Web前端开发者的内功修炼秘笈

4大社区鼎力推荐！

實戰

曹刘阳 著

Writing Solid Web Front-end Code

编写高质量代码①

Web前端开发修炼之道

机械工业出版社
China Machine Press

编写高质量代码

——Web 前端开发修炼之道

曹刘阳　著

机械工业出版社
China Machine Press

本书以网站重构为楔子，深刻而直接地指出了 Web 前端开发中存在的重要问题——代码难以维护。如何才能提高代码的可维护性？人是最关键的因素！于是本书紧接着全方位地解析了作为一名合格的前端开发工程师应该掌握的技能和承担的职责，这对刚加入前端开发这一行的读者来说有很大的指导意义。同时，还解读了制定规范和团队合作的重要性。

本书的核心内容是围绕 Web 前端开发的三大技术要素——HTML、CSS 和 JavaScript 来深入地探讨编写高质量的 HTML 代码、CSS 代码和 JavaScript 代码的方法、技巧、规范和最佳实践，从而为编写易于维护的 Web 前端代码打下坚实的基础。这不是一本单纯的“技术”书籍，没有系统地讲解 Web 前端开发的基础知识，它更专注于“技巧”，探索如何为“技术”提供最佳“技巧”。

本书包含了大量的开发思想和原则，都是作者在长期开发实践中积累下来的经验和心得，不同水平的 Web 前端开发者都会从中获得启发。尤其是对于那些中初级水平的读者而言，本书是一本不可多得的内功修炼秘籍。

图书在版编目（CIP）数据

编写高质量代码：Web 前端开发修炼之道 / 曹刘阳著. —北京：机械工业出版社，2010.5

ISBN 978-7-111-30595-8

Ⅰ. 编… Ⅱ. 曹… Ⅲ. 主页制作-代码-程序设计 Ⅳ. TP393.092

中国版本图书馆 CIP 数据核字（2010）第 082612 号

机械工业出版社（北京市西城区百万庄大街 22 号　邮政编码　100037）
责任编辑：陈佳媛
印刷
2010 年 6 月第 1 版第 1 次印刷
186mm×240mm • 18.75 印张（含 2.5 印张彩插）
标准书号：ISBN 978-7-111-30595-8
定价：49.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991；88361066
购书热线：（010）68326294；88379649；68995259
投稿热线：（010）88379604
读者信箱：hzjsj@hzbook.com

Foreword 推 荐 序

Web 前端开发是从网页制作演变而来的，名称上有很明显的时代特征。在互联网的演化进程中，网页制作是 Web 1.0 时代的产物，那时网站的主要内容都是静态的，用户使用网站的行为也以浏览为主。2005 年以后，互联网进入 Web 2.0 时代，各种类似桌面软件的 Web 应用大量涌现，网站的前端由此发生了翻天覆地的变化。网页不再只是承载单一的文字和图片，各种富媒体让网页的内容更加生动，网页上软件化的交互形式为用户提供了更好的使用体验，这些都是基于前端技术实现的。

以前会 Photoshop 和 Dreamweaver 就可以制作网页，现在只掌握这些已经远远不够了。无论是开发难度上，还是开发方式上，现在的网页制作都更接近传统的网站后台开发，所以现在不再叫网页制作，而是叫 Web 前端开发。Web 前端开发在产品开发环节中的作用变得越来越重要，而且需要专业的前端工程师才能做好，这方面的专业人才近两年来备受青睐。Web 前端开发是一项很特殊的工作，涵盖的知识面非常广，既有具体的技术，又有抽象的理念。简单地说，它的主要职能就是把网站的界面更好地呈现给用户。

如何才能做得更好呢？

第一，必须掌握基本的 Web 前端开发技术，其中包括：CSS、HTML、DOM、BOM、Ajax、JavaScript 等，在掌握这些技术的同时，还要清楚地了解它们在不同浏览器上的兼容情况、渲染原理和存在的 Bug。

第二，在一名合格的前端工程师的知识结构中，网站性能优化、SEO 和服务器端

的基础知识也是必须掌握的。

第三，必须学会运用各种工具进行辅助开发。

第四，除了要掌握技术层面的知识，还要掌握理论层面的知识，包括代码的可维护性、组件的易用性、分层语义模板和浏览器分级支持，等等。

可见，看似简单的网页制作，如果要做得更好、更专业，真的是不简单。这就是前端开发的特点，也是让很多人困惑的原因。如此繁杂的知识体系让新手学习起来无从下手，对于老手来说，也时常不知道下一步该学什么。

目前市面上关于 Web 前端开发的书主要都是针对单一技术的，本书与这些书有着本质的区别。它主要想实现两个目标：第一，为不太有经验的 Web 前端开发工程师建立大局观，让他们真正了解和理解这个职业；第二，帮助有一定 Web 前端开发经验的工程师修炼内功，通过编写高质量的代码来提高前端代码的可维护性。这是很多前端开发工程师感兴趣的内容。

本书的前两章讨论了网站重构和团队合作，这是很有必要的。网站重构的目的仅仅是为了让网页更符合 Web 标准吗？不是！重构的本质应该是构建一个前端灵活的 MVC 框架，即 HTML 作为信息模型（Model），CSS 控制样式（View），JavaScript 负责调度数据和实现某种展现逻辑（Controller）。同时，代码需要具有很好的复用性和可维护性。这是高效率、高质量开发以及协作开发的基础。建立了这种大局观后，学习具体技术的思路就更清晰了。

代码质量是前端开发中应该重点考虑的问题之一。例如，实现一个网站界面可能会有无数种方案，但有些方案的维护成本会比较高，有些方案会存在性能问题，而有些方案则更易于维护，而且性能也比较好。这里的关键影响因素就是代码质量。CSS、HTML、JavaScript 这三种前端开发语言的特点是不同的，对代码质量的要求也不同，但它们之间又有着千丝万缕的联系。本书中包含着很多开发的思想和经验，都是在长期的开发实践中积累下来的，不同水平的 Web 前端工程师都会从中获得启发。

张克军（著名 Web 前端开发工程师）

2010 年 4 月

Praise 赞　誉

这是一本能修炼前端开发人员内功的书，没有过多的名词解释和理论讲述，都是来自实战中的经验。本书强调的团队合作和开发习惯对于每一个前端开发团队来说都非常重要，尤其是大公司的前端开发团队，所有成员都应该了解一下这些内容。对于所有前端开发人员而言，本书是不可或缺的参考书。

——马波 百度前端开发组项目经理

组织前端技术交流活动是作者最大的喜好之一，对前端技术痴迷，乐于分享。本书由浅入深地讲解了开发高质量的、易于维护的 Web 前端应用的技巧和注意事项。是辅助初级 Web 前端开发者进阶的一本好书，强烈推荐。

——赵立元　新浪产品部前端开发技术经理

前端开发工程师是一个易学难精的职业，很多企业常常抱怨招聘不到合适的人才。市面上此类书籍质量参差不齐，要么是泛泛而谈的理论知识，要么是用大量简单的代码示例和插图充数。本书的作者是一位有着丰富经验的资深前端开发工程师，无论是他的理论功底，还是实战经验，你都能在本书中有所体会。本书适合于中初级前端开发工程师、技术经理和项目经理阅读，选择此书一定不会让你失望。

——罗金　网易的前端开发项目经理

对于经验不太丰富的 Web 前端开发者而言，本书可谓不可多得。书中丰富的编写高质量前端代码的技巧是在大量实践和总结中结出的硕果，极具学习和参考价值。

——周裕波 百度前端研发工程师、Web 标准化交流会发起人之一

如今的 Web 应用正朝着规模化和复杂化的方向发展，应用的可维护性变得越来越重要。本书从编写高质量代码的角度探讨了如何提高 Web 前端代码的可维护性，包含大量难得的编程技巧，值得学习和研究。

——Ajax 中国（www.okajax.com）

有经验的 Web 前端开发工程师都知道，如果要精通这一行，必须先“精通”十行。在 Web 前端开发工程师近乎庞杂的知识体系中，HTML、CSS、JavaScript 这三大技术是最基本的，也是最核心的。本书没有系统地介绍这三个方面的基础知识，重点是讲解了大量编写高质量 HTML、CSS 和 JavaScript 代码的技巧，旨在提高中初级开发者的水平。

——jQuery 中文社区（http://bbs.jquery.org.cn）

如果你只是初窥 Web 前端开发的门径，强烈建议你阅读本书，它能帮助你修炼内功。它不仅从知识体系的角度为准备从事 Web 前端开发工作的朋友指明了学习方向，而且还从技术的角度给出了大量的技巧和最佳实践。

——一起 Ext（http://www.17ext.com/）

HTML、CSS、JavaScript 是 Web 前端开发者必须精通的三把利器，本书是关于这三把利器的使用秘籍，如能善加利用，必定例无虚发。强烈推荐！

——CSS 中文社区

Preface 前　言

前端开发工程师是一个很新的职业，在国内乃至国际上真正开始受到重视的时间不超过 5 年。但是，随着 Web 2.0 概念的普及和 W3C 组织的推广，网站重构的影响力正以惊人的速度增长。XHTML+CSS 布局、DHTML 和 Ajax 像一阵旋风，铺天盖地席卷而来，包括新浪、搜狐、网易、腾讯、淘宝等在内的各种规模的 IT 企业都对自己的网站进行了重构。

为什么它们会对自己的网站进行重构呢？有两个方面的原因：

第一，根据 W3C 标准进行重构后，可以让前端的代码组织更有序，显著改善网站的性能，还能提高可维护性，对搜索引擎也更友好；

第二，重构后的网站能带来更好的用户体验，用 XHTML+CSS 重新布局后的页面，文件更小，下载速度更快。

DHTML 可以让用户的操作更炫，更吸引眼球；Ajax 可以实现无刷新的数据交换，让用户的操作更流畅。对于普通用户来说，一个网站是否专业、功能是否强大，服务器端是用 J2EE+Oracle 的强大组合，还是用 ASP+Access 的简单组合，并没有太明显的区别。但是，前端的用户体验却给了用户直观的印象。

随着人们对用户体验的要求越来越高，前端开发的技术难度越来越大，前端开发工程师这一职业终于从设计和制作不分的局面中独立出来。前端开发技术包括三个要素：HTML、CSS 和 JavaScript，但随着 RIA 的流行和普及，Flash/Flex、Silverlight、XML 和服务器端语言也是前端开发工程师应该掌握的。前端开发工程师既要与上游的交互设计

师、视觉设计师和产品经理沟通，又要与下游的服务器端工程师沟通，需要掌握的技能非常多。这就从知识的广度上对前端开发工程师提出了要求。**如果要精于前端开发这一行，也许要先精十行**。然而，全才总是少有的。所以，对于不太重要的知识，我们只需要“通”即可。但“通”到什么程度才算够用呢？对于很多初级前端开发工程师来说，这个问题是非常令人迷惑的。

前端开发的门槛其实非常低，与服务器端语言先慢后快的学习曲线相比，前端开发的学习曲线是先快后慢。所以，对于从事 IT 工作的人来说，前端开发是个不错的切入点。也正因为如此，前端开发领域有很多自学成“才”的同行，但大多数人都停留在会用的阶段，因为后面的学习曲线越来越陡峭，每前进一步都很难。另一方面，正如前面所说，前端开发是个非常新的职业，对一些规范和最佳实践的研究都处于探索阶段。总有新的灵感和技术不时闪现出来，例如 CSS sprite、负边距布局、栅格布局等；各种 JavaScript 框架层出不穷，为整个前端开发领域注入了巨大的活力；浏览器大战也越来越白热化，跨浏览器兼容方案依然是五花八门。为了满足“高可维护性”的需要，我们需要更深入、更系统地去掌握前端知识，这样才可能创建一个好的前端架构，保证代码的质量。

一位好的前端开发工程师在知识体系上既要有广度，又要有深度，所以很多大公司即使出高薪也很难招聘到理想的前端开发工程师。市面上有很多关于前端开发的书，这些书籍能很好地指导读者掌握前端开发的基础知识，能让读者达到会用的水平。然而，几乎还没有书能告诉开发者们如何才能用得更好，如何才能编写出高质量的前端代码，如何才能系统有效地组织前端架构……本书弥补了这方面的市场空白，它假定读者已经具有一定的 Web 前端开发基础，不会对基础知识进行详细介绍，主要精力放在如何编写和组织高质量代码上，从而提高代码的可维护性。

本书的重点不在于讲解技术，而是更侧重于对技巧的讲解。技术非黑即白，只有对和错，而技巧则见仁见智。本书是笔者个人的经验分享，尽信书不如无书，大家可以有选择地吸收，如果对书中的观点或技巧有不同的见解，非常欢迎与笔者讨论交流。大家可以通过邮箱 cly84920@gmail.com 与作者取得联系，也可以通过 QQ 群 8791223 参与到“如何编写高质量前端代码”的讨论中来。

在编写本书的过程中，如何组织目录一直是让笔者非常纠结的事情：HTML、CSS 和 JavaScript 是三门截然不同的语言，在实际应用过程中涉及的深度也各不相同，HTML 需注意的事项较少，CSS 次之，JavaScript 最为复杂。所以，本书虽然会同时对这三个方面进行探讨，但所用篇幅与它们的复杂度是成正比的。

致　　谢　Acknowledgment

感谢在本书写作过程中为我提供宝贵意见的朋友们，他们是周裕波、钟志、刘运周、李海玲、钟伟明、邹渤一、黄海宝、胡淑芳，没有你们的反馈，这本书将失色不少。

感谢克军为我写的推荐序，很怀念我们一起吃午饭的那段日子，下次音乐节我一定到。感谢卢海军为我提供了三张精美的插图，它们真的很漂亮。

感谢华章编辑们的细心工作，谢谢你们一直耐心友好地对待我一次又一次的拖稿，与你们合作非常愉快。

感谢我的家人，谢谢你们在我最没有耐心写下去的时候一直陪着我。谢谢我的老婆张霞，没有你每天的督促，真不知道这本书要写到哪天才写得完。

Contents 目　录

第 1 章 从网站重构说起

本章内容

- 糟糕的页面实现，头疼的维护工作
- Web 标准——结构、样式和行为的分离
- 前端的现状
- 打造高品质的前端代码，提前代码的可维护性——精简、重用、有序

1.1 糟糕的页面实现，头疼的维护工作

代码清单 1-1 一个糟糕老网页的实现

```
    <html>
    <head>
    <meta http-equiv="Content-Type" content="text/html; charset=gb2312">
    <style type=text/css>
    td,body {  font-size: 15px; font-family:arial,sans-serif,宋体;}
    body{margin-top:0px;margin-left:0px;margin-right:0px;background-color: #fcfff7}
    a:link{ color:#000000; text-decoration:none;padding-left:4px; }
a:visited{COLOR: #000000; TEXT-DECORATION: none;padding-left:4px;}
a:active{color:green;text-decoration:none;padding-left:4px;}a:hover{color:red;text-
decoration:underline;padding-left:4px;}
    a.m:link{ color:#000000; text-decoration:none;padding-left:0px;}
    a.m:visited{COLOR: #000000; TEXT-DECORATION: none;padding-left:0px;}
    a.m:active{color:green;text-decoration:none;padding-left:0px;}
    a.m:hover{color:red;text-decoration:underline;padding-left:0px;}
    .t1{border-width:1px 1px 1px 1px;border-style:solid;font-size:12px;text-align:
center}
    .bgg{border-color:#8AB78A; width:779px}
    .f9pt{font-size: 12px;}
    #sfont a,#sfont b{font-size:13px;}
    </style>
    <base target="_blank">
    <title>网址之家 - -动漫</title>
    <script src="js/usertrack.js"></script>
    </head>
    <body>
    <div align="center"><center>
    <table border="0" cellpadding="0" cellspacing="0" width="778" height="51">
       <tr>
         <td width="230" height="51"><a href="http://www.xxx.com" target="_self"
class="m"><img src="logo.gif" alt="xxx.com" width="168" height="63"
border="0"></a></td>
        <td height="51" align="center"><table width="100%" border=0  cellpadding=0
cellspacing=0>
          <form name=form1 action=http://www.baidu.com/s>
                <input type=hidden name=tn value=abc>
                <tr>
          <td colspan="2" id=sfont>
                <a  href=http://news.baidu.com>新 闻</a>
                <b>网 页</b>
                <a  href=http://tieba.baidu.com>贴 吧</a>
```

```
                <a  href=http://zhidao.baidu.com>知 道</a>
    </td>
         </tr>
         <tr>
         <td height="30" valign="top">
         <input type=text name=wd size=40 onMouseOver=this.focus()
onFocus=this.select() style="margin-bottom:-5px;
font-size:16px;height:1.78em;font-family:arial,sans-serif,宋体;padding-top:2px;
padding-left:1px" maxlength=100>
            <input type=submit style="height:1.82em;width:6.4em;font-size:16px;
margin-bottom:-5px; padding-top:2px" value="百度一下">
    </td>
        <td width="80" valign="top" class="f9pt"> </td>
    </tr>
        </form></table></td>
    </tr>
      </table>
    </center></div>
    <script language=javascript>
    <!--
    UserTrack.init(1,"动漫")
    document.form1.wd.focus()
    //-->
    </script>
    </body>
    </html>
```

1.2 Web 标准——结构、样式和行为的分离

代码清单 1-2 结构、样式和行为混杂的网页

```
    <td width="100%" height="20"  class="f9pt"  align="center">
         <font color="#346F0E">下载</font>
         <input type=text name=wd size=40 onMouseOver=this.focus()
onFocus=this.select() style="margin-bottom:-5px;
font-size:16px;height:1.78em;font-family:arial,sans-serif,宋体;padding-top:2px;
padding-left:1px" maxlength=100>
    </td>
```

代码清单 1-3 结构、样式和行为分别用单独文件分离开

```
    test.html 文件：
    <link rel=”stylesheet” type=”text/css” href=”test.css” />
    <td class="f9pt myTd">
         <span class=”myFont”>下载</span>
         <input type=”text” name=”wd” size=”40” id=”myInput” maxlength=”100” />
    </td>
    <script type=”text/Javascript” src=”test.js”></script>

    ================================================================

    test.css 文件：
    .myTd{width:100%;height:20px;text-align:center;}
    .myFont{color:#346F0E;}
    #myInput{margin-bottom:-5px;font-size:16px;height:1.78em;font-family:arial,sans-
serif,宋体;padding-top:2px;padding-left:1px}

    ================================================================
```

```
    test.js 文件
    var myInput = document.getElementById(“myInput”);
    myInput.onmouseover = function(){
      this.focus();
    }
    myInput.onfocus = function(){
      this.select();
    }
```

代码清单 1-4 样式和行为在网页内分离开

```
    <style type=”text/CSS”>
    .myTd{width:100%;height:20px;text-align:center;}
    .myFont{color:#346F0E;}
    #myInput{margin-bottom:-5px;font-size:16px;height:1.78em;font-family:arial,sans-serif,宋体;padding-top:2px;padding-left:1px}
    </style>
    <td class="f9pt myTd"><span class=”myFont”>下载</span><input type=”text” name=”wd” size=”40” id=”myInput” maxlength=”100” /></td>
    <script type=”text/Javascript”>
    var myInput = document.getElementById(“myInput”);
    myInput.onmouseover = function(){
      this.focus();
    }
    myInput.onfocus = function(){
      this.select();
    }
    </script>
```

1.3 前端的现状

1.4 打造高品质的前端代码，提高代码的可维护性——精简、重用、有序

第 2 章 团队合作

本章内容

- 揭密前端开发工程师
- 欲精一行，必选通十行
- 增加代码可读性——注释
- 提高重用性——公共组件和私有组件的维护
- 冗余和精简的矛盾——选择集中还是选择分散
- 磨刀不误砍柴功——前期的构思很重要
- 制订规范
- 团队合作最大的难度不是技术，是人

2.1 揭密前端开发工程师

2.2 欲精一行，必先通十行

2.3 增加代码可读性——注释

2.4 提高重用性——公共组件和私有组件的维护

2.5 冗余和精简的矛盾——选择集中还是选择分散

2.6 磨刀不误砍柴功——前期的构思很重要

2.7 制订规范

2.8 团队合作最大的难度不是技术，是人

第 3 章 高品质的 HTML

本章内容

- 标签的语义
- 为什么要使用语义化标签
- 如何确定你的标签是否语义良好
- 常见模块详解

3.1 标签的语义

3.2 为什么要使用语义化标签

代码清单 3-1 简单页面

```
    <!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>阿当的简单示例</title>
    </head>
    <body>
    <h1>h1 标签</h1>
    <h2>h2 标签</h2>
    <h3>h3 标签</h3>
    <p>p 标签</p>
    <div>div 标签</div>
    <span>span 标签</span>
    <strong>strong 标签</strong>
    <input type="text" value="input 标签" />
    <textarea>textarea 标签</textarea>
    <input type="button"  value="提交" />
    <ul>
         <li>1</li>
         <li>2</li>
         <li>3</li>
         <li>4</li>
         <li>5</li>
         <li>6</li>
         <li>7</li>
         <li>8</li>
    </ul>
    </body>
    </html>
```

代码清单 3-2 简单页面的 CSS 代码

```
    h1{font-size:12px;font-weight:normal;}
    h2{font-size:16px;}
    h3{font-size:20px;font-weight:normal;}
```

```
    p{size:normal;background:#333;color:#fff;padding:50px;font-size:30px;font-weight
:bold;}
    div{display:inline;}
    strong{display:block;padding:10px;border:1px dashed #000;margin:20px 50px;}
    .text{width:300px;height:100px;}
    textarea{width:150px;height:20px;border:4px solid
#999;background:#ccc;overflow:auto;}
    .btn{border:3px solid green;background:black;color:#fff;}
    ul{list-style:none;}
    li{padding:10px;border:1px dashed #ccc;float:left;margin:0 2px;}
```

3.3 如何确定你的标签是否语义良好

3.4 常见模块详解

3.4.1 标题和内容

代码清单 3-3　标题和内容模块实现方案一

```
    html 部分:
     <div class="h2">标签的语义<a href="#">更多&gt;&gt;</a></div>
    <div class="p">段落一的内容。。。<span class="strong">根据浏览器的默认样式 </span>。。。</div>
    <div class="p">段落二的内容</div>

    CSS 部分:
    .h2{position:relative;border-bottom:1px dashed #fff;}
    .h2 a{position:absolute;right:0;top:0;}
    .p{text-indent:2em;line-height:150%;margin:0 0 20px 0;}
    .strong{color:red;}
```

代码清单 3-4　标题和内容模块实现方案一的语义

```
    <分隔 class="h2">标签的语义<锚点 href="#">更多&gt;&gt;</锚点></分隔>
    <分隔>段落一的内容。。。<范围 class="strong">根据浏览器的默认样式 </范围>。。。</分隔>
    <分隔>段落二的内容</分隔>
```

代码清单 3-5　标题和内容模块实现方案二

```
    html 部分:
    <h2>标签的语义 <a href="#">更多&gt;&gt;</a> </h2>
    <p>段落一的内容。。。<strong>根据浏览器的默认样式 </strong>。。。</p>
    <p>段落二的内容</p>


    CSS 部分:
    h2{position:relative;border-bottom:1px dashed #fff;}
    h2 a{position:absolute;right:0;top:0;}
    p{text-indent:2em;line-height:150%;margin:0 0 20px 0;}
    strong{color:red;font-weight:normal}
```

代码清单 3-6　标题和内容模块实现方案二的语义

```
    <二级标题>标签的语义<锚点 href="#">更多&gt;&gt;</锚点> </二级标题>
    <段落>段落一的内容。。。<强调>根据浏览器的默认样式 </强调>。。。</段落>
    <段落>段落二的内容</段落>
```

代码清单 3-7　标题和内容模块实现方案三

```
    html 部分:
    <h2>标签的语义</h2>
    <a href="#">更多&gt;&gt;</a>
```

```
    <p>段落一的内容。。。<strong>根据浏览器的默认样式 </strong>。。。</p>
    <p>段落二的内容</p>

    CSS 部分:
    h2{}
    a{}
    p{text-indent:2em;line-height:150%;margin:0 0 20px 0;}
    strong{color:red;font-weight:normal}
```

代码清单 3-8　标题和内容模块实现方案三的语义

```
    <二级标题>标签的语义</二级标题>
    <锚点 href="#">更多&gt;&gt;</锚点>
    <段落>段落一的内容。。。<强调>根据浏览器的默认样式 </强调>。。。</段落>
    <段落>段落二的内容</段落>
```

代码清单 3-9　标题和内容模块实现方案三的改进版

```
    html 部分:
    <div class="title">
         <h2>标签的语义</h2>
         <a href="#">更多&gt;&gt;</a>
    </div>
    <p>段落一的内容。。。<strong>根据浏览器的默认样式 </strong>。。。</p>
    <p>段落二的内容</p>

    CSS 部分:
    .title{border-bottom:1px dashed #fff;text-align:right;}
    .title h2{float:left;}
    p{text-indent:2em;line-height:150%;margin:0 0 20px 0;}
    strong{color:red;font-weight:normal}
```

3.4.2 表单

代码清单 3-10　表单模块实现方案一

```
    <form action="" method="" class="fieldset">
         <div><span>帐号 ：</span> <input type="text" id="name" /></div>
         <div><span>密码 ：</span> <input type="password" id="pw" /></div>
         <input type="submit" value="登陆" class="subBtn" />
    </form>
```

代码清单 3-11　表单模块实现方案一的语义

```
    <表单 action="" method="" class="fieldset">
         <分隔><范围>帐号 ：</范围> <表单项 type="text" id="name" /></分隔>
         <分隔><范围>密码 ：</范围> <表单项 type="password" id="pw" /></分隔>
         <表单项 type="submit" value="登陆" class="subBtn" />
    </表单>
```

代码清单 3-12　表单模块实现方案二

```
    <form action="" method="">
         <fieldset>
                  <legend>登录表单</legend>
                  <p><label for="name">帐号 ：</label> <input type="text" id="name"
/></p>
                  <p><label for="pw">密码 ：</label> <input type="password" id="pw"
/></p>
                  <input type="submit" value="登陆" class="subBtn" />
         </fieldset>
    </form>
```

代码清单 3-13　表单模块实现方案二的语义

```
    <表单 action=“” method=“” class=“fieldset”>
         <域集>
                <域集名>登录表单</域集名>
                <段落><表单项说明 for=“name”>帐号 ：</表单项说明> <表单项 type=“text” id=
“name” /></段落>
                <段落><表单项说明 for=“pw”>密码 :</表单项说明 ><表单项 type=“password” id=
“pw” /></段落>
                <表单项 type="submit" value="登陆" class="subBtn" />
         </域集>
    </表单>
```

代码清单 3-14　表格模块实现方案一

```
    <div class="caption">几种页面实现的比较</div>
    <table border="1">
         <tr class="thead"><td class="th">实现方式</td><td class="th">代码量</td><td
class="th">搜索引擎友好</td><td class="th">特殊终端兼容</td></tr>
          <tr><td class="th">table 布局</td><td>多</td><td>差</td><td>一般</td></tr>
          <tr><td class="th">乱用标签的 CSS 布局</td><td>少</td><td>一般</td><td>差
</td></tr>
         <tr><td class="th">标签语义良好的 CSS 布局</td><td>少</td><td>好</td><td>好
</td></tr>
    </table>
```

代码清单 3-15　表格模块实现方案一的语义

```
    <分隔 class=“caption”>几种页面实现的比较</分隔>
    <表格 border="1">
         <表格行 class="thead"><表格单元格 class="th">实现方式</表格单元格><表格单元格
class="th">代码量</表格单元格><表格单元格 class="th">搜索引擎友好</表格单元格><表格单元格
class="th">特殊终端兼容</表格单元格></表格行>
          <表格行><表格单元格 class="th">table 布局</表格单元格><表格单元格>多</表格单元格><表
格单元格>差</表格单元格><表格单元格>一般</表格单元格></表格行>
          <表格行><表格单元格class="th">乱用标签的CSS布局</表格单元格><表格单元格>少</表格单元
格><表格单元格>一般</表格单元格><表格单元格>差</表格单元格></表格行>
         <表格行><表格单元格class="th">标签语义良好的CSS布局</td><表格单元格>少</表格单元格><
表格单元格>好</表格单元格><表格单元格>好</表格单元格></表格行>
    </表格>
         <分隔><范围>帐号 ：</范围> <表单项 type="text" id="name" /></分隔>
         <分隔><范围>密码 ：</范围> <表单项 type=“password” id=“pw” /></分隔>
         <表单项 type="submit" value="登陆" class="subBtn" />
    </表单>
```

代码清单 3-16　表格模块实现方案二

```
    <table border="1">
         <caption>几种页面实现的比较</caption>
         <thead>
                <tr><th>实现方式</th><th>代码量</th><th>搜索引擎友好</th><th>特殊终端兼容
</th></tr>
         </thead>
         <tbody>
                <tr><th>table 布局</th><td>多</td><td>差</td><td>一般</td></tr>
                <tr><th>乱用标签的 CSS 布局</th><td>少</td><td>一般</td><td>差</td></tr>
    <tr><th>标签语义良好的 CSS 布局</th><td>少</td><td>好</td><td>好</td></tr>
         </tbody>
    </table>
```

代码清单 3-17　表格模块实现方案二的语义

```
    <表格 border="1">
         <表格标题>几种页面实现的比较</表格标题>
         <表格头部>
```

```
                    <表格行> <表头>实现方式</表头><表头>代码量</表头><表头>搜索引擎友好</表头><表头>特殊终端兼容</表头></表格行>
          </表格头部>
           <表格主体>
                    <表格行><表头>table 布局</表头><表格单元格>多</表格单元格><表格单元格>差</表格单元格><表格单元格>一般</表格单元格></表格行>
                    <表格行><表头>乱用标签的 CSS 布局</表头><表格单元格>少</表格单元格><表格单元格>一般</表格单元格><表格单元格>差</表格单元格></表格行>
                    <表格行><表头>标签语义良好的 CSS 布局</表头><表格单元格>少</表格单元格><表格单元格>好</表格单元格><表格单元格>好</表格单元格></表格行>
          </表格主体>
    </table>
```

3.4.4 语义化标签应注意的一些其他问题

第 4 章 高品质的 CSS

本章内容

- 怪异模式和 DTD
- 如何组织 CSS
- 推荐的 base.css
- 模块化 CSS——引入编程的思想到 CSS
- 低权重原则——避免滥用子选择器
- CSS sprite
- CSS 的常见问题

4.1 怪异模式和 DTD

代码清单 4-1 HTML 中常见的 4 种 DTD 类型

```
    用于 HTML 4.01 的严格型
    <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">

    用于 HTML 4.01 的过渡型
    <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
    用于 XHTML 1.0 的严格型
    <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
    用于 XHTML 1.0 的过渡型
    <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

4.2 如何组织 CSS

代码清单 4-2 page.css 的注释

```
    /*首页*/
    .test{}
    .test2{}

    /*关于我们*/
    .test3{}
    .test4{}

    /*联系我们*/
    .test5{}
    .test6{}
```

4.3 推荐的 base.css

代码清单 4-3 base.css

```
    .mb5{margin-bottom:5px}/*CSS reset*/
    body,div,dl,dt,dd,ul,ol,li,h1,h2,h3,h4,h5,h6,pre,form,fieldset,input,textarea,p,
blockquote,th,td {margin:0;padding:0;}
    table {border-collapse:collapse;border-spacing:0;}
    fieldset,img {border:0}
    address,caption,cite,code,dfn,em,strong,th,var
{font-style:normal;font-weight:normal}
    ol,ul {list-style:none}
    caption,th {text-align:left}
    h1,h2,h3,h4,h5,h6 {font-size:100%;font-weight:normal}
    q:before,q:after {content:''}
    abbr,acronym { border:0}

    /*文字排版*/
    .f12{font-size:12px}
    .f13{font-size:13px}
    .f14{font-size:14px}
    .f16{font-size:16px}
    .f20{font-size:20px}
    .fb{font-weight:bold}
    .fn{font-weight:normal}
    .t2{text-indent:2em}
    .lh150{line-height:150%}
    .lh180{line-height:180%}
    .lh200{line-height:200%}
    .unl{text-decoration:underline;}
    .no_unl{text-decoration:none;}

    /*定位*/
    .tl{text-align:left}
    .tc{text-align:center}
    .tr{text-align:right}
    .bc{margin-left:auto;margin-right:auto;}
    .fl{float:left;display:inline}
    .fr{float:right;display:inline}
    .cb{clear:both}
    .cl{clear:left}
    .cr{clear:right}
    .clearfix:after{content:".";display:block;height:0;clear:both;visibility:hidden}
.clearfix{display:inline-block}* html .clearfix{height:1%}.clearfix{display:block}
    .vm{vertical-align:middle}
    .pr{position:relative}
    .pa{position:absolute}
    .abs-right{position:absolute;right:0}
    .zoom{zoom:1}
    .hidden{visibility:hidden}
    .none{display:none}

    /*长度高度*/
    .w10{width:10px}
    .w20{width:20px}
    .w30{width:30px}
    .w40{width:40px}
    .w50{width:50px}
    .w60{width:60px}
    .w70{width:70px}
```

```
    .w80{width:80px}
    .w90{width:90px}
    .w100{width:100px}
    .w200{width:200px}
    .w250{width:250px}
    .w300{width:300px}
    .w400{width:400px}
    .w500{width:500px}
    .w600{width:600px}
    .w700{width:700px}
    .w800{width:800px}
    .w{width:100%}
    .h50{height:50px}
    .h80{height:80px}
    .h100{height:100px}
    .h200{height:200px}
    .h{height:100%}

    /*边距*/
    .m10{margin:10px}
    .m15{margin:15px}
    .m30{margin:30px}
    .mt5{margin-top:5px}
    .mt10{margin-top:10px}
    .mt15{margin-top:15px}
    .mt20{margin-top:20px}
    .mt30{margin-top:30px}
    .mt50{margin-top:50px}
    .mt100{margin-top:100px}

    .mb10{margin-bottom:10px}
    .mb15{margin-bottom:15px}
    .mb20{margin-bottom:20px}
    .mb30{margin-bottom:30px}
    .mb50{margin-bottom:50px}
    .mb100{margin-bottom:100px}
    .ml5{margin-left:5px}
    .ml10{margin-left:10px}
    .ml15{margin-left:15px}
    .ml20{margin-left:20px}
    .ml30{margin-left:30px}
    .ml50{margin-left:50px}
    .ml100{margin-left:100px}
    .mr5{margin-right:5px}
    .mr10{margin-right:10px}
    .mr15{margin-right:15px}
    .mr20{margin-right:20px}
    .mr30{margin-right:30px}
    .mr50{margin-right:50px}
    .mr100{margin-right:100px}
    .p10{padding:10px;}
    .p15{padding:15px;}
    .p30{padding:30px;}
    .pt5{padding-top:5px}
    .pt10{padding-top:10px}
    .pt15{padding-top:15px}
    .pt20{padding-top:20px}
    .pt30{padding-top:30px}
    .pt50{padding-top:50px}
    .pb5{padding-bottom:5px}
```

```
.pb10{padding-bottom:10px}
.pb15{padding-bottom:15px}
.pb20{padding-bottom:20px}
.pb30{padding-bottom:30px}
.pb50{padding-bottom:50px}
.pb100{padding-bottom:100px}
.pl5{padding-left:5px}
.pl10{padding-left:10px}
.pl15{padding-left:15px}
.pl20{padding-left:20px}
.pl30{padding-left:30px}
.pl50{padding-left:50px}
.pl100{padding-left:100px}
.pr5{padding-right:5px}
.pr10{padding-right:10px}
.pr15{padding-right:15px}
.pr20{padding-right:20px}
.pr30{padding-right:30px}
.pr50{padding-right:50px}
.pr100{padding-right:100px}
```

4.4 模块化 CSS —— 引入面向对象的编程思想到 CSS

4.4.1 如何划分模块 —— 单一职责

4.4.2 CSS 的命名 —— 命名空间的概念

代码清单 4-4 css 命名方案一

```
<style type="text/CSS">
.timeList{xxxxxxx}
</style>
<ul class="timelist">
   <li>2009 年 08 月(3)</li>
   <li>2009 年 07 月(12)</li>
   <li>2009 年 06 月(3)</li>
   <li>2009 年 05 月(8)</li>
<li>2009 年 04 月(2)</li>
<li>2009 年 03 月(8)</li>
</ul>
```

代码清单 4-5 css 命名方案二

```
<style type="text/CSS">
.timeList{xxxxxxx}
.timeList li{border-bottom:xxxx;}
.timeList .last{border-bottom:none;}
</style>
<ul class="timeList">
   <li>2009 年 08 月(3)</li>
   <li>2009 年 07 月(12)</li>
   <li>2009 年 06 月(3)</li>
<li>2009 年 05 月(8)</li>
<li>2009 年 04 月(2)</li>
<li class="last">2009 年 03 月(8)</li>
</ul>
```

代码清单 4-6 css 命名方案三

```
<style type="text/CSS">
.timeList{xxxxxxx}
```

```
.timeList li{border-bottom:xxxx;}
.timeListLastItem{border-bottom:none;}
</style>
<ul class=”timeList”>
   <li>2009 年 08 月(3)</li>
   <li>2009 年 07 月(12)</li>
   <li>2009 年 06 月(3)</li>
<li>2009 年 05 月(8)</li>
<li>2009 年 04 月(2)</li>
<li class=”timeListLastItem”>2009 年 03 月(8)</li>
</ul>
```

代码清单 4-7　css 命名冲突

```
<style type=”text/CSS”>
/*made by 工程师 A*/
.timeList{xxxxxxx}
.timeList li{border-bottom:xxxx;}
.timeList-lastItem{border-bottom:none;}

...

/*made by 工程师 B*/
.timeList{}
.timeList li{border-bottom:xxxx;}

</style>
<!--made by 工程师 A-->
<ul class=”timeList”>
   <li>2009 年 08 月(3)</li>
   <li>2009 年 07 月(12)</li>
   <li>2009 年 06 月(3)</li>
<li>2009 年 05 月(8)</li>
<li>2009 年 04 月(2)</li>
<li class=”timeList-lastItem”>2009 年 03 月(8)</li>
</ul>

...

<!--made by 工程师 B-->
<ol class=”timeList”>
   <li>2009-08-07</li>
   <li>2009-08-06 </li>
   <li>2009-08-05</li>
</ol>
```

代码清单 4-8　css 命名加前辍

```
<style type=”text/CSS”>
/*made by 工程师 A*/
.ad-timeList{xxxxxxx}
.ad-timeList li{border-bottom:xxxx;}
.ad-timeList-lastItem{border-bottom:none;}

...

/*made by 工程师 B*/
.zx-timeList{}
.zx-timeList li{border-bottom:xxxx;}

</style>
<!--made by 工程师 A-->
```

```
<ul class=”ad-timeList”>
   <li>2009 年 08 月(3)</li>
   <li>2009 年 07 月(12)</li>
   <li>2009 年 06 月(3)</li>
<li>2009 年 05 月(8)</li>
<li>2009 年 04 月(2)</li>
<li class=”ad-timeList-lastItem”>2009 年 03 月(8)</li>
</ul>

...

<!--made by 工程师 B-->
<ol class=”zx-timeList”>
   <li>2009-08-07</li>
   <li>2009-08-06 </li>
   <li>2009-08-05</li>
</ol>
```

代码清单 4-9　不加前辍的命名方式

```
<div=”box”>
     <div class=”hd”></div>
     <div class=”bd”></div>
     <div class=”ft”></div>
</div>
```

代码清单 4-10　加前辍的命名方式

```
<div=”box”>
     <div class=”box-hd”></div>
     <div class=”box-bd”></div>
     <div class=”box-ft”></div>
</div>
```

4.4.3　挂多个 class 还是新建 class ——多用组合，少用继承

代码清单 4-11　三个简单模块实现方案一

```
<style type=”text/CSS”>
.numberList1{border:1px solid #ccc;padding:10px}
.numberList1 li{height:20px;line-height:20px;font-size:12px}
.numberList2{border:1px solid #ccc;padding:10px;}
.numberList2 li{height:20px;line-height:20px;font-size:16px}
.numberList3{border:1px solid #ccc;padding:10px;}
.numberList3 li{height:20px;line-height:20px;font-size:12px;color:red}
</style>

<body>
     <ul class="numberList1">
          <li>1111111111111111</li>
          <li>2222222222222222</li>
          <li>3333333333333333</li>
     </ul>
     <ul class="numberList2">
          <li>4444444444444444</li>
          <li>5555555555555555</li>
          <li>6666666666666666</li>
     </ul>
     <ul class="numberList3">
          <li>7777777777777777</li>
          <li>8888888888888888</li>
          <li>9999999999999999</li>
```

```
        </ul>
    </body>
```

代码清单 4-12　三个简单模块实现方案二

```
    <style type=”text/CSS”>
    .numberList1,.numberList2,.numberList3{border:1px solid #ccc;padding:10px}
    .numberList1 li,.numberList2 li,.numberList3
li{height:20px;line-height:20px;font-size:12px}
    .numberList2 li{font-size:16px}
    .numberList3 li{color:red}
    </style>

    <body>
        <ul class="numberList1">
            <li>111111111111111111</li>
            <li>222222222222222222</li>
            <li>333333333333333333</li>
        </ul>
        <ul class="numberList2">
            <li>444444444444444444</li>
            <li>555555555555555555</li>
            <li>666666666666666666</li>
        </ul>
        <ul class="numberList3">
            <li>777777777777777777</li>
            <li>888888888888888888</li>
            <li>999999999999999999</li>
        </ul>
    </body>
```

代码清单 4-13　三个简单模块实现方案三

```
    <style type=”text/CSS”>
    .f12{font-size:12px}
    .f16{font-size:16px}
    .red{color:red}
    .numberList{border:1px solid #ccc;padding:10px;}
    .numberList li{height:20px;line-height:20px;}
    </style>

    <body>
        <ul class="numberList f12">
            <li>111111111111111111</li>
            <li>222222222222222222</li>
            <li>333333333333333333</li>
        </ul>
        <ul class="numberList f16">
            <li>444444444444444444</li>
            <li>555555555555555555</li>
            <li>666666666666666666</li>
        </ul>
        <ul class="numberList f12 red">
            <li>777777777777777777</li>
            <li>888888888888888888</li>
            <li>999999999999999999</li>
        </ul>
    </body>
```

代码清单 4-14　方案二面对扩展

```
    <style type=”text/CSS”>
```

```
    .numberList1,.numberList2,.numberList3,.numberList4{border:1px solid
#ccc;padding:10px}
    .numberList1 li,.numberList2 li.,numberList3 li,numberList4 li
{height:20px;line-height:20px;font-size:12px}
    .numberList2 li,.numberList4 li{font-size:16px}
    .numberList3 li，.numberList4 li{color:red}
    </style>

    <body>
         <ul class="numberList1">
             <li>11111111111111111</li>
             <li>22222222222222222</li>
             <li>33333333333333333</li>
         </ul>
         <ul class="numberList2">
             <li>44444444444444444</li>
             <li>55555555555555555</li>
             <li>66666666666666666</li>
         </ul>
         <ul class="numberList3">
             <li>77777777777777777</li>
             <li>88888888888888888</li>
             <li>99999999999999999</li>
         </ul>
          <ul class="numberList4">
             <li>101010101010101010</li>
             <li>11111111111111111</li>
             <li>121212121212121212</li>
         </ul>
    </body>
```

代码清单 4-15　方案三面对扩展

```
    <style type=”text/CSS”>
    .f12{font-size:12px}
    .f16{font-size:16px}
    .red{color:red}
    .numberList{border:1px solid #ccc;padding:10px;}
    .numberList li{height:20px;line-height:20px;}
    </style>

    <body>
         <ul class="numberList f12">
             <li>11111111111111111</li>
             <li>22222222222222222</li>
             <li>33333333333333333</li>
         </ul>
         <ul class="numberList f16">
             <li>44444444444444444</li>
             <li>55555555555555555</li>
             <li>66666666666666666</li>
         </ul>
         <ul class="numberList f12 red">
             <li>77777777777777777</li>
             <li>88888888888888888</li>
             <li>99999999999999999</li>
         </ul>
         <ul class="numberList f16 red">
             <li>101010101010101010</li>
             <li>11111111111111111</li>
             <li>121212121212121212</li>
```

```
        </ul>
    </body>
```

代码清单 4-16　YUI3 中的 class

```
    <div class="yui-widget yui-overlay yui-widget-positioned yui-widget-stacked">
<!--Bounding Box-->

            <div class="yui-overlay-content yui-widget-stdmod">
<!--Content Box-->

                <div class="yui-widget-hd">Overlay Header Content</div>  <!--Header
Section-->
                <div class="yui-widget-bd">Overlay Body Content</div>   <!--Body
Section-->
                <div class="yui-widget-ft">Overlay Footer Content</div>  <!--Footer
Section-->

            </div>

            <iframe class="yui-widget-shim"></iframe>  <!-- Stacking shim, if
enabled-->

        </div>
```

4.4.4　如何处理上下 margin

代码清单 4-17　提取标题组件

```
    <style type=”text/CSS”>
    .title{border-bottom:1px dashed
#B2BCC6;color:#0066CF;font-size:16px;font-weight:bold;}
    </style>

    <h2 class=”title”>服务理念</h2>
    …
    <h2 class=”title”>服务宗旨</h2>
    …
    <h2 class=”title”>Gsns 产品的优势</h2>
    …
```

代码清单 4-18　包含 margin 的标题组件

```
    <style type=”text/CSS”>
    .title1{border-bottom:1px dashed
#B2BCC6;color:#0066CF;font-size:16px;font-weight:bold;margin-top:20px;}
    .title2{border-bottom:1px dashed
#B2BCC6;color:#0066CF;font-size:16px;font-weight:bold;margin-top:60px;}
    </style>

    <h2 class=”title1”>服务理念</h2>
    …
    <h2 class=”title1”>服务宗旨</h2>
    …
    <h2 class=”title2”>Gsns 产品的优势</h2>
    …
```

代码清单 4-19　应用类的组合

```
    <style type=”text/CSS”>
    .mt20{margin-top:20px;}
    .mt60{margin-top:60px;}
```

```
    .title{border-bottom:1px dashed
#B2BCC6;color:#0066CF;font-size:16px;font-weight:bold;}
    </style>

    <h2 class="title mt20">服务理念</h2>
    …
    <h2 class="title mt20">服务宗旨</h2>
    …
    <h2 class="title mt60">Gsns 产品的优势</h2>
    …
```

4.5 低权重原则 —— 避免滥用子选择器

代码清单 4-20 css 的层叠

```
    <style type="text/CSS">
    span{font-size:40px}
    .test{color:red}
    </style>

    <span class="test">1234567890</span>
```

代码清单 4-21 css 层叠有冲突的情况

```
    <style type="text/CSS">
    span{font-size:40px;color:green}
    .test{color:red}
    </style>

    <span class="test">1234567890</span>
```

代码清单 4-22 选择符权重相同的情况

```
    <style type="text/CSS">
    span{font-size:40px}
    .test{color:red}
    .test2{color:green}
    </style>

    <span class="test test2">1234567890</span>
```

代码清单 4-23 调换样式的位置

```
    <style type="text/CSS">
    span{font-size:40px}
    .test2{color:green}
    .test{color:red}
    </style>

    <span class="test test2">1234567890</span>
```

代码清单 4-24 需标红的代码

```
    <style type="text/CSS">
    #test{font-size:14px;}
    </style>
    <p id="test">CSS 选择符权重很重要</p>
```

代码清单 4-25 用子选择器

```
    <style type="text/CSS">
    #test{font-size:14px;}
    #test span{color:red;}
```

```
    </style>
    <p id="test">CSS 选择符权重<span>很重要</span></p>
```

代码清单 4-26　新建 class

```
    <style type="text/CSS">
    #test{font-size:14px;}
    .font{color:red;}
    </style>
    <p id="test">CSS 选择符权重<span class="font">很重要</span></p>
```

代码清单 4-27　添加新的文本

```
    <style type="text/CSS">
    #test{font-size:14px;}
    #test span{color:red;}
    </style>
    <p id="test">CSS 选择符权重<span>很重要</span>,我们要小心处理</p>
```

代码清单 4-28　需标绿的代码

```
    <style type="text/CSS">
    #test{font-size:14px;}
    #test span{color:red;}
    .font{color:green;}
    </style>
    <p id="test">CSS 选择符权重<span>很重要</span>,我们要<span class="font">小心处理
</span></p>
```

代码清单 4-29　被迫加重权重的选择器

```
    <style type="text/CSS">
    #test{font-size:14px;}
    #test span{color:red;}        /*选择符权重为 100+1=101*/
    #test .font{color:green;}     /*选择符权重为 100+10=110*/
    </style>
    <p id="test">CSS 选择符权重<span>很重要</span>,我们要<span class="font">小心处理
</span></p>
```

代码清单 4-30　新增 class 作为标绿容器的选择符

```
    <style type="text/CSS">
    #test{font-size:14px;}
    .font{color:red;}
    .font2{color:green;}
    </style>
    <p id="test">CSS 选择符权重<span class="font">很重要</span>,我们要<span class="font2">
小心处理</span></p>
```

4.6 CSS sprite

代码清单 4-31　用两张图实现高亮

```
    <style type="text/CSS">
    .nav li{float:left; display:inline; margin-right:10px; font-family:黑体;}
    .nav a{float:left; width:139px; height:31px; line-height:31px; font-size:24px;
color:#fff; text-decoration:none; text-align:center; background:url(img1.gif);}
    .nav a:hover{background:url(img2.gif);}
    </style>
    <ul  class="nav">
        <li><a href="#">联系我们</a></li>
        <li><a href="#">产品答疑</a></li>
        <li><a href="#">广告服务</a></li>
```

```
    </ul>
```

代码清单 4-32　利用一张大图的背景移动实现高亮

```
    <style type=”text/CSS”>
    .nav li{float:left; display:inline; margin-right:10px; font-family:黑体;}
    .nav a{float:left; width:139px; height:31px; line-height:31px; font-size:24px;
color:#fff; text-decoration:none; text-align:center; background:url(img3.gif);}
    .nav a:hover{background-position:0 -31px;}
    </style>
    <ul  class="nav">
        <li><a href="#">联系我们</a></li>
        <li><a href="#">产品答疑</a></li>
        <li><a href="#">广告服务</a></li>
    </ul>
```

4.7 CSS 的常见问题

4.7.1 CSS 的编码风格

代码清单 4-33　多行式的 css 编码风格

```
    .test{
        width:100px;
        height:50px;
        color:#ccc;
    }
    .demo{
        background-color:green;
        font-size:20px;
    }
```

代码清单 4-34　一行式的 css 编码风格

```
    .test{width:100px;height:50px;color:#ccc;}
    .demo{background-color:green;font-size:20px;}
```

4.7.2　id 和 class

4.7.3　CSS hack

代码清单 4-35　只在 IE 下生效

```
    <!--[if IE]>
    <link type="text/CSS" href="test.css" rel="stylesheet" />
    <![endif]-->
```

代码清单 4-36　只在 IE6 下生效

```
    <!--[if IE 6]>
    <link type="text/CSS" href="test.css" rel="stylesheet" />
    <![endif]-->
```

代码清单 4-37　只在 IE6 以上版本生效

```
    <!--[if gt IE 6]>
    <link type="text/CSS" href="test.css" rel="stylesheet" />
    <![endif]-->
```

代码清单 4-38　只在 IE7 上不生效

```
    <!--[if ! IE 7]>
    <link type="text/CSS" href="test.css" rel="stylesheet" />
```

```
<![endif]-->
```

代码清单 4-39　条件注释和 style 标签

```
<!--[if IE 6]>
<style type=”text/CSS”>
.test{}
</style>
<![endif]-->
```

代码清单 4-40　条件注释和 script 标签

```
<!--[if IE 6]>
<script type=”text/JavaScript”>
alert(“我是 IE 6”);
</script>
<![endif]-->
```

代码清单 4-41　ie6.css、ie7.css 和 ie8.css

```
1) ie6.css
.test{width:60px;}
2) ie7.css
.test{width:70px;}
3) IE 8.css
.test{width:80px;}
```

代码清单 4-42　针对不同 IE 版本分别导入样式

```
<!--[if IE 6]>
<link type="text/CSS" href="ie6.css" rel="stylesheet" />
<![endif]-->
<!--[if gt IE 7]>
<link type="text/CSS" href="ie7.css" rel="stylesheet" />
<![endif]-->
<!--[if gt IE 8]>
<link type="text/CSS" href="IE 8.css" rel="stylesheet" />
<![endif]-->
```

代码清单 4-43　选择符前缀 hack 法

```
<style type=”text/CSS”>
.test{width:80px;}                    /*IE 6,IE 7,IE 8*/
*html .test{width:60px;}              /*only for IE 6*/
*+html .test{width:70px;}             /*only for IE 7*/
</style>
```

代码清单 4-44　样式属性前缀 hack 法

```
<style type=”text/CSS”>
.test{width:80px;*width:70px;_width:60px;}
</style>
```

4.7.4 超链接访问过后 hover 样式就不出现的问题

代码清单 4-45　不正确的伪类顺序

```
<style type="text/CSS">
a:hover{color:yellow;}
a:visited{color:green;}
</style>
<a href=”#”>hello world</a>
```

代码清单 4-46　正确的伪类顺序

```
<style type="text/CSS">
```

```
    a:visited{color:green;}
    a:hover{color:yellow;}
    </style>
    <a href="#">hello world</a>
```

4.7.5 hasLayout

4.7.6 块级元素和行内元素的区别

代码清单 4-47 块级元素和行内元素

```
    <style type="text/CSS">
    p{background:red}
    div{background:yellow}
    span{background:blue}
    strong{background:green}
    </style>
    <p>块级元素 p</p><div>块级元素 div</div><span>行内元素 span</span><strong>行内元素
strong</strong>
```

代码清单 4-48 对块级元素和行内元素设置长宽

```
    <style type="text/CSS">
    p{background:red;width:200px;height:200px;}
    div{background:yellow;width:400px;height:100px;}
    span{background:blue;width:200px;height:200px;}
    strong{background:green;width:400px;height:100px;}
    </style>
    <p>块级元素 p</p><div>块级元素 div</div><span>行内元素 span</span><strong>行内元素
strong</strong>
```

代码清单 4-49 块级元素、行内元素的 margin 和 padding

```
    <style type="text/CSS">
    p{background:red;padding:20px;margin:20px;}
    div{background:yellow;padding:20px;}
    span{background:blue;padding:20px;margin:20px;}
    strong{background:green;padding:20px;margin:20px;}
    </style>
    <p>块级元素 p</p><div>块级元素 div</div><span>行内元素 span</span><strong>行内元素
strong</strong>
```

代码清单 4-50 改变块级元素和行内元素的显示

```
    <style type="text/CSS">
    p{background:red;display:inline;}
    div{background:yellow;display:inline;}
    span{background:blue;display:block;}
    strong{background:green;display:block;}
    </style>
    <p>块级元素 p</p><div>块级元素 div</div><span>行内元素 span</span><strong>行内元素
strong</strong>
```

4.7.7 display:inline-block 和 hasLayout

代码清单 4-51 设置 display:inline-block

```
    <style type="text/CSS">
    p{color:red;width:100px;background:#ccc;height:30px;display:inline-block;}
    </style>
    <body>
    abcdefg <p>12345</p>
    </body>
```

代码清单 4-52　触发 hasLayout 在 IE6、IE7 下模拟 inline-block

```
    <style type="text/CSS">
    span{color:red;width:100px;background:#ccc;height:30px;display:inline-block;}
    </style>
    <body>
    abcdefg <span>12345</span>
    </body>
```

代码清单 4-53　调整竖直排列的位置

```
    <style type="text/CSS">
    span{color:red;width:100px;background:#ccc;height:30px;display:inline-block;*
vertical-align:-10px;}
    </style>
    <body>
    abcdefg <span>12345</span>
    </body>
```

4.7.8　relative、absolute 和 float

4.7.9　居中

代码清单 4-54　行内元素的水平居中

```
    <style type="text/CSS">
    .wrap{background:#000; width:500px; height:100px; margin-bottom:10px; color:#fff;
text-align:center}
    </style>

    <div class=”wrap”>hello world</div>
    <div class=”wrap”><img src=”test.gif” /></div>
```

代码清单 4-55　确定宽度的块级元素的水平居中

```
    <style type="text/CSS">
    .wrap{background:#000; width:500px; height:100px}
    .test{background:red; width:200px; height:50px; margin-left:auto;
margin-right:auto}
    </style>

    <div class=”wrap”><div class=”test”></div></div>
```

代码清单 4-56 不确定宽度的块级元素的水平居中方法一

```
    <style type="text/CSS">
    ul{list-style:none; margin:0; padding:0}
    table{margin-left:auto; margin-right:auto;}
    .test li{float:left; display:inline; margin-right:5px;}
    .test a{float:left; width:20px; height:20px; text-align:center; line-height:20px;
background:#316AC5; color:#fff; border:1px solid #316AC5; text-decoration:none;}
    .test a:hover{background:#fff; color:#316AC5}
    </style>

    <div class=”wrap”>
        <table><tbody><tr><td>
                <ul class="test">
                        <li><a href="#">1</a></li>
                </ul>
        </td></tr></tbody></table>
        <table><tbody><tr><td>
                <ul class="test">
                        <li><a href="#">1</a></li>
```

```
                        <li><a href="#">2</a></li>
                        <li><a href="#">3</a></li>
                </ul>
        </td></tr></tbody></table>
        <table><tbody><tr><td>
                <ul class="test">
                        <li><a href="#">1</a></li>
                        <li><a href="#">2</a></li>
                        <li><a href="#">3</a></li>
                        <li><a href="#">4</a></li>
                        <li><a href="#">5</a></li>
                </ul>
        </td></tr></tbody></table>

    </div>
```

代码清单 4-57　不确定宽度的块级元素的水平居中方法二

```
    <style type="text/CSS">
    ul{list-style:none;margin:0;padding:0}
    .wrap{background:#000; width:500px; height:100px}
    .test{text-align:center;padding:5px;}
    .test li{display:inline;}
    .test a{padding:2px 6px;background:#316AC5;color:#fff;border:1px solid
#316AC5;text-decoration:none;}
    .test a:hover{background:#fff;color:#316AC5}
    </style>

    <div class="wrap">
    <ul class="test">
        <li><a href="#">1</a></li>
    </ul>
    <ul class="test">
        <li><a href="#">1</a></li>
        <li><a href="#">2</a></li>
        <li><a href="#">3</a></li>
    </ul>
    <ul class="test">
        <li><a href="#">1</a></li>
        <li><a href="#">2</a></li>
        <li><a href="#">3</a></li>
        <li><a href="#">4</a></li>
        <li><a href="#">5</a></li>
    </ul>
    </div>
```

代码清单 4-58　不确定宽度的块级元素的水平居中方法三

```
    <style type="text/CSS">
    ul{list-style:none; margin:0; padding:0}
    .wrap{background:#000; width:500px; height:100px}
    .test{clear:both; padding-top:5px; float:left; position:relative; left:50%;}
    .test li{float:left; display:inline; margin-right:5px; position:relative;
left:-50%;}
    .test a{float:left; width:20px; height:20px; text-align:center; line-height:20px;
background:#316AC5; color:#fff; border:1px solid #316AC5; text-decoration:none;}
    .test a:hover{background:#fff; color:#316AC5}
    </style>

    <div class="wrap">
    <ul class="test">
        <li><a href="#">1</a></li>
```

```
    </ul>
    <ul class="test">
        <li><a href="#">1</a></li>
        <li><a href="#">2</a></li>
        <li><a href="#">3</a></li>
    </ul>
    <ul class="test">
        <li><a href="#">1</a></li>
        <li><a href="#">2</a></li>
        <li><a href="#">3</a></li>
        <li><a href="#">4</a></li>
        <li><a href="#">5</a></li>
    </ul>
    </div>
```

代码清单 4-59　父元素高度不确定的文本、图片、块级元素的竖直居中

```
    <style type="text/CSS">
    .wrap{background:#000; width:500px; color:#fff; margin-bottom:10px;
padding-top:20px; padding-bottom:20px}
    .test{width:200px;height:50px;background:red;}
    </style>

    <div class="wrap">hello world</div>
    <div class="wrap"><img src="54.gif" /></div>
    <div class="wrap"><div class="test"></div></div>
```

代码清单 4-60　父元素高度确定的单行文本的竖直居中

```
    <style type="text/CSS">
    .wrap{background:#000; width:500px; color:#fff;height:100px;line-height:100px;}
    </style>

    <div class="wrap">hello world</div>
```

代码清单 4-61　父元素高度确定的多行文本、图片、块级元素的竖直居中方法一

```
    <style type="text/CSS">
    .wrap{background:#000; width:500px; color:#fff;height:100px}
    .test{width:200px;height:50px;background:red;}
    </style>

    <table><tbody><tr><td class="wrap">
    hello world<br />
    hello world<br />
    hello world
    </td></tr></tbody></table>

    <table><tbody><tr><td class="wrap">
    <img src="54.gif" />
    </td></tr></tbody></table>

    <table><tbody><tr><td class="wrap">
    <div class="test"></div>
    </td></tr></tbody></table>
```

代码清单 4-62　父元素高度确定的多行文本、图片、块级元素的竖直居中方法二

```
    <style type="text/CSS">
    .mb10{margin-bottom:10px}
    .wrap{background:#000; width:500px;
color:#fff;margin-bottom:10px;height:100px;display:table-cell;vertical-align:middle
;*position:relative}
```

```
.test{width:200px;height:50px;background:red}
.verticalWrap{*position:absolute;*top:50%}
.vertical{*position:relative;*top:-50%}
</style>
<div class="mb10">
    <div class="wrap">
        <div class="verticalWrap">
            <div class="vertical">
                hello world<br />
                hello world<br />
                hello world
            </div>
        </div>
    </div>
</div>
<div class="mb10">
    <div class="wrap">
        <div class="verticalWrap">
            <img src="54.gif" class="vertical" />
        </div>
    </div>
</div>
<div class="mb10">
    <div class="wrap">
        <div class="verticalWrap">
            <div class="test vertical"></div>
        </div>
    </div>
</div>
```

4.7.10 网格布局

代码清单 4-63　两栏式布局方案一

```
<style type="text/CSS">
.header{}
.footer{clear:both}
.sidebar{width:25%; float:left}
.main{width:70%; float:right}
</style>

<div class=”header”></div>
<div class=”content”>
    <div class=”sidebar”></div>
    <div class=”main”></div>
</div>
<div class=”footer”></div>
```

代码清单 4-64　两栏式布局方案二

```
<style type="text/CSS">
.header{}
.footer{clear:both}
.sidebar{width:25%; float:left}
.main{width:70%; float:right}
</style>

<div class=”header”></div>
<div class=”content”>
    <div class=”main”></div>
    <div class=”sidebar”></div>
```

```
</div>
<div class=”footer”></div>
```

代码清单 4-65　调换左右两栏的位置

```
<style type="text/CSS">
.header{}
.footer{clear:both}
.sidebar{width:25%; float:right}
.main{width:70%; float:left}
</style>

<div class=”header”></div>
<div class=”content”>
     <div class=”main”></div>
     <div class=”sidebar”></div>
</div>
<div class=”footer”></div>
```

代码清单 4-66　用类的组合的方式设定浮动方向

```
<style type="text/CSS">
.fl{float:left}
.fr{float:right}
.header{}
.footer{clear:both}
.sidebar{width:25%}
.main{width:70%}
</style>

<div class=”header”></div>
<div class=”content”>
     <div class=”main fr”></div>
     <div class=”sidebar fl”></div>
</div>
<div class=”footer”></div>
```

代码清单 4-67　用类的组合的方式面对简单修改

```
<style type="text/CSS">
.fl{float:left}
.fr{float:right}
.header{}
.footer{clear:both}
.sidebar{width:25%}
.main{width:70%}
</style>

<div class=”header”></div>
<div class=”content”>
     <div class=”main fl”></div>
     <div class=”sidebar fr”></div>
</div>
<div class=”footer”></div>
```

代码清单 4-68　用类的组合的方式面对复杂修改

```
<style type="text/CSS">
.fl{float:left}
.fr{float:right}
.content{clear:both}
.header{}
.footer{clear:both}
```

```
.sidebar{width:25%}
.main{width:70%}
</style>

<div class="header"></div>
<div class="content">
    <div class="main fl"></div>
    <div class="sidebar fr"></div>
</div>
<div class="content">
    <div class="main fr"></div>
    <div class="sidebar fl"></div>
</div>
<div class="footer"></div>
```

代码清单 4-69　将宽度提取出来

```
<style type="text/CSS">
.fl{float:left}
.fr{float:right}
.content{clear:both}
.header{}
.footer{clear:both}
.sidebar{}
.main{}
.w25{width:25%}
.w70{width:70%}
.w35{width:35%}
.w60{width:60%}
</style>

<div class="header"></div>
<div class="content">
    <div class="main fl w70"></div>
    <div class="sidebar fr w25"></div>
</div>
<div class="content">
    <div class="main fr w60"></div>
    <div class="sidebar fl w35"></div>
</div>
<div class="footer"></div>
```

代码清单 4-70　用子选择器应对复杂变化

```
<style type="text/CSS">
.content{clear:both}
.header{}
.footer{clear:both}
.main{}
.sidebar{}
.content-lr-7025 .main{float:left; width:70%}
.content-lr-7025 .sidebar{float:right; width:25%}
.content-rl-7025 .main{float:right; width:70%}
.content-rl-7025 .sidebar{float:left; width:25%}
.content-lr-6035 .main{float:left; width:60%}
.content-lr-6035 .sidebar{float:right; width:35%}
.content-rl-6035 .main{float:right; width:60%}
.content-rl-6035 .sidebar{float:left; width:35%}
</style>

<div class="header"></div>
<div class="content content-lr-7025">
```

```
        <div class="main"></div>
        <div class="sidebar"></div>
    </div>
    <div class="content content-rl-6035">
        <div class="main"></div>
        <div class="sidebar"></div>
    </div>
    <div class="footer"></div>
```

代码清单 4-71　组合类的方式面对扩展

```
    <style type="text/CSS">
    .fl{float:left}
    .fr{float:right}
    .content{clear:both}
    .header{}
    .footer{clear:both}
    .sidebar{}
    .main{}
    .w25{width:25%}
    .w70{width:70%}
    .w35{width:35%}
    .w60{width:60%}
    .w80{width:80%}
    .w15{width:15%}
    </style>

    <div class="header"></div>
    <div class="content">
        <div class="main fl w70"></div>
        <div class="sidebar fr w25"></div>
    </div>
    <div class="content">
        <div class="main fr w60"></div>
        <div class="sidebar fl w35"></div>
    </div>
    <div class="content">
        <div class="main fl w80"></div>
        <div class="sidebar fr w15"></div>
    </div>
    <div class="footer"></div>
```

代码清单 4-72　子选择器方式面对扩展

```
    <style type="text/CSS">
    .content{clear:both}
    .header{}
    .footer{clear:both}
    .main{}
    .sidebar{}
    .content-lr-7025 .main{float:left; width:70%}
    .content-lr-7025 .sidebar{float:right; width:25%}
    .content-rl-7025 .main{float:right; width:70%}
    .content-rl-7025 .sidebar{float:left; width:25%}
    .content-lr-6035 .main{float:left; width:60%}
    .content-lr-6035 .sidebar{float:right; width:35%}
    .content-rl-6035 .main{float:right; width:60%}
    .content-rl-6035 .sidebar{float:left; width:35%}
    .content-lr-8015 .main{float:left; width:80%}
    .content-lr-8015 .sidebar{float:right; width:15%}
    .content-rl-8015 .main{float:right; width:80%}
    .content-rl-8015 .sidebar{float:left; width:15%}
```

```
    </style>

    <div class="header"></div>
    <div class="content content-lr-7025">
        <div class="main"></div>
        <div class="sidebar"></div>
    </div>
    <div class="content content-rl-6035">
        <div class="main"></div>
        <div class="sidebar"></div>
    </div>
    <div class="content content-lr-8015">
        <div class="main"></div>
        <div class="sidebar"></div>
    </div>
    <div class="footer"></div>
```

代码清单 4-73　子选择器布局的嵌套使用

```
    <style type="text/CSS">
    .content{clear:both}
    .header{}
    .footer{clear:both}
    .main{}
    .sidebar{}
    .content-lr-7025 .main{float:left; width:70%}
    .content-lr-7025 .sidebar{float:right; width:25%}
    .content-rl-7025 .main{float:right; width:70%}
    .content-rl-7025 .sidebar{float:left; width:25%}
    </style>

    <div class="header"></div>
    <div class="content content-lr-7025">
        <div class="main content-lr-7025">
                <div class="main"></div>
                <div class="sidebar"></div>
        </div>
        <div class="sidebar"></div>
    </div>
    <div class="footer"></div>
```

4.7.11 z-index 和 Flash、IE 6 下的 select

代码清单 4-74　z-index 设置高低

```
    <style type="text/CSS">
    #one{width:300px;height:300px; background:black}
    #two{width:100px;height:100px; background:red;position:absolute; z-index:1;
left:100px; top:250px}
    #three{width:100px;height:100px; background:green; position:relative; z-index:2;
left:120px; top:-100px}
    </style>

    <div id="one"></div>
    <div id="two"></div>
    <div id="three"></div>
```

代码清单 4-75　调整 z-index 的高低

```
    <style type="text/CSS">
    #one{width:300px; height:300px; background:black}
```

```
    #two{width:100px; height:100px; background:red; position:absolute; z-index:3;
left:100px; top:250px}
    #three{width:100px; height:100px; background:green; position:relative; z-index:2;
left:120px; top:-100px}
    </style>

    <div id="one"></div>
    <div id="two"></div>
    <div id="three"></div>
```

代码清单 4-76　z-index 设为负数

```
    <style type="text/CSS">
    #one{width:300px;height:300px; background:black}
    #two{width:100px;height:100px; background:red;position:absolute; z-index:-1;
left:100px; top:250px}
    #three{width:100px;height:100px; background:green; position:relative; z-index:2;
left:120px; top:-100px}
    </style>

    <div id="one"></div>
    <div id="two"></div>
    <div id="three"></div>
```

代码清单 4-77　为 one、two、three 监听 click 事件

```
    <script type="text/JavaScript">
    var one = document.getElementById("one"), two = document.getElementById("two"),
three = document.getElementById("three");
    one.onclick = function(){
        alert("one");
    }
    two.onclick = function(){
        alert("two");
    }
    three.onclick = function(){
        alert("three");
    }
```

代码清单 4-78　z-index 值相同

```
    <style type="text/CSS">
    #two{width:100px;height:100px; background:red;position:absolute; z-index:2;
left:100px; top:250px}
    #three{width:100px;height:100px; background:green; position:relative; z-index:2;
left:120px; top:-100px}
    </style>
    <div id="two"></div>
    <div id="three"></div>
```

代码清单 4-79　负边距引起的竖直重叠

```
    <style type="text/CSS">
    #one{width:300px; height:300px; background:black}
    #two{width:100px; height:100px; background:red; margin-top:-50px}
    </style>

    <div id="one"></div>
    <div id="two"></div>
```

代码清单 4-80　负边距引起的水平重叠

```
    <style type="text/CSS">
    #one{width:300px; height:300px; background:black; float:left}
```

```
    #two{width:100px; height:100px; background:red; display:inline; float:left;
margin-left:-50px;}
    </style>

    <div id="one"></div>
    <div id="two"></div>
```

代码清单 4-81　设置 flash 的显示模式

```
    <object width="640" height="90" type="application/x-shockwave-Flash"
classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000">
        <param value="xxxxxx.swf" name="movie"></param>
        <param value="opaque" name="wmode"></param>
        <embed width="640" height="90" wmode="opaque"
type="application/x-shockwave-Flash"  src="xxxxxx.swf.swf"></embed>
    </object>
```

代码清单 4-82　select 表单元素

```
    <style type="text/CSS">
    #test{width:200px; height:200px; background:green; position:absolute; left:50px;
top:10px;}
    </style>
    <select><option>-请选择-</option></select>
    <div id="test"></div>
```

代码清单 4-83　用 iframe 解决 IE6 下 select 与浮起元素的显示 bug

```
    <style type="text/CSS">
    #test{width:200px; height:200px; background:green; position:absolute; left:50px;
top:10px; z-index:2}
    #testMask{width:200px; height:200px; position:absolute; left:50px; top:10px;
z-index:1;}
    </style>
    <select><option>-请选择-</option></select>
    <div id="test"></div>
    <iframe id="testMask" frameborder="0" scrolling="no"></iframe>
```

4.7.12 IE 6 和 png 图片

代码清单 4-84　插入 png 图片

```
    <img src=”test.png” />
```

代码清单 4-85　一次性解决 png 透明度在 IE6 下显示问题的脚本

```
    function correctPNG()
   {
   for(var i=0; i<document.images.length; i++)
      {
     var img = document.images[i]
     var imgName = img.src.toUpperCase()
     if (imgName.substring(imgName.length-3, imgName.length) == "PNG")
        {
       var imgID = (img.id) ? "id='" + img.id + "' " : ""
       var imgClass = (img.className) ? "class='" + img.className + "' " : ""
       var imgTitle = (img.title) ? "title='" + img.title + "' " : "title='" + img.alt
+ "' "
       var imgStyle = "display:inline-block;" + img.style.cssText
       if (img.align == "left") imgStyle = "float:left;" + imgStyle
       if (img.align == "right") imgStyle = "float:right;" + imgStyle
       if (img.parentElement.href) imgStyle = "cursor:hand;" + imgStyle
       var strNewHTML = "<span " + imgID + imgClass + imgTitle
       + " style=\"" + "width:" + img.width + "px; height:" + img.height + "px;" + imgStyle
```

```
+ ";"
        + "filter:progid:DXImageTransform.Microsoft.AlphaImageLoader"
       + "(src=\'" + img.src + "\', sizingMethod='scale');\"></span>"
       img.outerHTML = strNewHTML
       i = i-1
        }
      }
   }
function alphaBackgrounds(){
   var rslt = navigator.appVersion.match(/MSIE (d+.d+)/, '');
   var itsAllGood = (rslt != null && Number(rslt[1]) >= 5.5);
   for (i=0; i<document.all.length; i++){
      var bg = document.all[i].currentStyle.backgroundImage;
      if (bg){
         if (bg.match(/.png/i) != null){
            var mypng = bg.substring(5,bg.length-2);
            document.all[i].style.filter =
"progid:DXImageTransform.Microsoft.AlphaImageLoader(src='"+mypng+"',
sizingMethod='crop')";
            document.all[i].style.backgroundImage = "url('')";
         }
      }
   }
}
if (navigator.platform == "Win32" && navigator.appName == "Microsoft Internet Explorer"
&& window.attachEvent) {
window.attachEvent("onload", correctPNG);
window.attachEvent("onload", alphaBackgrounds);
}
```

4.7.13 多版本 IE 并存方案——CSS 的调试利器 IETester

第 5 章 高品质的 JavaScript

本章内容

- 养成良好的编程习惯
- JavaScript 分层概念和 JavaScript 库
- 编程实用技巧
- 面向对象编程
- 其他问题

5.1养成良好的编程习惯

5.1.1 团队合作——如何避免 js 冲突

代码清单 5-1 工程师甲编写功能 A

```
<div>
    xxxxxxxxxx
</div>
<script type="text/Javascript">
var a = 123 , b = "hello world";
…
</script>
<div>
    xxxxxxxxx
</div>
```

代码清单 5-2 工程师乙添加功能 B

```
<div>
    xxxxxxxxxx
</div>
<script type="text/JavaScript">
var a = 123 , b = "hello world";
…
</script>
<div>
    xxxxxxxxx
</div>
<script type="text/JavaScript">
var a , c = "abc";
…
</script>
<div>
    xxxxxxxxx
</div>
```

代码清单 5-3 使用匿名函数控制变量的作用域

```
<div>
    xxxxxxxxxx
</div>
<script type="text/JavaScript">
```

```
    (function(){
        var a = 123 , b = “hello world”;
        …
    })();
    </script>
    <div>
        xxxxxxxxxx
    </div>
    <script type=”text/JavaScript”>
    (function(){
        var a , c = ”abc”;
        …
    })();
    </script>
    <div>
        xxxxxxxxxx
    </div>
```

代码清单 5-4　功能 C 和功能 A 的通信

```
    <div>
        xxxxxxxxxxx
    </div>
    <script type=”text/JavaScript”>
    (function(){
        var a = 123 , b = “hello world”;
        …
        var d = “adang is very handsome!”;
        d = b + “, ” + d;
        …
    })();
    </script>
    <div>
        xxxxxxxxxx
    </div>
    <script type=”text/JavaScript”>
    (function(){
        var a , c = ”abc”;
        …
    })();
    </script>
    <div>
        xxxxxxxxxx
    </div>
```

代码清单 5-5　工程师丙添加功能 C

```
    <div>
        xxxxxxxxxxx
    </div>
    <script type=”text/JavaScript”>
    (function(){
        var a = 123 , b = “hello world”;
        …
    })();
    </script>
    <div>
        xxxxxxxxxx
    </div>
    <script type=”text/JavaScript”>
    (function(){
        var a , c = ”abc”;
```

```
    …
})();
</script>
<div>
    xxxxxxxxx
</div>
<script type="text/JavaScript">
(function(){
    var d = "adang is very handsome!";
    d = b + ", " + d;
    …
})();
</script>
```

代码清单 5-6　利用全局作用域的变量在各匿名函数间搭起桥梁

```
<div>
    xxxxxxxxxx
</div>
<script type="text/JavaScript">
var str;
</script>
<script type="text/JavaScript">
(function(){
    var a = 123 , str = b = "hello world";
    …
})();
</script>
<div>
    xxxxxxxxx
</div>
<script type="text/JavaScript">
(function(){
    var a , c = "abc";
    …
})();
</script>
<div>
    xxxxxxxxx
</div>
<script type="text/JavaScript">
(function(){
    var b = str;
    var d = "adang is very handsome!";
    d = b + ", " + d;
    …
})();
</script>
```

代码清单 5-7　添加新的全局变量

```
<div>
    xxxxxxxxxx
</div>
<script type="text/JavaScript">
var str,str2;
</script>
<script type="text/JavaScript">
(function(){
    var str2 = a = 123 , str = b = "hello world";
    …
})();
```

```
</script>
<div>
    xxxxxxxxxx
</div>
<script type="text/JavaScript">
(function(){
    var a , c = "abc";
    …
})();
</script>
<div>
    xxxxxxxxxx
</div>
<script type="text/JavaScript">
(function(){
    var a = str2 , b = str;
    var d = "adang is very handsome!";
    d = b + ", " + d + a;
    …
})();
</script>
```

代码清单 5-8　用 hash 对象作为全局变量

```
<div>
    xxxxxxxxxxx
</div>
<script type="text/JavaScript">
var GLOBAL = {};
</script>
<script type="text/JavaScript">
(function(){
    var a = 123 , b = "hello world";
    GLOBAL.str2 = a;
    GLOBAL.str = b;
    …
})();
</script>
<div>
    xxxxxxxxxx
</div>
<script type="text/JavaScript">
(function(){
    var a , c = "abc";
    …
})();
</script>
<div>
    xxxxxxxxxx
</div>
<script type="text/JavaScript">
(function(){
    var a = GLOBAL.str2 , b = GLOBAL.str;
    var d = "adang is very handsome!";
    d = b + ", " + d + a;
    …
})();
</script>
```

代码清单 5-9　全局变量的冲突

```
<div>
```

```
    xxxxxxxxxx
</div>
<script type="text/JavaScript">
var GLOBAL = {};
</script>
<script type="text/JavaScript">
(function(){
    var a = 123 , b = "hello world";
    GLOBAL.str2 = a;
    GLOBAL.str = b;
    …
})();
</script>
<div>
    xxxxxxxxxx
</div>
<script type="text/JavaScript">
(function(){
    var a , c = "abc";
    GLOBAL.str = c;
    …
})();
</script>
<div>
    xxxxxxxxxx
</div>
<script type="text/JavaScript">
(function(){
    var a = GLOBAL.str2 , b = GLOBAL.str;
    var d = "adang is very handsome!";
    d = b + ", " + d + a;
    …
})();
</script>
<div>
    xxxxxxxxxx
</div>
<script type="text/JavaScript">
(function(){
    var test = GLOBAL.str;
    alert(test);
    …
})();
</script>
```

代码清单 5-10　使用命名空间

```
<div>
    xxxxxxxxxx
</div>
<script type="text/JavaScript">
var GLOBAL = {};
</script>
<script type="text/JavaScript">
(function(){
    var a = 123 , b = "hello world";
    GLOBAL.A = {};
    GLOBAL.A.str2 = a;
    GLOBAL.A.str = b;
    …
})();
```

```
</script>
<div>
    xxxxxxxxxx
</div>
<script type="text/JavaScript">
(function(){
    var a , c = "abc";
    GLOBAL.B = {};
    GLOBAL.B.str = c;
    …
})();
</script>
<div>
    xxxxxxxxxx
</div>
<script type="text/JavaScript">
(function(){
    var a = GLOBAL.A.str2 , b = GLOBAL.A.str;
    var d = "adang is very handsome!";
    d = b + ", " + d + a;
    …
})();
</script>
<div>
    xxxxxxxxxx
</div>
<script type="text/JavaScript">
(function(){
    var test = GLOBAL.B.str;
    alert(test);
    …
})();
</script>
```

代码清单 5-11　使用多级命名空间

```
<div>
    xxxxxxxxxxx
</div>
<script type="text/JavaScript">
var GLOBAL = {};
</script>
<script type="text/JavaScript">
(function(){
    var a = 123 , b = "hello world";
    GLOBAL.A = {};
    GLOBAL.A.CAT = {};
    GLOBAL.A.DOG = {};
    GLOBAL.A.CAT.name = "mimi";
    GLOBAL.A.DOG.name = "wangcai";
    GLOBAL.A.CAT.move = function(){

    }
    GLOBAL.A.DOG.move = function(){

    }
    GLOBAL.A.str2 = a;
    GLOBAL.A.str = b;
    …
})();
</script>
```

代码清单 5-12　定义命名空间函数

```
<div>
     xxxxxxxxxx
</div>
<script type=”text/JavaScript”>
var GLOBAL = {};
GLOBAL.namespace = function(str){
     var arr = str.split(".“),o = GLOBAL;
     for (i=(arr[0] == "GLOBAL") ? 1 : 0; i<arr.length; i++) {
             o[arr[i]]=o[arr[i]] || {};
       o=o[arr[i]];
    }
}
</script>
<script type=”text/JavaScript”>
(function(){
     var a = 123 , b = “hello world”;
     GLOBAL.namespace(“A.CAT”);
     GLOBAL.namespace(“A.DOG”);
     GLOBAL.A.CAT.name = “mimi”;
     GLOBAL.A.DOG.name = “wangcai”;
     GLOBAL.A.CAT.move = function(){

     }
     GLOBAL.A.DOG.move = function(){

     }
     GLOBAL.A.str2 = a;
     GLOBAL.A.str = b;
     …
})();
</script>
```

代码清单 5-13　使用命名空间解决冲突的完整代码

```
<div>
     xxxxxxxxxx
</div>
<script type=”text/JavaScript”>
var GLOBAL = {};
GLOBAL.namespace = function(str){
     var arr = str.split(".“),o = GLOBAL;
     for (i=(arr[0] == "GLOBAL") ? 1 : 0; i<arr.length; i++) {
             o[arr[i]]=o[arr[i]] || {};
       o=o[arr[i]];
    }
}
</script>
<script type=”text/JavaScript”>
(function(){
     var a = 123 , b = “hello world”;
     GLOBAL.namespace(“A.CAT”);
     GLOBAL.namespace(“A.DOG”);
     GLOBAL.A.CAT.name = “mimi”;
     GLOBAL.A.DOG.name = “wangcai”;
     GLOBAL.A.CAT.move = function(){

     }
     GLOBAL.A.DOG.move = function(){
```

```
        }
        GLOBAL.A.str2 = a;
        GLOBAL.A.str = b;
        …
    })();
    </script>
    <div>
        xxxxxxxxxx
    </div>
    <script type=”text/JavaScript”>
    (function(){
        var a , c = ”abc”;
        GLOBAL.namespace(“B”);
        GLOBAL.B.str = c;
        …
    })();
    </script>
    <div>
        xxxxxxxxxx
    </div>
    <script type=”text/JavaScript”>
    (function(){
        var a = GLOBAL.A.str2 , b = GLOBAL.A.str;
        var d = “adang is very handsome!”;
        d = b + “, ” + d + a;
        …
    })();
    </script>
    <div>
        xxxxxxxxxx
    </div>
    <script type=”text/JavaScript”>
    (function(){
        var test = GLOBAL.B.str;
        alert(test);
        …
    })();
    </script>
```

代码清单 5-14　给代码添加注释

```
    <div>
        xxxxxxxxxxx
    </div>
    <script type=”text/JavaScript”>
    var GLOBAL = {};
    GLOBAL.namespace = function(str){
        var arr = str.split("."),o = GLOBAL;
        for (i=(arr[0] == "GLOBAL") ? 1 : 0; i<arr.length; i++) {
                o[arr[i]]=o[arr[i]] || {};
          o=o[arr[i]];
       }
    }
    </script>
    <script type=”text/JavaScript”>
    //=====================================================
    //   功能 A
    //   工程师甲
    //   email:cly84920@gmail.com  msn:cly84920@hotmail.com
    //   2009-11-04
    //=====================================================
```

```
(function(){
    var a = 123 , b = “hello world”;
    GLOBAL.namespace(“A.CAT”);
    GLOBAL.namespace(“A.DOG”);
    GLOBAL.A.CAT.name = “mimi”;
    GLOBAL.A.DOG.name = “wangcai”;
    GLOBAL.A.CAT.move = function(){

    }
    GLOBAL.A.DOG.move = function(){

    }
    GLOBAL.A.str2 = a;
    GLOBAL.A.str = b;
    …
})();
</script>
<div>
    xxxxxxxxx
</div>
<script type=”text/JavaScript”>
//=====================================================
//   功能 B
//   工程师乙
//   email:a@gmail.com        msn:a@hotmail.com
//   2009-11-05
//=====================================================
(function(){
    var a , c = ”abc”;
    GLOBAL.namespace(“B”);
    GLOBAL.B.str = c;
    …
})();
</script>
<div>
    xxxxxxxxx
</div>
<script type=”text/JavaScript”>
//=====================================================
//   功能 C
//   工程师丙
//   email:b@yahoo.com        msn:b@hotmail.com
//   2009-11-09
//=====================================================
(function(){
    var a = GLOBAL.A.str2 , b = GLOBAL.A.str;
    var d = “adang is very handsome!”;
    d = b + “, ” + d + a;
    …
})();
</script>
<div>
    xxxxxxxxx
</div>
<script type=”text/JavaScript”>
//=====================================================
//   功能 D
//   工程师丁
//   email:c@126.com  msn:c@hotmail.com
//   2009-11-20
```

```
//==================================================
(function(){
     var test = GLOBAL.B.str;
     alert(test);
     …
})();
</script>
```

5.1.2　给程序一个统一的入口——window.onload 和 DOMReady

代码清单 5-15　区分 javascript 的框架部分和应用部分

```
<div>
     xxxxxxxxxx
</div>
<script type=”text/JavaScript”>
var GLOBAL = {};
GLOBAL.namespace = function(str){
     var arr = str.split("."),o = GLOBAL;
     for (i=(arr[0] == "GLOBAL") ? 1 : 0; i<arr.length; i++) {
             o[arr[i]]=o[arr[i]] || {};
       o=o[arr[i]];
   }
}
</script>
<script type=”text/JavaScript”>
//==================================================
//   功能 A
//   工程师甲
//   email:cly84920@gmail.com  msn:cly84920@hotmail.com
//   2009-11-04
//==================================================
(function(){
     var a = 123 , b = “hello world”;
     GLOBAL.namespace(“A.CAT”);
     GLOBAL.namespace(“A.DOG”);
     GLOBAL.A.CAT.name = “mimi”;
     GLOBAL.A.DOG.name = “wangcai”;
     GLOBAL.A.CAT.move = function(){

     }
     GLOBAL.A.DOG.move = function(){

     }
     GLOBAL.A.str2 = a;
     GLOBAL.A.str = b;
     …
})();
</script>
<div>
     xxxxxxxxxx
</div>
<script type=”text/JavaScript”>
//==================================================
//   功能 B
//   工程师乙
//   email:a@gmail.com         msn:a@hotmail.com
//   2009-11-05
//==================================================
(function(){
```

```
        var a , c = ”abc”;
        GLOBAL.namespace(“B”);
        GLOBAL.B.str = c;
        …
    })();
    </script>
    <div>
        xxxxxxxxx
    </div>
    <script type=”text/JavaScript”>
    //=====================================================
    //   功能C
    //   工程师丙
    //   email:b@yahoo.com         msn:b@hotmail.com
    //   2009-11-09
    //=====================================================
    (function(){
        var a = GLOBAL.A.str2 , b = GLOBAL.A.str;
        var d = “adang is very handsome!”;
        d = b + “, ” + d + a;
        …
    })();
    </script>
    <div>
        xxxxxxxxx
    </div>
    <script type=”text/JavaScript”>
    //=====================================================
    //   功能D
    //   工程师丁
    //   email:c@126.com  msn:c@hotmail.com
    //   2009-11-20
    //=====================================================
    (function(){
        var test = GLOBAL.B.str;
        alert(test);
        …
    })();
    </script>
```

代码清单 5-16　给应用部分一个统一的“入口”

```
    <div>
        xxxxxxxxxx
    </div>
    <script type=”text/JavaScript”>
    var GLOBAL = {};
    GLOBAL.namespace = function(str){
        var arr = str.split("."),o = GLOBAL;
        for (i=(arr[0] == "GLOBAL") ? 1 : 0; i<arr.length; i++) {
                o[arr[i]]=o[arr[i]] || {};
          o=o[arr[i]];
       }
    }
    </script>
    <script type=”text/JavaScript”>
    function init(){
    //=====================================================
    //   功能A
    //   工程师甲
    //   email:cly84920@gmail.com  msn:cly84920@hotmail.com
```

```
    //   2009-11-04
    //=====================================================
    (function(){
        var a = 123 , b = “hello world”;
        GLOBAL.namespace(“A.CAT”);
        GLOBAL.namespace(“A.DOG”);
        GLOBAL.A.CAT.name = “mimi”;
        GLOBAL.A.DOG.name = “wangcai”;
        GLOBAL.A.CAT.move = function(){

        }
        GLOBAL.A.DOG.move = function(){

        }
        GLOBAL.A.str2 = a;
        GLOBAL.A.str = b;
        …
    })();

    //=====================================================
    //   功能B
    //   工程师乙
    //   email:a@gmail.com        msn:a@hotmail.com
    //   2009-11-05
    //=====================================================
    (function(){
        var a , c = ”abc”;
        GLOBAL.namespace(“B”);
        GLOBAL.B.str = c;
        …
    })();

    //=====================================================
    //   功能C
    //   工程师丙
    //   email:b@yahoo.com        msn:b@hotmail.com
    //   2009-11-09
    //=====================================================
    (function(){
        var a = GLOBAL.A.str2 , b = GLOBAL.A.str;
        var d = “adang is very handsome!”;
        d = b + “, ” + d + a;
        …
    })();

    //=====================================================
    //   功能D
    //   工程师丁
    //   email:c@126.com  msn:c@hotmail.com
    //   2009-11-20
    //=====================================================
    (function(){
        var test = GLOBAL.B.str;
        alert(test);
        …
    })();
    }
    </script>
    <div>
        xxxxxxxxx
```

```
    </div>
    <div>
        xxxxxxxxxx
    </div>
    <div>
        xxxxxxxxxx
    </div>
```

代码清单 5-17　在 DOM 节点加载进来之前就调用

```
    <script type="text/JavaScript">
    alert(document.getElementById("test").innerHTML);
    </script>
    <div id="test">hello world</div>
```

代码清单 5-18　调整脚本的位置

```
    <div id="test">hello world</div>
    <script type="text/JavaScript">
    alert(document.getElementById("test").innerHTML);
    </script>
```

代码清单 5-19　监听 window.onload

```
    <script type="text/JavaScript">
    window.onload = function(){
        alert(document.getElementById("test").innerHTML);
    }
    </script>
    <div id="test">hello world</div>
```

代码清单 5-20　定义初始化方法的名称为 init

```
    <script type="text/JavaScript">
    function init(){
        alert(document.getElementById("test").innerHTML);
    }
    window.onload = init;
    </script>
    <div id="test">hello world</div>
```

代码清单 5-21　jQuery 中监听 DOMReady

```
    <script type="text/JavaScript"
src="http://ajax.googleAPIs.com/ajax/libs/jquery/1.3/jquery.min.js"></script>
    <script type="text/JavaScript">
    function init(){
        alert(document.getElementById("test").innerHTML);
    }
    $(document).ready(init);
    </script>
    <div id="test">hello world</div>
```

代码清单 5-22　YUI 中监听 DOMReady

```
    <script type="text/JavaScript"
src="http://yui.yahooAPIs.com/combo?2.8.0r4/build/yahoo-dom-event/yahoo-dom-event.j
s"></script>
    <script type="text/JavaScript">
    function init(){
        alert(document.getElementById("test").innerHTML);
    }
    YAHOO.util.Event.onDOMReady(init);
    </script>
    <div id="test">hello world</div>
```

代码清单 5-23　模拟 DOMReady 效果

```
<script type="text/JavaScript">
function init(){
    alert(document.getElementById("test").innerHTML);
}
</script>
<div id="test">hello world</div>
…
<script type="text/JavaScript">
init();
</script>
</body>
```

代码清单 5-24　调用 init 函数

```
<div>
    xxxxxxxxxx
</div>
<script type=”text/JavaScript”>
var GLOBAL = {};
GLOBAL.namespace = function(str){
    var arr = str.split("."),o = GLOBAL;
    for (i=(arr[0] == "GLOBAL") ? 1 : 0; i<arr.length; i++) {
            o[arr[i]]=o[arr[i]] || {};
      o=o[arr[i]];
   }
}
</script>
<script type=”text/JavaScript”>
function init(){
//=====================================================
//   功能A
//   工程师甲
//   email:cly84920@gmail.com  msn:cly84920@hotmail.com
//   2009-11-04
//=====================================================
(function(){
    var a = 123 , b = “hello world”;
    GLOBAL.namespace(“A.CAT”);
    GLOBAL.namespace(“A.DOG”);
    GLOBAL.A.CAT.name = “mimi”;
    GLOBAL.A.DOG.name = “wangcai”;
    GLOBAL.A.CAT.move = function(){

    }
    GLOBAL.A.DOG.move = function(){

    }
    GLOBAL.A.str2 = a;
    GLOBAL.A.str = b;
    …
})();

//=====================================================
//   功能B
//   工程师乙
//   email:a@gmail.com         msn:a@hotmail.com
//   2009-11-05
//=====================================================
(function(){
```

```
        var a , c = ”abc”;
        GLOBAL.namespace(“B”);
        GLOBAL.B.str = c;
        …
    })();

    //=====================================================
    //   功能C
    //   工程师丙
    //   email:b@yahoo.com         msn:b@hotmail.com
    //   2009-11-09
    //=====================================================
    (function(){
        var a = GLOBAL.A.str2 , b = GLOBAL.A.str;
        var d = “adang is very handsome!”;
        d = b + “, ” + d + a;
        …
    })();

    //=====================================================
    //   功能D
    //   工程师丁
    //   email:c@126.com  msn:c@hotmail.com
    //   2009-11-20
    //=====================================================
    (function(){
        var test = GLOBAL.B.str;
        alert(test);
        …
    })();
    }
    </script>
    <div>
        xxxxxxxxxx
    </div>
    <div>
        xxxxxxxxxx
    </div>
    <div>
        xxxxxxxxxx
    </div>
    …
    <script type="text/JavaScript">
    init();
    </script>
    </body>
```

代码清单 5-25　框架部分的代码

```
    <div>
        xxxxxxxxxxx
    </div>
    <script type=”text/JavaScript”>
    var GLOBAL = {};
    GLOBAL.namespace = function(str){
        var arr = str.split("."),o = GLOBAL;
        for (i=(arr[0] == "GLOBAL") ? 1 : 0; i<arr.length; i++) {
                o[arr[i]]=o[arr[i]] || {};
          o=o[arr[i]];
       }
    }
```

```
    </script>
    <script type=”text/JavaScript”>
    function init(){
        //应用部分的 JavaScript
    }
    </script>
    <div>
        xxxxxxxxxx
    </div>
    <div>
        xxxxxxxxxx
    </div>
    <div>
        xxxxxxxxxx
    </div>
    …
    <script type="text/JavaScript">
    init();
    </script>
    </body>
```

代码清单 5-26　头部文件

```
    <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
    <html>
    <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>阿当制作</title>
    <link rel="stylesheet" type="text/CSS" href="xxx.css" />
    </head>
    <body>
    <div class="head">
        ...
    </div>
    <script type=”text/JavaScript”>
    var GLOBAL = {};
    GLOBAL.namespace = function(str){
        var arr = str.split("."),o = GLOBAL;
        for (i=(arr[0] == "GLOBAL") ? 1 : 0; i<arr.length; i++) {
                o[arr[i]]=o[arr[i]] || {};
          o=o[arr[i]];
        }
    }
    </script>
```

代码清单 5-27　尾部文件

```
    <div class="foot">
        ...
    </div>
    <script type=”text/JavaScript”>
        init();
    </script>
```

代码清单 5-28　主体文件

```
    <div>...</div>
    <script type=”text/JavaScript”>
        function init(){
                //应用部分的 JavaScript
        }
    </script>
```

代码清单 5-29　更健壮的 init 调用

```
    <div class="foot">
        ...
    </div>
    <script type=”text/JavaScript”>
        if(init){
                init();
        }
    </script>
```

5.1.3　CSS 放在页头，JavaScript 放在页尾

代码清单 5-30　有 JavaScript 和 CSS 的网页

```
    <script type=”text/JavaScript”>
    window.onload = function(){
        //非常多的 js 代码
        …
    }
    </script>
    <div id=”test”>xxxxx</div>
    <!—非常多的代码-->
    …
    <style type=”text/CSS”>
    #test{}
    </style>
```

代码清单 5-31　将 CSS 放在页头，JavaScript 放在页尾

```
    <style type=”text/CSS”>
    #test{}
    </style>
    <div id=”test”>xxxxx</div>
    <!—非常多的代码-->
    …
    <script type=”text/JavaScript”>
    window.onload = function(){
        //非常多的 JavaScript 代码
        …
    }
    </script>
```

5.1.4　引入编译的概念——文件压缩

代码清单 5-32　未压缩的 JavaScript 脚本

```
    //定义两个字符串
    var testStr = "hello";
    var testStr2 = "world";
    //连接字符串函数
    function addStr(str,str2){
       return str + " " + str2;
    }
    //调用函数，连接定义的两个字符串
    addStr(testStr,testStr2);
```

代码清单 5-33　压缩后的 JavaScript 脚本

```
    var testStr="hello";var testStr2="world";function addStr(a,b){return a+"
"+b}addStr(testStr,testStr2);
```

5.2 JavaScript 的分层概念和 JavaScript 库

5.2.1 JavaScript 如何分层

5.2.2 base 层

代码清单 5-34　简单的代码

```
<ul>
    <li id="item1"></li>
    <li id="item2"></li>
    <li id="item3"></li>
</ul>
<script type="text/JavaScript">
    var item1 = document.getElementById("item1");
    alert(item1.nextSibling.id);
</script>
```

代码清单 5-35　查看 IE 和 firefox 下的区别

```
<ul>
    <li id="item1"></li>
    <li id="item2"></li>
    <li id="item3"></li>
</ul>
<script type="text/JavaScript">
    var item1 = document.getElementById("item1");
    alert(item1.nextSibling.nodeType);
    alert(document.getElementsByTagName("ul")[0].childNodes.length);
</script>
```

代码清单 5-36　去掉空格

```
<ul><li id="item1"></li><li id="item2"></li><li id="item3"></li></ul>
<script type="text/JavaScript">
    var item1 = document.getElementById("item1");
    alert(item1.nextSibling.id);
</script>
```

代码清单 5-37　分别对 IE 和 firefox 写程序

```
<ul>
    <li id="item1"></li>
    <li id="item2"></li>
    <li id="item3"></li>
</ul>
<script type="text/JavaScript">
    var item1 = document.getElementById("item1");
    var nextNode = item1.nextSibling;
    //Firefox 不支持 document.all。这里用以判别浏览器类型
    if(!document.all){
            while(true){
                    if(nextNode.nodeType == 1){
                            break;
                    } else {
                            if(nextNode.nextSibling){
                                    nextNode = nextNode.nextSibling;
                            } else {
                                    break;
                            }
                    }
            }
```

```
        }
        alert(nextNode.id);
    </script>
```

代码清单 5-38　针对 IE 和 firefox 分别写程序如何面对扩展

```
    <ul>
        <li id="item1"></li>
        <li id="item2"></li>
        <li id="item3"></li>
    </ul>
    <script type="text/JavaScript">
        var item1 = document.getElementById("item1");
        var nextNode = item1.nextSibling;
        if(!document.all){
                while(true){
                        if(nextNode.nodeType == 1){
                                break;
                        } else {
                                if(nextNode.nextSibling){
                                        nextNode = nextNode.nextSibling;
                                } else {
                                        break;
                                }
                        }
                }
        }
        alert(nextNode.id);

        var item2 = document.getElementById("item2");
        var nextNode2 = item2.nextSibling;
        if(!document.all){
                while(true){
                        if(nextNode2.nodeType == 1){
                                break;
                        } else {
                                if(nextNode2.nextSibling){
                                        nextNode2 = nextNode2.nextSibling;
                                } else {
                                        break;
                                }
                        }
                }
        }
        alert(nextNode2.id);
    </script>
```

代码清单 5-39　用 getNextNode 函数封装 IE 和 firefox 的差异

```
    <ul>
        <li id="item1"></li>
        <li id="item2"></li>
        <li id="item3"></li>
    </ul>
    <script type="text/JavaScript">
        function getNextNode(node){
                node = typeof node == "string" ? document.getElementById(node) : node;
                var nextNode = node.nextSibling;
                if(!nextNode) return null;
                if(!document.all){
                        while(true){
                                if(nextNode.nodeType == 1){
```

```
                                        break;
                                } else {
                                        if(nextNode.nextSibling){
                                                nextNode = nextNode.nextSibling;
                                        } else {
                                                break;
                                        }
                                }
                        }
                }
                return nextNode;
        };
        var nextNode = getNextNode("item1");
        alert(nextNode.id);
        var nextNode2 = getNextNode("item2");
        alert(nextNode2.id);
    </script>
```

代码清单 5-40　针对 IE 和 firefox 分别设置透明度

```
    <style type="text/CSS">
    #test1{background:blue;height:100px;}
    #test2{background:green;height:100px;}
    </style>
    <div id="test1"></div>
    <div id="test2"></div>
    <script type="text/JavaScript">
    var test1 = document.getElementById("test1"),test2 =
document.getElementById("test2");
    if (document.all){
        test1.style.filter = 'alpha(opacity=20)';
        test2.style.filter = 'alpha(opacity=80)';
    } else {
        test1.style.opacity = 0.2;
        test2.style.opacity = 0.8;
    }
    </script>
```

代码清单 5-41　封装 setOpacity 函数

```
    <style type="text/CSS">
    #test1{background:blue;height:100px;}
    #test2{background:green;height:100px;}
    </style>
    <div id="test1"></div>
    <div id="test2"></div>
    <script type="text/JavaScript">
    function setOpacity(node, level){
        node = typeof node == "string" ? document.getElementById(node) : node;
        if (document.all){
           node.style.filter = 'alpha(opacity=' + level + ')';
        } else {
           node.style.opacity = level / 100;
        }
    }
    setOpacity("test1",20);
    setOpacity("test2",80);
    </script>
```

代码清单 5-42　点击事件和参数

```
    <input type="button" value="click me" id="btn" />
    <script type="text/JavaScript">
```

```
    var btn = document.getElementById("btn");
    btn.onclick = function(){
        alert(arguments.length);
    }
    </script>
```

代码清单 5-43　兼容 IE 和 firefox 的 event 对象

```
    <input type="button" value="click me" id="btn" />
    <script type="text/JavaScript">
    var btn = document.getElementById("btn");
    btn.onclick = function(e){
        e = window.event || e;
        //下面可以用 e 来做点什么事，e 在 IE 和 Firefox 下都指向了 event 对象
    }
    </script>
```

代码清单 5-44　兼容 srcElement 和 target

```
    <input type="button" value="click me" id="btn" />
    <span id="span">hello world</span>
    <script type="text/JavaScript">
    document.getElementById("btn").onclick = function(e){
        e = window.event || e;
        var el = e.srcElement || e.target;
        alert(el.tagName);
    }
    document.getElementById("span").onclick = function(e){
        e = window.event || e;
        var el = e.srcElement || e.target;
        alert(el.tagName);
    }
    </script>
```

代码清单 5-45　封装 getEventTarget 函数

```
    <input type="button" value="click me" />
    <span id="span">hello world</span>
    <script type="text/JavaScript">
    function getEventTarget(e){
        e = window.event || e;
        return e.srcElement || e.target;
    }
    document.getElementById("wrapper").onclick = function(e){
        var node = getEventTarget(e);
        alert(node.tagName);
    }
    document.getElementById("span").onclick = function(e){
        var node = getEventTarget(e);
        alert(node.tagName);
    }
    </script>
```

代码清单 5-46　event 对象的冒泡

```
    <style type="text/CSS">
    #infoBox{padding:5px;border:2px dashed
#ccc;margin-bottom:10px;width:136px;height:20px;line-height:20px;text-align:center;
}
    #wrapper{padding:20px 0;background:black;width:150px;text-align:center;}
    </style>
    <p id="infoBox"></p>
    <div id="wrapper">
```

```
        <input type="button" value="click me" id="btn" />
    </div>
    <script type="text/JavaScript">
    var infoBox = document.getElementById("infoBox") , wrapper =
document.getElementById("wrapper") , btn = document.getElementById("btn");
    wrapper.onclick = function(){
        infoBox.innerHTML = "你点击的是：div";
    }
    btn.onclick = function(){
        infoBox.innerHTML = "你点击的是：input";
    }
    </script>
```

代码清单 5-47　监测冒泡的过程

```
    <style type="text/CSS">
    #infoBox{padding:5px;border:2px dashed
#ccc;margin-bottom:10px;width:136px;height:20px;line-height:20px;text-align:center;
}
    #wrapper{padding:20px 0;background:black;width:150px;text-align:center;}
    </style>
    <p id="infoBox"></p>
    <div id="wrapper">
        <input type="button" value="click me" id="btn" />
    </div>
    <script type="text/JavaScript">
    var infoBox = document.getElementById("infoBox") , wrapper =
document.getElementById("wrapper") , btn = document.getElementById("btn");
    wrapper.onclick = function(){
        infoBox.innerHTML = "你点击的是：div";
        alert(1);
    }
    btn.onclick = function(){
        infoBox.innerHTML = "你点击的是：input";
        alert(2);
    }
    </script>
```

代码清单 5-48　阻止事件冒泡

```
    <style type="text/CSS">
    #infoBox{padding:5px;border:2px dashed
#ccc;margin-bottom:10px;width:136px;height:20px;line-height:20px;text-align:center;
}
    #wrapper{padding:20px 0;background:black;width:150px;text-align:center;}
    </style>
    <p id="infoBox"></p>
    <div id="wrapper">
        <input type="button" value="click me" id="btn" />
    </div>
    <script type="text/JavaScript">
    var infoBox = document.getElementById("infoBox") , wrapper =
document.getElementById("wrapper") , btn = document.getElementById("btn");
    wrapper.onclick = function(){
        infoBox.innerHTML = "你点击的是：div";
    }
    btn.onclick = function(e){
        infoBox.innerHTML = "你点击的是：input";
        e = window.event || e;
        if(document.all){
                e.cancelBubble=true;
        } else {
```

```
                e.stopPropagation();
        }
    }
    </script>
```

代码清单 5-49　封装 stopPropagation 函数

```
    <style type="text/CSS">
    #infoBox{padding:5px;border:2px dashed
#ccc;margin-bottom:10px;width:136px;height:20px;line-height:20px;text-align:center;
}
    #wrapper{padding:20px 0;background:black;width:150px;text-align:center;}
    </style>
    <p id="infoBox"></p>
    <div id="wrapper">
        <input type="button" value="click me" id="btn" />
    </div>
    <script type="text/JavaScript">
    function stopPropagation(e){
        e = window.event || e;
        if(document.all){
                e.cancelBubble=true;
        } else {
                e.stopPropagation();
        }
    }
    var infoBox = document.getElementById("infoBox") , wrapper =
document.getElementById("wrapper") , btn = document.getElementById("btn");
    wrapper.onclick = function(){
        infoBox.innerHTML = "你点击的是：div";
    }
    btn.onclick = function(e){
        infoBox.innerHTML = "你点击的是：input";
        stopPropagation(e);
    }
    </script>
```

代码清单 5-50　监听 click 事件

```
    <input type="button" value="click me" id="btn" />
    <script type="text/JavaScript">
    var btn = document.getElementById("btn");
    btn.onclick = function(){
        alert(1);
    }
    </script>
```

代码清单 5-51　监听两个 click 事件

```
    <input type="button" value="click me" id="btn" />
    <script type="text/JavaScript">
    var btn = document.getElementById("btn");
    btn.onclick = function(){
        alert(1);
    }
    btn.onclick = function(){
        alert(2);
    }
    </script>
```

代码清单 5-52　on xxx 在多人合作时的冲突问题

```
    <input type="button" value="click me" id="btn" />
```

```
    <script type="text/JavaScript">
    //============================
    // 功能A ，made by 工程师甲
    //============================
    (function(){
         …
         var btn = document.getElementById("btn");
         btn.onclick = function(){
                  alert(1);
         }
         …
    })();
    //============================
    // 功能B ，made by 工程师乙
    //============================
    (function(){
         …
         var btn = document.getElementById("btn");
         btn.onclick = function(){
                  alert(2);
         }
         …
    })();
    </script>
```

代码清单 5-53　使用 attachEvent 和 addEventListener 代替 on xxx

```
    <input type="button" value="click me" id="btn" />
    <script type="text/JavaScript">
    var btn = document.getElementById("btn");
    if(document.all){
         btn.attachEvent("onclick",function(){
                  alert(1);
         });
    } else {
         btn.addEventListener("click",function(){
                  alert(1);
         },false);
    }
    if(document.all){
         btn.attachEvent("onclick",function(){
                  alert(2);
         });
    } else {
         btn.addEventListener("click",function(){
                  alert(2);
         },false);
    }
    </script>
```

代码清单 5-54　封装 on 函数

```
    <input type="button" value="click me" id="btn" />
    <script type="text/JavaScript">
    function on(node,eventType,handler){
         node = typeof node == "string" ? document.getElementById(node) : node;
         if(document.all){
                  node.attachEvent("on"+eventType,handler);
         } else {
                  node.addEventListener(eventType,handler,false);
         }
    }
```

```
    var btn = document.getElementById("btn");
    on(btn,"click",function(){
        alert(1);
    });
    on(btn,"click",function(){
        alert(2);
    })
    </script>
```

代码清单 5-55　定义 trim 函数

```
    function trim(ostr){
        return ostr.replace(/^\s+|\s+$/g,"");
    }
    var str = " abc ";
    alert(trim(str).length);              // 3
    alert(" "=="");                       // false
    alert(trim(" ") == "");               // true
```

代码清单 5-56　定义类型判断函数

```
    function isNumber(s){
        return !isNaN(s);
    }
    function isString(s){
        return typeof s == "string";
    }
    function isBoolean(s){
        return typeof s == "boolean";
    }
    function isFunction(s){
        return typeof s == "function";
    }
    function isNull(s){
        return s == null;
    }
    function isUndefined(s){
        return typeof s == "undefined";
    }
    function isEmpty(s){
        return /^\s*$/.test(s);
    }
    function isArray(s){
        return s instanceof Array;
    }
```

代码清单 5-57　定义 get 和$函数

```
    function get(node){
        node = typeof node == "string" ? document.getElementById(node) : node;
        return node;
    }
    function $(node){
        node = typeof node == "string" ? document.getElementById(node) : node;
        return node;
    }
    alert(get("test1").innerHTML);        // Hello
    alert($("test2").innerHTML)           // World
```

代码清单 5-58　定义 getElementsByClassName 函数

```
    <span class="a">1</span>
    <span class="a">2</span>
```

```
    <p class="a">3</p>
    <div class="b">4</div>
    <strong class="b">5</strong>
    <div id="wrapper"><strong class="b">6</strong></div>
    <script type="text/JavaScript">
    function getElementsByClassName(str,root,tag){
         if(root){
                 root = typeof root == "string" ? document.getElementById(root) : root;
         } else {
                 root = document.body;
         }
         tag = tag || "*";
         var els = root.getElementsByTagName(tag),arr = [];
         for(var i=0,n=els.length;i<n;i++){
                 for(var j=0,k=els[i].className.split(" "),l=k.length;j<l;j++){
                         if(k[j] == str){
                                 arr.push(els[i]);
                                 break;
                         }
                 }
         }
         return arr;
    }
    var aEls = getElementsByClassName("a") , bEls =
getElementsByClassName("b","wrapper") , bEls2 =
getElementsByClassName("b",null,"strong");
    alert(aEls.length);           // 3     (1,2,3)
    alert(bEls.length);           // 1     (6)
    alert(bEls2.length);          // 2     (5,6)
```

代码清单 5-59　定义 extend 函数

```
    function extend(subClass,superClass){
        var F = function(){};
       F.prototype = superClass.prototype;
       subClass.prototype = new F();
       subClass.prototype.constructor = subClass;
       subClass.superclass = superClass.prototype;
       if(superClass.prototype.constructor == Object.prototype.constructor){
           superClass.prototype.constructor = superClass;
       }
    }

    function Animal(name){
         this.name = name;
         this.type = "animal"
    }
    Animal.prototype = {
         say : function(){
                 alert("I'm a(an) " + this.type + " , my name is " + this.name);
         }
    }
    function Bird(name){
         this.constructor.superclass.constructor.apply(this,arguments);
         this.type = "bird"
    }
    extend(Bird,Animal);
    Bird.prototype.fly = function(){
         alert("I'm flying");
    }
```

```
var canary = new Bird("xiaocui");
canary.say();                 // I’m a(an) bird , my name is xiaocui
canary.fly();                 // I’m flying
```

代码清单 5-60　使用命名空间

```
var GLOBAL = {};
GLOBAL.namespace = function(str){
     var arr = str.split("."),o = GLOBAL;
     for (i=(arr[0] == "GLOBAL") ? 1 : 0; i<arr.length; i++) {
             o[arr[i]]=o[arr[i]] || {};
       o=o[arr[i]];
    }
}

//Dom 相关
GLOBAL.namespace("Dom");
GLOBAL.Dom.getNextNode = function(node){
     node = typeof node == "string" ? document.getElementById(node) : node;
     var nextNode = node.nextSibling;
     if(!nextNode) return null;
     if(!document.all){
             while(true){
                     if(nextNode.nodeType == 1){
                             break;
                     } else {
                             if(nextNode.nextSibling){
                                     nextNode = nextNode.nextSibling;
                             } else {
                                     break;
                             }
                     }
             }
     }
     return nextNode;
};
GLOBAL.Dom.setOpacity = function(node, level){
    node = typeof node == "string" ? document.getElementById(node) : node;
    if (document.all){
       node.style.filter = 'alpha(opacity=' + level + ')';
    } else {
       node.style.opacity = level / 100;
    }
}
GLOBAL.Dom.get = function(node){
     node = typeof node == "string" ? document.getElementById(node) : node;
     return node;
}
GLOBAL.Dom.getElementsByClassName = function(str,root,tag){
     if(root){
             root = typeof root == "string" ? document.getElementById(root) : root;
     } else {
             root = document.body;
     }
     tag = tag || "*";
     var els = root.getElementsByTagName(tag),arr = [];
     for(var i=0,n=els.length;i<n;i++){
             for(var j=0,k=els[i].className.split(" "),l=k.length;j<l;j++){
                     if(k[j] == str){
                             arr.push(els[i]);
                             break;
```

```
                        }
                }
        }
        return arr;
    }

    //Event 相关
    GLOBAL.namespace("Event");
    GLOBAL.Eevent.getEventTarget = function(e){
        e = window.event || e;
        return e.srcElement || e.target;
    }
    GLOBAL.Event.stopPropagation = function(e){
        e = window.event || e;
        if(document.all){
                e.cancelBubble=true;
        } else {
                e.stopPropagation();
        }
    }
    GLOBAL.Event.on = function(node,eventType,handler){
        node = typeof node == "string" ? document.getElementById(node) : node;
        if(document.all){
                node.attachEvent("on"+eventType,handler);
        } else {
                node.addEventListener(eventType,handler,false);
        }
    }

    //Lang 相关
    GLOBAL.namespace("Lang");
    GLOBAL.Lang.trim = function(ostr){
        return ostr.replace(/^\s+|\s+$/g,"");
    }
    GLOBAL.Lang.isNumber = function(s){
        return !isNaN(s);
    }
    GLOBAL.Lang.extend = function(subClass,superClass){
        var F = function(){};
       F.prototype = superClass.prototype;
       subClass.prototype = new F();
       subClass.prototype.constructor = subClass;
       subClass.superclass = superClass.prototype;
       if(superClass.prototype.constructor == Object.prototype.constructor){
           superClass.prototype.constructor = superClass;
       }
    }
```

代码清单 5-61　引入 common.js

```
    <script type="text/JavaScript" src="base.js"></script>
    <script type="text/JavaScript" src="common.js"></script>
```

代码清单 5-62　设置 cookie

```
    document.cookie = "cookieName = cookieValue; expirationdate = timeValue; path =
pathValue";
```

代码清单 5-63　读写 cookie

```
    // 设置 cookie
    document.cookie = "name=adang; expires= Mon, 30 Nov 2009 05:49:47 GMT; path=/";
```

```
    document.cookie = "sex=male; expires= Mon, 30 Nov 2009 05:49:47 GMT; path=/";
    document.cookie = "blog=http://www.adanghome.com; expires= Mon, 30 Nov 2009 05:49:47
GMT; path=/";
    /* 读取 cookie
    ** 此时 cookie 里的值为"name=adang; sex=male; blog=http://www.adanghome.com"
    */
    var cookieStr = document.cookie;
    // 对字符进行操作，取出 name 对应的值
    var name = cookieStr.split("name")[1].split(";")[0].split("=")[1];
```

代码清单 5-64　封装 cookie 组件

```
    GLOBAL.namespace("Cookie");
    GLOBAL.Cookie={
        // 读取
        read : function(name){
         var cookieStr = "; "+document.cookie+"; ";
         var index = cookieStr.indexOf("; "+name+"=");
         if (index!=-1){
           var s = cookieStr.substring(index+name.length+3,cookieStr.length);
           return unescape(s.substring(0, s.indexOf("; ")));
         }else{
           return null;
         }
        },
        // 设置
        set : function(name,value,expires){
         var expDays = expires*24*60*60*1000;
         var expDate = new Date();
         expDate.setTime(expDate.getTime()+expDays);
         var expString = expires ? "; expires="+expDate.toGMTString() : "";
         var pathString = ";path=/";
         document.cookie = name + "=" + escape(value) + expString + pathString;
        },
        // 删除
        del : function(name){
         var exp = new Date(new Date().getTime()-1);
         var s=this.read(name);
         if(s!=null) {document.cookie= name +
"="+s+";expires="+exp.toGMTString()+";path=/"};
        }
    };
```

代码清单 5-65　引用可拖拽的 Msg 组件

```
    <script type="text/JavaScript" src="base.js"></script>
    <script type="text/JavaScript" src="common_drag.js"></script>
    <script type="text/JavaScript" src="common_msg.js"></script>
```

5.2.4　page 层

5.2.5　JavaScript 库

5.3编程实用技巧

5.3.1　弹性

代码清单 5-66　开始编写 Tab 组件

```
    <style type="text/CSS">
     ul{padding:0;margin:0}
```

```
    .tab{width:400px;}
    .tab-menuWrapper{padding-left:20px;}
    .tab-menuWrapper li{float:left;display:inline;padding:5px;border:1px solid
#333;border-bottom:none;margin-right:5px;}
    .tab-contentWrapper{border:1px solid #333;clear:left;padding:5px;}
    </style>
     <div class="tab">
        <ul class="tab-menuWrapper">
                <li>menu1</li>
                <li>menu2</li>
                <li>menu3</li>
        </ul>
        <div class="tab-contentWrapper">
                <div>content1</div>
                <div style="display:none">content2</div>
                <div style="display:none">content3</div>
        </div>
     </div>
```

代码清单 5-67　给 html 标签挂上 id

```
     …
     <div class="tab">
        <ul class="tab-menuWrapper">
                <li id="tab-menu1">menu1</li>
                <li id="tab-menu2">menu2</li>
                <li id="tab-menu3">menu3</li>
        </ul>
        <div class="tab-contentWrapper">
                <div id="tab-content1">content1</div>
                <div id="tab-content2" style="display:none">content2</div>
                <div id="tab-content3" style="display:none">content3</div>
        </div>
     </div>
     <script type="text/JavaScript">
        //获得 tabMenu 和 tabContent 的 DOM 节点，并保存在变量中
        var tabMenu1 = document.getElementById("tab-menu1") ,
            tabMenu2 = document.getElementById("tab-menu2") ,
            tabMenu3 = document.getElementById("tab-menu3") ,
            tabContent1 = document.getElementById("tab-content1") ,
            tabContent2 = document.getElementById("tab-content2") ,
    tabContent3 = document.getElementById("tab-content3");

        //让 tabMenu 监听点击 click 事件
        tabMenu1.onclick = function(){
                //显示 tabContent1，隐藏 tabContent2 和 tabContent3
                tabContent1.style.display = "block";
                tabContent2.style.display = "none";
                tabContent3.style.display = "none";
        }
        tabMenu2.onclick = function(){
                //显示 tabContent2，隐藏 tabContent1 和 tabContent3
                tabContent1.style.display = "none";
                tabContent2.style.display = "block";
                tabContent3.style.display = "none";
        }
        tabMenu3.onclick = function(){
                //显示 tabContent3，隐藏 tabContent1 和 tabContent2
                tabContent1.style.display = "none";
                tabContent2.style.display = "none";
                tabContent3.style.display = "block";
```

```
        }
    </script>
```

代码清单 5-68　提高 Tab 的扩展性

```
    …
    <div class="tab">
        <ul class="tab-menuWrapper" id="tab-menuWrapper">
                <li>menu1</li>
                <li>menu2</li>
                <li>menu3</li>
                <li>menu4</li>
        </ul>
        <div class="tab-contentWrapper" id="tab-contentWrapper">
                <div>content1</div>
                <div style="display:none">content2</div>
                <div style="display:none">content3</div>
                <div style="display:none">content4</div>
        </div>
    </div>
    <script type="text/JavaScript">
        //获得 tabMenu 和 tabContent 的节点，并以数组的形式保存在变量中
        var tabMenus =
document.getElementById("tab-menuWrapper").getElementsByTagName("li") ,
            tabContents =
document.getElementById("tab-contentWrapper").getElementsByTagName("div");

        //遍历数组，让 tabMenu 监听 click 事件
        for(var i=0;i<tabMenus.length;i++){
                tabMenus[i].onclick = function(){
                        //遍历数组，隐藏所有 tabContent
                        for(var j=0;j<tabContents.length;j++){
                                tabContents[j].style.display = "none";
                        }
                        //显示被点击的 tabMenu 对应的 tabContent
                        tabContents[i].style.display = "block";
                }
        }
    </script>
```

代码清单 5-69　循环中的 i

```
        for(var i=0;i<tabMenus.length;i++){
                tabMenus[i].onclick = function(){
                        //弹出当前索引
                        alert(i);
                        //遍历数组，隐藏所有 tabContent
                        for(var j=0;j<tabContents.length;j++){
                                tabContents[j].style.display = "none";
                        }
                        //显示被点击的 tabMenu 对应的 tabContent
                        tabContents[i].style.display = "block";
                }
        }
```

代码清单 5-70　解决循环 bug 方法一

```
        for(var i=0;i<tabMenus.length;i++){
                (function(_i){
                        tabMenus[_i].onclick = function(){
                                for(var j=0;j<tabContents.length;j++){
                                        tabContents[j].style.display = "none";
                                }
```

```
                                tabContents[_i].style.display = "block";
                        }
                })(i);
        }
```

代码清单 5-71　解决循环 bug 方法二

```
        for(var i=0;i<tabMenus.length;i++){
                tabMenus[i]._index = i;
                tabMenus[i].onclick = function(){
                        for(var j=0;j<tabContents.length;j++){
                                tabContents[j].style.display = "none";
                        }
                        tabContents[this._index].style.display = "block";
                }
        }
```

代码清单 5-72　两个标签的 Tab

```
    <!—- 两个标签 -->
    <div class="tab">
        <ul class="tab-menuWrapper" id="tab-menuWrapper">
                <li>menu1</li>
                <li>menu2</li>
        </ul>
        <div class="tab-contentWrapper" id="tab-contentWrapper">
                <div>content1</div>
                <div style="display:none">content2</div>
        </div>
      </div>
```

代码清单 5-73　五个标签的 Tab

```
    <!—- 五个标签 -->
    <div class="tab">
        <ul class="tab-menuWrapper" id="tab-menuWrapper">
                <li>menu1</li>
                <li>menu2</li>
                <li>menu3</li>
                <li>menu4</li>
                <li>menu5</li>
        </ul>
        <div class="tab-contentWrapper" id="tab-contentWrapper">
                <div>content1</div>
                <div style="display:none">content2</div>
                <div style="display:none">content3</div>
                 <div style="display:none">content4</div>
                 <div style="display:none">content5</div>
        </div>
      </div>
```

代码清单 5-74　结构稍复杂的 Tab

```
    <div class="tab">
        <ul class="tab-menuWrapper" id="tab-menuWrapper">
                <li>menu1</li>
                <li>menu2</li>
                <li>menu3</li>
        </ul>
        <div class="tab-contentWrapper" id="tab-contentWrapper">
                <div><div>content1</div><ul><li>abc</li></ul></div>
                <div style="display:none"><p>content2</p><div>abc</div></div>
                <div style="display:none">content3</div>
```

```
        </div>
      </div>
```

代码清单 5-75　复杂 Tab 结构带来的问题

```
    <div class="tab">
        <ul class="tab-menuWrapper" id="tab-menuWrapper">
                <li>menu1</li>
                <li>menu2</li>
                <li>menu3</li>
        </ul>
        <div class="tab-contentWrapper" id="tab-contentWrapper">
                <div><div>content1</div><ul><li>abc</li></ul></div>
                <div style="display:none"><p>content2</p><div>abc</div></div>
                <div style="display:none">content3</div>
        </div>
      </div>
      <script type="text/JavaScript">
         var tabMenus =
document.getElementById("tab-menuWrapper").getElementsByTagName("li") ,
             tabContents =
document.getElementById("tab-contentWrapper").getElementsByTagName("div");
             alert(tabContents.length);
             //弹出 5，不是 3
    …
```

5.3.2　getElementById 、getElementsByTagName 和 getElementsByClassName

代码清单 5-76　使用 getElementsByClassName

```
    …
      <div class="tab">
         <ul class="tab-menuWrapper" id="tab-menuWrapper">
                <li>menu1</li>
                <li>menu2</li>
                <li>menu3</li>
         </ul>
         <div class="tab-contentWrapper">
                <div class="J_tab-content"><div>content1</div><ul>abc</ul></div>
                <div class="J_tab-content"
style="display:none"><p>content2</p><div>abc</div></div>
                <div class="J_tab-content" style="display:none">content3</div>
         </div>
      </div>
      <script type="text/JavaScript">
         var GLOBAL = {};
         GLOBAL.namespace = function(str){
                var arr = str.split("."),o = GLOBAL;
                for (i=(arr[0] == "GLOBAL") ? 1 : 0; i<arr.length; i++) {
                        o[arr[i]]=o[arr[i]] || {};
                        o=o[arr[i]];
                }
         }
         GLOBAL.namespace("Dom");
         GLOBAL.Dom.getElementsByClassName = function(str,root,tag){
                if(root){
                        root = typeof root == "string" ?
document.getElementById(root) : root;
                } else {
                        root = document.body;
```

```
                }
                tag = tag || "*";
                var els = root.getElementsByTagName(tag),arr = [];
                for(var i=0,n=els.length;i<n;i++){
                        for(var j=0,k=els[i].className.split("
"),l=k.length;j<l;j++){
                                if(k[j] == str){
                                        arr.push(els[i]);
                                        break;
                                }
                        }
                }
                return arr;
        }


        var tabMenus =
document.getElementById("tab-menuWrapper").getElementsByTagName("li"),
        tabContents = GLOBAL.Dom.getElementsByClassName("J_tab-content");
        for(var i=0;i<tabMenus.length;i++){
                tabMenus[i]._index = i;
                tabMenus[i].onclick = function(){
                        for(var j=0;j<tabContents.length;j++){
                                tabContents[j].style.display = "none";
                        }
                        tabContents[this._index].style.display = "block";
                }
        }
    </script>
```

5.3.3 可复用性

代码清单 5-77 将 id 换为 class

```
    …
    <div class="tab">
        <ul class="tab-menuWrapper">
                <li class="J_tab-menu">menu1</li>
                <li class="J_tab-menu">menu2</li>
                <li class="J_tab-menu">menu3</li>
        </ul>
        …
    var tabMenus = GLOBAL.Dom.getElementsByClassName("J_tab-menu");
    …
```

代码清单 5-78 三个 Tab

```
    <div class="tab">
       <ul class="tab-menuWrapper">
                <li class="J_tab-menu">menu1-1</li>
                <li class="J_tab-menu">menu1-2</li>
                <li class="J_tab-menu">menu1-3</li>
        </ul>
        <div class="tab-contentWrapper">
                <div class="J_tab-content"><div>content1-1</div><ul>abc</ul></div>
                <div class="J_tab-content"
style="display:none"><p>content1-2</p><div>abc</div></div>
                <div class="J_tab-content" style="display:none">content1-3</div>
        </div>
    </div>
    <div class="tab">
```

```
        <ul class="tab-menuWrapper">
                <li class="J_tab-menu">menu2-1</li>
                <li class="J_tab-menu">menu2-2</li>
        </ul>
        <div class="tab-contentWrapper">
                <div class="J_tab-content"><div>content2-1</div><ul>abc</ul></div>
                <div class="J_tab-content"
style="display:none"><p>content2-2</p><div>abc</div></div>
        </div>
     </div>
     <div class="tab">
        <ul class="tab-menuWrapper">
                <li class="J_tab-menu">menu3-1</li>
                <li class="J_tab-menu">menu3-2</li>
                <li class="J_tab-menu">menu3-3</li>
                <li class="J_tab-menu">menu3-4</li>
                <li class="J_tab-menu">menu3-5</li>
        </ul>
        <div class="tab-contentWrapper">
                <div class="J_tab-content"><div>content3-1</div><ul>abc</ul></div>
                <div class="J_tab-content"
style="display:none"><p>content3-2</p><div>abc</div></div>
                <div class="J_tab-content" style="display:none">content3-3</div>
                <div class="J_tab-content"
style="display:none"><p>content3-4</p><div>abc</div></div>
                <div class="J_tab-content" style="display:none">content3-5</div>
        </div>
     </div>
```

代码清单 5-79　给 Tab 添加根结点挂钩

```
    …
     <div class="tab J_tab">
        <ul class="tab-menuWrapper">
                …
        </ul>
        <div class="tab-contentWrapper">
                …
        </div>
     </div>
     <div class="tab J_tab">
        <ul class="tab-menuWrapper">
                …
        </ul>
        <div class="tab-contentWrapper">
                …
        </div>
     </div>
     <div class="tab J_tab">
        <ul class="tab-menuWrapper">
                …
        </ul>
        <div class="tab-contentWrapper">
                …
        </div>
     </div>
     <script type="text/JavaScript">
        …

    function setTab(root){
        var tabMenus = GLOBAL.Dom.getElementsByClassName("J_tab-menu",root),
```

```
        tabContents = GLOBAL.Dom.getElementsByClassName("J_tab-content",root);
        for(var i=0;i<tabMenus.length;i++){
                tabMenus[i]._index = i;
                tabMenus[i].onclick = function(){
                        for(var j=0;j<tabContents.length;j++){
                                tabContents[j].style.display = "none";
                        }
                        tabContents[this._index].style.display = "block";
                }
        }
    }

    var tabs = GLOBAL.Dom.getElementsByClassName("J_tab");
    for(var i=0;i<tabs.length;i++){
        setTab(tabs[i]);
    }
    </script>
```

5.3.4 避免产生副作用

代码清单 5-80 高亮当前 Tab

```
    <style type="text/CSS">
     …
    .tab .tab-currentMenu{background-color:#333;color:#fff;}
     </style>
    <div class="tab J_tab">
        <ul class="tab-menuWrapper">
                <li class="J_tab-menu .tab-currentMenu">menu1-1</li>
                <li class="J_tab-menu">menu1-2</li>
                <li class="J_tab-menu">menu1-3</li>
        </ul>
        <div class="tab-contentWrapper">
                …
        </div>
    </div>
    <div class="tab J_tab">
        <ul class="tab-menuWrapper">
                <li class="J_tab-menu .tab-currentMenu">menu2-1</li>
                <li class="J_tab-menu">menu2-2</li>
        </ul>
        <div class="tab-contentWrapper">
                …
        </div>
    </div>
    <div class="tab J_tab">
        <ul class="tab-menuWrapper">
                <li class="J_tab-menu .tab-currentMenu">menu3-1</li>
                <li class="J_tab-menu">menu3-2</li>
                <li class="J_tab-menu">menu3-3</li>
                <li class="J_tab-menu">menu3-4</li>
                <li class="J_tab-menu">menu3-5</li>
        </ul>
        <div class="tab-contentWrapper">
                …
        </div>
    </div>
    <script type="text/JavaScript">
    …
    function setTab(root){
```

```
        var tabMenus = GLOBAL.Dom.getElementsByClassName("J_tab-menu",root),
        tabContents = GLOBAL.Dom.getElementsByClassName("J_tab-content",root);
        for(var i=0;i<tabMenus.length;i++){
                tabMenus[i]._index = i;
                tabMenus[i].onclick = function(){
                        for(var j=0;j<tabContents.length;j++){
                                tabContents[j].style.display = "none";
                        }
                        tabContents[this._index].style.display = "block";
                        //如果有当前选中标签，则去掉“tab-currentMenu”
                        var currentMenu =
GLOBAL.Dom.getElementsByClassName("tab-currentMenu",root)[0];
                        if(currentMenu){
                                currentMenu.className = "";
                        }
                        //给当前被点击的按钮挂上当前选中的 class
                        this.className = "tab-currentMenu";
                }
        }
    }
    …
    </script>
```

代码清单 5-81　修改 className

```
    if(currentMenu){
        currentMenu.className = "";
    }
    this.className = "tab-currentMenu";
```

代码清单 5-82　修改 className 引起的冲突

```
    <style type="text/CSS">
     …
    .tab .tab-currentMenu{background-color:#333;color:#fff;}
    .underLine{text-decoration:underline;}
    </style>
    …
    <div class="tab">
        <ul class="tab-menuWrapper">
                <li class="J_tab-menu">menu1-1</li>
                <li class="J_tab-menu">menu1-2</li>
                <li class="J_tab-menu underLine">menu1-3</li>
        </ul>
    …
```

代码清单 5-83　修改至无副作用

```
    if(currentMenu){
        currentMenu.className = currentMenu.className.replace(new
RegExp("(^|\\s+)tab-currentMenu"),"");
    }
    if(!new RegExp("(^|\\s+)tab-currentMenu").test(this.className)){
        this.className =this.className + " tab-currentMenu";
    }
```

代码清单 5-84　定义 addClass 和 removeClass 函数

```
    …
    GLOBAL.Dom.addClass = function(node,str){
        if(!new RegExp("(^|\\s+)"+str).test(node.className)){
                node.className = node.className + " " + str;
        }
```

```
    }
    GLOBAL.Dom.removeClass = function(node,str){
        node.className = node.className.replace(new RegExp("(^|\\s+)"+str),"");
    }
    …
    if(currentMenu){
        GLOBAL.Dom.removeClass(currentMenu,"tab-currentMenu");
    }
    GLOBAL.Dom.addClass(this,"tab-currentMenu");
    …
```

5.3.5 通过传参实现定制

代码清单 5-85 编写 setTab、setTab2 和 setTab3

```
    <style type="text/CSS">
     …
    .tab .tab-currentMenu{background-color:#333;color:#fff;}
    .tab .tab-currentMenu2{background-color:blue;color:#fff;}
    .underLine{text-decoration:underline;}
     </style>
    <div class="tab J_tab">
        <ul class="tab-menuWrapper">
                <li class="J_tab-menu">menu1-1</li>
                <li class="J_tab-menu">menu1-2</li>
                <li class="J_tab-menu">menu1-3</li>
        </ul>
        <div class="tab-contentWrapper">
                …
        </div>
    </div>
    <div class="tab J_tab">
        <ul class="tab-menuWrapper">
                <li class="J_tab-menu .tab-currentMenu">menu2-1</li>
                <li class="J_tab-menu">menu2-2</li>
        </ul>
        <div class="tab-contentWrapper">
                …
        </div>
    </div>
    <div class="tab J_tab">
        <ul class="tab-menuWrapper">
                <li class="J_tab-menu .tab-currentMenu2">menu3-1</li>
                <li class="J_tab-menu">menu3-2</li>
                <li class="J_tab-menu">menu3-3</li>
                <li class="J_tab-menu">menu3-4</li>
                <li class="J_tab-menu">menu3-5</li>
        </ul>
        <div class="tab-contentWrapper">
                …
        </div>
    </div>
    <script type="text/JavaScript">
    …
    function setTab(root){
        …
    //去掉高亮当前标签的功能
    /*
        var currentMenu =
GLOBAL.Dom.getElementsByClassName("tab-currentMenu2",root)[0];
```

```
        if(currentMenu){
                GLOBAL.Dom.removeClass(currentMenu,"tab-currentMenu");
        }
        GLOBAL.Dom.addClass(this,"tab-currentMenu");
    */
        …
    }
    function setTab2(root){
        …
    //当前标签挂上高亮样式 1
        var currentMenu =
GLOBAL.Dom.getElementsByClassName("tab-currentMenu",root)[0];
        if(currentMenu){
                GLOBAL.Dom.removeClass(currentMenu,"tab-currentMenu");
        }
        GLOBAL.Dom.addClass(this,"tab-currentMenu");
        …
    }
    function setTab3(root){
        …
    //当前标签挂上高亮样式 2
        var currentMenu =
GLOBAL.Dom.getElementsByClassName("tab-currentMenu2",root)[0];
        if(currentMenu){
                GLOBAL.Dom.removeClass(currentMenu,"tab-currentMenu2");
        }
        GLOBAL.Dom.addClass(this,"tab-currentMenu2");
        …
    }

    var tabs = GLOBAL.Dom.getElementsByClassName("J_tab");
    setTab(tabs[0]);
    setTab2(tabs[1]);
    setTab3(tabs[2]);
    </script>
```

代码清单 5-86　用参数写活变化的部分

```
    function setTab(root,currentClass){
        …
    if(currentClass){
        var currentMenu = GLOBAL.Dom.getElementsByClassName(currentClass,root)[0];
        if(currentMenu){
                GLOBAL.Dom.removeClass(currentMenu,currentClass);
        }
        GLOBAL.Dom.addClass(this,currentClass);
    }
        …
    }

    var tabs = GLOBAL.Dom.getElementsByClassName("J_tab");
    setTab(tabs[0]);
    setTab(tabs[1],"tab-currentMenu");
    setTab(tabs[2],"tab-currentMenu2");
    </script>
```

代码清单 5-87　写活事件触发方式

```
    …
    GLOBAL.namespace("Event");
    GLOBAL.Event.on = function(node,eventType,handler){
        node = typeof node == "string" ? document.getElementById(node) : node;
```

```
        if(document.all){
                node.attachEvent("on"+eventType,handler);
        } else {
                node.addEventListener(eventType,handler,false);
        }
    }
    …
    function setTab(root,currentClass,trigger){
        var tabMenus = GLOBAL.Dom.getElementsByClassName("J_tab-menu",root),
        tabContents = GLOBAL.Dom.getElementsByClassName("J_tab-content",root);
        //如果不传入激活类型，默认激活类型为点击
        trigger = trigger || "click";
        for(var i=0;i<tabMenus.length;i++){
                tabMenus[i]._index = i;
                //调用 base 层提供的 on 方法
                GLOBAL.Event.on(tabMenus[i],trigger,function(){
                        for(var j=0;j<tabContents.length;j++){
                                tabContents[j].style.display = "none";
                        }
                        tabContents[this._index].style.display = "block";
                        if(currentClass){
                                var currentMenu =
GLOBAL.Dom.getElementsByClassName(currentClass,root)[0];
                                if(currentMenu){
        GLOBAL.Dom.removeClass(currentMenu,currentClass);
                                }
                                GLOBAL.Dom.addClass(this,currentClass);
                        }
                });
        }
    }
    var tabs = GLOBAL.Dom.getElementsByClassName("J_tab");
    //将 Tab1 和 Tab3 的标签激活方式改为 mouseover
    setTab(tabs[0],null,"mouseover");
    setTab(tabs[1],"tab-currentMenu");
    setTab(tabs[2],"tab-currentMenu2","mouseover");
    </script>
```

5.3.6 控制 this 关键字的指向

代码清单 5-88 调试 bug

```
    …
    GLOBAL.Event.on(tabMenus[i],trigger,function(){
        for(var j=0;j<tabContents.length;j++){
                tabContents[j].style.display = "none";
        }
        alert(this._index);
        tabContents[this._index].style.display = "block";
```

代码清单 5-89 javascript 伪协议和内联事件

```
    // 弹出“A”
    <a href="#" onclick="alert(this.tagName)">click me</a>
    // 弹出“undefined”
    <a href="JavaScript:alert(this.tagName)">click me</a>
    // 弹出“true”
    <a href="JavaScript:alert(this == window)">click me</a>
```

代码清单 5-90 setTimeout 和 setInterval 改变 this 指向

```
    var name = "somebody";
```

```
    var adang = {
        name : "adang",
        say : function(){
                alert("I'm " + this.name);
        }
    };
    adang.say();                                        // I’m adang
    setTimeout(adang.say,1000);                         // I’m someBody
    setInterval(adang.say,1000);                        // I’m someBody
```

代码清单 5-91　onxxx 改变 this 指向

```
    <input type="button" value="click me" id="btn" name="BUTTON" />

    var name = "somebody";
    var btn = document.getElementById("btn");
    var adang = {
        name : "adang",
        say : function(){
                alert("I'm " + this.name);
        }
    };
    btn.onclick = adang.say;                            // I’m BUTTON
```

代码清单 5-92　匿名函数调整 this 指向

```
    <input type="button" value="click me" id="btn" name="BUTTON" />

    var name = "somebody";
    var btn = document.getElementById("btn");
    var adang = {
        name : "adang",
        say : function(){
                alert("I'm " + this.name);
        }
    };
    adang.say();                                        // I’m adang
    setTimeout(function(){adang.say()},1000);           // I’m adang
    setInterval(function(){adang.say()},1000);          // I’m adang
    btn.onclick = function(){adang.say()};              // I’m adang
    setTimeout(function(){alert(this == window)},1000); // true
    btn.onclick = function(){alert(this == btn)};       // true
```

代码清单 5-93　call 和 apply 调整 this 指向

```
    <input type="button" value="click me" id="btn" name="BUTTON" />

    var name = "somebody";
    var btn = document.getElementById("btn");
    var adang = {
        name : "adang",
        say : function(){
                alert("I'm " + this.name);
        }
    };
    adang.say.call(btn);                                    // I’m BUTTON
    setTimeout(function(){adang.say.call(btn)},1000);       // I’m BUTTON
    setInterval(function(){adang.say.apply(btn)},1000);     // I’m BUTTON
    btn.onclick = function(){adang.say.apply(btn)};         // I’m BUTTON
```

代码清单 5-94　将 this 指向的对象保存到变量

```
    <input type="button" value="click me" id="btn" name="BUTTON" />
```

```
    var name = "somebody";
    var adang = {
        name : "adang",
        say : function(){
                alert("I'm " + this.name);
        },
        init : function(){
                // this 指向 adang 对象
                var This = this;
                document.getElementById("btn").onclick = function(){
                        // this 指向 btn 的 Dom 节点，This 指向 adang 对象
                        This.say();             // I'm adang
                        this.say();             // 报错,this.say is not a function
                };
        }
    };
    adang.init();
```

代码清单 5-95　调试 bug

```
    …
    GLOBAL.Event.on(tabMenus[i],trigger,function(){
        for(var j=0;j<tabContents.length;j++){
                tabContents[j].style.display = "none";
        }
        alert(this == window);
        tabContents[this._index].style.display = "block";
```

代码清单 5-96　attachEvent、addEventListener 和 this 指针

```
    <input type="button" value="click me" id="btn" />
     <script type="text/JavaScript">
        var btn = document.getElementById("btn");
        if(document.all){
                // ie
                btn.attachEvent("onclick",function(){
                        alert(this.tagName);                    //undefined
                        alert(this == window);                  //true
                });

        } else {
                // Firefox
                btn.addEventListener("click",function(){
                        alert(this.tagName);                    //INPUT
                        alert(this == window);                  //false
                },false);
        }
     </script>
```

代码清单 5-97　加强 on 函数的功能

```
GLOBAL.Event.on = function(node,eventType,handler,scope){
    node = typeof node == "string" ? document.getElementById(node) : node;
    scope = scope || node;
    if(document.all){
        node.attachEvent("on"+eventType,function(){handler.apply(scope,arguments)})
;
    } else {
        node.addEventListener(eventType,function(){handler.apply(scope,arguments)},
false);
    }
```

```
}
```

5.3.7 预留回调接口

代码清单 5-98 设置回调函数

```
    …
function setTab(root,currentClass,trigger,handler){
        var tabMenus = GLOBAL.Dom.getElementsByClassName("J_tab-menu",root),
        tabContents = GLOBAL.Dom.getElementsByClassName("J_tab-content",root);
        trigger = trigger || "click";
        for(var i=0;i<tabMenus.length;i++){
                tabMenus[i]._index = i;
                GLOBAL.Event.on(tabMenus[i],trigger,function(){
                        for(var j=0;j<tabContents.length;j++){
                                tabContents[j].style.display = "none";
                        }
                        tabContents[this._index].style.display = "block";
                        if(currentClass){
                                var currentMenu =
GLOBAL.Dom.getElementsByClassName(currentClass,root)[0];
                                if(currentMenu){
        GLOBAL.Dom.removeClass(currentMenu,currentClass);
                                }
                                GLOBAL.Dom.addClass(this,currentClass);
                        }
                        if(handler){
                                handler(this._index);
                        }
                });
        }
}

var tabs = GLOBAL.Dom.getElementsByClassName("J_tab");
setTab(tabs[0],null,"mouseover");
setTab(tabs[1],"tab-currentMenu");
setTab(tabs[2],"tab-currentMenu2","mouseover",function(index){alert("您激活的是第
"+(index+1)+"个标签")});
```

5.3.8 don’t repeat yourself

代码清单 5-99 添加自动切换功能

```
    …
function setTab(root,currentClass,trigger,handler,autoPlay,playTime){
    var tabMenus = GLOBAL.Dom.getElementsByClassName("J_tab-menu",root),
    tabContents = GLOBAL.Dom.getElementsByClassName("J_tab-content",root);
    trigger = trigger || "click";
    playTime = playTime || 3000;
    var currentIndex = 0;
    function autoHandler(){
        currentIndex++;
        if(currentIndex >= tabMenus.length){
                currentIndex = 0;
        }
        for(var i=0;i<tabContents.length;i++){
                tabContents[i].style.display = "none";
        }
        tabContents[currentIndex].style.display = "block";
        if(currentClass){
```

```
                var currentMenu =
GLOBAL.Dom.getElementsByClassName(currentClass,root)[0];
                if(currentMenu){
        GLOBAL.Dom.removeClass(currentMenu,currentClass);
                }
                GLOBAL.Dom.addClass(tabMenus[currentIndex],currentClass);
        }
        if(handler){
                handler(currentIndex);
        }
    }
    if(autoPlay){
        setInterval(autoHandler,playTime);
    }
    for(var i=0;i<tabMenus.length;i++){
        tabMenus[i]._index = i;
        GLOBAL.Event.on(tabMenus[i],trigger,function(){
                for(var j=0;j<tabContents.length;j++){
                        tabContents[j].style.display = "none";
                }
                tabContents[this._index].style.display = "block";
                if(currentClass){
                        var currentMenu =
GLOBAL.Dom.getElementsByClassName(currentClass,root)[0];
                        if(currentMenu){
        GLOBAL.Dom.removeClass(currentMenu,currentClass);
                        }
                        GLOBAL.Dom.addClass(this,currentClass);
                }
                if(handler){
                        handler(this._index);
                }
                currentIndex = this._index;
        });
    }
}

var tabs = GLOBAL.Dom.getElementsByClassName("J_tab");
setTab(tabs[0],null,"mouseover");
setTab(tabs[1],"tab-currentMenu",null,null,true,5000);
setTab(tabs[2],"tab-currentMenu2","mouseover",function(index){alert("您激活的是第
"+(index+1)+"个标签")});
```

代码清单 5-100　重构代码，将相同代码提取出来

```
    …
function setTab(root,currentClass,trigger,handler,autoPlay,playTime){
    var tabMenus = GLOBAL.Dom.getElementsByClassName("J_tab-menu",root),
    tabContents = GLOBAL.Dom.getElementsByClassName("J_tab-content",root);
    trigger = trigger || "click";
    playTime = playTime || 3000;
    var currentIndex = 0;
    function showItem(n){
        for(var i=0;i<tabContents.length;i++){
                tabContents[i].style.display = "none";
        }
        tabContents[n].style.display = "block";
        if(currentClass){
                var currentMenu =
GLOBAL.Dom.getElementsByClassName(currentClass,root)[0];
```

```
                if(currentMenu){
        GLOBAL.Dom.removeClass(currentMenu,currentClass);
                }
                GLOBAL.Dom.addClass(tabMenus[n],currentClass);
        }
        if(handler){
                handler(n);
        }
    }
    function autoHandler(){
        currentIndex++;
        if(currentIndex >= tabMenus.length){
                currentIndex = 0;
        }
        showItem(currentIndex);
    }
    if(autoPlay){
        setInterval(autoHandler,playTime);
    }
    for(var i=0;i<tabMenus.length;i++){
        tabMenus[i]._index = i;
        GLOBAL.Event.on(tabMenus[i],trigger,function(){
                showItem(this._index);
                currentIndex = this._index;
        });
    }
}

var tabs = GLOBAL.Dom.getElementsByClassName("J_tab");
setTab(tabs[0],null,"mouseover");
setTab(tabs[1],"tab-currentMenu",null,null,true,5000);
setTab(tabs[2],"tab-currentMenu2","mouseover",function(index){alert("您激活的是第
"+(index+1)+"个标签")});
```

5.3.9 用 hash 对象传参

代码清单 5-101　普通方式传参和 hash 对象传参

```
    // 普通方式传参
    function test(a,b,c){
        var oA = a || 1, oB = b || 2, oC = c || 3;
    }
    test(4,5,6);
    test(null,7,8);
    test(null,null,9);

    // 用 hash 对象传参
    function test2(config){
        var oA = config.a || 1, oB = config.b || 2, oC = config.c || 3;
    }
    test2({a:4,b:5,c:6});
    test2({b:7,c:8});
    test2({c:9});
```

代码清单 5-102　改用 hash 对象传参

```
    …
function setTab(config){
    var root = config.root;
    var currentClass = config.currentClass;
    var trigger = config.trigger || "click";
```

```
    var handler = config.handler;
    var autoPlay = config.autoPlay;
    var playTime = config.playTime || 3000;
    var tabMenus = GLOBAL.Dom.getElementsByClassName("J_tab-menu",root),
    tabContents = GLOBAL.Dom.getElementsByClassName("J_tab-content",root);
    var currentIndex = 0;
    function showItem(n){
        for(var i=0;i<tabContents.length;i++){
                tabContents[i].style.display = "none";
        }
        tabContents[n].style.display = "block";
        if(currentClass){
                var currentMenu =
GLOBAL.Dom.getElementsByClassName(currentClass,root)[0];
                if(currentMenu){
        GLOBAL.Dom.removeClass(currentMenu,currentClass);
                }
                GLOBAL.Dom.addClass(tabMenus[n],currentClass);
        }
        if(handler){
                handler(n);
        }
    }
    function autoHandler(){
        currentIndex++;
        if(currentIndex >= tabMenus.length){
                currentIndex = 0;
        }
        showItem(currentIndex);
    }
    if(autoPlay){
        setInterval(autoHandler,playTime);
    }
    for(var i=0;i<tabMenus.length;i++){
        tabMenus[i]._index = i;
        GLOBAL.Event.on(tabMenus[i],trigger,function(){
                showItem(this._index);
                currentIndex = this._index;
        });
    }
}

var tabs = GLOBAL.Dom.getElementsByClassName("J_tab");
setTab({root:tabs[0],trigger:"mouseover"});
setTab({root:tabs[1],currentClass:"tab-currentMenu",autoPlay:true,playTime:5000});
setTab({root:tabs[2],currentClass:"tab-currentMenu2",trigger:"mouseover",handler:fu
nction(index){alert("您激活的是第"+(index+1)+"个标签")}});
```

5.4面向对象编程

5.4.1　面向过程编程和面向对象编程

代码清单 5-103　电话本程序

```
    // 定义电话本 1
    var phonebook = [
        {name:"adang",tel:"111111"},
        {name:"zhangxia",tel:"222222"},
        {name:"yangfuchuan",tel:"333333"},
```

```
        {name:"qiaojiwu",tel:"444444"},
        {name:"zhouyubo",tel:"555555"}
    ];

    // 定义电话本 2
    var phonebook2 = [
        {name:"niaoren",tel:"111111"},
        {name:"j2ee",tel:"222222"},
        {name:"baobao",tel:"333333"}
    ];

    // 查询电话
    function getTel(oPhonebook,oName){
        var tel = "";
        for(var i=0;i<oPhonebook.length;i++){
                if(oPhonebook[i].name == oName){
                        tel = oPhonebook[i].tel;
                        break;
                }
        }
        return tel;
    }

    // 添加记录
    function addItem(oPhonebook,oName,oTel){
        oPhonebook.push({name:oName,tel:oTel});
    }

    // 删除记录
    function removeItem(oPhonebook,oName){
        var n;
        for(var i=0;i<oPhonebook.length;i++){
                if(oPhonebook[i].name == oName){
                        n = i;
                        break;
                }
        }
        if(n != undefined){
                oPhonebook.splice(n,1);
        }
    }

    // 从电话本 1 中查询”niaoren”的电话
    var str = getTel(phonebook,"niaoren");
    alert(str);                             // ""

    // 在电话本 1 中添加”niaoren”的记录
    addItem(phonebook,"niaoren","666666");
    str = getTel(phonebook,"niaoren");
    alert(str);                             // "666666"

    // 在电话本 1 中删除”niaoren”的记录
    removeItem(phonebook,"niaoren");
    str = getTel(phonebook,"niaoren");
    alert(str);                             // ""

    // 在电话本 2 中查询”niaoren”的记录
    str = getTel(phonebook2,"niaoren");
    alert(str);                             // "111111"
```

代码清单 5-104　面对过程方式编程

```
var name = "adang";
var state = "awake";
var say = function(oName){
     alert("I'm " + oName);
}
var sleep = function(oState){
     oState = "asleep"
}
say(name);
sleep(state);
```

代码清单 5-105　面向对象方式编程

```
var adang = {
     name : "adang",
     state : "awake",
     say : function(){
              alert("I'm " + this.name);
     },
     sleep : function(){
              this.state = "asleep";
     }
};
adang.say();
adang.sleep();
```

5.4.2　JavaScript 的面向对象编程

代码清单 5-106　java 的类

```
// 定义 Animal 类
public class Animal{
    String name , type = "animal";
    public Animal(String a) {
              name = a;
     }
    public void say(){
              System.out.println("I'm a(an) " + type + " , my name is " + name);
    }
}
...
// 实例化 Animal 类
Animal myDog = new Animal("wangcai");
myDog.say();
```

代码清单 5-107　JavaScript 中的类

```
// 函数作为普通函数
function sayHi(){
     alert("hi");
}
// 调用函数
sayHi();

// 函数作为类
function Animal(name){
     this.name = name;
     this.type = "animal";
     this.say = function(){
              alert("I'm a(an) " + this.type + " , my name is " + this.name);
     }
}
```

```
// 实例化 Animal 类
var myDog = new Animal("wangcai");
myDog.say();
```

代码清单 5-108　构造函数和原型之间的引用

```
// 定义 Animal 类的构造函数
function Animal(){
    …
}
var a = Animal.prototype;          // a 指向 Animal 类对应的原型
var b = a.constructor;             // b 指向 a 对应的类的构造函数
alert(b == Animal)                 // true
```

代码清单 5-109　使用原型

```
// 定义 Animal 类
function Animal(){

}
// 修改 Animal 类的原型
Animal.prototype = {
    name : "xxx",
    type : "animal",
    say : function(){
            alert("I'm a(an) " + this.type + " , my name is " + this.name);
    }
}
// 实例化 Animal 类
var myDog = new Animal();
myDog.say();          // I'm a(an) animal , my name is xxx
```

代码清单 5-110　使用原型的另一种方法

```
// 修改 Animal 类的原型
Animal.prototype.name = "xxx";
Animal.prototype.type = "ANIMAL";
Animal.prototype.say = function(){
    alert("I'm a(an) " + this.type + " , my name is " + this.name);
};
```

代码清单 5-111　JavaScript 类的典型用法

```
// 定义 Animal 类
function Animal(name){
    this.name = name || "xxx";
    this.type = "animal";
}
// 修改 Animal 类的原型
Animal.prototype = {
    say : function(){
            alert("I'm a(an) " + this.type + " , my name is " + this.name);
    }
}
// 实例化 Animal 类
var myDog = new Animal("wangcai");
myDog.say();          // I'm a(an) animal , my name is wangcai
```

代码清单 5-112　JavaScript 中的公有和私有

```
// 定义 Animal 类
function Animal(name){
    // 公有属性
    this.name = name || "xxx";
```

```
        this.type = "animal";
        // 私有属性
        var age = 20;
        // 私有方法
        var move = function(){
                aler("I'm moving now");
        }
    }
    // 修改 Animal 类的原型
    Animal.prototype = {
        // 公有方法
        say : function(){
                alert("I'm a(an) " + this.type + " , my name is " + this.name);
        }
    }
    // 实例化 Animal 类
    var myDog = new Animal("wangcai");
    alert(myDog.name);    // wangcai
    alert(myDog.age);     // undefined
    myDog.move();         // 报错，myDog.move is not function
```

代码清单 5-113　私有属性的作用域

```
    // 定义 Animal 类
    function Animal(name){
        // 公有属性
        this.name = name || "xxx";
        this.type = "animal";
        // 私有属性
        var age = 20;
        // 私有方法
        var move = function(){
                aler("I'm moving now");
        }
    }
    // 修改 Animal 类的原型
    Animal.prototype = {
        // 公有方法
        say : function(){
                alert("I'm a(an) " + this.type + " , my name is " + this.name + " , I'm
" + age);
        },
        act : function(){
                move();
        }
    }
    // 实例化 Animal 类
    var myDog = new Animal("wangcai");
    myDog.say();          // 报错，age 未定义
    myDog.act();          // 报错，move is not defined
```

代码清单 5-114　行为访问私有属性的方法

```
    // 定义 Animal 类
    function Animal(name){
        // 公有属性
        this.name = name || "xxx";
        this.type = "animal";
        // 私有属性
        var age = 20;
        // 私有方法
        var move = function(){
```

```
                alert("I'm moving now");
        }
        this.say = function(){
                alert("I'm a(an) " + this.type + " , my name is " + this.name + " , I'm
" + age);
        }
        this.act = function(){
                move();
        }
    }
    // 修改 Animal 类的原型
    Animal.prototype = {
    }
    // 实例化 Animal 类
    var myDog = new Animal("wangcai");
    myDog.say();          // I'm a(an) animal , my name is wangcai , I'm 20
    myDog.act();          // I'm moving now
```

代码清单 5-115　使用命名约定来模拟“私有”属性

```
    // 定义 Animal 类
    function Animal(name){
        // 公有属性
        this.name = name || "xxx";
        this.type = "animal";
        // 私有属性
        this._age = 20;
    }
    // 修改 Animal 类的原型
    Animal.prototype = {
        // _开头，私有方法
        _move : function(){
                alert("I'm moving now");
        },
        // 公有方法
        say : function(){
                alert("I'm a(an) " + this.type + " , my name is " + this.name + " , I'm
" + this._age);
        },
        act : function(){
                this._move();
        }
    }
    // 实例化 Animal 类
    var myDog = new Animal("wangcai");
    myDog.say();          // I'm a(an) animal , my name is wangcai , I'm 20
    myDog.act();          // I'm moving now
    myDog._move();        // I'm moving now  (不推荐实例直接调用_move，违反命名约定)
    alert(myDog._age);    // 20   (不推荐实例直接调用_age，违反命名约定)
```

代码清单 5-116　通过 get 和 set 方法访问属性

```
    function Animal(name){
        var name;
        var type = "animal";
        var _age = 20;
        this.getName = function(){
                return name;
        }
        this.setName = function(o){
                name = o;
        }
```

```
        this.getType = function(){
                return type;
        }
        this.setType = function(o){
                type = o;
        }
        this._getAge = function(){
                return _age;
        }
        this._setAge = function(o){
                _age = o;
        }
        this.setName(name);
    }
    Animal.prototype = {
         _move : function(){
                alert("I'm moving now");
        },
        say : function(){
                alert("I'm a(an) " + this.getType() + " , my name is " + this.getName()
+ " , I'm " + this._getAge());
        },
        act : function(){
                this._move();
        }
    }


    var myDog = new Animal("wangcai");
    myDog.say();                    // I’m a(an) animal , my name is wangcai , I’m 20
    myDog.setType("dog");           // 通过 set 方法设置 type 属性的值
    alert(myDog.getType());         // 通过 get 方法获取 type 属性的值
```

代码清单 5-117　通过 set 方法保护属性

```
    function Animal(name){
        var name;
        var type = "animal";
        var _age = 20;
        this.getName = function(){
                return name;
        }
        this.setName = function(o){
                if(o != "wangcai" && o != "xiaoqiang"){
                        alert("您设置的 name 值不合要求");
                        return;
                }
                name = o;
        }
        this.getType = function(){
                return type;
        }
        this.setType = function(o){
                alert("赋值失败，Animal 类的 type 属性是只读的");
        }
        this._getAge = function(){
                return _age;
        }
        this._setAge = function(o){
                _age = o;
        }
```

```
        this.setName(name);
    }
    Animal.prototype = {
         _move : function(){
                alert("I'm moving now");
        },
        say : function(){
                alert("I'm a(an) " + this.getType() + " , my name is " + this.getName()
+ " , I'm " + this._getAge());
        },
        act : function(){
                this._move();
        }
    }

    var myDog = new Animal("wangcai");
    myDog.say();                    // I’m a(an) animal , my name is wangcai , I’m 20
    myDog.setName("abc");           // 您设置的 name 值不合要求
    alert(myDog.getName());         // wangcai
    myDog.setName("xiaoqiang");
    alert(myDog.getName());         // xiaoqiang
    myDog.setType("dog");           // 赋值失败，Animal 类的 type 属性是只读的
```

代码清单 5-118　监听属性的 valueChange

```
    function Animal(name){
        var name = name;
        var type = "animal";
        var _age = 20;
        // 添加 master 属性，默认值为 adang
        var master = "adang";
        this.getName = function(){
                return name;
        }
        this.setName = function(o){
                if(o != "wangcai" && o != "xiaoqiang"){
                        alert("您设置的 name 值不合要求");
                        return;
                }
                name = o;
                // 触发 name 属性的 valueChange 事件
                this._valueChangeHandler("name");
        }
        // master 属性的获取方法
        this.getMaster = function(){
                return master;
        }
        // master 属性的设置方法
        this.setMaster = function(o){
                master = o;
                // 触发 master 属性的 valueChange 事件
                this._valueChangeHandler("master");
        }
        this.getType = function(){
                return type;
        }
        this.setType = function(o){
                alert("赋值失败，Animal 类的 type 属性是只读的");
        }
        this._getAge = function(){
                return _age;
```

```
        }
        this._setAge = function(o){
                _age = o;
        }
    }
    Animal.prototype = {
         _move : function(){
                alert("I'm moving now");
        },
        say : function(){
                alert("I'm a(an) " + this.getType() + " , my name is " + this.getName()
+ " , I'm " + this._getAge());
        },
        act : function(){
                this._move();
        },
        // 公有行为，用于注册属性的 valueChange 事件
        onChange : function(valueName,fun){
                this["_" + valueName + "ChangeHadlers"] = this["_" + valueName +
"ChangeHadlers"] || [];
                this["_" + valueName + "ChangeHadlers"].push(fun);
        },
        // 私有行为，属性 valueChange 的处理函数
        _valueChangeHandler : function(valueName){
                var o = this["_" + valueName + "ChangeHadlers"];
                if(o){
                        for(var i=0,n=o.length;i<n;i++){
                                var methodName = "get" +
valueName.charAt(0).toUpperCase()+valueName.slice(1);
                                o[i](this[methodName]());
                        }
                }
        }
    }

    var myDog = new Animal("wangcai");
    // 给 myDog 注册 name 属性的 valueChange 事件
    myDog.onChange("name",function(o){
        // 回调函数接收新的 value 值作为参数，可以对其进行匹配
        if(o == "xiaoqiang"){
                alert(1);
        } else {
                alert(2);
        }
    });
    // 给 myDog 换个新名字 xiaoqiang
    myDog.setName("xiaoqiang");             // 1
    //给 myDogg 再注册一个 name 属性的 valueChange 事件
    myDog.onChange("name",function(o){
        alert("my new name is " + o);
    });
    // 给 myDog 换个新名字 wangcai
    myDog.setName("wangcai");               // 2 ， my new name is wangcai
    // 给 myDog 注册 master 属性的 valueChange 事件
    myDog.onChange("master",function(o){
        alert("my new master is " + o);
    })
    // 给 MyDog 换个主人
    myDog.setMaster("Yang Fuchuan");        // my new master is Yang Fuchuan
    var myDog2 = new Animal("xiaoqiang");
```

```
// 给 myDog2 注册 master 属性的 valueChange 事件
myDog2.onChange("master",function(o){
    alert(o + " is my new master");
})
myDog2.setMaster("Zhou Yubo");        // Zhou Yubo is my new master
```

代码清单 5-119　定义 Animal 类

```
// 定义 Animal 类
function Animal(name){
    this.name = name;
    this.type = "animal";
}
Animal.prototype = {
    say : function(){
            alert("I'm a(an) " + this.type + " , my name is " + this.name);
    }
}
```

代码清单 5-120　定义 Bird 类

```
// 定义 Animal 类
function Animal(name){
    this.name = name;
    this.type = "animal";
}
Animal.prototype = {
    say : function(){
            alert("I'm a(an) " + this.type + " , my name is " + this.name);
    }
}

// 定义 Bird 类
function Bird(){

}
Bird.prototype = {

}
```

代码清单 5-121　继承构造函数中的属性和行为

```
// 定义 Animal 类
function Animal(name){
    this.name = name;
    this.type = "animal";
}
Animal.prototype = {
    say : function(){
            alert("I'm a(an) " + this.type + " , my name is " + this.name);
    }
}

// 定义 Bird 类
function Bird(name){
    this.name = name;
    this.type = "animal";
}
Bird.prototype = {

}

// 实例化 Bird 对象
```

```
    var myBird = new Bird("xiaocui");
    alert(myBird.type);                    // animal
```

代码清单 5-122　改进构造函数中的继承

```
    // 定义 Animal 类
    function Animal(name){
         this.name = name;
         this.type = "animal";
    }
    Animal.prototype = {
         say : function(){
                  alert("I'm a(an) " + this.type + " , my name is " + this.name);
         }
    }

    // 定义 Bird 类
    function Bird(name){
         Animal(name);
    }
    Bird.prototype = {

    }

    // 实例化 Bird 对象
    var myBird = new Bird("xiaocui");
    alert(myBird.type);                    // undefined
```

代码清单 5-123　实际效果

```
    // 定义 Animal 类
    function Animal(name){
         this.name = name;
         this.type = "animal";
    }
    Animal.prototype = {
         say : function(){
                  alert("I'm a(an) " + this.type + " , my name is " + this.name);
         }
    }

    // 定义 Bird 类
    function Bird(name){
         window.name = name;
         window.type = "animal";
    }
    Bird.prototype = {

    }

    // 实例化 Bird 对象
    var myBird = new Bird("xiaocui");
    alert(myBird.type);                    // undefined
    alert(type);                           // animal （window.type 可省略写成 type）
```

代码清单 5-124　使用 call 方法指定 this 指向

```
    // 定义 Animal 类
    function Animal(name){
         this.name = name;
         this.type = "animal";
    }
    Animal.prototype = {
```

```
        say : function(){
                alert("I'm a(an) " + this.type + " , my name is " + this.name);
        }
    }

    // 定义 Bird 类
    function Bird(name){
        Animal.call(this,name);
    }
    Bird.prototype = {

    }

    // 实例化 Bird 对象
    var myBird = new Bird("xiaocui");
    alert(myBird.type);                    // animal
```

代码清单 5-125　原型的继承

```
    // 定义 Animal 类
    function Animal(name){
        this.name = name;
        this.type = "animal";
    }
    Animal.prototype = {
        say : function(){
                alert("I'm a(an) " + this.type + " , my name is " + this.name);
        }
    }

    // 定义 Bird 类
    function Bird(name){
        Animal.call(this,name);
    }
    Bird.prototype = Animal.prototype;

    // 实例化 Bird 对象
    var myBird = new Bird("xiaocui");
    myBird.say();         // I'm a(an) animal , my name is xiao cui
```

代码清单 5-126　添加 fly 行为

```
    // 定义 Animal 类
    function Animal(name){
        this.name = name;
        this.type = "animal";
    }
    Animal.prototype = {
        say : function(){
                alert("I'm a(an) " + this.type + " , my name is " + this.name);
        }
    }

    // 定义 Bird 类
    function Bird(name){
        Animal.call(this,name);
    }
    Bird.prototype = Animal.prototype;
    Bird.prototype.fly = function(){
        alert("I'm flying");
    }
```

```
// 实例化 Bird 对象
var myBird = new Bird("xiaocui");
myBird.say();        // I'm a(an) animal , my name is xiao cui
myBird.fly();// I’m flying
var myDog = new Animal("wangcai");
myDog.fly();         // I’m flying
```

代码清单 5-127　传值与传址

```
var a = 10;                 // 基本数据类型
var b = a;                  // 将变量 a 保存的值复制一份，传给变量 b，a 和 b 各保存一份数据
var c = [1,2,3];            // 复杂数据类型
var d = c;                  // 将变量 c 指向的数据的内存地址传给变量 d，c 和 d 指向同一份数据
b++;
d.push(4);
alert(a);    // 10
alert(b);    // 11             变量 b 保存的数据更改不会影响到变量 a
alert(c);    // 1,2,3,4        变量 c 和 d 指向同一份数据，数据更改会互相影响
alert(d);    // 1,2,3,4
```

代码清单 5-128　对复杂数据类型进行传值

```
var a = [1,2,3] , b = {name:"adang",sex:"male",tel:"1234567"};
var c = [] , d = {};
for(var p in a){
    c[p] = a[p];
}
for(var p in b){
    d[p] = b[p];
}
c.push("4");
d.email = "xxx@gmail.com";
alert(a);                   // 1,2,3
alert(c);                   // 1,2,3,4
alert(b.email);             // undefined
alert(d.email);             // xxx@gmail.com
```

代码清单 5-129　数组传值的简单方法

```
var a = [1,2,3];
var b = a.slice() , c = a.concat();
b.pop();
c.push(4);
alert(a);            // 1,2,3
alert(b);            // 1,2
alert(c);            // 1,2,3,4
```

代码清单 5-130　改进原型继承

```
// 定义 Animal 类
function Animal(name){
    this.name = name;
    this.type = "animal";
}
Animal.prototype = {
    say : function(){
            alert("I'm a(an) " + this.type + " , my name is " + this.name);
    }
}

// 定义 Bird 类
function Bird(name){
    Animal.call(this,name);
```

```
}
Bird.prototype = new Animal();
Bird.prototype.constructor = Bird;
Bird.prototype.fly = function(){
     alert("I'm flying");
}

// 实例化 Bird 对象
var myBird = new Bird("xiaocui");
myBird.say();         // I'm a(an) animal , my name is xiao cui
myBird.fly();         // I’m flying
var myDog = new Animal("wangcai");
myDog.fly();          // 报错，myDog.fly is not a function
```

代码清单 5-131　定义 extend 函数

```
function extend(subClass,superClass){
    var F = function(){};
    F.prototype = superClass.prototype;
    subClass.prototype = new F();
    subClass.prototype.constructor = subClass;
    subClass.superclass = superClass.prototype;
    if(superClass.prototype.constructor == Object.prototype.constructor){
        superClass.prototype.constructor = superClass;
    }
}

function Animal(name){
     this.name = name;
     this.type = "animal";
}
Animal.prototype = {
     say : function(){
              alert("I'm a(an) " + this.type + " , my name is " + this.name);
     }
}
function Bird(name){
     this.constructor.superclass.constructor.apply(this,arguments);
     this.type = "bird"
}
extend(Bird,Animal);
Bird.prototype.fly = function(){
     alert("I'm flying");
}
var canary = new Bird("xiaocui");
canary.say();                 // I’m a(an) bird , my name is xiaocui
canary.fly();                 // I’m flying
```

5.4.3　用面向对象方式重写代码

代码清单 5-132　用面向对象方式重写电话本程序

```
// 定义电话本管理类
function PhonebookManager(){
     this._phonebook = null;
}
PhonebookManager.prototype = {
     // 查询电话
     getTel : function(){

     },
```

```
        // 添加记录
        addItem : function(){

        },
        // 删除记录
        removeItem : function(){

        }
    }
```

代码清单 5-133　定义行为的参数

```
    // 定义电话本管理类
    function PhonebookManager(o){
        this._phonebook = o;
    }
    PhonebookManager.prototype = {
        // 查询电话
        getTel : function(oName){

        },
        // 添加记录
        addItem : function(oName,oTel){

        },
        // 删除记录
        removeItem : function(oName){

        }
    }
```

代码清单 5-134　编写具体的行为代码

```
    // 定义电话本管理类
    function PhonebookManager(o){
        this._phonebook = o;
    }
    PhonebookManager.prototype = {
        // 查询电话
        getTel : function(oName){
                var tel = "";
                for(var i=0;i<this._phonebook.length;i++){
                        if(this._phonebook[i].name == oName){
                                tel = this._phonebook[i].tel;
                                break;
                        }
                }
                return tel;
        },
        // 添加记录
        addItem : function(oName,oTel){
                this._phonebook.push({name:oName,tel:oTel});
        },
        // 删除记录
        removeItem : function(oName){
                var n;
                for(var i=0;i<this._phonebook.length;i++){
                        if(this._phonebook[i].name == oName){
                                n = i;
                                break;
                        }
                }
```

```
                if(n != undefined){
                        this._phonebook.splice(n,1);
                }
        }
    }
```

代码清单 5-135　使用面向对象的电话本程序

```
    // 实例化两个电话本管理对象
    var myPhonebookManager = new PhonebookManager([
        {name:"adang",tel:"111111"},
        {name:"zhangxia",tel:"222222"},
        {name:"yangfuchuan",tel:"333333"},
        {name:"qiaojiwu",tel:"444444"},
        {name:"zhouyubo",tel:"555555"}
    ]),
       myPhonebookManager2 = new PhonebookManager([
        {name:"niaoren",tel:"111111"},
        {name:"j2ee",tel:"222222"},
        {name:"baobao",tel:"333333"}
    ]);


    // 从电话本 1 中查询"niaoren"的电话
    var str = myPhonebookManager.getTel("niaoren");
    alert(str);                             // ""

    // 在电话本 1 中添加"niaoren"的记录
    myPhonebookManager.addItem("niaoren","666666");
    str = myPhonebookManager.getTel("niaoren");
    alert(str);                             // "666666"

    // 在电话本 1 中删除"niaoren"的记录
    myPhonebookManager.removeItem("niaoren");
    str = myPhonebookManager.getTel("niaoren");
    alert(str);                             // ""

    // 在电话本 2 中查询"niaoren"的记录
    str = myPhonebookManager2.getTel("niaoren");
    alert(str);                             // "111111"
```

代码清单 5-136　用面向对象方式重构 Tab

```
    var GLOBAL = {};
    GLOBAL.namespace = function(str){
        var arr = str.split("."),o = GLOBAL;
        for (i=(arr[0] == "GLOBAL") ? 1 : 0; i<arr.length; i++) {
                o[arr[i]]=o[arr[i]] || {};
                o=o[arr[i]];
        }
    }
    GLOBAL.namespace("Dom");
    GLOBAL.Dom.getElementsByClassName = function(str,root,tag){
        if(root){
                root = typeof root == "string" ? document.getElementById(root) : root;
        } else {
                root = document.body;
        }
        tag = tag || "*";
        var els = root.getElementsByTagName(tag),arr = [];
        for(var i=0,n=els.length;i<n;i++){
                for(var j=0,k=els[i].className.split(" "),l=k.length;j<l;j++){
```

```
                        if(k[i] == str){
                                arr.push(els[i]);
                                break;
                        }
                }
        }
        return arr;
}
GLOBAL.Dom.addClass = function(node,str){
        if(!new RegExp("(^|\\s+)"+str).test(node.className)){
                node.className = node.className + " " + str;
        }
}
GLOBAL.Dom.removeClass = function(node,str){
        node.className = node.className.replace(new RegExp("(^|\\s+)"+str),"");
}
GLOBAL.namespace("Event");
GLOBAL.Event.on = function(node,eventType,handler,scope){
        node = typeof node == "string" ? document.getElementById(node) : node;
        scope = scope || node;
        if(document.all){

        node.attachEvent("on"+eventType,function(){handler.apply(scope,arguments)})
;
        } else {

        node.addEventListener(eventType,function(){handler.apply(scope,arguments)},
false);
        }
}

function Tab(config){
    this._root = config.root;
    this._currentClass = config.currentClass;
    var trigger = config.trigger || "click";
    this._handler = config.handler;
    var autoPlay = config.autoPlay;
    var playTime = config.playTime || 3000;
    this._tabMenus = GLOBAL.Dom.getElementsByClassName("J_tab-menu",this._root);
    this._tabContents =
GLOBAL.Dom.getElementsByClassName("J_tab-content",this._root);
    this.currentIndex = 0;
    var This = this;
    if(autoPlay){
        setInterval(function(){This._autoHandler()},playTime);
    }
    for(var i=0;i<this._tabMenus.length;i++){
        this._tabMenus[i]._index = i;
        GLOBAL.Event.on(this._tabMenus[i],trigger,function(){
                This.showItem(this._index);
                this.currentIndex = this._index;
        });
    }
}
Tab.prototype = {
    showItem : function(n){
        for(var i=0;i<this._tabContents.length;i++){
                this._tabContents[i].style.display = "none";
        }
        this._tabContents[n].style.display = "block";
```

```
        if(this._currentClass){
                var currentMenu =
GLOBAL.Dom.getElementsByClassName(this._currentClass,this._root)[0];
                if(currentMenu){
                        GLOBAL.Dom.removeClass(currentMenu,this._currentClass);
                }
                GLOBAL.Dom.addClass(this._tabMenus[n],this._currentClass);
        }
        if(this._handler){
                this._handler(n);
        }
    },
    _autoHandler : function(){
        this.currentIndex++;
        if(this.currentIndex >= this._tabMenus.length){
                this.currentIndex = 0;
        }
        this.showItem(this.currentIndex);
    }
}

var tabs = GLOBAL.Dom.getElementsByClassName("J_tab");
new Tab({root:tabs[0],trigger:"mouseover"});
new Tab({root:tabs[1],currentClass:"tab-currentMenu",autoPlay:true,playTime:5000});
new Tab({root:tabs[2] , currentClass:"tab-currentMenu2" , trigger:"mouseover" ,
handler:function(index){alert("您激活的是第"+(index+1)+"个标签")}});
```

5.5其他问题

5.5.1 prototype 和内置类

代码清单 5-137 调用内置类

```
  var a = new String("hello world");      // 通过 new String()实例化 string 类型对象
  var b = "hello world";                  // 直接通过""实例化 string 类型对象
  alert(a.length);
  alert(b.length);
  var c = new Array(1,2,3);               // 通过 new Array()实例化 array 类型对象
  var d = [1,2,3];                        // 直接通过[]实例化 array 类型对象
  c.push(4);
  d.pop();
  alert(c);
  alert(d);
```

代码清单 5-138 扩展内置类的行为

```
  Array.prototype.each = function(fun){
        for(var i = 0,n = this.length;i<n;i++){
                fun(this[i],i);
        }
  }
  Array.prototype.clone = function(){
        var o = [];
        this.each(function(v,k){
                o[k] = v;
        });
        return o;
  }
  Array.prototype.map = function(fun){
        var o = [];
```

```
        this.each(function(v,k){
                o[k] = fun(v,k);
        });
        return o;
}
//因为 IE 中 delete 是保留字，所以方法名改用 Delete
Array.prototype.Delete = function(a){
        var o = this.clone();
        for(var i=o.length,n=0;i>n;i--){
                if(o[i] == a){
                        o.splice(i,1);
                }
        }
        return o;
}
var a = [1,2,3,2,4,5];
var str = "";
a.each(function(v,k){
        str += k + " : " + v + "\n";
});
alert(str);                             // 0 : 1 1:2 2:3 3:2 4:4 5:5
var b = a.map(function(v,k){
        return v * 10;
});
alert(a);                               // 1,2,3,2,4,5
alert(b);                               // 10,20,30,20,40,50
var c = b.Delete(20);
alert(c);                               // 10,30,40,50
```

代码清单 5-139　重写内置类的方法

```
var a = [1,2,3];
alert(a);                                       // 1,2,3
Array.prototype.toString = function(str){
        return "I'm an array";
}
alert(a);                                       // I’m an array
```

代码清单 5-140　自定义 toString 方法

```
function Dog(o){
        this.name = o;
}
var myDog = new Dog("wang cai");
alert(myDog);           // [object Object]
Dog.prototype.toString = function(){
        return "my name is " + this.name;
}
alert(myDog);           // my name is wang cai
var me = {
        name : "adang",
        email : "cly84920@gmail.com",
        toString : function(){
                return "I'm adang,my email is cly84920@gmail.com";
        }
}
alert(me);              // I'm adang,my email is cly84920@gmail.com
```

代码清单 5-141　重写属性

```
Array.prototype.length = 1;
String.prototype.length = 1;
alert([1,2,3].length);          // 3
```

```
   alert("abc".length);           // 3
```

代码清单 5-142　扩展 Object 类

```
   Object.prototype.test = function(){
         alert("hello world");
   }
   var a = [1,2,3],b = "abc",c = {},d = true,e = function(){};
   a.test();
   b.test();
   c.test();
   d.test();
   e.test();
   function Dog(o){
         this.name = o;
   }
   Dog.prototype.toString = function(){
         return "my name is " + this.name;
   }
   var f = new Dog("wangcai");
   f.test();
```

代码清单 5-143　使用自定义类

```
   function myArray(o){
         this.getArray = function(){
                  return o;
         };
   }
   myArray.prototype = {
         each : function(fun){
                  var o = this.getArray();
                  for(var i=0,n=o.length;i<n;i++){
                           fun(o[i],i);
                  }
         }
   }
   var a = new myArray([1,2,3]),str = "";
   a.each(function(v,k){
         str += k + " : " + v + "\n";
   });
   alert(str);                 // 0 : 1 1 : 2 2 : 3
```

5.5.2　标签的自定义属性

代码清单 5-144　html 标签和 js 的映射

```
   <a id="a" class="b" title="" href="http://www.adanghome.com"
onclick="alert(this.href);return false;">my blog</a>
  <script type="text/JavaScript">
   var node = document.getElementById("a");
   alert(typeof node);                              // object
  </script>
```

代码清单 5-145　在 JavaScript 中获得 DOM 节点对象的属性

```
   <a id="a" class="b" title="" href="http://www.adanghome.com"
onclick="alert(this.href);return false;">my blog</a>
  <script type="text/JavaScript">
   var node = document.getElementById("a");
   alert(node.getAttribute("id"));         // a
   alert(node.id)                          // a
```

代码清单 5-146　IE 和 firefox 获得 DOM 节点对象属性的差异

```
   <a id="a" class="b" title="我的博客" href="http://www.adanghome.com"
onclick="alert(this.href);return false;">my blog</a>
  <script type="text/JavaScript">
   var node = document.getElementById("a");

   alert(node.getAttribute("id"));
   /*
      IE 和 Firefox : a
   */

   alert(node.getAttribute("class"));
   /*
      IE      : null
      Firefox : b
   */

   alert(node.getAttribute("className"));
   /*
      IE      : b
      Firefox : null
   */

   alert(node.getAttribute("title"));
   /*
      IE 和 Firefox : 我的博客
   */

   alert(node.getAttribute("href"));
   /*
      IE 和 Firefox : http://www.adanghome.com
   */

   alert(node.getAttribute("onclick"));
   /*
      IE      : function onclick(){
                          alert(this.href);return false;
                }
      Firefox : alert(this.href);return false;
   */

   alert(node.getAttribute("innerHTML"));
   /*
      IE      : my blog
      Firefox : null
   */

   alert(node.id);
   /*
      IE 和 Firefox : a
   */

   alert(node.className);
   /*
      IE 和 Firefox : b
   */

   alert(node.title);
   /*
```

```
    IE 和 Firefox : 我的博客
*/

alert(node.href);
/*
    IE 和 Firefox : http://www.adanghome.com
*/

alert(node.onclick);
/*
    IE      : function onclick(){
                        alert(this.href);return false;
              }
    Firefox : function onclick(event){
                        alert(this.href);
                        return false;
              }
*/

alert(node.innerHTML));
/*
    IE 和 Firefox : my blog
*/
```

代码清单 5-147　JavaScript 和 html 间的映射

```
<style type="text/CSS">
      .red{color:red;}
</style>
<span id="a">hello world</span>
<script type="text/JavaScript">
      var node = document.getElementById("a");
      node.className = "red";
</script>
```

代码清单 5-148　自定义属性的获取

```
<a id="a" href="http://www.adanghome.com" blogName="阿当的博客" blogType="前端开发">my
blog</a>
<script type="text/JavaScript">
      var node = document.getElementById("a");

      alert(node.blogName);
      /*
          IE      : 阿当的博客
          Firefox : undefined
      */

      alert(node.blogType);
      /*
          IE      : 前端开发
          Firefox : undefined
      */

      alert(node.getAttribute("blogName"));
      /*
          IE 和 Firefox : 阿当的博客
      */

      alert(node.getAttribute("blogType"));
      /*
          IE 和 Firefox : 前端开发
```

```
        */
  </script>
```

代码清单 5-149　自定义属性的反序列化

```
    <a id="a" href="http://www.adanghome.com" blogInfo="{name:'阿当的博客',type:'前端开发'}">my blog</a>

    <script type="text/JavaScript">
        var node = document.getElementById("a");
        var info = node.getAttribute("blogInfo");
        alert(typeof info);               // string
        alert(info.name);                 // undefined
        alert(info.type);                 // undefined
        info = eval("("+info+")");
        alert(typeof info);               // object
        alert(info.name);                 // 阿当的博客
        alert(info.type);                 // 前端开发
    </script>
```

5.5.3　标签的内联事件和 event 对象

代码清单 5-150　简单的点击事件

```
    <input type="button" id="btn" value="click me" />

    <script type="text/JavaScript">
    document.getElementById("btn").onclick = function(){
        alert(arguments.length);
    };
    </script>
```

代码清单 5-151　内联事件中的点击事件

```
    <input type="button" id="btn" value="click me" onclick = "handler()" />

    <script type="text/JavaScript">
    function handler(){
        alert(arguments.length);
    };
    </script>
```

代码清单 5-152　替换可能一

```
    btn.onclick = handler;
    function handler(){
        alert(arguments.length);
    }
```

代码清单 5-153　替换可能二

```
    btn.onclick = function(){
        handler();
    }
    function handler(){
        alert(arguments.length);
    }
```

代码清单 5-154　内联事件和参数

```
    <input type="button" id="btn" value="click me" onclick = "alert(arguments[0].type)" />
```

代码清单 5-155　event 对象的兼容处理

```
    <input type="button" id="btn" value="click me" />

    <script type="text/JavaScript">
    document.getElementById("btn").onclick = function(e){
        e = window.event || e;                  //e 兼容 IE 和 Firefox，指向 event 对象
    };
    </script>
```

代码清单 5-156　在内联事件中兼容 event 对象

```
    <input type="button" id="btn" value="click me" onclick="alert(event.type);" />
```

代码清单 5-157　内联事件的使用

```
    //只弹出 1
    <input type="button" id="btn" value="click me"
onclick="alert(1);//alert(2);alert(3);" />

    //弹出 1 和 3
    <input type="button" id="btn" value="click me"
onclick="alert(1);/*alert(2);*/alert(3);" />

    //弹出“string”
    <input type="button" id="btn" value="click me" onclick="var a='abc';alert(typeof
a);"/>
```

代码清单 5-158　在两处同时监听事件

```
    <input type="button" id="btn" value="click me" onclick="alert(123);" />

    <script type="text/JavaScript">
    document.getElementById("btn").onclick = function(){
        alert(456);
    };
    </script>
```

代码清单 5-159　等同效果

```
    <input type="button" id="btn" value="click me" />

    <script type="text/JavaScript">
    document.getElementById("btn").onclick = function(){
        alert(123);
    };
    document.getElementById("btn").onclick = function(){
        alert(456);
    };
    </script>
```

代码清单 5-160　改进方案，同时监听两处事件

```
    <input type="button" id="btn" value="click me" onclick="alert(123);" />

    <script type="text/JavaScript">
    function handler(){
         alert(456);
    }
    if(document.all){
         btn.attachEvent("onclick",handler);
    } else {
         btn.addEventListener("click",handler,false);
    }
    </script>
```

5.5.4　利用事件冒泡机制

代码清单 5-161　打分程序

```
    <!—- 编写 rate 相关的 html 结构 -->
    <h1>阿当饭店</h1>
    <p>卫生：<p>
    <p class="J_rate">
         <img src="star.gif" title="很烂" />
         <img src="star.gif" title="一般" />
         <img src="star.gif" title="还好" />
         <img src="star.gif" title="较好" />
         <img src="star.gif" title="很好" />
    </p>
    <p>价格：<p>
    <p class="J_rate">
         <img src="star.gif" title="很烂" />
         <img src="star.gif" title="一般" />
         <img src="star.gif" title="还好" />
         <img src="star.gif" title="较好" />
         <img src="star.gif" title="很好" /></p>
    <p>味道：<p>
    <p class="J_rate">
         <img src="star.gif" title="很烂" />
         <img src="star.gif" title="一般" />
         <img src="star.gif" title="还好" />
         <img src="star.gif" title="较好" />
         <img src="star.gif" title="很好" />
    </p>
    <script type="text/JavaScript">
         // 封装 DOM、Event 接口
         var GLOBAL = {};
         GLOBAL.namespace = function(str){
                  var arr = str.split("."),o = GLOBAL;
                  for (i=(arr[0] == "GLOBAL") ? 1 : 0; i<arr.length; i++) {
                           o[arr[i]]=o[arr[i]] || {};
                           o=o[arr[i]];
                  }
         }
         GLOBAL.namespace("Dom");
         GLOBAL.Dom.getElementsByClassName = function(str,root,tag){
                  if(root){
                           root = typeof root == "string" ?
document.getElementById(root) : root;
                  } else {
                           root = document.body;
                  }
                  tag = tag || "*";
                  var els = root.getElementsByTagName(tag),arr = [];
                  for(var i=0,n=els.length;i<n;i++){
                           for(var j=0,k=els[i].className.split("
"),l=k.length;j<l;j++){
                                    if(k[j] == str){
                                             arr.push(els[i]);
                                             break;
                                    }
                           }
                  }
                  return arr;
         }
```

```
        GLOBAL.namespace("Event");
        GLOBAL.Event.on = function(node,eventType,handler,scope){
                node = typeof node == "string" ? document.getElementById(node) : node;
                scope = scope || node;
                if(document.all){

   node.attachEvent("on"+eventType,function(){handler.apply(scope,arguments)});
                } else {

   node.addEventListener(eventType,function(){handler.apply(scope,arguments)},false)
;
                }
        }


        // 定义 Rate 类
        function Rate(rateRoot){
                var root = typeof rateRoot == "string" ?
document.getElementById(rateRoot) : rateRoot;
                var items = root.getElementsByTagName("img");
                var imgs = ["star.gif","star2.gif"];
                var rateFlag;
                for(var i=0,n=items.length;i<n;i++){
                        items[i].index = i;
                        GLOBAL.Event.on(items[i],"mouseover",function(){
                                if(rateFlag) return;
                                for(var j=0;j<n;j++){
                                        if(j<=this.index){
                                                items[j].src = imgs[1];
                                        } else {
                                                items[j].src = imgs[0];
                                        }
                                }
                        });
                        GLOBAL.Event.on(items[i],"mouseout",function(){
                                if(rateFlag) return;
                                for(var j=0;j<n;j++){
                                        items[j].src = imgs[0];
                                }
                        });
                        GLOBAL.Event.on(items[i],"click",function(){
                                if(rateFlag) return;
                                rateFlag = true;
                                alert("您打了"+(this.index+1)+"分");
                        });
                }
        }

        // 实例化 Rate 类
        var rateNodes = GLOBAL.Dom.getElementsByClassName("J_rate");
        for(var i=0,n=rateNodes.length;i<n;i++){
                new Rate(rateNodes[i]);
        }
      </script>
```

代码清单 5-162　等同效果

```
<p class="J_rate">
    <img src="star.gif" title="很烂" onmouseover="…" onmouseout="…" onclick="…" />
    <img src="star.gif" title="一般" onmouseover="…" onmouseout="…" onclick="…" />
    <img src="star.gif" title="还好" onmouseover="…" onmouseout="…" onclick="…" />
```

```
    <img src="star.gif" title="较好" onmouseover="…" onmouseout="…" onclick="…" />
    <img src="star.gif" title="很好" onmouseover="…" onmouseout="…" onclick="…" />
</p>
```

代码清单 5-163　利用事件冒泡进行优化

```
        // 定义 Rate 类
        function Rate(rateRoot){
                var root = typeof rateRoot == "string" ?
document.getElementById(rateRoot) : rateRoot;
                var items = root.getElementsByTagName("img");
                var imgs = ["star.gif","star2.gif"];
                var rateFlag;
                for(var i=0,n=items.length;i<n;i++){
                        items[i].index = i;
                }
                GLOBAL.Event.on(root,"mouseover",function(e){
                        if(rateFlag) return;
                        var target = e.target || e.srcElement;
                        if(target.tagName.toLowerCase() != "img") return;
                        for(var i=0;i<n;i++){
                                if(i<=target.index){
                                        items[i].src = imgs[1];
                                } else {
                                        items[i].src = imgs[0];
                                }
                        }
                });
                GLOBAL.Event.on(root,"mouseout",function(e){
                        if(rateFlag) return;
                        var target = e.target || e.srcElement;
                        for(var i=0,n=items.length;i<n;i++){
                                items[i].src = imgs[0];
                        }
                });
                GLOBAL.Event.on(root,"click",function(e){
                        if(rateFlag) return;
                        rateFlag = true;
                        var target = e.target || e.srcElement;
                        alert("您打了"+(target.index+1)+"分");
                });
        }
```

代码清单 5-164　等同效果

```
<p class="J_rate" onmouseover="…" onmouseout="…" onclick="…">
    <img src="star.gif" title="很烂" />
    <img src="star.gif" title="一般" />
    <img src="star.gif" title="还好" />
    <img src="star.gif" title="较好" />
    <img src="star.gif" title="很好" />
</p>
```

5.5.5　改变 DOM 样式的三种方式

代码清单 5-165　改变 DOM 节点样式的方法一

```
   <span id="test">hello world</span>
   <script type="text/JavaScript">
         var node = document.getElementById("test");
         node.style.color = "red";
   </script>
```

代码清单 5-166　等同效果

```
   <span id="test" style="color:red">hello world</span>
```

代码清单 5-167　用方法一改变多个样式

```
   <span id="test">hello world</span>
   <script type="text/JavaScript">
         var node = document.getElementById("test");
         node.style.color = "red";
         node.style.backgroundColor = "black";
         node.style.fontSize = "40px";
         node.style.fontWeight = "bold";
   </script>
```

代码清单 5-168　等同效果

```
   <span id="test"
style="color:red;background-color:black;font-size:40px;font-weight:bold">hello
world</span>
```

代码清单 5-169　改变 DOM 节点样式的方法二

```
   <style type="text/CSS">
   .testStyle{color:red;background-color:black;font-size:40px;font-weight:bold;}
   </style>
   <span id="test">hello world</span>
   <script type="text/JavaScript">
         var node = document.getElementById("test");
         node.className = "testStyle";
   </script>
```

代码清单 5-170　等同效果

```
   <style type="text/CSS">
   .testStyle{color:red;background-color:black;font-size:40px;font-weight:bold;}
   </style>
   <span id="test" class="testStyle">hello world</span>
```

代码清单 5-171　需设置样式的 DOM 节点

```
   <span id="test">hello world</span>
   <span>aaaaaa</span>
   <span>bbbbbb</span>
   <span>cccccc</span>
```

代码清单 5-172　等同效果一

```
   <span id="test" style="font-size:40px;background:#000;color:red">hello world</span>
   <span style="font-size:40px;background:#000;color:#fff">aaaaaa</span>
   <span style="font-size:40px;background:#000;color:#fff">bbbbbb</span>
   <span style="font-size:40px;background:#000;color:#fff">cccccc</span>
```

代码清单 5-173　等同效果二

```
   <style type="text/CSS">
         .testStyle{font-size:40px;background:#000;color:#fff}
         #test{color:red}
   </style>
   <span id="test" class="testStyle">hello world</span>
   <span class="testStyle">aaaaaa</span>
   <span class="testStyle">bbbbbb</span>
   <span class="testStyle">cccccc</span>
```

代码清单 5-174　改变 DOM 节点样式的方法三

```
   <span id="test">hello world</span>
```

```
   <span>aaaaaa</span>
   <span>bbbbbb</span>
   <span>cccccc</span>
   <script type="text/JavaScript">
         function addStyleNode(str){
                  var styleNode = document.createElement("style");
                  styleNode.type = "text/CSS";
                  if(styleNode.styleSheet){
                           styleNode.styleSheet.cssText = str;
                  } else {
                           styleNode.innerHTML = str;
                  }
                  document.getElementsByTagName("head")[0].appendChild(styleNode);
         }
         addStyleNode("span{font-size:40px;background:#000;color:#fff}
#test{color:red}");
   </script>
```

代码清单 5-175　等同效果

```
   <style type="text/CSS">
         span{font-size:40px;background:#000;color:#fff}
         #test{color:red}
   </style>
   <span id="test">hello world</span>
   <span>aaaaaa</span>
   <span>bbbbbb</span>
   <span>cccccc</span>
```

附录 前端规范

附录 A 写在规则前面的话

附录 B 命名规则

```
html:
<ul class=”textList”>
<li class=”textList_firstItem”>1)XXXXXXXXXXXXXX</li>
<li>2)XXXXXXXXXXXXXX</li>
<li>3)XXXXXXXXXXXXXX</li>
</u1>

CSS:
.textList{ } V              .text_list{ } X
.textList_firstItem{ } V    .textListFirstItem{ } X
.textList_firstItem{ } V
.textList .firstItem{ } X
```

附录 C 分工安排

附录 D 注释规则

```
/**
* 文件用途说明
* 作者姓名
* 联系方式
* 制作日期
**/
//================
```

```
// 代码用途
//=================
//代码说明
//姓名
var name = "abc";        V

var name ="abc"; //姓名   X
```

附录 E HTML 规范

```
<!--头部开始{-->
<div class="head">
<div class="header clearfix">
   <h1 class="fr">阿当制作</h1>
   <h2 class="fb fl">阿当测试站点<span class="gray fn">的页面管理模式</span></h2>
   <input type="button" value="完成并退出" class="ml50 fl" />
</div>
</div>
<!--}头部结束-->
```

附录 F CSS 规范

```
html:
<p>12345</p>
<ul class="textList">
<li>1)XXXXXXXXXXX</li>
<li>2)XXXXXXXXXXX</li>
</ul>
<p>abcde</p>
<ul class="textList2">
<li>1)XXXXXXXXXXX</li>
<li>2)XXXXXXXXXXX</li>
</ul>
CSS:
.textList,.textList2{margin-top:10px;XXXXXXXXXXXXX}
.textList2{margin-top:20px;}
X

=====================================================

html:
<p>12345</p>
<ul class="textList marginTop10">
<li>1)XXXXXXXXXXX</li>
<li>2)XXXXXXXXXXX</li>
</ul>
<p>abcde</p>
<ul class="textList marginTop20">
<li>1)XXXXXXXXXXX</li>
<li>2)XXXXXXXXXXX</li>
</ul>
CSS:
.textList{XXXXXXXXXXXXX}
.marginTop10{margin-top:10px;}
.marginTop20{margin-top:20px;}
V
<p>XXXXXXXXXXXX</p>
<p class="marginTop10 marginBottom10">XXXXXXXXXXX</p>
<p>XXXXXXXXXXXX</p>
X
```

```
=====================================================

<p>XXXXXXXXXXXXX</p>
<p class="marginTop10">XXXXXXXXXXXXX</p>
<p class="marginTop10">XXXXXXXXXXXXX</p>
V
.menu{margin:0;float:left;font-weight:bold;}   V
.menu{
     margin:0;
     float:left;
     font-weight:bold;
}                                               X
```

附录 G JavaScript 规范

互联网进入Web 2.0时代以后，Web应用敲响了传统桌面应用的丧钟，它一路摧城拔寨，如今几乎所有的应用都打上了“Web”的烙印。与之相应的，Web开发技术得到了空前的发展，尤其是前端技术。近年来，随着用户对使用体验的要求越来越高，前端开发的技术难度越来越大，昔日设计和制作不分的网页设计师这一职位终于“拆分”成了视觉设计师和前端开发工程师两个职位，分别向着艺术和技术的方向纵深发展。

Web前端开发工程师是一个很新的职业，在国内乃至国际上真正开始受到重视的时间也不超过5年，这类专业人才一直供不应求。从知识体系上讲，前端开发工程师需要掌握和了解的知识非常之多，甚至可以用庞杂来形容。作为一名没有太多经验的前端开发工程师，我们应该如何去全面认识自己的工作，如何找准自己的定位，如何从合格成为优秀，最后迈向卓越？本书尝试从如何编写易于维护的、高质量的Web前端代码的角度给出答案。

如果你在思考下面这些问题，也许本书就是你想要的！

- 作为一名合格的Web前端开发工程师，究竟需要具备哪些技能和素质？为什么说如果要精通Web前端开发这一行，必须先精通十行？
- 在Web应用的实现代码中，有哪些技术因素会导致应用难以维护？
- 高质量的Web前端代码应该满足哪些条件？如何才能提高Web前端代码的可读性和可重用性？
- 在HTML代码中，为何要使用语义化标签？如何检查你使用的标签是否语义良好？语义化标签时应该注意哪些问题？
- 如何编写CSS代码和JavaScript代码可以避免团队合作时产生冲突？
- 如何组织CSS文件才能让它们更易于管理？如何让CSS模块化，从而提高代码的重用率？CSS的命名应该注意哪些问题？何谓优良的CSS编码风格？
- 如何在CSS编码中引入面向对象的编程思想？这样做有哪些好处？
- 原生JavaScript和JavaScript类库之间有何关系？如何编写自己的JavaScript类库？
- JavaScript有哪些常见的跨浏览器兼容问题？如何解决这些问题？
- 如何组织JavaScript才能让代码的结构更清晰有序，从而更易于维护？如何才能编写出弹性良好的JavaScript代码？编写过程中应该注意哪些问题？
- JavaScript的面向对象编程是如何实现的？如何用面向对象的方式重写原有的代码？
- 编写高质量的JavaScript代码有哪些实用的技巧？又有哪些常见的问题需要注意？
- 为了提高Web前端代码的可维护性，我们应该遵循哪些规范？

上架指导：计算机/程序设计
ISBN 978-7-111-30595-8
9 787111 305958
定价：49.00元

客服热线：(010) 88378991, 88361066
购书热线：(010) 68326294, 88379649, 68995259
投稿热线：(010) 88379604
读者信箱：hzjsj@hzbook.com

华章网站 http://www.hzbook.com
网上购书：www.china-pub.com

封面设计：陈子平