p>')
```

在这个代码中，jQuery 执行多个有用的操作：它将字符串中的 HTML 解析成一个 DOM 片断，选中这个片断，就像一个普通的 jQuery 选择器的做法一样。这意味着它立即就绪，可以做进一步的 jQuery 处理。例如，要向新创建的元素添加一个 class，只要这样写：

```
$('<p>A new paragraph!</p>').addClass('new');
```

新段落现在被赋予名为 new 的 CSS class。用这种方法，可以通过 jQuery 创建任何新元素，而不必在 HTML 标签中定义它们。这样，我们就可以实现页面渐进增强的目标。

① http://developer.yahoo.com/blogs/ydn/posts/2010/10/how-many-users-have-javascript-disabled/

innerHTML

从内部机制上讲，在解析 HTML 字符串时，先创建一个简单的元素（例如 ），然后将 的 innerHTML 属性设为所提供的标签。有些传递进来的内容转换起来不这么容易，所以最好保持 HTML 片断尽可能简单。

创建新元素之后，需要有种方法将它插入页面中需要的地方。有多个 jQuery 函数可以用于这个目的。我们首先要介绍的是 insertAfter 函数。insertAfter 接受当前的 jQuery 选择（在本示例中是指新创建的元素），然后将它插到另一个选中元素的后面，另一个元素作为参数传递给这个函数。

下面的例子以最简单的方式显示了它是如何工作的。使用 jQuery 创建切换按钮的方法如下：

chapter_02/17_insert_after/script.js *(excerpt)*

```
$('<input type="button" value="toggle" id="toggleButton">')
  .insertAfter('#disclaimer');
$('#toggleButton').click(function() {
  $('#disclaimer').toggle();
});
```

如图 2.5 所示，按钮在页面中插在免责声明之后，就像是在 HTML 文件中放在那里一样。

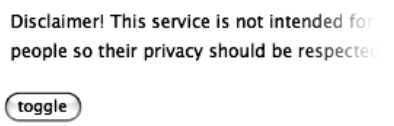

图 2.5 用 jQuery 创建和插入的按钮

insertAfter 函数将新元素作为兄弟元素直接添加在免责声明元素（disclaimer）之后。如果希望按钮出现在 disclaimer 元素之前，可以将新元素放在 disclaimer 之前的元素的后面，并使用 insertAfter；或者，更合理地使用 insertBefore 方法。insertBefore 也会将新元素作为现有元素的兄弟元素，但会将其放在现有元素之前：

chapter_02/18_insert_before/script.js *(excerpt)*

```
$('<input type="button" value="toggle" id="toggleButton">')
  .insertBefore('#disclaimer');
```

快速复习：当我们讲到 DOM 的时候，兄弟元素是指 DOM 层次结构中处于同一级别上的元素。如果一个 中包含两个元素，这两个元素就是兄弟元素。

如果想将新元素作为现有元素的子元素加入（即新元素出现在现有元素的内部），可以使用 prependTo 或 appendTo 函数：

chapter_02/19_prepend_append/script.js *(excerpt)*

```
$('<strong>START!</strong>').prependTo('#disclaimer');
$('<strong>END!</strong>').appendTo('#disclaimer');
```

如图 2.6 所示，两个新元素分别添加到免责声明的<p>的开始和末尾，不在之前，也不在之后。插入和删除元素还有更多操作，但在这轮改动中还不需要，我们将在后面介绍它们。

START! Disclaimer! This service is not intended for the those with criminal intent. Celebrities are kind of like people so their privacy should be respected. **END!**

图 2.6 prependTo 和 appendTo 的效果

插入多个元素

对于每个与选择器匹配的元素，都会插入一次新项目。例如，如果选择器与每个段落标签匹配，insertAfter 操作会在每个段落标签之后插入新元素。这真是一个相当强大的函数！

避免重复

如果你一路走到这里，那么将额外送给你条创建元素的高级技巧。当在这里、那里都插入了一些 HTML 后，你可能厌烦了创建元素、选择元素、在元素上添加事件处理器的重复过程。一种解决办法就是使用高级的创建语法，在创建元素的时候，直接将信息添加到新元素：

```
$('<div>', {
    id: 'specialButton',
    text: 'Click Me!',
    click: function(){
        alert("Advanced jQuery!")
    }
}).insertBefore('#disclaimer');
```

在使用这个高级语法时，还是同往常一样创建元素（在示例中是个简单的 ），但是要使用键/值对对象的形式添加元素属性，就像本章开始设置 CSS 样式时一样。属性中包括属于 HTML 元素的普通属性，但也可以直接在创建时添加事件处理器。

2.4.4 删除现有元素

我们告诉客户团队，会有 3%的用户可能没有 JavaScript 能力，因此会错过我们正在构建的一些高级功能。客户团队的成员可能会问我们能否添加一条消息，对此类用户推荐使用 JavaScript。显然，对于有 JavaScript 的用户应该隐藏这条消息。

正确的解决方案是将消息放在专为这个目的而提供的<noscript>标签内。另一方面，本着自我教育的目的，打破这条规则对我们来说是件好事，因为这看起来是一个学习如何用 jQuery 从页面上删除 HTML 元素的完美机会。我们将消息放在 HTML 内，然后用 jQuery 删除它，这样，只有不支持 JavaScript 的访客才会看到它。

那就让我们行动，给 HTML 页面新增一条警告吧：

chapter_02/20_removing_elements/index.html *(excerpt)*

```
<p id="no-script">
  We recommend that you have JavaScript enabled!
</p>
```

现在我们需要运行代码从页面上删除这个元素。如果用户禁用了 JavaScript，我们的 jQuery 语句

不会运行，于是这条消息会保留在屏幕上。要用 jQuery 删除元素，先要用选择器选择它们（同往常一样），然后调用 remove 方法：

chapter_02/20_removing_elements/script.js *(excerpt)*

```
$('#no-script').remove();
```

remove 操作会从 DOM 中删除选中的全部元素，还会删除这些元素上附加的任何事件处理器或数据。remove 操作不需要任何参数，不过可以指定一个表达式来进一步细化选择。请试试下面的示例：

chapter_02/21_removing_with_selector/script.js *(excerpt)*

```
$('#celebs tr').remove(':contains("Singer")');
```

这个代码没有删除#celebs 中的每一行（tr），而只是删除其中包含文本“Singer”的那些行。下一章在学习一些高级特效时，这会很方便。

多亏了这些变化，我们的页面对 3%潜在的不支持 JavaScript 的用户也能表现良好，对其余 97%的用户表现得更好。这是一个非常简单的渐进增强示例，通过它可以很好地理解这个概念的基本想法：不要用 jQuery 作为用户界面的基础，要用它给已经正常工作的体验锦上添花。这样做的话，可以确保不会遗漏任何一个人。

本着让示例代码短小、精悍的原则，我们不再深入讲解这个主题。但你自己应该做些研究：这可以让你成为更好的 Web 开发人员。

2.4.5 修改内容

现在，我们对元素想做什么都可以：显示它们、隐藏它们、添加新元素、删除旧元素、按我们喜欢的样子设置它们的样式……但是，如果我们想修改元素的内容呢？jQuery 提供了两种专门做这件事的方法：text 和 html。

text 和 html 的操作类似，都用来设置选中元素的内容。只需要给函数传递一个字符串：

chapter_02/22_modifying_content/script.js *(excerpt)*

```
$('p').html('good bye, cruel paragraphs!');
$('h2').text('All your titles are belong to us');
```

在上述两个示例中，匹配元素的内容将改成我们提供的字符串：页面中的每个段落和<h2>标签都会被新内容覆盖。如果我们在内容字符串中添加一些 HTML，就会看到 text 和 html 之间的区别：

chapter_02/23_text_vs_html/script.js *(excerpt)*

```
$('p').html('<strong>Warning!</strong> Text has been replaced … ');
$('h2').text('<strong>Warning!</strong> Title elements can be …');
```

在上述示例中，段落中会包含一些粗体文本，但只有<h2>标签才会原样包含整个字符串内容，包括标签。修改内容时使用哪种操作取决于你的需求：text 适用于纯文本，html 适用于 HTML。

你可能想知道：“这些新操作只能设置内容么？”到了现在，你应该不会感到奇怪，我们还能用相同的操作从 jQuery 选择中获取内容：

chapter_02/24_get_content/script.js *(excerpt)*

```
alert($('h2:first').text());
```

我们使用没有参数的 text 操作返回页面上的第一个<h2>标签的文本内容（“欢迎!”）。与其他获取值的操作类似，它对条件语句特别有用，对于向用户互动添加一些基本信息也非常有用。

2.4.6 基本动画：用 Flair 隐藏和显示

虽然这些显示、隐藏、变化很有用，但在视觉效果上给人的印象不深。现在该介绍一些更高级点的 jQuery 技术了，或者说：动画。

核心 jQuery 库包含少量特效，可以用来给我们的页面加点料。如果这些满足不了你，你可以去 jQuery 插件库中寻找，在那里可以找到成百上千更加疯狂的特效。

保持合理

在 Web 上处理特效和动画时，最好运用自己的良好品味。记请住，<blink>标签（的闪烁效果）曾一度被认为完全可以接受!

淡入、淡出

jQuery 中最普遍（而且不过时）的特效之一，就是内置的渐变效果。淡入效果最简单的形式，只要将 show 用 fadeIn 代替，或将 hide 用 fadeOut 代替即可：

chapter_02/25_fade_in_out/script.js *(excerpt)*

```
$('#hideButton').click(function() {
  $('#disclaimer').fadeOut();
});
```

还有几个可选参数可以用来修改特效，第一个用来控制渐变完成需要的时间。许多 jQuery 特效和动画都接受时间参数，时间可以用字符串或整数形式传入。

我们可以以下面预定义的单词，以字符串形式指定时间跨度：slow（慢）、fast（快）、normal（正常）。例如：fadeIn('fast')。如果希望对动画的时长做更细致的控制，可以以毫秒为单位指定时间，例如：fadeIn(1000)。

切换特效和动画

jQuery 还包含一个 fadeToggle 操作，在元素隐藏的时候，将元素淡入，在元素显示的时候，将元素淡出。另外，我们之前讲解的 toggle 操作也有一些我们想象之外的技巧。如果我们给它传递一个时间跨度参数，就会发现 toggle 具有动画的能力：

chapter_02/26_toggle_fade/script.js *(excerpt)*

```
$('#toggleButton').click(function() {
  $('#disclaimer').toggle('slow');
});
```

可以看到整个元素的宽度、高度、不透明度都动了起来。如果这个动画效果对你有点多，还有另外一个核心的 jQuery 动画效果也包含内置的切换操作：滑动。

滑动会让元素在视图中慢进慢出，就像从暗格中滑出一样，它的实现方式与渐进相同，不过采

用的是 slideDown（滑下）、slideUp（滑上）和 slideToggle（滑动切换）三个操作。同渐进效果一样，也可以指定时间跨度：

chapter_02/27_slide_toggle/script.js *(excerpt)*

```
$('#toggleButton').click(function() {
  $('#disclaimer').slideToggle('slow');
});
```

2.4.7 回调函数

许多特效（包括滑动和渐变特效）都接受一个特殊的参数：**回调**。回调可以指定在特效完成之后需要运行的代码，执行特效完成之后需要做的操作。在下面的示例中，在滑动完成时，将运行回调代码：

chapter_02/28_callback_functions/script.js *(excerpt)*

```
$('#disclaimer').slideToggle('slow', function() {
  alert('The slide has finished sliding!')
});
```

回调作为第二个参数传递给特效操作，作为一个匿名函数，传递的方式与我们提供函数作为事件处理器参数的方式基本相同。

匿名函数

在 JavaScript 中，内联定义的函数（如回调和事件处理器）称为**匿名函数**。之所以称其为“匿名”，就是因为它们没有名称！匿名函数用在只需要函数在一个特定位置运行的时候。

在使用匿名函数的地方，也可以传递进一个函数名称，然后在其他地方定义函数。如果同一函数需要在多个地方调用，则最好这样做。在上述示例这种简单的情况下，这种做法会让代码不好理解，所以在这种情况下，我们坚持使用匿名函数。

现在让回调函数做些实际的事情。如果我们想在免责声明已经滑出视野的时候隐藏按钮，可以这样做：

chapter_02/29_callback_functions_2/script.js *(excerpt)*

```
$('#disclaimer').slideUp('slow', function() {
  $('#hideButton').fadeOut();
});
```

免责声明会向上滑动，在动画完成之后，按钮会从视野中淡出。

2.5 几条小技巧

现在我们已经解决了几项客户事项列表中高优先级的请求，下面我们再卖弄一些，给网站加些

额外的炫耀效果。我们将用迄今为止所学的知识增加一些特效和可视的突出效果，并将介绍一些新的构造和操作，所以如果您刚刚接触 jQuery，后续的操作值得您一一练习。

2.5.1 鼠标悬停时突出显示

客户真的很关心斑马条纹效果的可用性。客户团队要求，除了改变行的颜色外，当用户的鼠标经过表格（<table>）时，还应该有一些额外的突出效果。

我们可以通过给<table>元素添加事件处理器，处理鼠标经过（mouseover）和鼠标离开（mouseout）事件，来实现这个效果。然后我们可以在鼠标悬停的元素上添加或删除包含背景颜色的 CSS class。这与使用普通旧版的 JavaScript 的做法基本相同。

chapter_02/30_hover_highlight/script.js *(excerpt)*

```
$('#celebs tbody  tr').mouseover(function() {
  $(this).addClass('zebraHover');
});
$('#celebs tbody  tr').mouseout(function() {
  $(this).removeClass('zebraHover');
});
```

请记住，$(this)是指选中的对象，所以当用户鼠标徘徊于表格上的时候，我们就会针对每一行添加和删除 zebraHover 类。现在我们只需要在 CSS 文件中添加一条样式规则：

chapter_02/30_hover_highlight/zebra.css *(excerpt)*

```
tr.zebraHover {
  background-color: #FFFACD;
}
```

在浏览器中进行测试，可以看到效果有多好。但是，有一种更简单的方式可以实现同样的效果：jQuery 包含一个 hover 操作，它将 mouseover 和 mouseout 组合到了一个事件处理器内：

chapter_02/31_hover_action/script.js *(excerpt)*

```
$('#celebs tbody tr').hover(function() {
  $(this).addClass('zebraHover');
}, function() {
  $(this).removeClass('zebraHover');
});
```

有没有注意到 hover 事件处理器的一些奇怪的地方？它要求两个函数作为参数：一个负责处理 mouseover 事件，另一个负责处理 mouseout 事件。

回调知多少？

有两个 jQuery 事件处理器要求不同数量的函数。例如，toggle 事件处理器可以接受任意数量的函数，每次事件触发时，它都在每个回调上逐一循环。hover 操作可以接受一个或两个函数：如果传递一个函数，那么鼠标经过和鼠标离开两个事件都会调用它。阅读每个操作的文档有助于发现可能漏掉的炫酷功能。

在适应了添加和删除 class 属性之后，是时候该学习另一个很有帮助的与类有关的操作：toggleClass。顾名思义，它的作用是切换类。它非常有用，如果元素没有某个类，就加上它，如果已经有这个类，则删除它。

例如，假设我们希望用户能够在表格中选择多行。在一个表格行上单击一次应该将其高亮显示，再次单击则取消高亮。用新的 jQuery 技能可以很容易实现这一功能：

chapter_02/32_toggle_class/script.js *(excerpt)*

```
$('#celebs tbody tr').click(function() {
  $(this).toggleClass('zebraHover');
});
```

现在单击表格行，效果很酷吧？toggleClass 接受也可以接受多个类名，中间以空格分隔。全部类都会根据当前的状态进行切换。

2.5.2 小道消息揭秘器

StarTrackr!网站最新的新闻区，提供一批名人最新的小道消息。小道消息才是网站的真实卖点：多数用户每天回来，就是为了得到最新更新。客户很愿意在这个小道消息形成的炒作基础上添油加醋，因此客户团队向我们请求帮助。我们建议加一个小道消息揭秘器，让用户在单击答案之前，先猜一猜是关于哪个名人的小道消息。

这类功能对于包含电影评论的网站来说，会起到锦上添花的效果。你可以隐藏那部分泄漏影片细节的部分，只有用户看过电影之后才允许他们揭开它们。

要设置小道消息揭秘器，我们需要在网站的新闻区增加一个新元素。默认隐藏的“秘密”都封装在 span 元素内，并附加一个名为 spolier 的 class：

chapter_02/33_spoiler_revealer/index.html *(excerpt)*

```
Who lost their recording contract today?
<span class='spoiler'>The Zaxntines!</span>
```

下面将脚本需要完成的操作分解说明：首先，我们隐藏答案，添加一个新元素，使它们可以在用户需要的时候显示。在单击这个元素时，我们揭开答案。隐藏？添加？处理单击？这些我们都知道该如何处理：

chapter_02/33_spoiler_revealer/script.js *(excerpt)*

```
$('.spoiler').hide();
$('<input type="button" class="revealer" value="Tell Me!"/>')
  .insertBefore('.spoiler');
$('.revealer').click(function(){
  $(this).hide();
  $(this).next().fadeIn();
});
```

这里的代码很多，有些是新内容，但是如果一次只读一行，就会看懂它们的意思。首先，我们立即隐藏全部 spoiler 元素，然后用 insertBefore 操作在每个 spoiler 元素前添加一个新按钮。这时，页

面会显示新的“Tell Me!”按钮，原来属于 spoiler 类的 span 都将被隐藏。

接下来，选择刚刚添加的新按钮，在上面附加 click 事件处理器。单击其中任何一个按钮，都会显示新的 revealer 元素（由$(this)指向），淡出下一个 spoiler 元素。next 操作是个新操作，它用来对 DOM 进行遍历，访问元素的下一个兄弟元素（即同一容器内的下一元素）。

如果查看修改后的 DOM（如图 2.7 所示），可以看到 spoiler 的 span 是揭秘按钮的下一个元素。next 操作只是将选择移到这个元素。jQuery 还提供了 prev 操作，它的作用是将选择转移到当前选择元素前面的一个元素。

```
▼ <p>
    Who lost their recording contract today?
    <input class="revealer" type="button" value="Tell Me!"/>
    <span class="spoiler" style="display: none;">The Zaxntines!</span>
  </p>
```

图 2.7　修改后的 DOM

实际上，jQuery 有十几个不同的操作可以用来在 DOM 中移动；prev 和 next 只是特别有用的两个而已。在本书后续的学习过程中，将看到更多操作，您也可以参考 jQuery API 的文档[①]查看全部操作。

由于隐藏的 spoiler 元素现在都在 jQuery 的控制下，所以我们只要调用 fadeIn 就能用一个平滑的过渡显示名人的小道消息。

2.6　到下一课之前

在开始两章我们已经介绍了这么多内容，所以你现在应该了解了 jQuery 的结构和威力。如果幸运的话，可能你已经计划在自己当前的项目中使用 jQuery。如果是这样，请大胆使用吧！不论是用它解决一个恶性问题，还是仅想四处加点小花絮，身体力行是巩固所学知识的最佳途径。

只有一个小小的警告，请记住一句谚语：“当你只有锤子这一个工具时，每件事看起来都像是钉子”。jQuery 是个很棒的工具，但在某些情况下可能并不适用。如果某个问题只要简单地修改 CSS 或 HTML 就能更好地解决，就应该那么办。当然，在学习期间，可以用 jQuery 做任何事，只要记住，到了将你的技能投入实践的时候，就应该坚持用最好的工具来完成工作。

在后面的课程中，我们将用这里学习的简单的 jQuery 构造块构建一些你可以立即开始使用的非常酷的窗口小部件、特效以及用户交互。

① http://docs.jquery.com/Traversing