第一章　认识一下 jQuery

随着 JavaScript 的兴起，一系列 JavaScript 库也蓬勃发展起来。从早期的 Prototype、Dojo 到 2006 年的 jQuery，再到 2007 年 ExtJs。可以发现，互联网正在掀起一场 JavaScript 风暴。在这场风暴中，jQuery 以其独特优雅的姿态，始终处于这场风暴的中心，受到越来越多的人的追捧。

1.1　jQuery 简介

jQuery 是继 Prototype 之后又一个优秀的 JavaScript 库，它由 John Resig 创建于 2006 年 1 月。它简化了遍历 HTML 文档、操作 DOM、处理事件、执行动画和 Ajax 的操作。它独特而又优雅的代码风格改变了 JavaScript 程序员编写程序的设计方式和思路。

不管你是网页设计师、后台开发者、业余爱好者还是项目管理者，也不管你是 JavaScript 初学者还是 JavaScript 高手，你都有很多理由去学习 jQuery，因为它是面向任何人的。

1.2　加入 jQuery

1.2.1　JavaScript 简介

JavaScript 是为了适应动态网页制作的需要而诞生的一种编程语言。它是由 Netscape 公司开发的一种脚本语言（scripting language）。JavaScript 的出现使得网页和用户之间实现了一种实时性的、动态的、交互性的关系，使网页包含更多活跃的元素和更加精彩的内容。然而，几乎每个 Web 开发人员都曾有过诅咒 JavaScript 的经历。这个备受争议的语言受累于其复杂的称为文档对象模型 (DOM)的编程模型、糟糕的实现和调试工具以及不一致的浏览器实现。直到现在，很多开发人员还认为 JavaScript 是一门令人厌恶的语言。

随着 WEB2.0 的兴起，作为广泛应用于 Web 开发的脚本语言，JavaScript 开始日益重要起来，JavaScript 的复苏使一些业界领袖人物也不得不开始重新审视这种编程语言。诸如 Ajax (Asynchronous JavaScript + XML) 这样的编程技术让 Web 网页更加迷人，而完整的 Web 开发框架，比如 Apache Cocoon（一种使用而且充分利用了 XML 强大功能的发布框架），则让 JavaScript 的应用越来越多，使其不止限于是一种用于制作 Web 页面的简单脚本。

1.2.2　JavaScript 库作用及对比

JavaScript 库能帮助你轻松建立有高难度互动的 web2.0 特性的富客户端页面，它带有很

多预定义的对象和实用函数。下面是目前几种流行的 JavaScript 库的介绍和对比：

Prototype (http://www.prototypejs.org/)

图 1-1

这个算是最早成型的 JavaScript 库之一，它对 JavaScript 的内置对象（如 String 对象、Array 对象等）做了大量的扩展。现在很多项目中都使用它，但这很大程度上是由于以前项目用了，现在不得不继续沿用。这个库可以看做是把很多好的有用的 JavaScript 的方法组合在一起的一个 JS 库，你甚至可以在你需要的时候随时将其中的几段代码抽出来放进自己的脚本里。但也正是由于它成型年代早，在整体对于面向对象的编程思想把握上并不是很到位，导致了结构的松散。

Dojo (http://dojotoolkit.org/)

图 1-2

Dojo 强大之处在于它提供了很多其他 JavaScript 库所没有提供的功能。比如离线存储的 API、生成图标的组件、基于 SVG/VML 的适量图形库、Comet 支持等等很多优点。是非常适合企业级应用的一款 JavaScript 库。同时它也得到了一些大公司的支持，如 IBM、SUN、BEA 等。同时它的缺点也是很显著的：学习曲线陡，文档不齐全，最要命的就是 API 不稳定，每次升级都可能导致已有的程序失效。但从它的 1.0 版以后看起来，情况有所好转。未来是个很有潜力的库。

YUI (http://developer.yahoo.com/yui/)

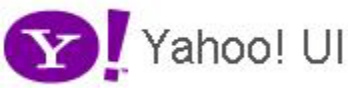

图 1-3

这套库是 Yahoo 打造出来的 JavaScript 库。全名是 The Yahoo! User Interface Library 。它提供了一些比较丰富的关于 DOM 操作、Ajax 应用等一系列的封装。同时它还包括了几个核心的 CSS 等。是一套比较齐全完备的富交互网页程序工具集。本身的文档极其完备，以至于很少看到第三方写相应的文章。本身的代码编写也非常的规范，扩展性也很不错的一套库。

ExtJS (http://www.extjs.com/)

图 1-4

ExtJS，也常简称 Ext。原本是对 YUI 的一个扩展，主要是用于创建前端用户界面，它提供了极其丰富的组件。如今已经发展到可以利用包括 jQuery 在内的多种 JavaScript 框架作为基础库，而 Ext 作为界面的扩展库来使用。但由于侧重于界面，所以本身比较臃肿，不压缩的话文件上兆(MB)，所以使用之前请先考虑。请注意，Ext 并非完全免费的，如果用于商

业用途的话，是要付费获得授权许可的。

MooTools (http://mootools.net/)

图 1-5

这是一套轻量级的 JavaScript 库，是一个简洁，模块化，面向对象的 JavaScript 框架。其语法几乎跟 Prototype 一样，但却提供了更强大的功能和更好的扩展性及兼容性。其模块化思想非常优秀，核心代码只有 8K。用到什么模块可即时导入，即使是完整版也不超过 160K。还有它彻底完全的面向对象的编程思想，语法简洁直观，文档完善。

jQuery (http://jquery.com)

图 1-6

本书的重点 jQuery 也同样是一个轻量级的库，它拥有强大的选择器，出色的 DOM 操作，可靠的事件处理、出色的兼容性，以及链式操作等等，这些优点吸引了一批批 JavaScript 开发者去学习它、研究它。下一节将具体介绍 jQuery 的优势。

1.2.3 jQuery 的优势

jQuery 强调的理念是写的少，做的多(write less, do more)。其独特的选择器、链式的 DOM 操作方式、事件绑定机制、封装完善的 Ajax 都是其它 JavaScript 库望尘莫及的。

（1）轻量级。jQuery 非常轻巧，采用 Dean Edwards 的 Packer(http://dean.edwards.name/packer/)压缩后，只有不到 30KB 的大小，如果服务器端启用 gzip 压缩后，甚至只有 16KB 的大小！

（2）强大的选择器。jQuery 可以让操作者使用从 CSS 1 到 CSS 3 几乎所有的选择器，以及 jQuery 独创的高级而复杂的选择器。如果你需要，还可以加入插件使其支持 XPath 选择器！下一章我们将为你详细讲解 jQuery 中强大的选择器。

（3）出色的 DOM 操作的封装。jQuery 封装了大量常用 DOM 操作，使你编写 DOM 操作相关程序的时候能够得心应手，优雅的完成各种原本非常复杂的操作，让 JavaScript 新手也能写出出色的程序。第三章将为你重点介绍 jQuery 中优雅的 DOM 操作。

（4）可靠的事件处理机制。jQuery 的事件处理机制吸取了 JavaScript 专家 Dean Edwards 编写的事件处理函数的精华，使得 jQuery 处理事件绑定的时候相当的可靠。在预留退路(graceful degradation)方面，jQuery 也做的非常不错。第四章将为你重点介绍 jQuery 中的事件处理。

（5）完善的 Ajax。jQuery 将所有的 Ajax 操作封装到一个函数$.ajax 里，使得我们处理 Ajax 的时候能够专心处理业务逻辑而无需关心复杂的浏览器兼容性和 XMLHttpRequest 对象的创建和使用的问题。第六章将为你重点介绍 jQuery 中的 Ajax 处理。

（6）不污染顶级变量。jQuery 只建立一个名为 jQuery 的对象，其所有的方法都在这个对象之下。另外的一个别名$也是可以随时交出控制权的。绝对不会污染其它的对象！

（7）出色的浏览器兼容性。作为一个流行的 JavaScript 库，浏览器的兼容性自然是必

须具备的条件之一。jQuery 能够在 IE 6.0+、FF 2+、Safari 2.0+和 Opera 9.0+下正常运行。同时修复了一些浏览器之间的差异。使你不用在开展项目前忙于建立一个浏览器兼容库而焦头烂额。

（8）链式操作方式。jQuery 中最有特色的莫过于它的链式操作方式——即对发生在同一个 jQuery 对象上的一组动作，可直接连写而无需重复获取对象。这一点使 jQuery 的代码无比优雅。请注意，在章节 1.3.3 中，我们将要讨论相应代码风格的问题。

（9）行为层与结构层的分离。开发者不需要再去 html 调用事件，而是直接使用 jQuery 选择器选中元素，然后直接给元素添加事件。

（10）丰富的插件支持。任何事物的壮大，如果没有很多人的支持，是永远发展不起来的。jQuery 的易扩展性，吸引了来自全球的开发者来共同编写 jQuery 的扩展插件。目前已经有超过几百种的官方插件支持。在第七章，我们将介绍目前流行的几款插件并指导大家动手编写自己的插件。

（11）完善的文档。jQuery 的文档是非常丰富的，现阶段多为英文文档，而中文文档较少。当然，很多热爱 jQuery 的团队都在为这个努力，比如图灵教育翻译的《Learning jQuery》。

（12）开源。jQuery 是一个开源的产品，任何人都可以自由的使用。

既然 jQuery 拥有这么多优势，我们有什么理由不去学习它呢？下面我们就开始正式的接触它，使用它。

1.3　编写简单 jQuery 代码

1.3.1 配置 jQuery 环境

- 获取 jQuery:最新版本

  进入 jQuery 的官方网站，网址是：http://jquery.com/。在如图 1-7 所示左下的 Download jQuery 区域，下载最新的 jQuery 库（编者注：目前是 1.2.6 版）文件，本书所有的 jQuery 实例都是基于 1.2.6 版本进行编写。

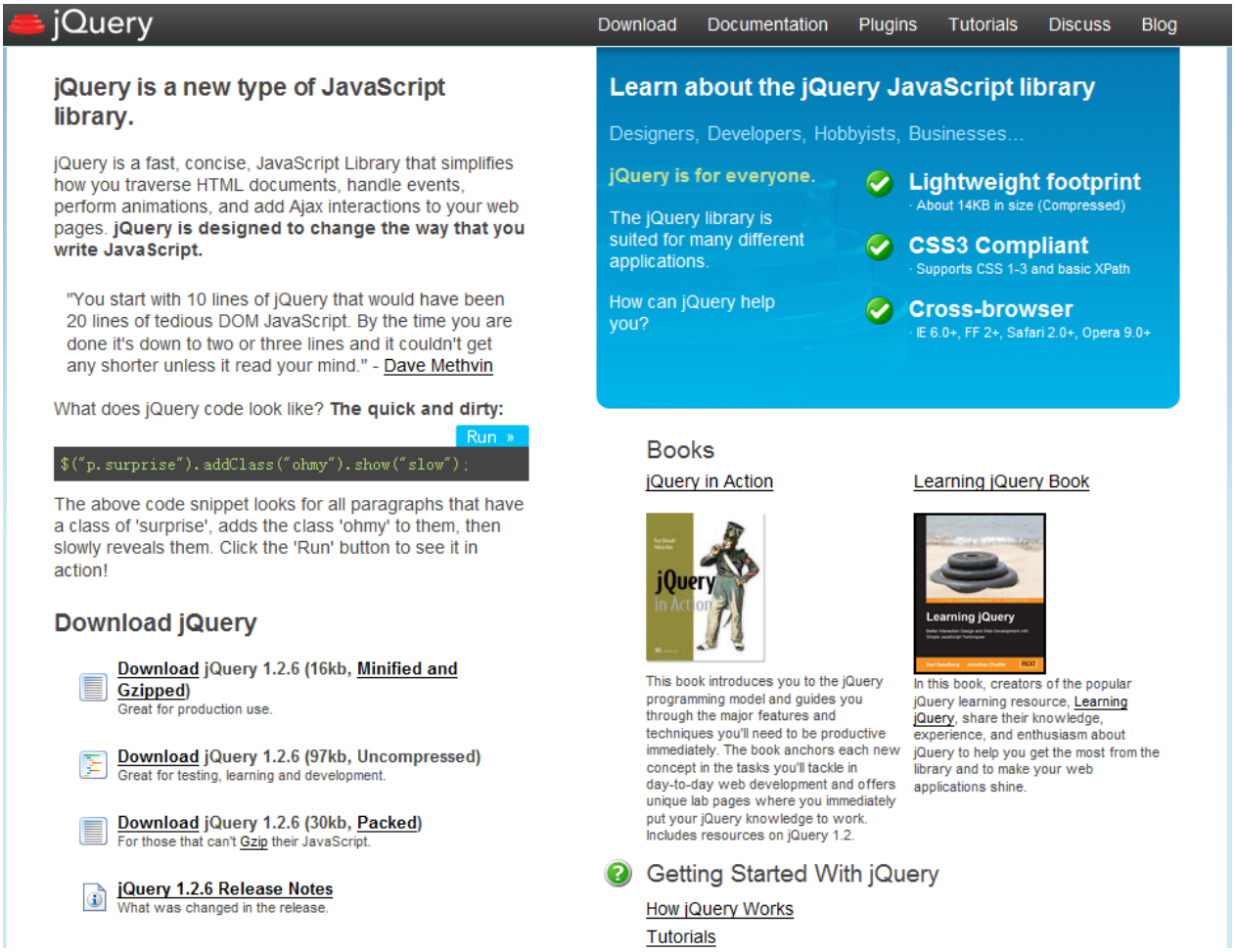

图 1-7　jQuery 官方网站截图

- jQuery:库类型说明

在 Download jQuery 区域，一共有三种类型的 jQuery:库，分别是 jQuery 1.2.6 (16kb, Minified and Gzipped)、jQuery 1.2.6(97kb, Uncompressed)和 jQuery 1.2.6 (30kb, Packed)，区别如表 1-1 所示：

表 1-1　　jQuery 库类型对比表

名称	大小	说明
jQuery 1.2.6 (16kb, Minified and Gzipped)	16kb	经过 gzip 压缩，体积最小，为应用在产品、项目而准备的版本
jQuery 1.2.6 (97kb, Uncompressed)	97kb	完整无压缩版本，为测试，学习和开发而准备的版本
jQuery 1.2.6 (30kb, Packed)	30kb	经过 Packer 压缩，为不支持 gzip 的服务器而准备的版本

表 1-1　几种 jQuery 库类型对比

为统一本书的讲解，建议选择下载 jQuery 1.2.6 (97kb, Uncompressed)版本。

- jQuery 环境配置

jquery-1.2.6.js 下载完毕，将其放置在具体项目目录下即可方便地引用 jQuery 库。

- 在页面中引入 jQuery

本书约定：本书将 jquery-1.2.6.js 放在目录 scripts 下，所提供的 jQuery 例子中为

了方便调试，引用时使用相对路径。在实际项目中，读者可以根据实际需要调整 jQuery 库路径。

如下程序所示，在<head>标签内引入 jQuery 库：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
    <head>
        <!-- 在 head 标签内 引入 jQuery  -->
        <script src="../scripts/jquery-1.2.6.js" type="text/javascript"></script>
    </head>
    <body>
    </body>
</html>
```

注意，在本书的后面所有章节中，如果没有特别说明，jQuery 库都是默认导入的。

1.3.2 编写简单的 jQuery 代码

环境配置好后，你也许迫不及待的想试用一下 jQuery。在我们开始编写第一个 jQuery 程序之前，先明确一点：在 jQuery 库中，$就是 jQuery 的一个简写形式，比如$("#foo")和 jQuery("#foo")是等价的，$.ajax 和 jQuery.ajax 是等价的。如果没有特别说明，程序中的$符号都是 jQuery 的一个简写形式。

下面我们开始编写我们第一个 jQuery 程序：

```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>1-1</title>
         <!-- 引入 jQuery -->
         <script src="../scripts/jquery-1.2.6.js" type="text/javascript"></script>
         <script type="text/javascript">
        $(document).ready(function(){          //等待 dom 元素加载完毕.
            alert("Hello World!");             //弹出一个框。
        });
        </script>
    </head>
    <body>
    </body>
</html>
```

运行后，测试结果如图 1-8 所示：

图 1-8　输出 Hello World!

代码中有一个也许我们从来没用过的东西，那就是：

```
$(document).ready(function(){
......
});
```

那么这段话是什么意思呢？其实它类似于 window.onload，不过跟 window.onload 还是有些区别。

请看下面的表格对比：

表 1-2　　window.onload 和$(document).ready()对比表

	window.onload	$(document).ready()
执行时机	必须等待网页中所有的内容加载完毕后(包括图片)才能执行.	网页中所有 DOM 结构绘制完毕后就执行.

编写个数	不能同时编写多个.
	比如：
	window.onload = function(){
	alert("test1")
	} ;
	window.onload = function(){
	alert("test2")
	} ;
	结果只会输出”test2”。
		能编写多个.
		比如：
		$(document).ready(function(){
		alert("Hello World!");
		});
		$(document).ready(function(){
		alert("Hello again!");
		});
		结果两次都输出。

简化写法	无	$(document).ready(function(){
		...
		});
		可以简写成:
		$(function(){
		...
		});

表格 1-2　window.onload 和$(document).ready()对比

1.3.3 jQuery 代码风格

用 jQuery 写项目很长时间了，期间也看到有朋友抱怨，说 jQuery 的链式操作看上去很优雅，用起来也很舒服，好多操作一行搞定，那种感觉是无与伦比的。但等日后回过头来再去改自己的代码的时候，不由眉头一皱，大量一行搞定的代码，单个看起来很优雅，多了就看起来非常的凌乱，时间久了，自己都不知道自己写了些什么。

举我实际一个项目中的代码为例(一个导航栏)：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>1-3</title>
        <style>
        #menu {
        width:300px;
        }
          .has_children{
        background : #555;
        color :#fff;
        cursor:pointer;
          }
          .highlight{
        color : #fff;
        background : green;
          }
          ul{
        list-style:none;
        padding:0;
          }
          ul li{
        background : #888;
        display : none;
          }
          </style>
    </head>
    <body>
        <div id="menu">
        <ul class="has_children">第 1 章-认识一下 jQuery
            <li>1.1-jQuery 简介</li>
            <li>1.2-加入 jQuery</li>
            <li>1.3-编写 jQuery 代码</li>
            <li>1.4-jQuery 对象和 DOM 对象</li>
            <li>1.5-解决 jQuery 和其它库的冲突</li>
            <li>1.6-小结</li>
        </ul>
        <ul class="has_children">第 2 章-jQuery 之选择器
            <li>2.1-什么是选择器</li>
            <li>2.2-jQuery 选择器的优势</li>
            <li>2.3-jQuery 选择器</li>
            <li>2.4-选择器中带特殊符号的处理</li>
```

```
                <li>2.5-javascript 中的一些对比操作势</li>
                <li>2.6-做一个图片展示效果</li>
                <li>2.7-小结</li>
            </ul>
            <ul class="has_children">第 3 章-jQuery 之 DOM 操作
                <li>3.1-介绍 DOM 操作的重要性</li>
                <li>3.2-jQuery 的 DOM 操作</li>
                <li>3.3-打造一个图片馆</li>
                <li>3.4-小结</li>
            </ul>
            </div>
        </body>
</html>
```

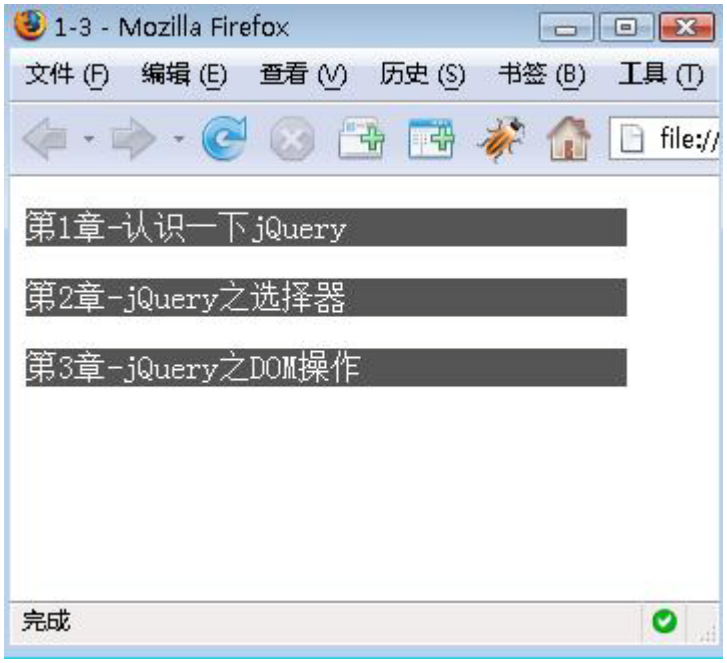

图 1-9 导航栏初始化

项目需求是做一个导航栏，点击不同的章节，显示相应的内容，同时高亮当前选择的章节。

然后我们用 jQuery 实现：

```
$(".has_children").click(function(){
$(this).addClass("highlight").children("li").show().end().siblings().removeClass("highlight").children("li").hide();
});
```

这段代码的作用是，当鼠标点击到 class 中含有.has_children 的元素上的时候，给其添加一个名为 highlight 的 class，然后将其内部 li 子元素都显示出来，并且被点击的.has_children 元素的同辈元素都去掉一个名为 highlight 的 class，以及内部的 li 子元素统统隐藏。

效果如下：

图 1-11 效果 1

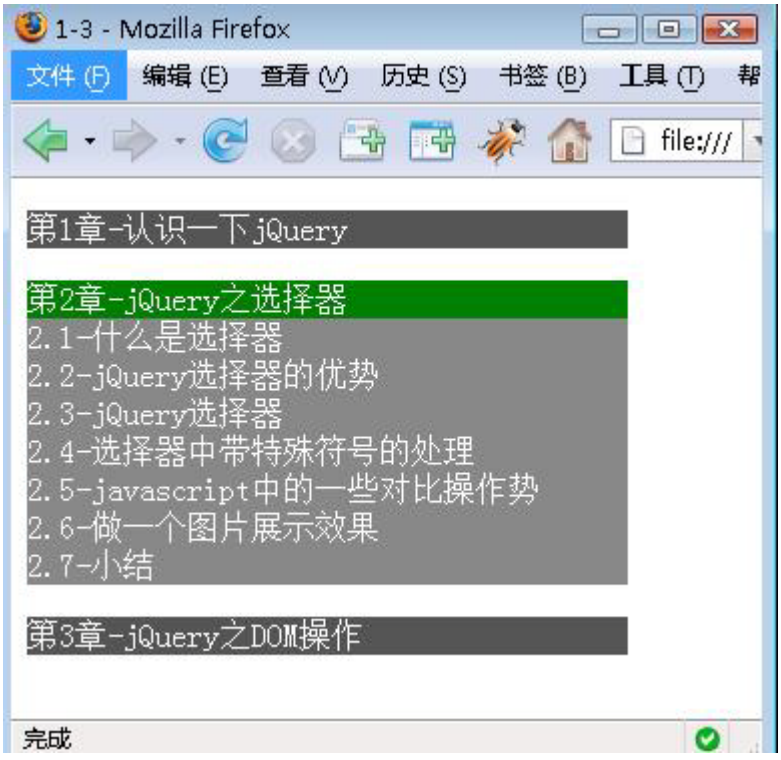

图 1-12 效果 2

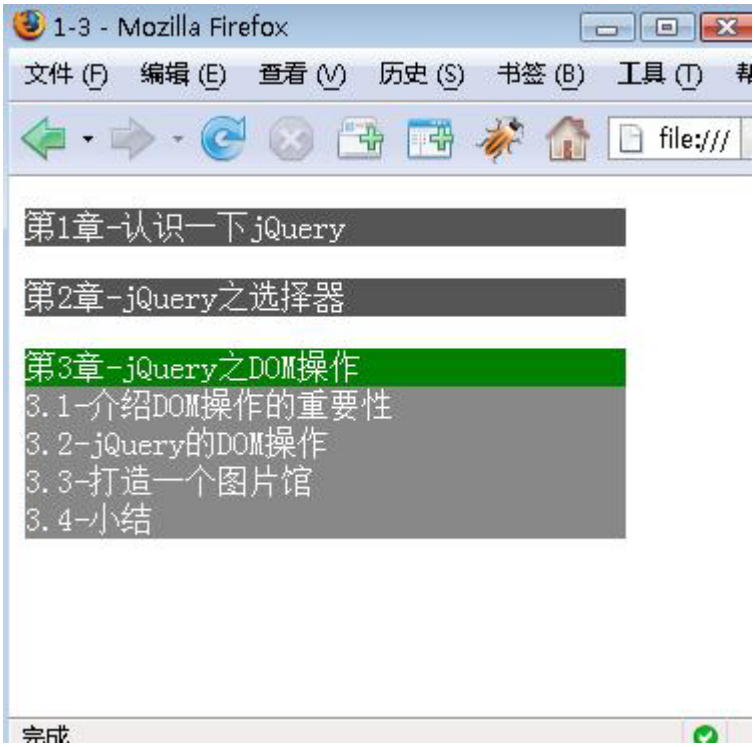

图 1-13 效果 3

看吧，这就是 jQuery 强大的链式操作，一行代码就把导航栏给做出来了。但是，请注意了，这样的写法只是在你自己编写时有畅快淋漓之感，但会给后期维护和团队协作带来无尽的苦难。

所以，我们更推荐的是一种带有适当的格式的代码风格：

```
$(".has_children").click(function(){
    $(this).addClass("highlight")                    //为当前元素增加 highlight 类
        .children("li").show().end()                 //将子节点的 li 元素显示出来并重新定位到上次操作的元素
    .siblings().removeClass("highlight")             //获取元素的兄弟元素，并去掉他们的 highlight 类
        .children("li").hide();                      //将兄弟元素下的 li 元素隐藏
});
```

对吧，是不是看上去好些了？

也许你看了上面的代码可能还是不明白其中的要领，我们在这里总结了 4 种情况：

(1) 对于一个对象不超过 3 个的操作的，可以直接写成一行，例如：

```
$("li").show().unbind("click");
```

(2) 对于多个对象的操作的少量操作，可以每个对象一行，如果涉及到子元素，可以考虑适当的缩进。比如上面提到的代码：

```
$(this).addClass("highlight")
    .children("li").show().end()
.siblings().removeClass("highlight")
    .children("li").hide();
```

(3) 对于一个对象的较多操作，建议每行写一个，或者按功能块区分。例如：

```
$(this).removeClass("mouseout")
    .addClass("mouseover")
    .stop()
    .fadeTo("fast",0.6)
    .fadeTo("fast",1)
```

```
    .unbind("click")
    .click(function(){
    // do something ….
    });
```

对于上面的代码，如果你嫌代码行数过多，那以功能块来换行也是个不错的主意。前两个是对 class 的操作，接下来三个是动画的操作，最后是取消并重新绑定 click 的事件的处理函数，所以写成这样也是不错的选择：

```
$(this).removeClass("mouseout").addClass("mouseover")
    .stop().fadeTo("fast",0.6).fadeTo("fast",1)
    .unbind("click").click(function(){
    // do something ….
     });
```

(4) 对于多个对象的较多操作，建议结合 2、3 条来做。

jQuery 还以其强大的选择器著称，有的时候很复杂的问题却可以用一行选择器轻松搞定。但对其进行代码编写时也应该注意一个问题。请看下面的例子，这个例子是 jQuery 中文论坛中曾经有人问起的一个问题：

```
$("#table>tbody>tr:has(td:last:has(:checkbox:enabled))").css("background","red");
```

这行是什么意思？呵呵，恐怕就算是经验丰富的 jQuery 程序员也得反应上好几秒才能看出这句代码的意思。

这句的作用是，在一个表格的 tbody 中，每行最后一列中的 checkbox 如果没有被禁用，则把这一行的背景设为红色。jQuery 的选择器很强大不是么？用普通的 JavaScript 写是不是要很多行代码？但是，请各位在日后写出一个引以为豪的选择器的时候，请千万不要忘记给这一段加上注释，这很重要，利人利己！下面这样加上注释就很好了：

```
//在一个 id 为 table 的表格的 tbody 中，每行最后一列中的 checkbox 如果没有被禁用，则把这行的背景设为红色
$("#table>tbody>tr:has(td:last:has(:checkbox:enabled))").css("background","red");
```

1.4 jQuery 对象和 DOM 对象

1.4.1 简介 DOM 对象和 jQuery 对象

第一次学习 jQuery，经常会搞不清楚哪些是 jQuery 对象、哪些是 DOM 对象，所以我们在这里重点讲一下。

1． DOM 对象

文档对象模型(DOM，Document Object Model)。

每一份 DOM 都可以表示一棵树。下面我们构建一个非常基本的网页：

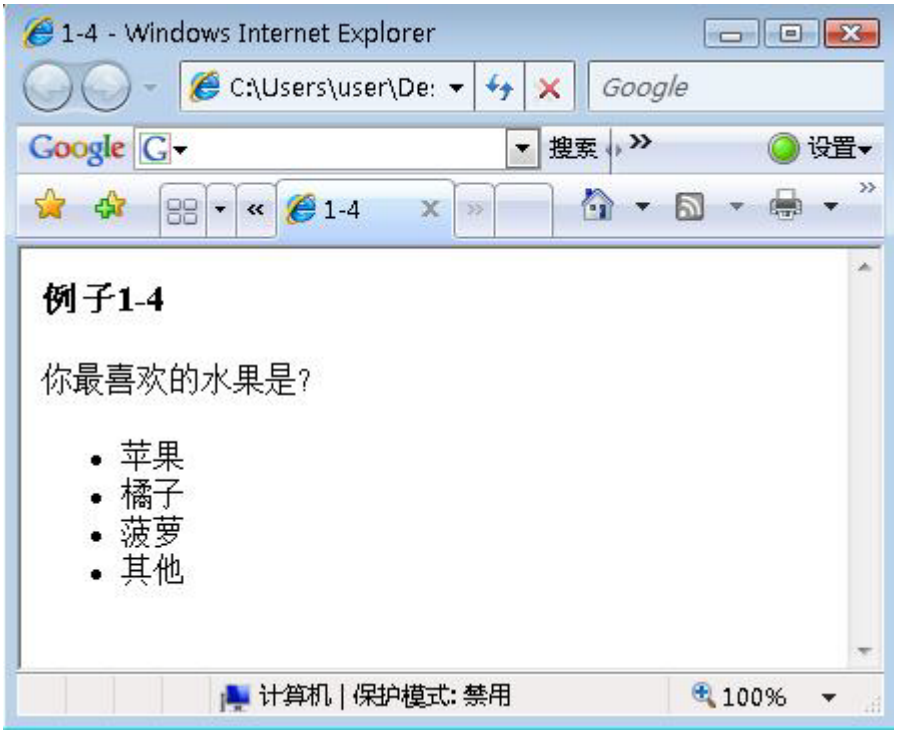

图 1-14 一个非常基本的网页

对应网页代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
      <head>
            <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
            <title>1-4</title>
      </head>
      <body>
            <h3>例子 1-4</h3>
            <p title="选择你最喜欢的水果." >你最喜欢的水果是?</p>
            <ul>
              <li>苹果</li>
              <li>橘子</li>
              <li>菠萝</li>
              <li>其他</li>
            </ul>
      </body>
</html>
```

我们可以构建上面网页结构的 DOM 树。如图 1-15 所示：

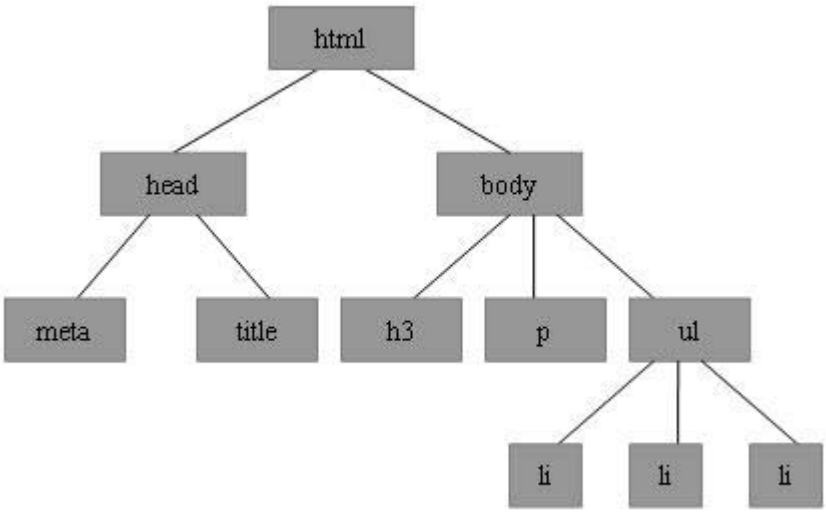

图 1-15 把网页元素表示为文档树

在这棵 DOM 树中，<h3> 、<p>、 以及<ul>的 3 个<li>子节点都是 DOM 元素节点。我们可以通过 JavaScript 中的 getElementsByTagName 来获取它们。像这样通过 getElementById 或者 getElementsByTagName 得到的 DOM 元素就是 DOM 对象。DOM 对象可以使用 JavaScript 中的方法，比如：

```
var   domObj     =     document.getElementById("id");   // 获得 DOM 对象
var   ObjHTML   =      domObj.innerHTML;          //使用 JavaScript 中的方法 innerHTML
```

2 ． jQuery 对象

那什么是 jQuery 对象呢？

jQuery 对象就是通过 jQuery 包装 DOM 对象后产生的对象。

jQuery 对象是 jQuery 独有的。如果一个对象是 jQuery 对象，那么它就可以使用 jQuery 里的方法。

比如：

```
$("#foo").html();   // 获取 id 为 foo 的元素内的 html 代码。.html()  是 jQuery 里的方法。
```

这段代码等同于：

```
document.getElementById("id").innerHTML;
```

注意：jQuery 对象将无法使用 DOM 对象的任何方法。比如$("#id").innerHTML、$("#id").checked 之类的写法都是错误的。同样，DOM 对象也不能使用 jQuery 的里方法，

比如 document.getElementById("id").html()也会报错，

只能用 document.getElementById("id").innerHTML;

特别要注意，用#id 作为选择符取得的是 jQuery 对象而并非 document.getElementById("id")所得到的 DOM 对象，两者并不等价。关于“#”选择符的运用，可以参考下一章。

从学习 jQuery 一开始就应当树立正确的观念，认识 jQuery 对象和 DOM 对象之间的区别，一旦能够跨越这道坎，之后学习 jQuery 之路就轻松多了。

1.4.2 jQurey 对象和 DOM 对象的相互转换

在讨论 jQurey 对象和 DOM 对象的相互转换之前，我们先约定定义变量的风格。

约定：如果一个获取的是 jQuery 对象，那么我们在变量前面加上$，如：

```
var  $variable  =  jQuery 对象 ;
```

如果获取的是 DOM 对象，则这么定义：

```
var  variable  =  DOM 对象;
```

本书中的例子均会以这种方式呈现，以方便读者阅读。

- **jQuery 对象转成 DOM 对象**

前面说了，jQuery 对象不能使用 DOM 中的方法， 但如果你对 jQuery 对象所提供的方法不熟悉，或者 jQuery 没有封装你想要的方法，这时你不得不用一下 DOM 对象的时候，你该怎么办呢？

在这里提供两种方式来将一个 jQuery 对象转换成 DOM 对象：

[index] 和 .get(index) ；

（1）jQuery 对象是一个数组对象，可以通过 [index] 的方式，来得到相应的 DOM 对象。

比如：

```
var  $cr  =  $("#cr");          // jQuery 对象
var  cr  =  $cr[0];             // DOM 对象
alert(cr.checked)               //检测这个 checkbox 是否被选中了
```

（2）另一种方式是 jQuery 本身提供的，通过.get(index) 方式，来得到相应的 DOM 对象。

比如：

```
var $cr  =  $("#cr");          // jQuery 对象
var cr  =  $cr.get(0);         // DOM 对象
alert(cr.checked)              //检测这个 checkbox 是否被选中了
```

- **DOM 对象转成 jQuery 对象**

对于已经是一个 DOM 对象的，我们只需要用$( ) 把 DOM 对象包装起来，就可以获得一个如假包换的 jQuery 对象了。比如：$(DOM 对象) ;

```
var cr  =  document.getElementById("cr");     //DOM 对象
var $cr  =  $(cr);                            // jQuery 对象
```

转换后，就可以任意使用 jQuery 中的方法了。

通过以上提供的方法，我们可以任意的相互转换 jQuery 对象和 DOM 对象。最后再次强调下，DOM 对象才能使用 DOM 中的方法，jQuery 对象是不可以用 DOM 中的方法的，但 jQuery 对象提供了一套更加完善的工具用于操作 DOM，可以参考第三章。

1.4.3 实例

下面我们举个简单的例子，来加深对 jQuery 对象和 DOM 对象的理解。

大家应该都玩过论坛，有的论坛注册的时候，你先要同意他的规章制度，才可以进行

其他操作。

图 1-16 某论坛注册截图

这个是某论坛的注册页面，中间你必须选中接受注册协议，否则不能注册。
我们做个简单的例子，来实现这个功能。
构建 html 代码:

```
<input type="checkbox" id="cr"/><label for="cr">我已经阅读了上面制度.</label>
```

效果如图 1-17 所示：

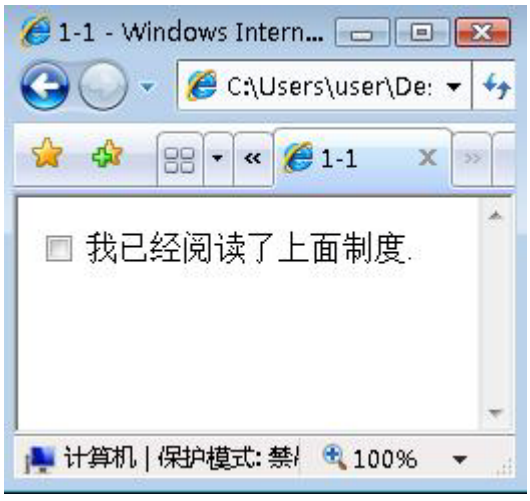

图 1-17 初始化状态

然后我们编写 JavaScript 部分，前面说过了，没有特殊声明，jQuery 库是默认导入的。下面我们用的是 DOM 的判断方式：

```
$(document).ready(function(){                  //等待 dom 元素加载完毕.
        var $cr = $("#cr");                    //jQuery 对象
        var cr = $cr.get(0);                   //DOM 对象，或者 $cr[0]
        $cr.click(function(){
            if(cr.checked){                    //DOM 方式判断
                alert("感谢你的支持!你可以继续操作!");
            }
        })
 })
```

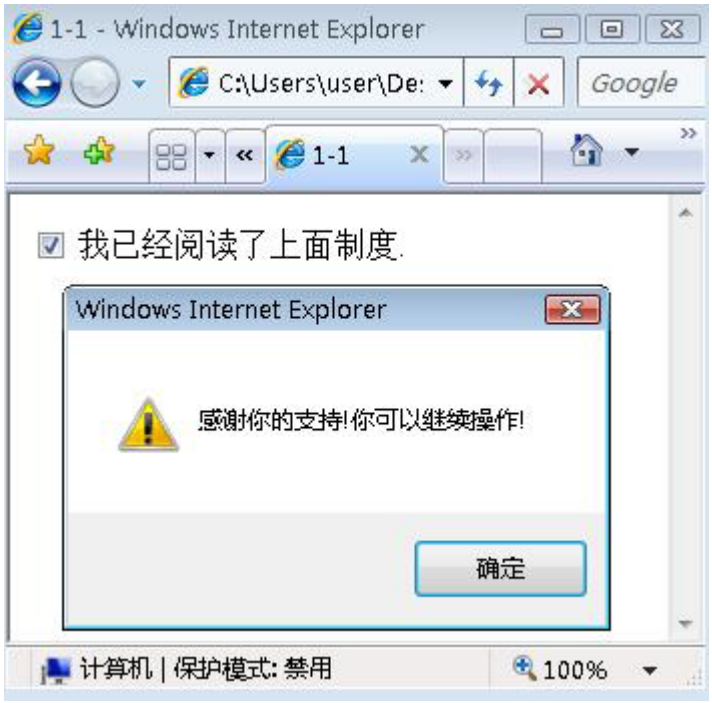

图 1-18　点击多选框，同意后

当然也可以直接用 jQuery 中的判断方式：

```
$(document).ready(function(){                  //等待 dom 元素加载完毕.
        var $cr = $("#cr");                    //jQuery 对象
        $cr.click(function(){
            if($cr.is(":checked")){            //jQuery 方式判断
                alert("感谢你的支持!你可以继续操作！");
            }
        })
})
```

编者注 ：**.is(":checked")** 是 jQuery 中的方法，来判断 jQuery 对象是否被选中，返回 boolean 值。

上面的例子就是一个简单的 jQuery 对象和 DOM 对象，分别使用了 jQuery 里的方法和 DOM 的方法。

1.5　解决 jQuery 和其它库的冲突

如果你的项目中有多种 JavaScript 库，比如同时存在 prototype.js 和 jquery.js。为了避免$对象的冲突，我们可以使用 jQuery 中的.noConflict()来解决冲突，需要注意引入 JavaScript 库的顺序。

写法 1：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
    <head>
        <!-- 引入 Prototype  -->
        <script src="prototype-1.6.0.2.js" type="text/javascript"></script>
        <!-- 引入 jQuery  -->
        <script src="../scripts/jquery-1.2.6.js" type="text/javascript"></script>
        <script type="text/javascript">
            jQuery.noConflict();                    //将变量$的控制权让渡给 prototype.js
            jQuery(function(){                      //使用 jQuery
                jQuery("p").css("color","red");     //使用 jQuery
                $("pp").style.display = "none";     //这里的$符号是 Prototype 的方法
            });
        </script>
    </head>
    <body>
        <p id="pp">testpp<p>
        <p>test<p>
    </body>
</html>
```

写法 2：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
    <head>
        <!-- 引入 jQuery  -->
        <script src="../scripts/jquery-1.2.6.js" type="text/javascript"></script>
        <script language="javascript">
            jQuery.noConflict();                        //将变量$的控制权让渡给 prototype.js
            jQuery(function(){                      //使用 jQuery
                jQuery("p").css("color","red");//使用 jQuery
                $("pp").style.display = "none";     //这里的$符号是 Prototype 的方法。
            });
        </script>
        <!-- 引入 Prototype  -->
        <script src="prototype-1.6.0.2.js" type="text/javascript"></script>
    </head>
    <body>
        <p id="pp">testpp<p>
        <p>test<p>
    </body>
</html>
```

切记不要把顺序弄错！

还有两种不使用.noConflict()来避免冲突常见方式。

一种是如果你的 jQuery(function(){…})内部不使用其它库的代码，通过如下形式构建代码，先引入 jQuery，再引入其它库，之后，继续使用$作为 jQuery 的简写形式：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
    <head>
        <!-- 引入 jQuery  -->
        <script src="../scripts/jquery-1.2.6.js" type="text/javascript"></script>
        <!-- 引入 Prototype  -->
        <script src="prototype-1.6.0.2.js" type="text/javascript"></script>
        <script type="text/javascript">
            jQuery(function($){
                $("p").css("color","red");              //函数内部的$还是 jQuery 的$。
            });
            $("pp").style.display = "none";             //函数外部的$是 Prototype 的$。
        </script>
    </head>
    <body>
        <p id="pp">testpp<p>
        <p>test<p>
    </body>
</html>
```

另一种是利用闭包的特性，你在任意地方建立一个闭包，在其内部，可以既使用$作为 jQuery 的缩写，又不用担心冲突，通常 jQuery 的插件都是采用这种形式来写的。当然也是先引入 jQuery，再引入其它库：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
    <head>
        <!-- 引入 jQuery  -->
        <script src="../scripts/jquery-1.2.6.js" type="text/javascript"></script>
        <!-- 引入 Prototype  -->
        <script src="prototype-1.6.0.2.js" type="text/javascript"></script>
        <script type="text/javascript">
            (function($){
                $("p").css("color","red");              //函数内部的$还是 jQuery 的$。
            })(jQuery);
            $("pp").style.display = "none";             //函数外部的$是 Prototype 的$。
        </script>
    </head>
    <body>
        <p id="pp">testpp<p>
        <p>test<p>
    </body>
</html>
```

这是最简单的解决冲突的方法，当然还有其它方式，但换汤不换药，这里我们就介绍这些方法。

1.6 小结

这章前半部分介绍了 jQuery 的由来和优势，同时也对目前流行的几个 JavaScript 库进行了介绍和对比。另外还自己编写了一个最简单的 jQuery 程序——在程序中，我们认识了 $(document).ready() ，此外我们还约定了 jQuery 的代码风格和变量风格。

后半部分重点介绍了 jQuery 对象和 DOM 对象的区别和它们之间的相互转换，中间插入了一个简单的论坛注册同意的实例用来加强对 jQuery 对象和 DOM 对象的理解。最后讲解了如何解决 jQuery 和其它 JavaScript 库冲突的问题，帮助那些项目上已经有其它 JavaScript 库的朋友。

第一章中我们特意强调了代码风格和变量风格，jQuery 对象和 DOM 对象，希望能引起初学者的注意。