网页设计与制作课程教案

杨志宏[①]

二〇一六年五月十六日

[①]西北民族大学新闻传播学院副教授，Email：yangjh@yeah.net。

目录

> You can code. They cannot. That is pretty damn cool.
>
> *Zed A. Shaw《Learn Python The Hard Way》*

第 1 章　如何学习编程

如今编程成为了一个越来越重要的能力：作为设计师，懂一些编程可能会帮你更好地理解自己的工作内容；作为创业者，技术创始人的身份则会让你的很多工作显得更容易；作为研究者，学会编程能让你按照自己的意图获取数据和处理数据。无论哪个行业，都面临着如何同互联网相融合的机遇和挑战。具备一定的编程能力，无疑会提高在互联网时代的竞争实力。

刚开始学习编程的新手，常面临这样的问题：面对各种学习资料，不知道从哪里开始；好不容易入门后，发现需要学习的内容越来越多、越来越难，陡峭的学习曲线使人望而却步；虽然具有一定的基础知识，但是不知道如何进行一个真正的项目。

幸运的是，我们现在处于互联网蓬勃发展的时代，初学者面临的这些问题，已经有诸多的解决方案，在这一章节中，我们一起了解前人们的经验和教训，从而提高学习效率。

1.1　调整心态

首先要明确一点，编程不是一件轻轻松松就能学会的技能。虽然能在书店中看到大量类似《21 天学通 XXX》的书籍，但一旦你开始学习的时候，就会有这样的体验：所谓的捷径是不存在的，尤其是在编程领域，只要有一处错误，有一处你没有真正掌握的知识点，你就会卡在哪里，只有排除错误后，你才能继续下一个任务。一句话：学习编程意味着你将需要投入大量的时间和精力。

在学习编程的过程中，一些人容易放弃的原因主要有三点：一是没有目标，不知道掌握了编程技能后能解决什么具体问题。二是乏味，在学习编程语言的基础知识（如语法规则、编程模式）时提不起兴趣。三是觉得难，有些知识点的确不是一下就能理解，尝试后知难而退。

如何解决这些学习过程中遇到的困惑呢？根据我个人的经验和别人的总结，提出如下建议：

1.1.1　明确目标

在你学习编程之前思考一下你的目标，你想要用这门语言写什么？网站？游戏？手机 APP？还是想自动完成一些乏味重复的任务？亦或你只是想找个好工作。当然，

所有的这些都是有价值的目标，这些目标都是你学习的驱动力，没有驱动力，很有可能在略显枯燥的漫长学习之旅中半途而废。

行为科学家认为，人的驱动力分为三种：第一驱动力是生物性需求，即对食物、性欲等等的需求；第二驱动力则来自外在动机，做出特定行为时环境会带来的奖励或惩罚；第三驱动力来自于内在动机，我们有“发现新奇事物、进行挑战、拓展并施展才能以及探索和学习的内在倾向”。[1] 第三种驱动力比另外两种更脆弱，它只有在合适的环境中才能存在。人们发现，以乐为本的内在动机，也就是**感受到自己的创造力是最强大的动机**，才是真正能驱动人们进行自我管理、不断前进的动力。Linux 操作系统的主创人员——大神李纳斯 · 特沃兹在自传《乐者为王》中也谈到类似的观点：很多顶尖的程序员写程序的目的，并不是为了赚钱或者得到公众的奉承，而是觉得写程序本身很有趣。[2]

1.1.2　保持兴趣

赫伯特 · 亚历山大 · 西蒙[①]认为，人是有限理性的动物，体现在学习中，就是情境理性。即“在哪里用，就在哪里学”，人的学习受到情景制约或者促进。你要学习的东西将实际应用在什么情境中，那么你就应该在什么样的情境中学习这些东西。比如，你要学习讨价还价的技巧，就应该在实际的销售场合学习，因为这一技巧最终是用在销售场合的。

了解到这些行为科学的研究成果后，我们应该将编程和自己或者他人的实际需求结合起来进行学习和实践，也就是说，我们应该利用计算机解决实际的问题，这样才能保持足够的驱动力。当然，人在不同阶段的需求会有变化。比如我刚毕业时，刚接触互联网，觉得很神奇，就利用业余时间建立了一个班级网站，方便同学之间联系。后来，在复习考研的过程中，发现很多背单词软件都满足不了自己的需求，又在复习的间隙，写了一个背单词的工具。

总之，编程是连接理论与实践的纽带，是计算机科学与计算机应用技术相交融的领域。正确的编程学习方法应该是：通过自顶而下的探索与项目实践，获得编程直觉与推动力；从自底向上的打基础过程中，获得最重要的通用方法并巩固编程思想的理解。

1.2　如何学习

1.2.1　快速迭代

迭代是一个重复反馈过程的活动，每一次迭代的结果都会作为下一次迭代的初始值，从而不断逼近目标或结果。迭代的实质思想是每次循环不求完美，但求不断发

[①]赫伯特 · 亚历山大 · 西蒙（英语：Herbert Alexander Simon，1916 年 6 月 15 日 – 2001 年 2 月 9 日），美国著名学者，计算机科学家和心理学家，研究领域涉及认知心理学、计算机科学、公共行政、经济学、管理学和科学哲学等多个方向。1975 年图灵奖得主，1978 年，获得诺贝尔经济学奖。

现新问题，迅速求解，获取和积累新知识。具体到编程学习中，就是先快速了解一门编程语言的基本语法，之后就开始用这门语言解决问题，做出一个能用的版本，然后再不断优化。

尽快用学习的编程工具作出实际项目，还有个好处，那就是能在学习的过程中，不断感受到自己的创造力，按照之前提到的理论，这种内在动机，也就是因一件事很有趣、很有挑战性、很令人着迷而去做的驱动力，对于从事创造性工作的人（艺术家、科学家、发明家和在校学生等等）至关重要。

1.2.2　内化输出

在学习的过程中，做好笔记。对不太熟悉的知识点，结合项目经验，简单总结后发布在博客或者 Github、Oschina 等等版本管理站点上。当知识点能形成知识网络时，利用思维导图工具将知识点之间的逻辑关系表述出来。这样做，一方面能在日积月累中熟悉知识，方便日后查询温习；另一方面通过内容的创作，能有效提高学习的成就感，进而增强学习兴趣，进入学习的正向循环。

做笔记的时候，最好不要粘贴、复制网络现成的资料，具体可参考理查德 · 费曼[①]提出的以下步骤：[3]

1. 选择你想要学习理解的概念；
2. 假设你正在给别人讲解这个概念，写出这个概念的解释。当你试着给别人讲解这个概念时，可促进对这个概念的深入理解，也能及时发现对这个概念的模糊之处；
3. 如果你卡在某个地方，回头查阅资料重新理解概念，只到你能达到上一步的要求；
4. 简化你的表述。使用你自己的语言，而不是资料上的原话，简洁清晰地表述概念。如果你的解释比较混乱，那就意味着你可能并没有真正理解这个概念。

1.2.3　善用网络

与其它领域的学习不同，学习编程有着高质量的网络资源，从官方文档到各种教程，从入门级别的练手代码到行业顶尖人员的作品，甚至开发工作中常见错误的解决办法，都可以从网络中获取。互联网本身就是一个友善的集体智慧创作群体的成果。我们应利用互联网，积极参与人类智慧的集体进化，这是最简单的让自己变得更智慧的方法。

1. 在线教学网站。网络上有许多高质量的教学网站，如慕课网、Code.org等等，这些网站积累了大量优质教学资源。
2. 官方网站。不管何种语言或者框架、库之类，其官方网站应该是学习者解决疑惑的首选；

[①]理查德 · 费曼（1918 年 5 月 11 日 – 1988 年 2 月 15 日），美国物理学家。1965 年诺贝尔物理奖得主。

3. 专业问答网站。遇到问题时，去SegmentFault.com、Stack Overflow等专业编程问答网站查找或者提问；
4. 代码托管网站。在开始项目前，最好到Github等代码托管网站查看有无相近的项目。
5. 利用 Google 等搜索引擎。

最后，请同学们喝了这碗鸡汤：大多数学校不会教的东西，以便精神饱满地继续学习。

本章节的写作，受到知乎用户萧井陌《编程入门指南》[4] 的启发，在此致谢。

参考文献

[1] 丹尼尔·平克. 驱动力 [M]. 龚怡屏, 译. 北京 : 中国人民大学出版社, 2012.

[2] 李纳斯·托沃兹, 大卫·戴蒙. 乐者为王 [M]. 王秋海, 译. 北京 : 中国青年出版社, 2001.

[3] 费曼. 别闹了，费曼先生 [M]. 吴程远, 译. 北京 : 生活·读书·新知三联书店, 2005.

[4] 萧井陌. 编程入门指南 v1.4 - 萧井陌的专栏 - 知乎专栏 [EB/OL]. 2015 [2015-12-25]. http://zhuanlan.zhihu.com/xiao-jing-mo/19959253.

能真正让自己变强大的，不是去征服别人，而是全力引导合作。

罗伯特·阿克塞尔罗德《合作的进化》

第 2 章　Web 标准

HTML 语言自 1989 年诞生以来，数以万计的网站使用 HTML 语言建立了起来，HTML 语言及浏览器的发明，大大提高了互联网的使用效率，将互联网从原来只在大学、军队等专业机构小范围的使用，普及到了更为广泛的一般用户，随着上网用户的增加，由互联网带来的商业机会也涌现了出来，伴随而来的，是众多不同厂商的浏览器试图建立对 HTML 语言的影响力。

尽管 HTML 对于所有计算机都是可用的，但是这并不意味着用户都能以相同的方式体验它，这些页面的实际显示效果取决于计算机的类型、显示器、网速以及查看页面的软件（浏览器），当今最流行的浏览器是 Internet Explorer、Firefox、Chrome、Safari 等等，同时，用手机上网的用户也逐渐增多并已成为主流。不幸的是，这些浏览器显示页面的方式并不完全相同。

造成这些不同的主要原因在于对商业利益的追逐和保护。1994 年，网景公司（Netscape Communication）在万维网上建立起了第一道栅栏，开始了所谓的“浏览器之争”。为了吸引用户，网景公司创建了一套只有 NetScape 能够处理的 HTML 扩展，使用 NetScape 浏览器冲浪的用户可以查看到改进的页面，如标题比其他文本大且粗、段落上下有间距、无序列表的每个项目之前有小黑点等等。许多人喜欢这些扩展，所以 NetScape 浏览器一度成为最为流行的浏览器。微软公司发现自己在这个市场上处于不利地位时，他们决定用自己的浏览器 Internet Explorer 来参与竞争，除了使用与操作系统捆绑销售的策略之外，同样为了吸引用户，他们增加了只有微软的 IE 浏览器能够识别的扩展。就这样，两个公司借助于网络设计者和开发者，为 HTML 增加了只适用于各自浏览器的扩展，更糟糕的是，其他一些浏览器开发商也加入了这场用户争夺战，这样导致结果就是，对于网络内容提供者而言，要试图创建一个适用于所有浏览器的网站变成一个很让人头疼的问题。

万维网创始人蒂姆·伯纳斯-李爵士希望万维网能像人的大脑那样在信息之间建立广泛的联系，并且应该对所有计算机开放，这个愿景被人们称为万维网的普适性原则。而这种浏览器的专有属性的不断推出，违背了蒂姆·伯纳斯-李所推崇的普适性的愿望。蒂姆·伯纳斯-李所领导的万维网“联合国”W3C（World Wide Web Consortium 万维网联盟）制订了一系列标准，将一些扩展吸收到了正式规范中，而将其他扩展完全取消了，推出了所谓的 Web 标准。[1]Web 标准的目标是使 Web 社区意识到普适性的重要意义，同时尽可能满足开发优美页面的愿望，努力拆除现有私有扩展，避免万维网出现分崩离析的局面。

早期版本的 HTML 将内容、结构和格式化指令组合在一个文档中，这虽然比较简单，但不够强大。W3C 设计了一个新系统——CSS，在这个系统中，格式化指令与内容和结构分开保存，因此可以根据需要将格式化指令或者样式应用到单一段落或者整个网站。并将 HTML 中那些用来表现内容形式的元素标记为“已废弃”，不鼓励人们去使用它们。

CSS 的理想是实现内容和表现方式的完全分离。将文档中的内容和表现相分离具有很多好处，人们可以轻松地增加、移除或更新内容，而不影响布局，还可以简单地改变整个站点的外观而不影响内容，从而使得网站的建设工作更有效率。

就目前而言，Web 标准表示的是由 W3C 组织定义并维护的、非私有化的一系列标准和规范。在近些年，Web 标准常常指的是将网页中的内容、表现、行为三者相分离的思想和相关的实现方法。按照 Web 标准，构建精良的 Web 文档应该有三层各自分离的资料层（见图2-1）。[2]

图 2-1　Web 标准

第一层是结构层，它包括文档的内容以及表示文本各部分的语义信息，如标题、段落、列表、导航条等等。这一部分的实现通过 HTML 或者 XHTML 来实现。

第二层是表现层，它决定了网页以何种方式显示，包括布局、排版、颜色、装饰、图片等等详细信息，在一些非视觉设备如屏幕阅读器上可能通过声音来呈现文字，这一部分使用用 CSS 来实现的。

除了内容层和表现层，还有一层是行为层。这一层主要是根据用户的动作，使用脚本来更新、增添或移除文档中的某些项目，已达到更好的交互效果。

内容层、表现层和行为层是如何协作的呢？我们可以看看网站中常常使用的幻灯片。如图2-2所示：

内容层使用 HTML 用来标记图片、新闻标题等内容，如上图2-2对应的 HTML 代码如下：

```
<div class="tabBody" bossZone="vision"> <div class="focus_wrap"> <div
    id="uedFocus5716281" class="fs_H">
<div id="uedFocus5716281_img" class="ued_focus_main">
</div>
<div id="uedFocus5716281_tit" class="ued_focus_text">
</div>
<div class="ued_focus_sub">
<div id="uedFocus5716281_dot ">
```

图 2-2 网页中的幻灯片

```
</div>
```

CSS 则将幻灯片排放在网页的具体位置，并为图片设置边框，为文字设置大小和对齐方式，调整文字和图片之间的间距等。图2-2所对应的部分 CSS 代码如下：

```
#card1C .tabBody {
  padding: 8px 10px 0;
  overflow: hidden;
  width: 100%;
}
```

JavaScript 脚本则让图片和文字每隔固定时间进行切换，当用户的鼠标移动到某个按钮上时，JavaScript 脚本还会显示对应的图片和文字。图2-2所对应的部分脚本如下：

```
<script type="text/javascript">
 Qfast(false,'alljs',function(){
    var uedFocus571628 = new uedFoucs();
     uedFocus571628.Content =[{img1:'http://mat1.gtimg.com/news/
        huozhe/12.03.16balaobf/3-228.jpg',img2:'http://mat1.gtimg.com
        /news/huozhe/12.03.16balaobf/3-68.jpg',title:'旱村标本',slink
        :'http://news.qq.com/photon/huozhe/balao.htm'},
        uedFocus571628.imgShowId = 'uedFocus571628_img'; }
</script>
```

我们最终看到的页面通常都是通过 HTML、CSS 和 JavaScript 配合来实现的，其中 HTML 和 CSS 是必不可少的手段，JavaScript 脚本在需要的时候才会使用。

参考文献

[1] W3C. Standards - W3C[EB/OL]. 2014 [2014-10-27]. http://www.w3.org/standards/.

[2] JEFFREY Z. 网站重构——应用 Web 标准进行设计 [M]. 傅捷, 王宗义, 祝军, 译. 北京 : 电子工业出版社, 2005.

这是给所有人的。

蒂姆·伯纳斯-李

第 3 章　HTML

3.1　HTML 发展简史

互联网最底层是计算机，没有计算机就没有互联网，1941 年德国人祖斯发布可编程的计算机，1976 年苹果发布面向个人用户的微型计算机，计算机保有量达到一定规模后，计算机之间如何沟通就成为问题，网络脱胎于麻省理工学院的一个教授乔治·威利，在 1956 年，当时为了防止美国遭受核爆炸，建立了雷达检测网，有了计算机网络的雏形，然后到 1973 年的以太网，发明者是麦特卡尔夫。计算机网络发展到一定阶段，不同的计算机网络之间的互联互通又成为重要课题，互联网应用而生，1969 年的阿帕网到 1975 年瑟夫和卡恩发明的 TCP/IP。互联网在发展过程中出现资料访问繁琐、学习成本高昂的问题，万维网和浏览器就是上述问题的解决方案。从布什、纳尔逊、恩格尔巴特来到 1990 年的伯纳斯-李，然后有了所谓的 HTTP 和之后的 Mosaic 浏览器。互联网万维站点的爆发，使得信息过剩，如何准确快速地获取信息又成为主要问题，搜索引擎开始崭露头角，搜索引擎最早是 1990 年的 Archie，到 1998 年谷歌的出现。最后才来到今天大家所见到的丰富多彩的内容，这个内容是建立在万维网基础上的。

互联网的诞生早于万维网 15 年，但起初因使用技术复杂难以普及，互联网的用户局限在大学、军事、科研机构等有限的群体。万维网的发明是公认的促使互联网迅速发展的重要因素。万维网借助与 HTML 文档，使用超文本链接，把不同电脑上的文本、图像、声音等文档链接在一起，使人们不必受电脑操作系统类别和地域等限制，就可以自由浏览和分享信息，互联网的操作因而大大简化。

HTML（HyperText Markup Language，超文本标记语言）是为“网页创建和其它可在网页浏览器中看到的信息”设计的一种标记语言。HTML 是万维网的基础，通过结合其他 Web 技术，可以创造出功能强大的网站，我们通过浏览器看到的信息就是用 HTML 呈现出来的。

HTML 最初是由万维网之父——蒂姆·伯纳斯-李给出了原始定义。1991 年，蒂姆·伯纳斯-李在互联网上公布了名为“HTML tag”的文档，详细说明了 HTML 组织信息的方式。之后，其它互联网组织和公司不断丰富 HTML 的功能，使之能够以直观的方式显示互联网中的文字、图片等信息，并通过超链接进行访问。随着互联网的爆炸式发展，商业利益的争夺也体现在 HTML 规则的制定中，以微软为代表的企业开始在浏览器中加入私有标签，这在相当程度上造成了 HTML 规则的混乱。

为了防止 HTML 受到商业利益的侵蚀，蒂姆·伯纳斯-李发起成立了万维网联盟（World Wide Web Consortium，W3C），接管了 HTML 的维护工作，并于 1997 年发布了 HTML3.2 版本和 4.0 版本。1999 年，万维网联盟推出了 HTML4.01 版本，加入了表格、表单和对象等特征。HTML4.01 版本是一个广泛采用的版本，至今仍然是 W3C 所推荐的 HTML 版本之一。

HTML4.01 发布之后，蒂姆·伯纳斯-李认为互联网的未来应该更加智能化，语义化，所以调整 HTML 的发展方向，停止了对 HTML 的升级，在 2000 年推出了 XHTML1.0，XHTML 是在 HTML4.01 基础上的优化和改进，XHTML 和 HTML4.01 之间只有极小但重要的区别。但是广大的互联网用户对于 HTML 的热情和需求不减反增，蒂姆·伯纳斯-李和万维网联盟最终推翻了之前的想法，认为 HTML 还是有很大的发展潜力，并于 2008 年推出了 HTML5 的草案，将工作重心重新调整到 HTML 的完善上。

2014 年 10 月 28 日，HTML5 正式发布，[1] 国内外一些互联网巨头已经将其应用到各自的网站中，比如 Google、Youtube、百度和淘宝等等。

3.2　HTML 基础概念及规则

3.2.1　HTML 基本结构

HTML 文件后缀名为“.htm”或“.html”，是一种包含了 HTML 标记的文本文件，我们通常所说的网页实质上就是 HTML 文件，HTML 的编辑非常简单，学习成本很低，而且所有网页的源代码都可通过浏览器直接查看。正如蒂姆·伯纳斯－李所说：“（互联网）起飞原因在于，全球的人可随意融入参与。”[2]

创建网页的方式有很多，我们可以新建一个空白的文本文件，编辑之后保存为后缀名为“.htm”的文件即可。在大多数情况下，我们使用专门的网页编辑软件来编辑网页，如：

```
<!DOCTYPE html>
<html>
<head>
  <meta charset=utf-8" />
  <title>文档标题</title>
</head>
<body>
</body>
</html>
```

在上面的 HTML 文件中我们可以看出，HTML 文档可以分为文档声明、网页头部分和网页主体部分三部分。

网页头部分指的是 <head> 和 </head> 部分，在这个区域一般放置网页相关的内容，比如网页编码方式、关键词、网页内容描述和网页相关联的文件等等。这些包含

在 <head> 和 </head> 之间的内容一般不会显示在用户的浏览器中。

网页主体部分指的是包含在 <body> 和 </body> 之间的部分，这部分内容是显示在浏览器中的内容，大部分 HTML 的编辑工作都是在主体部分中完成，如图片、文字、超链接等等。

头部分和主体部分又包含在 <html> 和 </html> 之中。浏览器会把 <html> 和 </html> 之中的内容视为网页。

3.2.2　元素及标签

元素是构成网页内容的基础单位，元素由起始标签、内容和结束标签构成，如：

```
<h1>一级标题</h1>
```

其中 <h1> 是 h1 元素的起始标签，标签以“<”开始，以“>”结束，</h1> 是结束标签，结束标签中含有“/”，位于起始标签和结束标签之间的“一级标题”是 h1 元素标记的内容。

元素可以嵌套，比如在段落 p 中插入图片 img。这时，img 元素作为整体构成段落元素 p 的内容。

```
<p>
  <img src="http://img1.gtimg.com/63530531.jpg" />
</p>
```

自闭和元素

并非所有元素都包含起始标签和结束标签。如上例中，img 元素就只有一个起始标签，没有结束标签，这一类元素被称为自闭和元素。除 img 元素之外，常见的自闭和元素还有：
、<embed>、<hr>、<input>、<link>、<meta>、<source>、<wbr>。

3.2.3　属性及属性值

属性（Attributes）及其值（Values）对于某些元素而言是必不可少的，比如新闻网页中经常使用图片来传递信息，图片在 HTML 中使用 img 元素来标记，但必须使用相应的属性及其值，告诉浏览器图片存放的位置、图片的大小等信息，其中图片存放的位置用 src 属性来表示，而图片的大小用 width 和 height 属性来表示，属性的值为字符串。如：

```
<img src="63530531.jpg" alt="一周图片精选">
```

上例中，img 元素拥有 src 属性，表示图片来源，src 属性的值为 63530531.jpg，表示图片存放的具体位置。

以下属性是全局属性，即几乎所有 HTML 元素都支持的属性：

accesskey 为元素指定快捷键
class 为元素指定类型
contenteditable 指定元素内容的可编辑性
dir 指定元素内容的文字方向
hidden 指定元素的可见性
id 为元素指定唯一名称
lang 指定元素内容的语言
spellcheck 为元素指定拼写检查
style 为元素指定样式
tabindex 为元素指定 tab 顺序
title 为元素指定提示文字
translate 为元素内容提供指定语言的翻译内容

3.3　HTML 常用元素

HTML5 中定义的元素共有 100 多个，[3]。常见的元素有段落元素 p、标题元素 h1、图片元素 img 等等 20 个左右。明白每个元素的用途才能恰当使用元素，比如，文章的标题就可以用 h1-h6 元素，文章中内容的段落就得放在段落 p 中，在文章中有想强调的文本，就可以使用 em 标签表示强调等等。恰当地使用元素标记内容，就是所谓的语义化，语义化编码描述了页面中内容的含义，而不是内容的表现形式，例如下面的代码：

```
<font size="6"><b>这是一个标题</b></font>
```

虽然上面的代码显示的效果和标题的效果没有区别，但是从语义上来说，上面的代码应该这样写：

```
<h1>This is a heading</h1>
```

语义化的好处是网页内容结构清晰，这样更容易被搜索引擎收录，也使其他程序，如屏幕阅读器，更容易理解文章内容，还有，语义化的 HTML 文档更容易管理，其内部结构也更为清晰。[4]

3.3.1　根元素

html 元素是 HTML 文档的根元素，其他所有元素都是 html 元素的后代。

最好通过 lang 属性为 html 指定特定的语言，以方便翻译工具、语音朗读工具确定合适的规则。

```
<!DOCTYPE html>
<html lang="en">
  <head>
  <title>Swapping Songs</title>
  </head>
  <body>
    <h1>Swapping Songs</h1>
    <p>Tonight I swapped some of the songs I wrote with some friends,
        who
    gave me some of the songs they wrote. I love sharing my music.</p
       >
  </body>
</html>
```

3.3.2　文档元数据相关元素

head

head 元素是 html 元素的第一个子元素，用来标记 HTML 文档的一系列元数据（如文档标题、作者、关键词、相关样式表、相关脚本等等）。

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
 </head>
 <body>
 ...
```

title

title 元素位于 head 元素之中，用来标记整个 HTML 文档的标题或名称，title 元素不得重复出现。

base

base 元素位于 head 元素之中，通过 href 属性设定文档基准 URL、通过 target 属性设定文档中所有超级链接的默认打开方式。如：

```
<!DOCTYPE html>
<html>
    <head>
```

```
        <title>This is an example for the &lt;base&gt; element</title
           >
        <base href="http://www.example.com/news/index.html">
    </head>
    <body>
        <p>Visit the <a href="archives.html">archives</a>.</p>
    </body>
</html>
```

上例3.3.2中 base 元素 href 的值必须为绝对地址。p 元素中的超级链接最终实际地址为：

```
 "http://www.example.com/news/archives.html"
```

以下代码将使得网页中超级链接的打开方式设定为新建窗口打开：

```
<!DOCTYPE html>
<html>
    <head>
        <title>This is an example for the &lt;base&gt; element</title
           >

        <base target="_blank" />
    </head>
    <body>
        <p>Visit the <a href="http://www.baidu.com">archives</a>.</p>
        <p>Visit the <a href="http://www.baidu.com" target="_self">
           archives</a>.</p>
    </body>
</html>
```

上例中的第一个超级链接将在新建窗口打开，第二个超级链接将在自身窗口打开。

link

link 元素位于 head 元素中，用来连接和当前文档相关的外部资源。link 元素必须指定 rel 属性。如：

```
<link rel="stylesheet" href="main.css" type="text/css">
```

meta

meta 元素用来标记不能被 title、base、link、style 和 script 元素标记的其他各种元数据，如网页关键词、版权信息、页面编码信息等等，最常见的是通过 meta 元素

设定页面的编码信息：

```
<meta charset="UTF-8">
```

meta 元素除了拥有全局性属性外，还可设定 name、http-equiv、content 值。

```
<!DOCTYPE html>
<html lang="zh">
<head>
  <!-- 指定页面编码方式 -->
  <meta charset="utf-8">
  <!-- 设定浏览器使用最新内核 -->
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <!-- 设定HTML文档的关键词 -->
  <meta name="keywords" content="HTML5技术">
  <!-- 设定HTML文档的描述信息 -->
  <meta name="description" content="HTML5技术学习及练习">
  <!-- 设定文档每隔3秒自动刷新频率 -->
  <meta http-equiv="Refresh" content="3">
  <!-- 设定文档在1秒后跳转到指定网址 -->
  <meta http-equiv="Refresh" content="1; URL=page4.html">
  <title>meta元素实例</title>
</head>
<body>

</body>
</html>
```

style

style 元素是 head 元素的子元素，用来设定 HTML 文档的内部样式表。

```
<!DOCTYPE html>
<html lang="en-US">
 <head>
  <title>My favorite book</title>
  <style>
   body { color: black; background: white; }
  </style>
 </head>
 <body>
  <p>My <em>favorite</em> book of all time has <em>got</em> to be
  <cite>A Cat's Life</cite>. It is a book by P. Rahmel that talks
  about the <i lang="la">Felis Catus</i> in modern human society.</p>
 </body>
```

```
</html>
```

3.3.3　块元素与行内元素

绝大多数 HTML 元素是块元素（Block-level element）或者行内元素（Inline-level element）。这两者之间有什么不同?

块元素开始于新的一行，占据指定的宽度。块元素还可以嵌套块元素，还可以在内部使用行内元素。最常见的块元素就是段落 <p>、 等等。

行内元素并不会开始于新的一行。它们按照字符的顺序依次再文档中显示，如从左到右，一个挨着一个，填满所在容器的宽度后才会换行。行内元素也可以嵌套使用，但是行内元素内部不能包含块元素。常见的行内元素有 <a>、<img> 等等。

3.3.4　区域元素

body

Body 元素用来标记全部文档内容。在 HTML 文档中，应该只能有一个 body 元素。

article

article 元素表示网页中完整的信息，比如论坛中的帖子、报纸中的报道、博客文章、用户评论或者其他独立完整的内容。

section

section 元素表示文档或者应用中一般意义上的分区，比如二级标题及段落的组合，就是一个分区。

```
<section>
  <h3>Red Delicious</h3>
  <p>These bright red apples are the most common found in many
  supermarkets.</p>
 </section>
 <section>
  <h3>Granny Smith</h3>
  <p>These juicy, green apples make a great filling for
  apple pies.</p>
 </section>
```

nav

nav 元素用来表示一组连接到外部网页的链接，即导航条。

```
<nav>
   <h1>Navigation</h1>
   <ul>
    <li><a href="articles.html">Index of all articles</a></li>
    <li><a href="today.html">Things sheeple need to wake up for today
       </a></li>
    <li><a href="successes.html">Sheeple we have managed to wake</a
       ></li>
   </ul>
</nav>
```

aside

Aside 元素用来表示和当前文档内容相关的区块，比如广告、相关链接以及其他和当前文档相关的内容。

h1,h2,...h6

这些元素用来标记对应区块的标题，其中 h1 为最高级别，h6 为最低级别的子标题。合理使用标题元素，也有利于搜索引擎对页面内容的理解。

header

header 元素不同于 head 元素，header 用来标记文档的简介或者导航区域。

footer

footer 元素通常包括版权信息、相关文档、机构简介等等内容。

address

address 元素用来表示联系方式。

```
<footer>
 <address>
  要了解更多信息，请联系<a href="mailto:js@example.com">作者</a>.
 </address>
 <p><small>&copy; copyright 2038 Example Corp.</small></p>
</footer>
```

HTML 转义字符

HTML 中，大于号 >、小于号 <、引号" 等一些字符，是有特殊意义的，比如 HTML 的标签就是以 < 开始，以 > 结束，如果直接使用，容易引起歧义。因此，HTML 中提供了输出特殊字符的机制，使用 & 字符开始，以 ; 结束，可以用来输出特殊字符，例如：> 将输出字符">"，© 将输出字符 ©。在 & 字符和 ; 之间，可以是名词或者数字编码。例如：" 表示引号。完整的 HTML 转义字符列表参见http://tool.oschina.net/commons?type=2。

3.3.5　内容组织元素

p

p 元素用来表示段落，列表元素 ol 和 ul 不能包含于 p 元素中。

hr

hr 元素表示段落级别的语义中断，例如，一个故事中的场景切换，或者文章中的主题切换。

pre

Pre 元素表示预定义格式的文本块，常用来标记诗歌、代码等等内容。

blockquote

Blockquote 元素表示从别处引用的内容。

ol

Ol 元素表示一组有序列表，所谓有序列表，就是列表项目的顺序是有意义的，如菜谱中的工序。列表项目使用 li 元素标记。

ul

Ul 元素表示一组无序列表，所谓无序，就是列表的顺序可以随意改变，列表项目使用 li 元素标记。

li

Li 元素表示列表中的项目。

dl

Dl 元素表示一组包含“名称-值”的自定义列表，其中名称使用 dt 标记，值可以是一个或者多个 dd 元素。

```
<dl>
 <dt> 作者 </dt>
 <dd> Adam </dd>
 <dd> Cui </dd>
 <dt> 编辑 </dt>
 <dd> Fang </dd>
</dl>
```

dt

Dt 元素用来表示自定义列表 dl 中的名称、术语。

dd

Dd 元素用来表示自定义列表 dl 中名称 dt 的解释、定义或者值。

figure

Figure 元素用来表示对插图、图表、照片、代码块等内容的引用。

```
<figure>
 <img src="bubbles-work.jpeg">
 <figcaption>图片标题</figcaption>
</figure>
```

figcaption

Figcaption 元素用来表示 figure 元素中的标题。

div

Div 元素本身并没有特殊含义，它是 HTML 中用来标记内容结构的最常用元素，用来将相关的元素组织在一起，形成逻辑上的整体。

main

Main 元素表示文档或者应用的最重要或者核心部分。main 元素可与 header、footer 一起标记网页的内容。

main 元素不要放在 article、aside、footer、header、或者 nav 元素中。

```
<!DOCTYPE html>
  <html>
  <head>
    <title>…… </title>
  </head>
  <body>

  <header>
  <nav>
  <ul>
  <li>…… </li>
  <li>…… </li>
  <li>…… </li>
  </ul>
  </nav>
  </header>

  <main>
  ……
  ……
  ……
  </main>

  <footer>…… </footer>

  </body>
  </html>
```

3.3.6　文本语义元素

a

A 元素用来表示超链接或者文档内部锚点。如果 a 元素有 href 属性，它就表示超链接。

当 a 元素表示超链接时，可以使用 target 属性指定链接打开方式，如：

```
<a href="http://www.baidu.com" target="_blank">百 度</a>
```

之前在3.3.3我们提到，行内元素不能包含块元素，但是 a 元素是个例外，a 元素内部可以是文字、图片等行内元素，也可以是段落、标题等块内容。

相对路径与绝对路径

链接通常可以分为两类：指向本站内部的链接和指向外部站点的链接。这些链接通过 a 元素的 href 属性来指定。我们常使用相对路径来指定网站内部的链接，相对路径中不包含域名（如.com，.org，.edu，等等）信息，因为链接指向的网页位于同一站点，因此 href 属性的值只需要包含网页文件所在的路径和文件名即可，如:`<a href="about.html">About</a>`；指向外部站点页面的链接必须使用绝对路径，绝对路径包含完整的 URL 信息。通常是以 http 开头，包含主机域名以及文件路径和文件名称。如`<a href="http://www.google.com/">Google</a>`

a 元素还可以通过在 href 属性中添加`#`创建指向文档内部具体位置的超级链接，如：

```
<body id="top">
  ...
  <a href="#top">Back to top</a>
  ...
</body>
```

上述代码将创建一个返回到 body 开始位置的超级链接，实现返回到页首的效果。

em

Em 元素表示对其标记内容的强调，在语义上强调与其他内容的不同。

strong

Strong 元素用来表示其标记内容非常重要、非常紧急，在语义上表示重要性。

small

Small 元素表示诸如注释、说明等等不同于正文的内容。

```
<dl>
<dt>标准间</dt>
<dd>199元。<small>不含早餐</small></dd>
<dd>229元。<small>含双早</small></dd>
</dl>
```

s

S 元素用来标记不再准确或者已不相关的内容。

```
<dl>
<dt>标准间</dt>
<dd><s>199元。<small>不含早餐</small></s></dd>
<dd>189元。<small>含早餐</small></dd>
</dl>
```

cite

Cite 用来表示对作品（如书籍、电影、歌曲、新闻等等）的引用，cite 中的内容必须包含作品名称或者作者名称或者 URL 地址。

```
<cite><a href="http://world.people.com.cn/n1/2016/0106/c1002
   -28019941.html">聚焦朝鲜的历次核试验</a>. 人民网. </cite>
```

q

Q 元素表示对外部资料的直接引用。

```
<p>那个男人说<q>事情不能再拖了</q>。我也赞同他的观点。</p>
```

可以使用 cite 属性来指定外部资料的来源，如：

```
<p>万维网联盟在其 <cite>About W3C</cite> 页面中表明，它的使命是 <q
   cite="http://www.w3.org/Consortium/">通过制定协议和规范，引导万维
   网发挥其全部潜力，确保万维网的长期发展。</q>. </p>
```

dfn

dfn 元素用来表示术语的定义。如：

```
<p>doit.im是一款<dfn><abbr title="Get Thing Done">GTD</abbr></dfn>软
   件，用来帮助用户进行任务管理。</p>
```

abbr

Abbr 元素表示某个术语的缩写。使用 title 属性来指定术语的全部名称。

data

Data 元素用来标记数据。

time

Time 元素用来标记时间。

code

Code 元素用来标记代码。

var

Var 元素用来标记变量。

samp

Samp 元素用来标记程序或者计算机的输出结果。

kbd

Kbd 元素用来标记用户输入的内容，尤其是键盘输入。

```
使用快捷键<kbd>Ctrl + S </kdb>保存文档。
```

sub 和 sup

Sub 表示下标，sup 表示上标。

i

I 元素表示不同于正文的可替换声音、情绪或其他语言的内容等等。

b

B 元素用来表示诸如关键字、产品名称等等需要引起注意的内容。

u

U 元素用来标记不能非常清楚表达的内容，如汉语诗歌等。

mark

Mark 元素用来标记高亮内容，以表示其与其他内容的区别。

ruby

Ruby 元素用来为东亚字符添加注音。

span

Span 元素用来在逻辑结构上对文本内容进行区分，比如在新闻信息中，我们可以将日期、作者信息、消息来源等内容，用 span 元素加以标记，结合 class 属性，进行文本区分。

br

Br 元素表示另起一行。

ins

Ins 元素表示文档的追加内容。datetime 属性用来说明追加内容的时间。

del

Del 元素表示文档中的移除内容。datetime 属性用来说明移除内容的时间。

```
<h1>To Do</h1>
<ul>
 <li>Empty the dishwasher</li>
 <li><ins datetime="2009-10-11T01:25-07:00">Watch Walter Lewin's
    lectures</ins></li>
 <li><del datetime="2009-10-10T23:38-07:00">Download more tracks</del
    ></li>
 <li>Buy a printer</li>
</ul>
```

3.3.7　嵌入内容元素

img

Img 元素用来在文档中插入图片。src 属性用来指定图片来源，alt 属性用来说明图片内容。

iframe

Iframe 元素生成内嵌框架，引用另外一个网页的内容。src 属性用来指定内嵌网页的地址。

```
<iframe src="http://ads.example.com/" width="468" height="60"></
   iframe>
```

embed

Embed 元素用来嵌入外部非 HTML 文档内容，比如 flash。

object

Object 元素用来嵌入外部对象，插入的对象视为图片元素。

video

Video 元素用来插入视频文件。通过 src, preload, autoplay, mediagroup, loop, muted, 以及 controls 属性控制视频内容及播放方式。

audio

Audio 元素用来插入音频文件。音频文件的播放控制通过属性 src, preload, autoplay, mediagroup, loop, muted, 以及 controls 进行设置。

source

Source 元素的 src 属性指定媒体文件的来源。

```
<video controls autoplay>
 <source src='video.mp4' type='video/mp4; codecs="avc1.42E01E, mp4a
    .40.2"'>
 <source src='video.ogv' type='video/ogg; codecs="theora, vorbis"'>
 ...
</video>
```

track

Track 元素为媒体文件提供基于时间线的文本信息，如不同语言的字幕。

```
<video src="brave.webm">
 <track kind=subtitles src=brave.en.vtt srclang=en label="English">
 <track kind=captions src=brave.en.hoh.vtt srclang=en label="English
    for the Hard of Hearing">
 <track kind=subtitles src=brave.fr.vtt srclang=fr lang=fr label="
    Français">
 <track kind=subtitles src=brave.de.vtt srclang=de lang=de label="
    Deutsch">
</video>
```

map

Map 元素将图片和区域组合起来形成地图。

```
<IMG SRC="/images/menu.gif" USEMAP="#NAV">
<MAP NAME="NAV">
  <P>
   <A HREF="/clothes/">Clothes</A>
   <AREA ALT="Clothes" COORDS="0,0,100,50" HREF="/clothes/"> |
   <A HREF="/toys/">Toys</A>
   <AREA ALT="Toys" COORDS="0,0,100,50" HREF="/toys/"> |
   <A HREF="/food/">Food</A>
   <AREA ALT="Food" COORDS="0,0,100,50" HREF="/food/"> |
   <A HREF="/books/">Books</A>
   <AREA ALT="Books" COORDS="0,0,100,50" HREF="/books/">
</MAP>
```

area

Area 元素为图片的指定区域添加超级链接。

math

Math 元素用来在 HTML 文档中插入数学公式。

svg

Svg 元素用来嵌入 svg 格式图片。

3.3.8　表格元素

table

Table 元素用来生成表格。表格拥有行、列。

caption

Caption 元素为表格元素添加标题或者说明信息，caption 应该包含在 table 元素中。

```
<caption>
<p>表1.</p>
<p>表格说明文字。</p>
</caption>
```

tbody

Tbody 元素用来标记表格主体。

thead

thead 元素用来标记表格的表头。

tfoot

Tfoot 元素用来标记表格的脚部，通常都是合计之类的信息。

tr

Tr 元素用来标记表格的行。

td

Td 元素用来标记表格的单元格。

th

Th 元素表示表头的单元格。

```
<table>
   <caption>表格说明文字</caption>
 <thead>
  <tr> <th> ID <th> Measurement <th> Average <th> Maximum
 <tbody>
  <tr> <td> <th scope=rowgroup> Cats <td> <td>
  <tr> <td> 93 <th scope=row> Legs <td> 3.5 <td> 4
  <tr> <td> 10 <th scope=row> Tails <td> 1 <td> 1
 </tbody>
 <tbody>
  <tr> <td> <th scope=rowgroup> English speakers <td> <td>
  <tr> <td> 32 <th scope=row> Legs <td> 2.67 <td> 4
  <tr> <td> 35 <th scope=row> Tails <td> 0.33 <td> 1
 </tbody>
</table>
```

3.3.9　表单元素

form

Form 元素用来标记一组和表单相关的元素，如文本框、提交按钮等等内容，是服务器和用户进行交互的最重要元素。

form 元素最重要的两个属性是 action 和 method，分别对应表单提交后的处理程序和表单提交方式。

```
<form action="http://www.bing.com/search" method="get">
 <label>Bing: <input type="search" name="q"></label> <input type="
    submit" value="Search...">
</form>
```

label

Label 元素用来标记表单交互元素的标签，是一个辅助说明性的元素，label 元素往往对应特定的元素。

```
<p><label>年 龄 : <input name=age type=number min=0></label></p>
```

input

Input 元素通过 type 属性，可生成各种交互元素，如文本框、密码框、按钮等等。

Input 元素的用法示例如下：

```
<input type="range" name="a" list="a-values">
<datalist id="a-values">
 <option value="10" label="Low">
 <option value="90" label="High">
</datalist>
```

button

Button 元素生成一个按钮，可通过 type 属性控制按钮类型。type 的值有 reset、submit 和 button，分别对应重置按钮、提交按钮和普通按钮。

select

Select 元素生成一个下拉菜单控制器，菜单列表由 option 构成。

表 3-1　input 元素的 type 属性值及其意义

type	类型	返回值
hidden	隐藏文本域	字符串
text	文本框	单行文本
search	搜索框	单行文本
tel	电话号码文本框	单行文本
url	URL 地址框	URL 地址
email	邮件地址框	邮箱地址或多个邮箱地址
password	密码框	单行文本
date	日期	无时区的日期
time	时间	无时区的时间
number	数字	数字
range	数字范围	数字
color	颜色选择器	RGB 颜色值
checkbox	复选框	列表值
radio	单选按钮	数字值
file	文件选择器	文件信息
submit	提交按钮	预设值
image	图片按钮	预设值
reset	重设按钮	n/a
button	按钮	n/a

```
<select name="unittype" required>
 <option value=""> Select unit type </option>
 <option value="1"> Miner </option>
 <option value="2"> Puffer </option>
 <option value="3"> Snipey </option>
 <option value="4"> Max </option>
 <option value="5"> Firebot </option>
</select>
```

datalist

Datalist 可为指定的表单元素，如文本框，生成一个数据列表，方便用户直接选择。

```
<label>
 Sex:
 <input name=sex list=sexes>
 <datalist id=sexes>
  <option value="Female">
  <option value="Male">
 </datalist>
```

```
</label>
```

optgroup

Optgroup 元素结合 select 元素使用，将生成列表分组。例如：

```
<form action="courseselector.dll" method="get">
 <p>Which course would you like to watch today?
 <p><label>Course:
  <select name="c">
   <optgroup label="8.01 Physics I: Classical Mechanics">
    <option value="8.01.1">Lecture 01: Powers of Ten
    <option value="8.01.2">Lecture 02: 1D Kinematics
    <option value="8.01.3">Lecture 03: Vectors
   <optgroup label="8.02 Electricity and Magnestism">
    <option value="8.02.1">Lecture 01: What holds our world together?
    <option value="8.02.2">Lecture 02: Electric Field
    <option value="8.02.3">Lecture 03: Electric Flux
   <optgroup label="8.03 Physics III: Vibrations and Waves">
    <option value="8.03.1">Lecture 01: Periodic Phenomenon
    <option value="8.03.2">Lecture 02: Beats
    <option value="8.03.3">Lecture 03: Forced Oscillations with
       Damping
  </select>
 </label>
 <p><input type=submit value="Play">
</form>
```

option

Option 元素为 select、optgroup、datalist 元素生成列表项目。

textarea

Textarea 元素表示能输入多段文字的文本框。

```
<p>如果您有任何意见，烦请告知我们：  <textarea cols=80 name=comments
   ></textarea></p>
```

keygen

Keygen 元素表示密钥生成器，当表单提交时，一个密钥将被提交到服务器。

```
<form action="" method="post" enctype="multipart/form-data">
        <p><keygen name="key"></p>
        <p><input type=submit value="Submit key..."></p>
</form>
```

output

Output 元素表示计算结果或者用户交互的结果。

```
<form onsubmit="return false" oninput="o.value = a.valueAsNumber + b.
   valueAsNumber">
        <input name=a type=number step=any> +
        <input name=b type=number step=any> =
        <output name=o for="a b"></output>
    </form>
```

progress

Progress 元素表示进度条。

```
<section>
        <h2>Task Progress</h2>
        <p>Progress:
            <progress id="p" max=100><span>0</span>%</progress>
        </p>
        <script>
        var progressBar = document.getElementById('p');

        function updateProgress(newValue) {
            progressBar.value = newValue;
            progressBar.getElementsByTagName('span')[0].textContent =
                newValue;
        }
        updateProgress(50);
        </script>
</section>
```

上述代码将生成一个进度为 50% 的进度条。

meter

Meter 元素表示在一定范围内的图形化比值。

```
<meter min=0 max=20 value=12>12cm</meter>
```

filedset

Filedset 元素表示表单中的一组表单元素集合。

```
<fieldset name="numfields">
  <legend> <label>
   <input type=radio checked name=clubtype onchange="form.numfields.
      disabled = !checked">
   My card has numbers on it
  </label> </legend>
  <div><label>Card number: <input name=clubnum required pattern
     ="[-0-9]+"></label></div>
 </fieldset>
```

legend

Legend 元素在 fieldset 元素使用，表示 fieldset 的标题。

3.3.10　脚本元素

脚本元素可为文档增加用户交互性。

script

元素 script 能为 HTML 文档包含动态脚本和数据块，包含在 script 元素中的内容不会直接显示给用户。

当使用 script 元素包含动态脚本时，脚本内容既可以直接嵌入在行内，也可以通过 scr 属性导入外部独立的脚本文件。

script 元素除全局性属性之外，还拥有以下几个属性：

src　外部脚本文件的地址

type　内嵌资源的类型。type 的默认值是"text/javascript"。如果脚本语言不是 JavaScript，则必须指定 type 的值。

charset　外部脚本文件的字符编码方式

async　使浏览器可以尽快地执行脚本，而不用在下载脚本时阻塞文档解析（异步）。在不支持 async 的浏览器中，通过动态创建 <script> 元素并把它插入文档中，来实现脚本的异步载入和执行。

defer　使得浏览器延迟脚本的执行，直到文档的载入和解析完成，并可以操作。

crossorigin 设置元素处理跨域请求的方式。

noscript

Noscript 元素当浏览器支持脚本时，其包含的内容不被显示；而当浏览器禁用脚本或者不支持脚本时，将会显示其内容。

template

Template 表示 HTML 模版片段，结合 JavaScript 能生成基于模版的动态内容。

canvas

Canvas 元素表示可实时生成内容的画布，结合 JavaScript，可生成动画、背景、游戏场景等等图片。

3.4　嵌入媒体

在网络中，除了文字信息之外，我们还见到图片、视频、音频以及其他形式的内容，图片使用之前介绍的 img 元素就可插入，而要插入视频、音频则有着不同的实现途径：

3.4.1　嵌入音频

在 HTML5 之前，在页面中嵌入视音频，绝大多数情况下都需要借助 Flash 播放器，而 HTML5 提供了一种快速、简单的添加音频文件的途径，即使用 audio 元素。和 img 元素类似，audio 元素通过 src 属性指定文件所在位置，但与 img 不同的是，audio 元素有闭合标签，例如：

```
<audio src="Backroad.ogg"></audio>
```

上例的代码将播放当前页面所在目录中的 Backroad.ogg 音频文件。

.ogg 格式

Ogg 全称是 OGG Vorbis，是一种音频压缩格式，类似于 MP3 等的音乐格式。但有一点不同的是，它是完全免费、开放和没有专利限制的。MP3 格式是受专利保护的，如果你想使用 MP3 格式发布自己的作品，则需要付给 Fraunhofer（发明 MP3 的公司）专利使用费。

设置音频播放属性

audio 元素除了具有 src 属性外，还有其他几个非常实用的属性，包括：

autoplay　自动播放当前文件
controls　显示音频播放器
loop　循环播放
preload　预载入音频文件，有三个值：none、auto、metadata，分别表示没有预载入的值、载入所有音频信息值和指定的音频信息值（如作者、时长等等）。

除 preload 之外，autoplay、controls 和 loop 都是布尔属性，他们可以不设置值，当它们出现在属性声明中时，它们的值默认为 true。

audio 元素在默认情况下，并不会显示在页面中。如果 autoplay 属性出现在 audio 元素中，则当页面载入的时候，就开始播放音频。如果要显示控制器，则需要添加 controls 属性，如：

```
<audio src="Backroad.ogg" controls></audio>
```

指定多种音频格式

不同浏览器对音频格式的支持是不一致的，绝大多数浏览器都支持常见的音频格式，如.mp3、.wav 以及.ogg。HTML5 中可以使用 source 元素，指定多种格式，浏览器会依次判断并读取其支持的格式，例如：

```
<audio controls>
  <source src="Backroad.ogg" type="audio/ogg">
  <source src="Backroad.mp3" type="audio/mpeg">
  <source src="Backroad.wav" type="audio/wav">
</audio>
```

苹果公司的 Safari 浏览器不支持 ogg 格式，但支持 MP3 格式，因此，我们为同一个音频提供 ogg 和 mp3 格式，基本上就能确保大多数情况下音频文件的播放。

3.4.2　嵌入视频

在 HTML5 中添加视频和添加音频的方式非常类似。使用 video 元素，配合 src 属性，就可以在页面中嵌入视频，其他属性如 autoplay、controls、loop 以及 preload 都和 audio 元素类似。与 audio 元素不同，vedio 元素还可以通过 poster 属性设定预览图片。例如：

```
<video src="back.ogv" controls poster="screenshot.jpg"></video>
```

浏览器视频支持情况

.ogv 影片格式一个开放标准的视频格式，Firefox、Chrome、Opera 支持此格式，但 Safari 和 IE9 不支持，支持常用的.mp4 视频格式的浏览器有 Safari（iPad、Windows、Mac OS）、Chrome、IE9，firefox 不支持.mp4 格式。

同音频一样，我们可以通过设定不同格式的文件，达到对不同浏览器的兼容：

```
<video controls>
  <source src="back.ogv" type="video/ogg">
  <source src="back.mp4" type="video/mp4">
</video>
```

3.4.3　嵌入 flash

尽管 flash 内容在移动平台日渐式微，但由于已经创作了大量的 flash 内容，因此，我们也需要掌握在页面中嵌入 flash 的方法。可通过 embed 元素嵌入 flash，如下：

```
<embed src="media/css-change-colors.swf" type="application/x-
   shockwave-flash" width="1024" height="798"> </embed>
```

embed 元素的 src 属性用来指定 flash 所在位置，type 属性用来声明嵌入的内容为 flash，witdh 为宽度，height 为高度。

嵌入视频平台的内容

在国内外，有许多视频网站提供免费视频存储及在线播放服务，如优酷、youtube 等等，这些网站几乎都提供将其内容嵌入到其它页面的办法，以优酷为例，嵌入其内容的办法如下：

```
<embed src="http://player.youku.com/player.php/Type/Folder/Fid
   /27236393/Ob/1/sid/XMTU3MTQ1MjM3Mg==/v.swf" quality="high" width
   ="480" height="400" align="middle" allowScriptAccess="always"
   allowFullScreen="true" mode="transparent" type="application/x-
   shockwave-flash"></embed>
```

究其本质，还是在页面中嵌入 flash。

3.5　在线学习资源及工具

1. 慕课网 HTML+CSS 基础课程

2. Learn to Code HTML & CSS
3. W3C HTML5 官方文档
4. 浏览器兼容性列表
5. HTML 语法验证工具

参考文献

[1] W3C. HTML5[EB/OL]. 2014 [2015-5-17]. http://www.w3.org/TR/html5/.

[2] 新华社. “万维网之父”回顾设计留遗憾 [EB/OL]. 2009 [2014-04-21]. http://www.yznews.com.cn/yzwb/html/2009-03/15/content_502176.htm.

[3] W3C. HTML5[EB/OL]. 2015 [2015-4-25]. http://www.w3.org/TR/2014/REC-html5-20141028/single-page.html.

[4] BOAG P. Semantic code: What? Why? How?[EB/OL]. 2005 [2016-04-21]. https://boagworld.com/dev/semantic-code-what-why-how/.

剪红情，裁绿意，花信上钗股。

节选自吴文英《祝英台近·除夕立春》

第 4 章　CSS

CSS(Cascading Style Sheets) 层级样式表，用以控制页面样式，实现页面中元素的表现形式、区块的布局、不同设备的适应等等，是 Web 标准中实现内容和表现相分离的唯一机制。虽然 W3C 目前正式推荐的版本是其在 2011 年发布的 CSS2.1，但由于还处于起草阶段的 CSS3 拥有更为便利和强大的表现手段，因此浏览器厂商为争夺用户，大都支持 CSS3 的特性。目前的实际工作中，CSS3 的使用已较为普遍，故而我们在学习 CSS 时，以 CSS2.1 版本为基础，兼顾 CSS3 的新特性。

通过 CSS，我们可以进行页面布局，设置元素样式，还可将这些样式或者布局应用到多个元素或者多个页面中。

4.1　使用样式表的三种方式

为网页添加 CSS 样式表的方式主要有三种，分别是：外部样式表、内部样式表和内嵌样式表。最好使用外部样式表，因为外部样式表更加符合 Web 标准所主张的内容和表现相分离思想。

4.1.1　外部样式表

外部样式表使用 link 元素将独立的样式表文件与网页连接起来，这样创作人员就可以用一个样式表文件为多个 html 文件指定样式，大大提高工作效率。link 元素必须放在 HTML 文档的 head 元素里面。就像这样：

```
<head>
<link href="global_v1.5.1.css" rel="stylesheet" type="text/css" media
   ="screen"/>
</head>
```

这行代码的意思就是以 global_v1.5.1.css 文件作为 HTML 文档的 CSS 来源，我们将这种 CSS 文档称为外部样式表。

link 元素有这样几个属性：href 用来指定外部样式表存放的地址，rel 描述的是 HTML 文件和它相连的文件之间的关系，如上例中就表示 global_v1.5.1.css 是网页的样式表。type 属性用来表述外部资源的类型，而 media 属性用来描述样式表适合的媒体类型（完整的媒体类型参见4.2.1）。可以通过指定 media 的值，来为打印机、屏幕

阅读器、投影仪、电视机、手机等不同的上网设备分别指定适合的样式表，这也体现了 CSS 的强大之处。

也可以在多种媒体中使用同一个样式表，只需要在 media 属性中以列出多种媒体名称，各个媒体名称之间用“, ”隔开。例如：

```
<link href="global_v1.5.1.css" rel="stylesheet"  type="text/css"
   media="screen, projection"/>
```

以上代码的意思是使用 global_v1.5.1.css 作为文档在计算机屏幕和投影仪上显示时的样式表。

4.1.2　内部样式表

除了使用 link 元素链接外部样式表之外，CSS 还可以使用 style 元素将样式信息同网页结合起来，这种方式的样式表叫做内部样式表。

style 元素可以直接在页面中嵌入 CSS 样式信息，而不用将它们单独作为外部样式表保存，因此并不符合内容和表现分离的 Web 标准。内部样式表在页面不多时比较方便，因为样式信息被存放在网页内部，当页面数量增加后，一旦修改，就要分别修改网页内容，这不但效率低下，而且还存在潜在的不一致风险。

但如果我们就想基于全站的样式针对某个特定页面进行微调，那么使用 style 元素标记内部样式表则是不错的选择。内部样式表也需要在 head 元素中使用，例如：

```
<head>
<style type="text/css" media="screen">
……
</style>
</head>
```

4.1.3　内嵌样式表

除了外部样式表和内部样式表之外，CSS 还提供了一种叫做内嵌样式表的机制。内嵌样式表是在具体的需要添加样式的元素中使用 style 属性。使用内嵌样式表的场合并不多见，因为同外部样式表和内部样式表相比，内嵌样式表更加不易修改。内嵌样式表的例子如下：

```
<div style="font-size:38px">加入微博，记录点滴</div>
```

4.2　CSS 语法与规则

4.2.1　语法声明

CSS 由一系列声明构成。语法声明分成两类：at 规则 (at-rules) 和 CSS 规则集 (rule sets)。[1] 声明之间可由空白字符连接。

at 规则

At 规则以“@”关键字开始，之后紧跟标志符。比如：“@import”。

“@import”规则的作用是从其他样式表文件中导入样式格式。@import 关键字之后必须跟随要引入到当前文件中的样式表 URI 地址，不过也可以仅用字符串表示。例如：

```
@import "mystyle.css";
```

上述语法声明等同于：

```
@import url("mystyle.css");
```

都表示要引用 mystle.css 样式表。

注意：“@import”不能放在语法块中，也不能放在“@charset”或者“@import”规则之后。例如：

```
@import "subs.css";
h1 { color: blue }
@import "list.css";
```

上述代码中最后的 @import 语句将被忽略。

除了“@import”之外，常用的 @ 规则还有“@media”，“@media”表示为特定媒体（多个媒体之间可用逗号分隔）声明样式，例如：

```
@media print {
    body { font-size: 10pt }
  }
```

表示当页面打印时字体为 10pt。

CSS 支持的媒体类型有：

all　适合所有设备。
braille　适合布莱叶盲文触摸设备。
embossed　适合布莱叶盲文打印设备。
handheld　适合小屏幕、带宽有限的手持设备。
print　适合打印设备。
projection　适合投影仪。

screen　适合计算机显示屏幕。
speech　适合语音阅读设备。
tv　适合早期低分辨率的电视。

规则集

规则集，也称规则，由选择符和跟随其后的声明块组成。声明块以“{”开始，以“}”结束，其中的声明以“;”分隔。声明由属性名称和属性值组成，属性名称和属性值之间用“:”连接。例如：

```
h1 { color: red; }
p { color: blue; }
em em { font-style: normal; }
```

4.2.2　CSS 注释

在 CSS 中，注释以“/*”开始，以“*/”结束，注释之内的内容会被浏览器忽略。注释不能嵌套使用。

4.3　CSS 选择符

CSS 是一门用来描述 HTML、XML 文档在屏幕、纸张或者朗读设备等等上如何表现的语言，CSS 使用选择符（selector）将样式（style）属性和文档中的特定元素（element）进行绑定。[2] 随着相关技术的演进，CSS 中的选择符也越来越丰富，功能也越来越强大。

4.3.1　简单选择符

简单选择符（Simple Selector）指的是选择符本身再无法分解，是构成组合选择符、群组选择符的选择符，这类选择符是 CSS 选择符的基础。

类型选择符

类型选择符（Type Selector），也叫做元素选择符，该选择符能代表文档树中的所有指定的元素。

```
h1 {color:red}
```

上述规则将使文档中所有的 h1 元素字体颜色为红色。

通配选择符

通配选择符（Universal Selector），将选定所有元素。通常不建议使用通配选择符，因为它会遍历并命中文档中所有的元素，出于性能考虑，需酌情使用。

```
* {color:red}
```

上述规则将使文档中所有元素的前景色为红色。

属性选择符

属性选择符（Attribute Selector）将选定那些拥有匹配属性的元素。具体如下表4-1：

表 4-1　属性选择符及其含义

属性选择符	含义
E[foo]	选择具有“foo”属性的元素
E[foo=”bar”]	选择具有“foo”属性并且属性值完全等于“bar”的元素
E[foo~=”bar”]	选择具有“foo”属性，且值中其中一个等于“bar”的 E 元素（包含只有一个值且该值等于“bar”的情况）。
E[foo^=”bar”]	选择具有“foo”属性，并且属性值以“bar”开头的元素
E[foo$=”bar”]	选择具有“foo”属性，并且属性值以“bar”结尾的元素
E[foo*=”bar”]	选择具有“foo”属性，并且属性值包含“bar”的元素
E[foo|=”en”]	选择具有“foo”属性，属性值并且以“en”开头并用连接符”-” 分隔的字符串的 E 元素

类选择符

类选择符（Class Selector）将选择那些 class 属性值为指定值的元素。

```
.bar       /*将选择所有具有class="bar"属性及值的元素*/
h1.bar     /*将选择具有class="bar"的所有h1元素*/
.bar.foo   /*将选择class值中同时包含bar、foo的元素*/
```

带有元素名称的类选择符和使用 class 属性的属性选择符是等价的。如 div.value 就等同于 div[class =value]。

ID 选择符

ID 选择符（ID Selector）将选择 ID 属性等于指定值的元素。按照 W3C 标准，ID 属性在 DOM 中的值应该是唯一的。

```
#bar       /*将选择所有具有id="bar"属性值的元素*/
h1#bar     /*将选择具有id="bar"的h1元素*/
```

4.3.2　简单选择符之伪类选择符

伪类（pseudo-class）的概念是指那些不在文档树中或者不能使用其他简单选择符选择的内容。伪类以“:”开头，后面跟随伪类名称。根据类别不同，伪类选择符又分为：链接伪类选择符、用户行为伪类选择符、目标伪类选择符、语言伪类选择符、UI 元素伪类选择符、结构伪类选择符、否定伪类选择符等。由于伪类选择符内容较多，虽然其从概念上将还是简单选择符，我们单独阐述。

链接伪类选择符

浏览器通常会将未访问的超级链接和访问过的超级链接加以区分，CSS 提供了:link 和:visited 链接伪类选择符用以区别它们。

:link　:link 伪类将应用于那些没有被访问过的内容。

:visited　:visited 伪类将应用于那些已经被访问过的内容。

用户行为伪类选择符

浏览器通常会在用户交互时用不同形式来表现交互状态。CSS 使用以下三个伪类用以和用户交互：

:hover　该伪类选择符表示用户将鼠标（或其他交互设备）移动到元素之上时的状态。

:active　该伪类表示用户激活（如点击）元素时的状态。

:focus　该伪类表示获得焦点（如键盘输入、鼠标点击）时的状态。

目标伪类选择符

有时 URI 只需资源内部的某个部分，这种类型的 URI 的尾部含有数字标记“#”, 在“#”之后是锚点识别符（也叫片段识别符）。例如下例中的 section2：

```
http://example.com/html/top.html#section2
```

使用:target 伪类选择符，就可以对资源内部中片段识别符（如上例中 section2）对应的元素设置样式。

语言伪类选择符

语言伪类选择符能对特定 lang 属性的元素设置样式。如：

```
html:lang(zh) /*选中lang属性值为zh的html元素*/
```

UI 元素伪类选择符

UI 元素伪类选择符可以针对 UI 元素的状态进行选择。

:enabled　选择 UI 元素中处于 enabled 状态的元素。
:disabled　选择 UI 元素中处于 disabled 状态的元素。
:checked　使用:checked 伪类选择符，可以选择被用户切换到选中状态 radio、checkbox 等元素。

结构伪类选择符

结构伪类选择符基于文档树中元素的父子兄弟关系进行选择。

:root　表示文档的根元素，在 HTML4 中，该选择符将始终选择 html 元素。
:nth-child()　代表同一父级元素的第 N 个元素。如：

```
tr:nth-child(2n+1) /* 表示表格中的奇数行 */
tr:nth-child(odd)  /* 同上 */
tr:nth-child(2n+0) /* 表示表格中的偶数行 */
tr:nth-child(even) /* 同上 */
tr:nth-child(5)    /* 表示表格中的第5行 */
```

:nth-last-child()　表示在同一父元素中，从后往前计算的第 N 个元素。
:nth-of-type()　表示同类型兄弟元素中的第 N 个元素。
:nth-last-of-type()　表示同类型兄弟元素中，从后往前计算的第 N 个元素。
:first-child　表示同一个父元素中的第一个子元素。
:last-child　表示同一个父元素中的最后一个子元素。
:first-of-type　表示同类型兄弟元素中的第一个元素。
:last-of-type　表示同类型兄弟元素中的最后一个元素。
:only-child　匹配父元素仅有的一个子元素。
:only-of-type　匹配同类型中的唯一的一个同级兄弟元素。
:empty　匹配没有任何子元素（包括 text 节点）的元素。

否定伪类选择符

否定伪类选择符:not(X)，将匹配不含有 X 选择符的元素。例如：

```
.demo li:not(:last-child) { border-bottom: 1px solid #ddd; }
```

上述规则将匹配给定列表项（除最后一项之外），使其有一条红色的下划线。

4.3.3　伪元素选择符

伪元素能创造出 HTML 语言无法创造的抽象元素，如段落首行、首字母、元素前、元素后等等。伪元素选择符以“::”开始，[①]后面跟元素元素名称。

::first-line　设置元素内的第一行的样式。此伪元素仅作用于块元素。

::first-letter　设置对象内的第一个字符的样式。此伪对象仅作用于块对象。

::before　设置在对象前（依据对象树的逻辑结构）发生的内容。和 content 属性一起使用，并且必须定义 content 属性。

::after　设置在对象后（依据对象树的逻辑结构）发生的内容。和 content 属性一起使用，并且必须定义 content 属性。

::placeholder　设置对象文字占位符的样式。

::selection　设置对象被选择时的颜色。

4.3.4　组合选择符

组合选择符能根据文档树中选择符所代表的父子兄弟关系进行更加准确的选择，其在实际工作中使用率很高。

后代选择符

两个简单选择符之间用空格组合在一起，就表示后代选择符，如：

```
h1 em
```

表示选择位于 h1 元素中的 em 元素。

子元素选择符

子元素选择符使用“>”将两个简单选择符组合在一起，选择具有父子关系的子元素。不同于后代选择符，子元素选择符只能命中子元素，而不能命中孙辈。例如：

```
.demo > div
```

表示选择 class 为 demo 的元素中的 div 子元素。

相邻选择符

两个元素用“+”连接在一起，如 E+F，表示选择出现在 E 元素之后的 F 元素。

[①]CSS3 将伪对象选择符前面的单个冒号 (:) 修改为双冒号 (::) 用以区别伪类选择符，但以前的写法仍然有效。

兄弟选择符

两个元素用“~”连接在一起，如 E~F，兄弟选择符会命中所有符合条件的兄弟元素，而不强制是紧邻的元素。

4.3.5　群组选择符

在 CSS 中，使用“,”将选择符连接在一起，表示这些选择符共享相同的样式声明。例如：

```
h1,h2,h3 {color : red;}
```

上述例子表示，h1、h2、h3 元素的颜色都为红色。群组选择符，能降低 CSS 样式书写时的重复规则，提高开发效率。

4.3.6　选择符优先级别

层级样式表之所以叫作“层级”，就是规则可以重叠、可以嵌套。这样就产生了新的问题：当规则重叠时，CSS 是如何决定采用哪个规则呢？总的原则是，最后出现的规则有效，越具体的选择符优先。如：

```
p {
  background: orange;
  font-size: 24px;
}
p {
  background: green;
}
```

浏览器在渲染页面时，会从上至下执行 CSS 规则，这样，上述代码中一开始声明的段落橙色背景，会被随后声明的段落绿色背景取代。

除了先后顺序外，CSS 还按照选择符的具体程度来决定优先级别，具体计算规则如下：

1. 计算选择符中的 ID 选择符数量 (= a)。
2. 计算选择符中的类选择符、属性选择符和伪类选择符数量 (= b)。
3. 计算元素选择符和伪元素选择符的数量 (= c)
4. 忽略通配选择符。
5. 将 a-b-c 的值连接在一起就得到选择符的优先级别，该值越大，优先级别越高。

例如：

```
*               /* a=0 b=0 c=0 -> specificity =   0 */
li              /* a=0 b=0 c=1 -> specificity =   1 */
ul li           /* a=0 b=0 c=2 -> specificity =   2 */
```

```
ul ol+li        /* a=0 b=0 c=3 -> specificity =   3 */
h1 + *[rel=up]  /* a=0 b=1 c=1 -> specificity =  11 */
ul ol li.red    /* a=0 b=1 c=3 -> specificity =  13 */
li.red.level    /* a=0 b=2 c=1 -> specificity =  21 */
#x34y           /* a=1 b=0 c=0 -> specificity = 100 */
#s12:not(foo)   /* a=1 b=0 c=1 -> specificity = 101 */
```

4.4　CSS 的取值

4.4.1　长度

长度是对距离的测量。长度由数字和紧随其后的单位组成。如果长度为 0，则单位可以忽略。有些属性还允许负值。

长度可分为相对长度和绝对长度。相对长度指的是相对与其他长度属性而言，相对长度又分为文本相对长度和视图窗口相对长度。

文本相对长度

em	相对于当前对象内文本的字体尺寸
ex	相对于字符“x”的高度。通常为字体高度的一半
ch	数字“0”的宽度
rem	相对于根元素 (即 html 元素)font-size 计算值的倍数

em 是常用的相对长度单位，em 的长度会基于元素的字体大小进行计算，最常用的用法是首行缩进两个字符：

```
text-indent:2em;
```

上述的例子中，如果元素中的字体大小为 16px，则文本首行缩进的量为 32px。

视图窗口相对长度

vw	相对于视口的宽度。视图窗口被均分为 100 单位的 vw
vh	相对于视口的高度。视图窗口被均分为 100 单位的 vh
vmax	相对于视图窗口的宽度或高度，总是相对于大的那个。视图窗口的宽度或高度被均分为 100 单位的 vmax
vmin	相对于视图窗口的宽度或高度，总是相对于小的那个。视图窗口的宽度或高度被均分为 100 单位的 vmin

绝对长度

绝对长度是最简单的长度单位，绝对长度单位有：

cm	厘米
mm	毫米
q	0.25 毫米
in	英寸，1 英寸 = 2.54 厘米
pt	点，1pt = 1/72in
pc	派卡，1pc = 12pt
px	像素，1px = 1/96in

其中经常使用的绝对单位是像素 px。1 英寸相当于 96 个像素，像素虽然是绝对单位，但是在不同的设备，尤其是高密度显示设备和低密度显示设备上的效果并非完全一致。如图4-1所示：

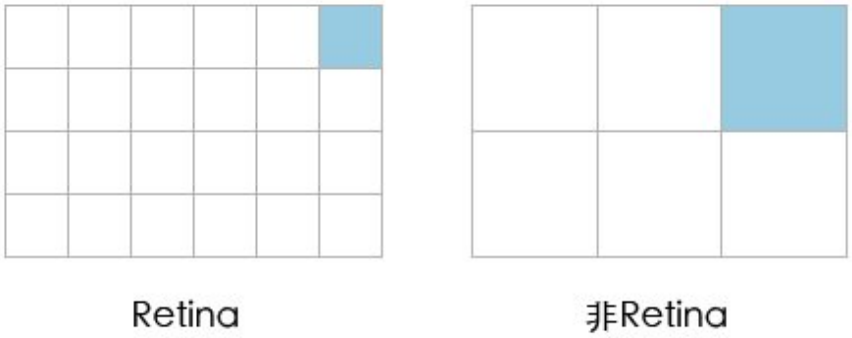

图 4-1　不同显示密度中的像素

px 与 pt 是两个看起来很像、却完全不一样的单位 (在某些场合他们是 1：1 的)，在很多时候却常常被搞混，或是制作过程根本没有分清楚、导致结果没有很精确。pt 是常见的标示文字尺寸的单位，在绘图以及文书软件等几乎都是使用 pt 作为字体尺寸的单位。px 像素，是屏幕上所显示的最小单位，当制作的内容是供屏幕浏览时，使用 px 可以精确的控制画面上显示效果。但也因为每个屏幕分辨率不同，像素的大小也不固定。在分辨率高的屏幕上，一个像素可能会小到肉眼无法辨识的大小。

pt 与 px 理解起来其实不难，在应用的时候其实也相当单纯，在大部分的情况下适用的一个原则是：当设计的目的是用于供屏幕浏览，则趋向于使用 px 以方便掌握细节；而如果是为了做输出打印的需求, 使用 pt 则是较好的选择。因此，在网站前端开发中，人们经常使用的单位是 px，而不是 pt。[3]

4.4.2　百分比

百分比由数字和紧随其后的“%”符号构成。百分比值始终是相对于另外一个值而言的，比如相对于长度。例如：

```
p { font-size: 10px }
p { line-height: 120% }
```

上面例子中的 line-height 值，最终计算出来应该为 12px。

4.4.3　颜色

在 CSS 中，可通过 color 属性为元素指定字体颜色。在 CSS2 中，颜色由三种表示方法：颜色名称、十六进制颜色值和 rgb 函数形式的颜色值。其中颜色名称只能在预设的 16 种颜色名称中选择，这 16 种预设的颜色名称分别是：black、silver、gray、white、maroon、red、purple、fuchsia、green、lime、oliver、yellow、navy、blue、teal、aqua。如：

```
p {
  color: black;
}
```

上述规则将使段落的颜色为黑色。

color 属性的值还可以采用 16 进制来表示，如：

```
p {
  color: #333333;
}
```

表示其后的数值为 16 进制，通常情况下是 6 位数字，分别表示 RRGGBB，即红色分量的颜色值、绿色分量的颜色值、蓝色分量的颜色值。如果 #RRGGBB 分别都由重复数字组成，那么还可以简写成为 #RGB。

除了使用颜色名称、十六进制之外，颜色值还可以通过 rgb 函数表示，如：

```
p {
  color: rgb(25%,25%,30%);
}
```

其中 rbg 函数的三个参数值分别表示的是红色、绿色和蓝色的颜色分量。

在 CSS3 中，还新增了 RGBA, HSL, HSLA 三种颜色表示方法。[4]

RGBA 模式与 RGB 相同，只是在 RGB 模式上新增了 Alpha 透明度。如：

```
p {
  color: rgba(25%,25%,30%,0.2);
}
```

最后一个参数值表示颜色的透明程度，0 表示完全透明，1 表示不透明。

HSL 是另外一种常用的颜色表示体系，其中：

H　Hue(色调)。0(或 360) 表示红色，120 表示绿色，240 表示蓝色，也可取其他数值来指定颜色。取值为：0 - 360

S　Saturation(饱和度)。取值为：0.0% - 100.0%

L　Lightness(亮度)。取值为：0.0% - 100.0%

例如：

```
.task {
  background: hsl(0, 100%, 25%);
}
```

HSLA 颜色体系，增加了透明度，例如：

```
.task {
  background: hsla(0, 100%, 25%, .25);
}
```

4.5　文本样式属性

使用 CSS 可以修改文本的字体、大小、粗细、倾斜、行间距、缩进方式、颜色等等，页面大部分内容都是以文字的形式出现，因此，文本的相关样式表的属性和值需要熟练掌握。

4.5.1　设定字体

通常情况下，中文页面的字体为宋体字，但新闻标题的字体会指定为更加醒目的黑体字或其他字体。在 CSS 中通过 font-family 属性来实现字体的指定，其中字体名称应该用单引号或者双引号包围起来。如：

```
h1 {
  font-family: "黑体";
}
```

虽然在语法上我们可以指定任何字体，但是在用户浏览时，只会看到他们的系统中已经安装的字体，没有安装的字体会被系统默认的字体代替，在中文系统中，默认的字体为宋体字，常见的字体还有黑体、楷体、仿宋。有时为了兼顾各种设备，我们需要为文字指定不止一种字体，这些字体在不同的设备中名称可能不同。如下：

```
body {
  font-family: "宋体","sans-serif";
}
```

浏览器会首先使用宋体来显示文本，当系统中没有安装宋体字时，才会使用替代字体 sans-serif。替代字体可以有多个，分别用逗号隔开。

4.5.2　创建斜体

斜体常常用来表示引述、人名、外语单词等内容的表示。斜体效果使用 font-style 属性来控制，font-style 有 3 个值：italic、oblique、normal，分别表示使用字体的倾斜版本来显示字体、使用计算机动态倾斜文字以及按正常字体显示。如：

```
em {
  font-style: italic;
}
```

4.5.3　应用粗体

在页面中，加粗的显示效果比较常见，是常规的、有效的强调内容的手段。

使用样式表的 font-weight 属性可以灵活地控制字体的粗细，如：

```
em {
  font-weight: bold;
}
```

font-weight 的只有两种表示方法，一种是从 100-900 之间的 100 的倍数；一种是 lighter、normal、bold 以及 bolder。

4.5.4　设置字体大小

在 CSS 中，使用 font-size 属性为字体指定大小，例如：

```
p {
  font-size: 10px;
}
body {
  font-size: 62.5%;
}
```

通常情况下，浏览器的默认字体大小为 16px，而 16px 的 62.5% 恰好也为 10px。

4.5.5　设置行间距

为了增强文字的易读性，我们通常要调整段落中的文字行间距，使用比较大的行间距能够提高文字的可识别性。对于一些不重要的信息，我们可以缩小行间距，使之显得更加紧凑。在 CSS 中，使用 line-height 属性来控制行间距的大小，它的值除了可以采用相对单位、绝对单位和百分比之外，还可以是没有单位的数字。例如：p font-size：16px; line-height: 1.75; 上例中段落中文字的行间距最终等于 28（16*1.75）像素。

4.5.6　同时设置所有字体值

除了能够使用 font-family、font-size、line-height、font-weight 分别设置字体字型、字体大小、行间距和粗细外，CSS 还提供了 font 属性用以快速设置字体属性。font 属性的设定非常灵活，可以是上述几种属性的自由组合，但字体大小的属性值出现在设置字体系列的前面，如果有行间距的话，行间距必须直接出现在字体大小后面，用斜杠连接。如：

```
p {
  font: bold 16px/1.75 “宋体”;
}
```

上述规则设定段落元素的字体加粗、字体大小为 16 像素，行间距为 28（16*1.75）像素，采用宋体字型。再如：

```
p {
  font: 16px “宋体” ;
}
```

上述规格规定段落元素采用宋体字型、字体大小为 16px，其他属性为 normal。

4.5.7　设置字体颜色

在 CSS 中，可通过 color 属性为元素指定字体颜色。如：

```
p {
  color: black;
}
```

4.5.8　控制字符间距

通常情况下，中文字符的字间距不用做额外的调整，但有时由于版面空间有限，需要将较长的标题安排在某个固定的空间内，在不删除字数的情况下，我们可以利用 CSS 提供的 letter-spacing 属性来增加或缩小中文字符之间的间距。letter-spacing 的值是带单位的数字。如：

```
h2 {
letter-spacing: -2px;
}
```

上述规则将使二级标题中字符之间的距离在原有基础上缩小 2 个像素。和 letter-spacing 相关联的另外一个属性是 word-spacing，word-spacing 用来调整单词之间的距离，由于汉字中的单个汉字被认定为字母，因此 word-spacing 在中文网页中的应用非常少。

4.5.9　首行缩进

CSS 使用 text-indent 属性来控制文本的首行缩进。text-indent 的值必须是有单位的数值，如 2em 或 18px 等。如果数值为正，则是首行缩进，如果数值为负，则是悬挂缩进。大多数情况下，新闻的正文内容中的段落都要求首行缩进两个字符，则可以这样设置规则：

```
p {
    text-indent: 2em;
}
```

4.5.10　文本对齐

CSS 中使用 text-align 来控制文本的对齐方式。text-align 的取值可以是 left、right、center、justify，分别对应左对齐、右对齐、居中以及两端对齐。如：p text-align: left; 表示左对齐段落中的文字。

4.5.11　文本装饰

CSS 还提供了对文本进行简单装饰的属性 text-decoration。该属性的只有 5 个：underline、overline、line-through、blink、none，分别表示下划线、上划线、删除线、闪烁 * 以及取消装饰效果。例如：

```
a:hover {
    text-decoration: underline;
}
```

上述样式的效果是当鼠标滑过或停留在超级链接上时，超级链接中的文字出现下划线。

除了以上装饰效果外，CSS3 还可以通过 text-shadow 属性为文字添加阴影。在以前，阴影效果一般都是通过图片实现，现在直接使用 text-shadow 属性来实现阴影。这个属性可以有两个作用，产生阴影和模糊主体。这样在不需要图片的情况下就能给文字增加质感。语法如下：

```
text-shadow:[Color X-Offset Y-Offset Blur],[Color X-Offset Y-Offset
   Blur]...
或者
text-shadow:[X-Offset Y-Offset Blur Color],[X-Offset Y-Offset Blur
   Color]...
```

X-Offset 表示阴影的水平偏移距离，其值为正值时阴影向右偏移，如果其值为负值时，阴影向左偏移；Y-Offset 是指阴影的垂直偏移距离，如果其值是正值时，阴影向下偏移反之其值是负值时阴影向顶部偏移；Blur 是指阴影的模糊程度，其值不能是

负值，如果值越大，阴影越模糊，反之阴影越清晰，如果不需要阴影模糊可以将 Blur 值设置为 0；Color 是指阴影的颜色，其可以使用 rgba 色。例如：

```
.demo1 {
   text-shadow: red 0 1px 0;
}
```

text-shadow 可以给一个对象应用一组或多组阴影效果，方式如前面的语法显示一样，用逗号隔开。例如：

```
.demo3 {
  text-shadow: 0 0 5px #fff, 0 0 10px #fff, 0 0 15px #fff, 0 0 40px #
     ff00de, 0 0 70px #ff00de;
}
```

4.5.12　其他文本样式

文本除了上述属性外，还有些中文环境下不大常用的属性，如 white-sapce、text-transform、word-spacing、font-variant。感兴趣的读者可自行查找参考资料学习。

4.6　背景

CSS 中提供了为元素设定背景的功能，我们看到许多设计精良的网页，在很大程度上是灵活运用 CSS 背景实现装饰效果的。在 CSS 中，不但可以为整个网页设定背景，也可为具体元素设定背景，任何可显示出来的元素都可以设定背景样式。

4.6.1　设置背景色

CSS 通过 background-color 属性为元素设定背景颜色，颜色值的设定同 color 属性。background-color 属性的默认值为 transparent（透明）。如下例中的规则将使段落中的内容颜色为黑色，段落的整体背景色为浅灰色。

```
p {
  background-color: #eee;
  color: black;
}
```

4.6.2　设置背景图片

一般情况下，我们会认为色块过于单调，缺少变化，因此，我们可以使用 CSS 提供的图像背景来装饰页面。在 CSS 中和图像背景有关的属性有：

background-image 用来指定作为背景使用的图片所在的地址，它的值为 URL 地址或者渐变色，默认值为 none。此外，当同时定义了背景颜色和背景图像时，背景图像覆盖在背景颜色之上。

background-repeat 来设定背景图片的重复方式，background-repeat 的值有：repeat（沿水平和垂直方向平铺）、repeat-x（沿水平方向平铺）、repeat-y（沿垂直方向平铺）或者 no-repeat（不平铺显示），默认值为 repeat。

background-position 用来设定背景图像的位置，位置信息有两个，第一个用来指定横向坐标，第二个用来指定纵向坐标。坐标的表示可以使用百分数、带单位的长度以及关键字（left、center、right、top、middle、bottom），默认值为 0。

background-attachment 用来设定背景图像的滚动模式。值有三个，其中 scroll 表示背景图像会随着页面窗口滚动；fix 表示固定在窗口，不随其滚动；local 表示元素随元素滚动时背景图像也会跟着滚动，因为背景图像总是要跟着内容。local 值是 CSS3 新增的值。background-attachment 默认值为 scroll。

例如：

```
div#footer {
  background-image: url(bg.png);
  background-repeat: no-repeat;
  background-attachment: scroll;
  background-position: center bottom;
}
```

上述规则为 id 等于 footer 的 div 元素指定背景图像，背景图像的来源为当前网页同级目录中的 bg.png 文件，该图片在元素背景范围内部重复，并且随网页滚动，图片的出现在 div 元素背景范围内居中靠下的位置。

其中，background-position 属性，可以使用 top、right、bottom、left 和 center 关键字以及像素、百分比或者其他长度单位。CSS 在定位时，坐标原点位于左上角，如下图所示：

图 4-2　CSS 坐标体系

通常情况下，我们将 background-color、background-image、background-position、

和 background-repeat 属性简写到 background 属性中。简写时，属性的顺序有多种，普遍采用 background-color、background-image、background-position、和 background-repeat 的顺序。例如：

```
div {
  background: #b2b2b2 url("alert.png") 20px 10px no-repeat;
}
```

4.6.3　设置线性渐变背景

许多年以来，设计师和开发人员若要实现背景渐变效果，只能使用图像处理软件制作渐变背景图片。但在 CSS3 中，我们现在可以使用渐变背景属性[5] 来代替以往繁琐的做法，只需一条语句，就可以实现漂亮的渐变背景效果。

CSS 的线性渐变背景是在 background 或者 background-image 属性中使用 linear-gradient() 函数实现，例如：

```
div {
  background: #466368;
  background: linear-gradient(#648880, #293f50);
}
```

linear-gradient() 函数必须包括至少两个值，第一个值为渐变起始值，第二个值为渐变结束值，浏览器会计算并平滑渲染出介于这两个颜色值之间的内容。

在渐变色背景之前，我们还声明了背景色，这样做的目的是当浏览器不支持渐变色属性时，使用指定的背景色作为背景。

默认情况下，线性渐变的方向为从上至下，我们可以在颜色值之前，使用关键词或者角度来改变渐变的方向。例如：

```
div {
  background: #466368;
  background: linear-gradient(to right bottom, #648880, #293f50);
}
```

上例中，我们生成了一个从左上角渐变到右下角的渐变填充背景。

除了使用关键字之外，还可以使用角度，例如，如果我们想要生成和上例类似的渐变填充，可以使用 135deg 调整渐变角度：

```
.deg {
    background: #466368;
    background: linear-gradient(135deg, black, white);
    border-radius: 6px;
    height: 120px;
}
```

4.6.4　设置径向渐变

线性渐变非常适合从不同方向之间的渐变，不过有时候，我们还需要径向渐变。径向渐变使用 radial-gradient()，其他类似于线性渐变。例如：

```
div {
  background: #466368;
  background: radial-gradient(#648880, #293f50);
}
```

4.6.5　使用多个色标

要实现渐变效果，最少需要两个颜色值。但我们还可以使用多个色标，生成不同颜色值之间的渐变效果，不同颜色值会在指定的方向上平均分布，如：

```
div {
  background: #648880;
  background: linear-gradient(to right, #f6f1d3, #648880, #293f50);
}
```

CSS3 渐变效果在线生成器

对于新手而言，手工生成复杂的渐变效果还是比较困难的，幸运的是，网络上有很多可以在线生成渐变效果的工具，如CSS3 gradient generators，借助于这些工具，新手可以学习复杂渐变的生成代码。

4.7　盒模型

在理解CSS是如何控制页面显示效果的时候，我们必须要掌握盒模型（Box Model）和定位（position）机制。CSS 借助于盒模型和定位机制，结合文档树，能够精确、高效地控制内容在页面中的位置，从而实现页面的布局。

4.7.1　盒模型的概念

所有 HTML 元素，在页面的呈现过程中，都遵循 CSS 制定的盒模型（Box Model），盒模型是一个包含外边距、边框线、内边距以及内容的矩形容器。具体如下图所示：[6]

从图中我们可以看到，元素的盒模型由外边距 (margin)、边框线 (border)、内边距（padding）以及元素的内容（content）构成，CSS 对外边距、边框线、内边距的控制可以分方向进行，也可以整体控制，如上图中的 TM 就表示上侧外边距、LM 表示左侧外边距、RM 表示右侧外边距、BM 表示底侧外边距。

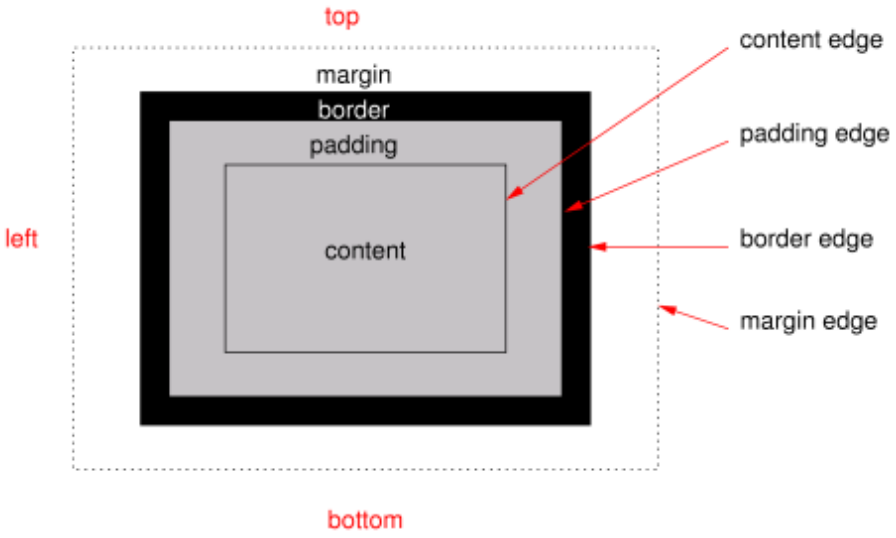

图 4-3　盒模型

在 CSS 中，直接用来描述盒模型的属性有 margin、border、padding、width、height。需要注意的是，CSS 提供的宽度属性（width）和高度属性（height）指的是内容区域（content）的宽度和高度，而不是整个盒模型的宽度和高度。整个盒模型的大小应该包括内外边距及边框的宽度。

盒模型的宽度 = ”margin-left” + ”border-left” + ”padding-left” + ”width” + ”padding-right” + ”border-right” + ”margin-right”

盒模型的高度 = ”margin-top” + ”border-top” + p”adding-top” + ”height” + p”adding-bottem” + ”border-bottom” + ”margin-right”

4.7.2　设置盒子大小

在 CSS 中，可以使用盒模型的 width 和 height 属性为除行内元素之外的大多数元素设置高度和宽度。行内元素的宽度和高度取决于自身内容。

宽度和高度的值可以为百分比、带单位的长度或者是 auto。如：

```
#content {
    width: 90%;
    height: 300px;
}
```

百分比的计算是按照父元素的大小来计算，而不是按照本身的大小。如果没有指定宽度，盒模型就是用默认值 100%，也就是说和所在容器的宽度一样，如果没有指定高度，浏览器则会根据内容自动计算出高度大小。

除了 width 和 height 之外，CSS 还提供了 max-height、max-width、min-height、min-width 属性。

4.7.3　设置外边距

外边距用来控制元素之间的距离，在 CSS 中，使用 margin 属性来控制外边距，外边距是透明的空间量。内容之间适当的空间能够增加内容的可读性。除行内元素不接受上下外边距的设定外，其他元素都可以设定外边距。

margin 属性的值可以是带单位的长度、百分比和 auto。设置外边距的方式如下：

```
div {
  margin: 10px;
}
```

上述规则表示 div 元素四侧的外边距为 10 个像素。

```
div {
  margin: 10px 20px;
}
```

上述规则表示 div 元素上下侧的外边距为 10 个像素，左右侧的外边距为 20 像素。

```
div {
  margin: 10px 20px 5px;
}
```

上述规则表示 div 元素上侧的外边距为 10 像素，左右两侧的外边距为 20 像素，下侧的外边距为 5 像素。

```
div {
  margin: 10px 20px 0 5px;
}
```

上述规则表示 div 元素上侧的外边距为 10 像素，右侧的外边距为 20 像素，下侧外边距为 0，左侧外边距为 5 像素。

除了使用 margin 统一设定元素外边距外，CSS 还提供了 margin-top、margin-right、margin-bottom 以及 margin-left 四个属性分别设定各侧的外边距。

需要特别说明的是，当元素左右两侧的外边距取值 auto 时，这个元素就会在所在容器中居中。如：

```
div {
  margin: 0 auto;
}
```

4.7.4　设定内边距

CSS 中使用 padding 表示内边距，内边距和外边距在很多方面是相似的。padding 的值可以是带单位的长度或者是百分比。padding 属性值中没有 auto。padding 属性值可以是 1-4 个值，其意义与 margin 相同。padding 也可以分侧指定，如 padding-top、padding-right、padding-bottom、padding-left。

4.7.5　设置边框

边框是进行信息组织是的一种有效手段，通过边框的使用，能够区分不同类型的信息，而且边框还是一种装饰手段，能在组织信息的同时美化页面。CSS 提供了 border-style、border-width 和 border-color 以及 border 元素来控制边框的样式、宽度以及颜色。如：

```
#footer {
  border-style: dashed;
  border-width: 1px;
  border-color: #ccc;
}
```

上述规则将使得 id 为 footer 的元素四周拥有灰色、1 个像素宽度的虚线边框。

其中 border-style 的属性值用来指定边框线的样式，默认值为 none，也就是没有边框，因此，在定义边框属性时，border-style 实际上是必须要指定的，常用的值有：solid, double, dashed, dotted, and none。

border-width 的属性用来指定边框先的宽度，宽度值为带单位的长度（如 1px）或关键字（thin、medium、thick），宽度默认值为 medium。

border-color 属性用来指定边框颜色，其值可以为文本、16 进制颜色值和 rgb 函数值，颜色的默认继承元素内容的颜色。

CSS 还提供了快速设定边框样式的属性——border，如上述的样式可以这样简写：

```
#footer {
  border: dashed 1px #ccc;
}
```

border 的值中必须要指定的是 border-style，其他两个可以任意组合，并且对出现的先后顺序也无要求。和 margin、padding 属性类似，border-style、border-width 以及 border-color 可以接受 1-4 个值，用以分别指定不同侧面的边框样式，如下面的样式规则使得 id 为 footer 的元素上下两侧没有边框，左右两侧的边框样式为虚线、宽度为 medium、颜色和该元素的内容颜色一致：

```
#footer {
  border-style: none dashed;
```

```
}
```

最后，CSS 也提供了分别指定不同方向上边框的机制：border-top、border-right、border-bottom 以及 border-left。如下面的样式规则将使得 id 为 footer 的元素拥有 1 个像素宽的红色虚线上边框：

```
#footer {
  border-top: dashed 1px red;
}
```

4.7.6　圆角

border-radius 属性，允许我们为元素设置圆角效果。

border-radius 属性接受长度单位，包括百分比和像素。border-radius 如果只有一个值，则设定四个角，俩值、三个值和四个值的情况与 margin、padding 类似，都按顺时针方向设定。例如：

```
.border-football {
  border-radius: 15px 75px;
}
```

上述代码设定左上、右下角为半径 15 像素的圆角，而右上和左下的圆角半径为 75 像素。

4.8　定位及布局

4.8.1　信息流

浏览器在呈现信息时会按照元素的类型进行处理，它将块元素从上到下显示（块元素与块元素之间另起一行），将行内元素按语言方向水平显示（如汉字、英语是从左到右，维吾尔语、阿拉伯语等有些语言是从右到左），行内元素直到到达容器边缘时才换行显示，这种显示元素的方式叫做页面的正常流。

常见的大多数元素属于块元素，如 p、table、div、li、ul、ol、object、h1-h6 等等，行内元素有 a、span、img、b、strong 等等。需要注意的是，匿名内容（即没有使用元素标签标注的内容）也按行内元素处理。

4.8.2　display

元素如何显示，可以通过 display 属性进行指定。每一个元素都有一个默认的 display 属性，但这个属性值可以修改。块元素和行内元素可通过 display 属性改变它们的显示方式，比如在某些情景，编辑人员需要给行内元素添加高度，以改进元素的

显示效果，行内元素是无法直接通过 width 属性指定高度的，但可以通过 display 属性将其变为块元素，再为其添加高度。

display 有很多属性值，最常用的是 block、inline、inline-block 以及 none。其中 block 将元素按照块元素方式显示；inline 将元素按照行内元素显示；而 inline-block 模式将使元素像行内元素那样显示，但同时又具有 bolock 元素的特征，可以赋予宽度和高度等等；none 模式将元素整体隐藏起来，相当于该元素不存在一样，元素一旦声明其 display 的值为 none，包含在内部的内容及其后代元素都会隐藏，并且其内容和后代不能再通过 display 改变显示方式。

4.8.3　浮动

除了相对定位和绝对定位能控制元素在正常流的位置之外，CSS 还提供了浮动（float）机制来控制元素在正常流中的位置，在实际工作中，float 属性是主要的定位、排版手段。从页面栏目的划分到图片的定位，大都通过 float 属性实现。

浮动的值有 left、right 和 none，其中左浮动将会使得块元素浮动向所在父元素的左侧，其后续的内容将出现在该元素的右侧，并和该元素的顶端对齐。右浮动刚好相反，none 则没有浮动效果。

先看一段 HTML 代码：

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<title>浮动</title>
<style>
#head {
  border:1px #CCC solid;
  width:410px;
  height:100px;
  margin-bottom:10px;
}
#main {
  border:1px #CCC solid;
  width:400px;
  height:200px;
  margin-bottom:10px;
  padding:5px;
}
#left {
  border:1px #CCC solid;
  width:198px;
  height:198px;
```

```
}
#right {
  border:1px #CCC solid;
  width:193px;
  height:198px;
  margin-left:5px;
}
</style>
</head>
<body>
<div id="head">头 部 </div>
<div id="main">
  <div id="left">左 侧 </div>
  <div id="right">右 侧 </div>
</div>
</body>
</html>
```

这段代码的显示效果如下：

下面，我们为 left 和 right 添加浮动属性。float 属性有三个值：none（不浮动）、left（左浮动）、right（右浮动），其中 none 是默认值。盒元素必须为其指定明确的宽度值，才能应用浮动属性。

```
#left {
  border:1px #CCC solid;
  width:198px;
  height:98px;
  float:left;
}
#right {
  border:1px #CCC solid;
  width:193px;
  height:98px;
  margin-left:5px;
  float:left;
}
```

显示效果如下：

我们看到 id 为 right 的 div 元素现在和 id 为 left 的 div 元素在同一水平位置上显示，并非另起一行。浮动的方向是针对使用了浮动元素的位置而言的，并不应用于其后的元素。另外一定要注意，如果一个元素没有宽度大小，则不能指定浮动效果。

常见的图文混排效果，就是通过浮动来实现。

```
<div class="item">
```

图 4-4　没有使用浮动效果的元素按正常流显示

图 4-5　使用了浮动效果的元素

```
<div class="img_area">
<a target="_blank" href="http://zhibo.qq.com/mbask/1844/index.html">
<img src="http://t3.qlogo.cn/mbloghead/72f0353448bac74e1d86/50" alt="
   罗崇敏：安全不保 何谈教育" title="罗崇敏：安全不保 何谈教育">
</a>
</div>
<div class="text_area">
<h5>
罗崇敏，中共云南省委高校工委书记，云南省教育厅长
<a class="more" target="_blank" href="http://zhibo.qq.com/mbask/1844/
   index.html">[详细]</a>
</div>
</div>
<div class="item">
<div class="img_area">
<a target="_blank" href="http://zhibo.qq.com/mbask/1833/index.html">
<img src="http://t3.qlogo.cn/mbloghead/ca99366b03bb20d692bc/50" alt="
   王旭明：如何避免校车惨剧一再发生" title="王旭明：如何避免校车惨剧
   一再发生">
</a>
</div>
<div class="text_area">
<h5>
<a target="_blank" href="http://zhibo.qq.com/mbask/1833/index.html">
   王旭明：如何避免惨剧再发生</a>
```

```
</h5>
王旭明，教育部语文出版社社长，原教育部新闻发言人
<a class="more" target="_blank" href="http://zhibo.qq.com/mbask/1833/
   index.html">[详细]</a>
</div>
</div>
```

在这个例子中，图片和文字分别作为内容放在 div 元素中，这两个 div 容器为“img_area”和“text_area”，“img_area”和“text_area”是 div 容器的类名称（使用了 class 属性），这两个 div 元素又是类名为 item 的容器的内容，类名为 item 的 div 元素可以重复使用，如上例中，就有两个类名为 item 的 div 元素。如果不使用浮动，文字没有办法完美环绕在图片周围，将图片放入类名为 img_area 的 div 元素中，并把文字放在类名为 text_area 的 div 元素中，使用如下样式，即可达到效果：

```
.img_area {
    float: left;
    padding: 3px 10px 0 0;
    width: auto;
}
.text_area {
    line-height: 20px;
}
.item {
    padding: 0 5px 5px;
    text-align: left;
}
```

其中花括弧外面的“.img_area”为选择符，.img_area 表示该选择符为类选择符，对应 HTML 中 class 为 img_area 的元素，“float：left”表示选择符对应的元素将左浮动，“padding: 3px 10px 0 0;”语句声明选择符对应元素上右下左的内边距分别为 3 像素、10 像素、0 和 0，最后 width 表示该元素的宽度将根据所包含内容的大小计算而来。另外两个 CSS 规则用来设定文字区域的行间距和整体的内边距、对齐方式。

在这个例子中，我们看到了浮动的特点，首先，类名为 img_area 的 div 元素声明了左浮动的规则，这样，这个 div 元素就改变了它默认的块元素显示方式，而是尽可能得向父元素（类名为 item 的 div 元素）的左上角靠齐，并且，它右侧的空间将会被后续的元素所填充，因此，尽管类名为 text_area 的 div 元素没有声明浮动，但它还是占据了类名为 img_area 元素的右侧空间。这样，我们就达到了图文混编的效果。

4.8.4　清除浮动

从上面的例子中，我们清楚地看到，使用了浮动属性的元素将会影响到其后的内容。这种影响有时候是我们期望的，有时候是我们所不乐见的。如下图所示：

图中的 HTML 及 CSS 如下：

图 4-6　有清除浮动时，左浮动后面的内容会尽可能得向左向上靠齐

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>浮动清除实例</title>
<style type="text/css">
body {
  font-size:14px;
  text-align:center;
}
.item {
  height:115px;
  width:500px;
  border:1px #ccc solid;
  padding:10px;
  margin:20px;
}
.float_area {
  border:1px #ccc solid;
  width:100px;
  float:left;
  margin:0 10px 10px 0;
}
.normal_area {
  border:1px #ccc solid;
  width:210px;
}
.float_area,.normal_area {
  height:50px;
  line-height:50px;
}
</style>
</head>
<body>
<div class="item">
```

```
  <div class="float_area">左浮动模块</div>
  <div class="float_area">左浮动模块</div>
  <div class="normal_area">希望按正常流显示的模块</div>
</div>
</body>
</html>
```

在上面的代码中，类名为 float_area 的 div 元素使用了左浮动，但是我们不希望类名为 normal_area 的 div 元素正常显示，也就是另起一行显示时，我们就必须使用 clear 属性来清除浮动，如下图 6-24 所示。

clear 的值有 left、right、both 和 none，其中 left 表示清除左浮动，right 表示清除右浮动，both 表示清除左、右两侧的浮动影响，而 none 表示不清除浮动。

图 4-7　清除浮动后，左浮动后面的内容位置正常

上图中的 HTML 和 CSS 如下：

```
<!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>浮动清除实例</title>
<style type="text/css">
body {
  font-size:14px;
  text-align:center;
}
.item {
  height:115px;
  width:500px;
  border:1px #ccc solid;
  padding:10px;
  margin:20px;
}
.float_area {
  border:1px #ccc solid;
  width:100px;
  float:left;
```

```
  margin:0 10px 10px 0;
}
.normal_area {
  border:1px #ccc solid;
  width:210px;
  clear:both;
}
.float_area,.normal_area {
  height:50px;
  line-height:50px;
}
</style>
</head>
<body>
<div class="item">
  <div class="float_area">左浮动模块</div>
  <div class="float_area">左浮动模块</div>
  <div class="normal_area">希望按正常流显示的模块</div>
</div>
</body>
</html>
```

在本例中，我们需要类名为 normal_area 的 div 元素不和左浮动模块并排显示，而是另起一行，因此，我们为其声明了“clear：both”的 CSS 规则，该规则表示将类名为 normal_area 的 div 元素不受之前左右浮动元素的影响，具体到上例中，“clear：both”完全可以用“clear：left”代替，显示效果是一致的。除了上述的方法外，我们还可以在需要清除浮动的地方插入空白 div，为其声明内嵌样式表，在样式表中声明 clear 属性即可。如下例所示：

```
<div class="item">
  <div class="float_area">左浮动模块</div>
  <div class="float_area">左浮动模块</div>
  <div style=” clear:both;” ></div>
  <div class="normal_area">希望按正常流显示的模块</div>
</div>
```

这种方法对于初学者容易理解和掌握，但内容为空的 div 元素，在语义上并不十分完美，实际工作中广泛采用的是名为“clearfix”的一种技巧：

```
.clearfix:before,.clearfix:after {
  display: table;
  content: " ";
}
.clearfix:after {
  clear:both;
```

```
}
```

这种 clearfix 方案，生成了两个伪元素，并将其 display 设置成 table。这将创建一个匿名的 table-cell 和一个新的块状区域，这意味着：:before 伪元素阻止了顶部边缘塌陷。而:after 伪元素清除了浮动。其结果是，只需要在包含浮动元素的容器上添加 class=“clearfix”属性，没有必要专门添加 HTML 元素，减少清楚浮动所需的代码量，提高了工作效率。

在 CSS 中，和定位相关的常用属性有 position、top、right、bottom、left、z-index。通过这些属性，设计人员可以控制元素的精确定位。

4.8.5　position

position 属性用来设定对象的定位方式。它的值有：

static　对象遵循常规流。top，right，bottom，left 等属性不会被应用,static 是 position 属性的默认值。

relative　对象遵循常规流，并且参照自身在常规流中的位置通过 top，right，bottom，left 属性进行偏移时不影响常规流中的任何元素。

absolute　对象脱离常规流，使用 top，right，bottom，left 等属性进行绝对定位，盒子的偏移位置不影响常规流中的任何元素，其 margin 不与其他任何 margin 折叠。

fixed　对象脱离常规流，使用 top，right，bottom，left 等属性以窗口为参考点进行定位，当出现滚动条时，对象不会随着滚动。

center　对象脱离常规流，使用 top，right，bottom，left 等属性指定盒子的位置或尺寸大小。盒子在其包含容器垂直水平居中。盒子的偏移位置不影响常规流中的任何元素，其 margin 不与其他任何 margin 折叠。（CSS3 新增属性）

page　盒子的位置计算参照 absolute。盒子在分页媒体或者区域块内，盒子的包含块始终是初始包含块，否则取决于每个 absolute 模式。（CSS3 新增属性）

sticky　对象在常态时遵循常规流。它就像是 relative 和 fixed 的合体，当在屏幕中时按常规流排版，当卷动到屏幕外时则表现如 fixed。该属性的表现是现实中你见到的吸附效果。（CSS3 新增属性）

其中 relative 和 absolute 方式使用较多，它们的区别可从下面的例子中看出：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http
   ://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf
       -8">
```

```
    <title>相对定位与绝对定位</title>
    <style type="text/css">
      body {
        width:960px;
        margin: 0 auto;
        font-family: 微软雅黑;
        font-size: 18px;
        line-height: 1.8em;
      }
      div {
        width: 300px;
        float: left;
      }
      .relative {
        position: relative;
        top:150px;
      }
      .absolute {
        position: absolute;
        top:150px;
      }
      .bg {
        background-color: #fcc;
      }
    </style>
  </head>
  <body>
    <div>
      <h2>正常信息流</h2>
      <p>孩子，你真是快活啊，一早晨坐在泥土里，耍着折下来的小树枝。</
         p>
      <p>我微笑着看你在那里耍着那根折下来的小树枝。</p>
      <p class="bg">我正忙着算账，一小时一小时在那里加叠数字。</p>
      <p>也许你在看我，想到：“这种好没趣的游戏，竟把你的一早晨的好时
         间浪费掉了!"</p>
      <p>孩子，我忘了聚精会神玩耍树枝与泥饼的方法了。</p>
    </div>
    <div>
      <h2>相对定位</h2>
      <p>孩子，你真是快活啊，一早晨坐在泥土里，耍着折下来的小树枝。</
         p>
      <p>我微笑着看你在那里耍着那根折下来的小树枝。</p>
      <p class="bg relative">我正忙着算账，一小时一小时在那里加叠数
         字。</p>
```

```
      <p>也许你在看我，想到："这种好没趣的游戏，竟把你的一早晨的好时
          间浪费掉了!"</p>
      <p>孩子，我忘了聚精会神玩耍树枝与泥饼的方法了。</p>
      <code>.relative { position: relative; top:150px;}</code>
    </div>
    <div>
      <h2>绝对定位</h2>
      <p>孩子，你真是快活啊，一早晨坐在泥土里，耍着折下来的小树枝。</
          p>
      <p>我微笑着看你在那里耍着那根折下来的小树枝。</p>
      <p class="bg absolute">我正忙着算账，一小时一小时在那里加叠数
          字。</p>
      <p>也许你在看我，想到："这种好没趣的游戏，竟把你的一早晨的好时
          间浪费掉了!"</p>
      <p>孩子，我忘了聚精会神玩耍树枝与泥饼的方法了。</p>
      <code>.absolute {
        position: absolute;
        top:150px;}
      </code>
    </div>
  </body>
</html>
```

上例的运行结果如下图4-8所示：

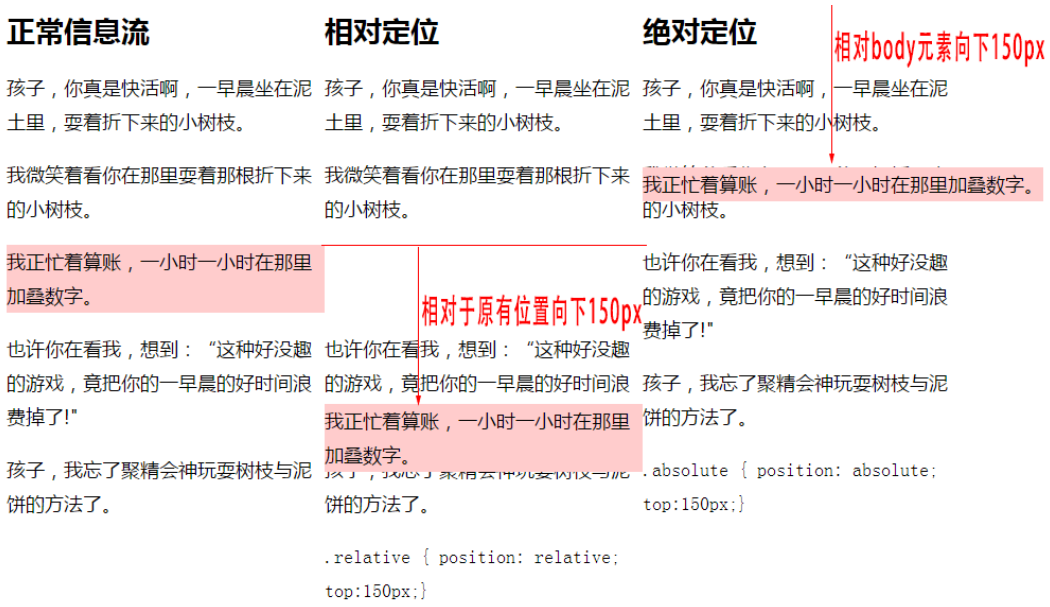

图 4-8　相对定位、绝对定位的区别

需要说明的是，absolute 方式定位的坐标原点默认为 body 元素的坐标原点，在上例中，使用绝对定位的元素虽然包含在第三个 div 元素中，但计算它的位置时，还是从 body 元素的顶部开始。

如果想要使 absolute 相对于所在父元素定位，则需要使父元素的 position 取值 relative。如下例：

```
<!DOCTYPE HTML>
<html>

<head>
    <meta charset="utf-8">
    <title>绝对定位的参照系</title>
    <style type="text/css">
    body {
        width: 960px;
        margin: 0 auto;
        font-family: 微软雅黑;
        font-size: 18px;
        line-height: 1.8em;
    }

    div {
        width: 400px;
        float: left;
        margin: 0 20px 0 0
    }

    .absolute {
        position: absolute;
        left: 50px;
    }

    .bg {
        background-color: #fcc;
    }

    code {
        background-color: #ccc;
        height: 50px;
        line-height: 50px;
        display: block;
    }
    </style>
</head>

<body>
    <div>
```

```
        <h2>以body为参照</h2>
        <p>孩子，你真是快活啊，一早晨坐在泥土里，耍着折下来的小树枝。
           </p>
        <p>我微笑着看你在那里耍着那根折下来的小树枝。</p>
        <p class="bg absolute">我正忙着算账，一小时一小时在那里加叠数
           字。</p>
        <p>也许你在看我，想到："这种好没趣的游戏，竟把你的一早晨的好
           时间浪费掉了!"</p>
        <p>孩子，我忘了聚精会神玩耍树枝与泥饼的方法了。</p>
        <code>.absolute { position: absolute; left:50px;}
      </code>
    </div>
    <div>
        <div style="position:relative">
            <h2>以父元素为参照</h2>
            <p>孩子，你真是快活啊，一早晨坐在泥土里，耍着折下来的小树
               枝。</p>
            <p>我微笑着看你在那里耍着那根折下来的小树枝。</p>
            <p class="bg absolute">我正忙着算账，一小时一小时在那里加
               叠数字。</p>
            <p>也许你在看我，想到："这种好没趣的游戏，竟把你的一早晨
               的好时间浪费掉了!"</p>
            <p>孩子，我忘了聚精会神玩耍树枝与泥饼的方法了。</p>
            <code>.absolute { position: absolute; left:50px;}
        </code>
        </div>
    </div>
</body>

</html>
```

上例的运行结果见4-9：

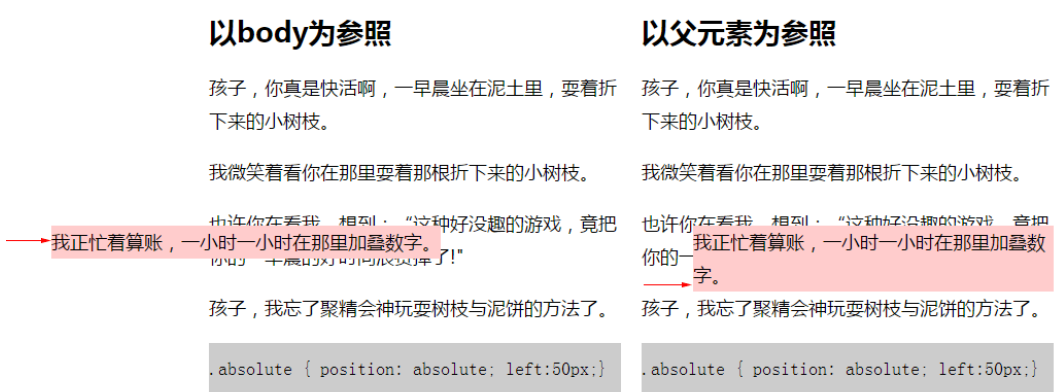

图 4-9　绝对定位的参照原点

4.8.6　top、right、bottom、left

top、right、bottom、left 属性用来设置对象参照相对物顶边界向下、向左、向上、向右偏移的位置，这四个属性的值可以是正值，也可以是负值，可以是整数，也可以是百分比。必须定义 position 属性值为 relative 、absolute 、fixed 、center 、page，此属性方可生效。

```
top:50%;
right:10px;
bottom:100px;
left:0;
```

4.8.7　z-index

z-index 属性用来设置对象的层叠顺序。同一个层叠上下文中，层叠级别（即 z-index 属性值）大的显示在上面，反之显示在下面。该属性对定义了 position 为 relative 、absolute 、fixed 、center 、page 、sticky 的元素有效，如果只设定 z-index 属性，但不使用 position 属性，则 z-index 属性无效。例如：

```
<!DOCTYPE HTML>
<html lang="zh">

<head>
    <meta charset="utf-8">
    <title>z-index属性演示</title>
    <style type="text/css">
    body {
        width: 960px;
        margin: 0 auto;
        font-family: 微软雅黑;
        font-size: 18px;
        line-height: 1.8em;
    }

    h1 {
        color: green;
    }

    div {
        width: 100px;
        height: 100px;
        text-align: center;
    }
```

```
    .index1 {
        position: absolute;
        left: 150px;
        top: 150px;
        background-color: #aaa;
    }

    .index2 {
        background-color: #ccc;
        position: absolute;
        left: 190px;
        top: 190px;
    }

    .index3 {
        position: absolute;
        left: 230px;
        top: 230px;
        z-index: 3;
        background-color: #eee;
    }

    .index4 {
        z-index: 4;
        background-color: #999;
    }
    </style>
</head>

<body>
    <h1>z-index属性只对使用了position属性的元素有效</h1>
    <div class="index4">z-index: 4</div>
    <div class="index1">z-index: 1</div>
    <div class="index2">z-index: 2</div>
    <div class="index3">z-index: 3</div>
</body>

</html>
```

上例中虽然为 div.index4 设定了 z-index 的值，但从运行效果（如下图4-10所示）来看，这个 z-index 值显然没有起作用。

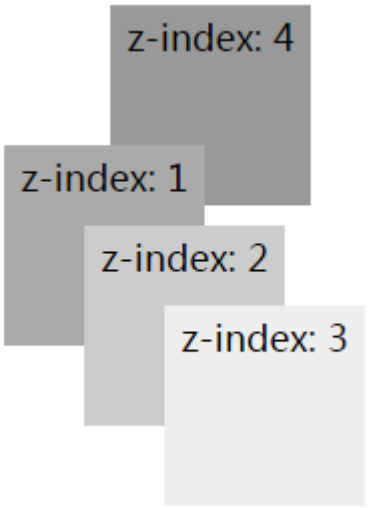

图 4-10　z-index 属性演示

4.8.8　CSS 中的居中

前面已经介绍，文字水平居中的方法为：

```
text-align:center;
```

而元素的水平居中，通过设定 margin-left 和 margin-right 的值为 auto 实现，如：

```
margin:0 auto;
```

单行文字的垂直居中，通过设定文字的行间距等于容器高度可以实现，例如：

```
.center-text-trick {
  height: 100px;
  line-height: 100px;
}
```

多行文字的的垂直居中，可通过利用表格的垂直居中属性 vertical-align 来实现，例如：

```
<div class="center-table">
  <p>I'm vertically centered multiple lines of text in a CSS-created
     table layout.</p>
</div>
.center-table {
  display: table;
  height: 250px;
  background: white;
  width: 240px;
  margin: 20px;
}
.center-table p {
```

```
  display: table-cell;
  margin: 0;
  background: black;
  color: white;
  padding: 20px;
  border: 10px solid white;
  vertical-align: middle;
}
```

元素的垂直居中，在不知道元素高度的情况下，可以通过绝对定位来实现，例如：

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>元素垂直居中实例</title>
    <style>
    main {
        background: white;
        height: 300px;
        margin: 20px;
        width: 300px;
        position: relative;
        resize: vertical;
        overflow: auto;
    }

    main div {
        position: absolute;
        top: 50%;
        left: 20px;
        right: 20px;
        background: black;
        color: white;
        padding: 20px;
        transform: translateY(-50%);
        resize: vertical;
        overflow: auto;
    }
    </style>
</head>

<body>
```

```
    <main>
        <div>
            我是一个块级元素，高度未知，我在父元素中垂直居中。
        </div>
    </main>
</body>

</html>
```

4.9　重置样式表

在现实中，用户使用的浏览器五花八门，比如 IE、Firefox、Chrome、Safari 等等，每个浏览器都有各自默认的显示 HTML 元素的样式，这些默认的样式可能并不相同。为了确保浏览器的兼容性，前端开发人员普遍采用重置样式表（CSS Reset）的办法。

重置样式表的思路非常简单：即为 HTML 元素预设一套样式，然后将其应用到所有浏览器。所谓重置样式，通常指的是移除元素的大小、外边距、内边距和其他样式。因为 CSS 可以级联，只要确保重置样式位于自定义样式之前，那么不同浏览器再渲染使用了重置样式表的页面时，会先重置样式表，然后再按照前端开发人员设置的样式来显示页面，这样达到了最大化统一不同浏览器显示效果的目的。例如：

```
    <link rel="stylesheet" href="reset.css">
    <link rel="stylesheet" href="main.css">
```

在网络上，我们可以看到有大量存在细微差异的重置样式表的方法，其中最流行的是Eric Meyer 提供的 CSS Reset 文件。除了强制重置样式之外，我们还可以选择不那么激进的 Normalize.css 方法，这种方法保留了大多数浏览器默认样式。

4.10　在线学习资源及工具

1. CSS 中文参考手册。http://css.doyoe.com/
2. CSS2.1 官方文档。https://www.w3.org/TR/2011/REC-CSS2-20110607/
3. CSS Color Module Level 3。https://www.w3.org/TR/css3-color/
4. 学习 CSS 布局。http://zh.learnlayout.com/
5. W3Cplus。http://www.w3cplus.com/
6. CSS 渐变生成器。http://www.cssmatic.com/gradient-generator
7. CSS 语法验证工具。http://jigsaw.w3.org/css-validator/

参考文献

[1] W3C. Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification[EB/OL]. 2016 [2016-3-17]. https://www.w3.org/TR/2011/REC-CSS2-20110607/.

[2] W3C. Selectors Level 3[EB/OL]. 2011 [2015-5-8]. http://www.w3.org/TR/2011/REC-css3-selectors-20110929/.

[3] CHANG D. 他真的不是我兄弟：像素跟點大不同 [EB/OL]. 2014 [2016-04-22]. http://blog.fourdesire.com/2014/12/11/ta-zhen-de-bu-shi-wo-xiong-di-xiang-su-gen-dian-da-bu-tong/.

[4] W3C. CSS Color Module Level 3[EB/OL]. 2011 [216-04-22]. https://www.w3.org/TR/css3-color/.

[5] W3C. CSS Image Values and Replaced Content Module Level 3[EB/OL]. 2012. https://www.w3.org/TR/css3-images/.

[6] BOS B, ETEMAD E J, KEMPER B. CSS Backgrounds and Borders Module Level 3[EB/OL]. 2014 [2016-4-19]. https://www.w3.org/TR/css3-background/.

大盈若冲, 其用不穷。

节选自老子《道德经》

第 5 章　JavaScript

JavaScript 是一种脚本语言，所谓脚本实际上就是一段嵌入到其他文档中的程序，用来完成特定的功能。JavaScript 脚本经常用来检验浏览器，相应用户动作、验证表单数据及实现动态特效等。在网站中，JavaScript 扮演的角色逐渐增多，正在蚕食着原本由 Flash 所占据的领地，如新闻幻灯片、焦点图、事件时间线、新闻地图等等。在 Web 标准中，行为层的实现目前是以 JavaScript 脚本为主，可以这样说，JavaScript 是事实上的浏览器端脚本。

尽管 JavaScript 刚开始的设计初衷是作为给非程序人员的脚本语言，但其本质是一门编程语言。[1]

5.1　JavaScript 简介

5.1.1　JavaScript 的特点

JavaScript 是一门可以运行在浏览器端的脚本语言。它与浏览器的结合使它成为世界上最为流行的编程语言之一，但它不是所谓的主流编程语言，在对这门语言没有太多了解，甚至对编程都没有什么了解的情况下，用户也能用它解决工作中的一些问题。

JavaScript 是事件驱动的语言。当用户在网页中进行某种操作时，就产生了一个“事件”。事件几乎可以是任何事情：单击一个网页元素、鼠标的移动等等均可视为事件。JavaScript 是事件驱动的，当事件发生时，它可以对之作出响应。具体如何响应某个时间由编写的事件响应处理程序完成。

JavaScript 是一种基于对象的语言，基于对象的语言含有面向对象语言的编程思想，但比面向对象语言更简单。它本身已包含一些创建完成的对象，通常情况下都是使用这些已创建好的对象。

JavaScript 是一种容易学习和掌握的编程语言。与其他编程语言相比，JavaScript 学习难度较低，学习资源丰富，由于 JavaScript 大多数情景都是在浏览器端使用，因此，用户可以通过查看源代码来学习，很多网站将所使用的 JavaScript 脚本都是开放的，可以下载学习并使用。

Javascript 是一种跨平台的脚本语言。JavaScript 的运行依赖于浏览器本身，与操作环境无关，只要能运行浏览器的计算机，并支持 javascript 的浏览器就可正确执行。

5.1.2 JavaScript 的用途

JavaScript 可以完成以下任务：

1. JavaScript 为 HTML 提供了一种程序工具，弥补了 HTML 语言在功能上的局限。HTML 只是一种标记语言，它的作用是将网页的内容通过标记组织起来，JavaScript 为其提供了实现动态交互和输出的一种途径，它可以很好地结合 HTML 标记语言实现复杂的应用。
2. JavaScript 可以为 HTML 页面动态添加内容。用户可以传递数据至 JavaScript 脚本，再由 JavaScript 修改 HTML 内容。
3. JavaScript 能响应一定的事件。当某些事件发生时，如键入内容、鼠标滑过、页面装载完毕等等，JavaScript 都可以根据事件作出响应。
4. JavaScript 可以动态地获取和改变 HTML 的元素属性和 CSS 属性，从而动态地创建内容和改变内容的显示。
5. JavaScript 可以检验数据，这在验证表单时候特别有用，在用户提交数据之前就可以保证数据的正确性，既能减轻服务器压力，又能使用户有更好的体验。
6. JavaScript 可以检验用户的浏览器，从而为用户提供更好的页面显示效果。
7. JavaScript 可以创建和读取 Cookie，根据保留在 cookie 中的数据，调整页面内容，提供更加个性化的服务。

尽管 JavaScript 能够完成很多复杂的应用，但由于浏览器安全性的考虑，JavaScript 有一些固有的限制，这些限制包括：

1. JavaScript 不允许读写客户端文件。唯一的例外是，JavaScript 可以读写 Cookie 文件，但是也有一些限制。这样限制能避免恶意程序传播病毒木马、窃取用户信息。
2. JavaScript 不允许写服务器端文件。尽管网页中有大量数据需要从服务器端读取和写入，如用户提交的投票信息、新闻评论、点击量等等，但是 JavaScript 不允许向服务器写入文件，通常都是利用服务器端脚本，如 PHP、ASP 程序来完成服务器文件的操作。
3. JavaScript 不能从读取已经打开的其它窗口的信息，因此，JavaScript 无法知晓浏览当前网页的用户还在访问哪些其它站点。
4. JavaScript 不能操纵不是由它打开的窗口，这是为了避免一个站点关闭其他任何站点的窗口，从而独占浏览器，强制用户接受信息。
5. JavaScript 调整浏览器窗口的大小和位置时，不能将窗口设置的很小或者移出屏幕之外。

5.1.3 JavaScript 开发环境

进行 JavaScript 的开发，需要开发人员具备以下环境：

文本编辑器 JavaScript 脚本是纯文本文件，因此，开发人员可选择自己熟悉的文本编辑器，推荐使用 Sublime Text 编辑器。

浏览器 由于 JavaScript 是网页内容的一部分，要预览脚本的运行效果，自然少不了浏览器，根据项目的不同，开发人员应该选择合适的浏览器，一般而言，至少需要 IE、Firefox 浏览器。

调试工具 JavaScript 脚本在开发过程中难免出现预料之外的运行结果，为提高开发效率，需要安装 JavaScript 调试工具，建议使用 Firefox 浏览器中的 firebug 组件，下载地址http://getfirebug.com/。

除上述工具之外，开发人员最好还应该有权威准确、更新及时的参考手册，推荐以下学习资源：

1. https://developer.mozilla.org/zh-CN/docs/Web/JavaScript
2. 《精通 JavaScript》[2]

5.1.4 在网页中使用 JavaScript 的方法

就像在 HTML 中使用 CSS 一样，必须通过适当的方法将 JavaScript 和 HTML 结合起来使用，JavaScript 只有被嵌入到 HTML 中时才能对网页产生作用。和使用 CSS 的方式类似，在 HTML 中使用 JavaScript 的方式有三种，即嵌入式、行内式和链接式。

嵌入式

所谓嵌入式使用 JavaScript 的方法就是利用 script 元素将脚本嵌入到网页中。Script 元素是 HTML 语言为了引入脚本程序而定义的一个标记，插入脚本的具体方法是：把脚本用 script 元素标记后，放在 HTML 文件中的 head 部分或者 body 部分。虽然 script 脚本既可以放在 head 部分和 body 部分，但比较好的做法是将所有包含预定义函数的脚本放在 head 部分，因为 HTML 的内容在浏览器中处理时是从上到下解释的，放在 head 中的脚本比放在 body 中的脚本先处理。这样，浏览器在处理 body 中的元素内容时需要用到的 JavaScript 功能已经预先装载，减少出错的可能。同样的道理，如果 JavaScript 脚本需要等到页面完全装载完毕才要使用，则将其放置在 body 部分末尾。

使用 script 元素标记标本的语法如下：

```
<script >
  脚本内容
</script>
```

下面的 HTML 代码创建了一行文本，当用户单击文本时，文字会改变为其它内容。

```
<!DOCTYPE html>
<html lang="zh">

<head>
```

```
    <meta charset="utf-8" />
    <title>嵌入式JavaScript案例</title>
    <script>
    function hello() {
        document.getElementById("demo").innerHTML = "你好，JavaScript
            世界！";
    }
    </script>
</head>

<body>
    <p id="demo" onclick="hello()">单击这段文字!</p>
</body>

</html>
```

在上述代码中，onclick=”hello()” 表示当鼠标单击时执行 hello() 函数功能，而 display 函数就是用 script 元素标记的嵌入式脚本。

行内式

所谓行内式脚本应用就是将脚本内容写在 HTML 元素的标记中，在这些标记中，将事件和相应内容写在一起。如上例的 HTML 代码，如果用行内式的写法可以修改如下，而执行结果是完全相同的：

```
<!DOCTYPE html>
<html lang="zh">

<head>
    <meta charset="utf-8" />
    <title>行内式JavaScript案例</title>
</head>

<body>
    <p id="demo" onclick="javascript:alert('Hello!');">单击这段文字
       !</p>
</body>

</html>
```

从上述代码可以看到，行内式的写法似乎更加简洁，但是，在实际工作中，我们要尽量避免这种写法，因为行内式嵌入的脚本可读性很差，不容易维护，不符合 Web 标准中将内容、表现、行为三者相分离的原则。而且，但从效率的角度来讲，如果某个功能要反复使用，则行内式的写法将会过于繁琐，反倒是其它写法更加简洁。故

而，行内式使用脚本的方式仅限于较为简单的情况。

链接式

所谓链接式就是将 JavaScript 脚本文件单独写在后缀名为 js 的文件中，然后在需要使用此脚本的网页中加入脚本文件的路径名称即可。这种方式使用的脚本可以链接到多个网页甚至多个网站中，提高了代码的重用性，也方面了维护，同时也能降低网站主机访问次数，提高网站性能。

要引用外部 js 文件，需要在合适的地方，一般为 HTML 文件的 head 部分，使用 script 元素来指定外部脚本存放的路径和名称。

还是以上述中的 HTML 为例，如果改写成链接式引用，则需要将脚本存放成单独的文件，HTML 文档代码如下：

```
<!DOCTYPE html>
<html lang="zh">

<head>
    <meta charset="utf-8" />
    <title>链接式JavaScript案例</title>
    <script src="js/hello.js"></script>
</head>

<body>
    <p>链接式JavaScript案例</p>
</body>

</html>
```

其中使用 src 属性引用的 hello.js 文件存放在 HTML 文件的同一级目录中，外部 hello.js 文件内容如下：

```
alert('Hi');
```

5.2　JavaScript 核心语法

5.2.1　注释

JavaScript 的注释和 CSS 注释类似，分多行和单行，多行注释使用/* ... */，如：

```
/*
Everything in between is a comment.
*/
```

单行注释使用“//”表示，如：

```
// This is a comment
```

5.2.2　变量

所谓变量就是程序中数据的临时存放场所。JavaScript 的变量定义都是使用“var”关键字，示例如下：

```
var author = "yangjh";
var age = 35;
var Age = 36;
var male = true;
```

上述三行代码中，var 关键字表示声明变量，变量名紧跟其后，通过等于号将变量赋予变量初始值。每一行结束的分号可以不写，但书写分号后，能增强代码的可读性。

变量的名称必须遵循以下规则：

1. 首字符必须是字母、下划线或美元符号。
2. 其余字母可以是下划线、美元符号、任意字母或者数字。
3. 变量名称不能是关键字或保留字，具体保留字的内容参见附录C。
4. 变量名对大小写敏感，例如：变量 age 和 Age 是不同的变量。
5. 变量名中不能有空格、回车符或其他标点字符。

5.2.3　常量

常量意味着其中保存的值是固定不变的，JavaScript 使用关键字 const 声明常量，习惯上，常量名称使用大写字母。

```
const A = 7;
console.log("A is " + A + ".");
```

上面的例子将输出”A is 7.”。

5.2.4　运算符

运算符用于将一个或多个值运算成结果。在 Javascript 中，常用的运算符有以下几种类别：

算术运算符

算术运算符能对操作数进行运算，返回一个数值型的值。常见的有“+”、“-”、“*”、“/”。

关系运算符

关系运算符（见5-1）通常用于检查两个操作数之间的关系，返回值为 true 或 false。

表 5-1　关系运算符

运算符	说明	例子	结果
==	是否相等	5==”5”	TRUE
===	是否全等	5==”5”	FALSE
！=	是否不等	5！=”5”	FALSE
!==	是否不全等	5！=”5”	TRUE
>	是否大于	5>8	FALSE
<	是否小于	5<8	TRUE
>=	是否大于等于	5>=8	FALSE
<=	是否小于等于	5<=8	TRUE

赋值运算符

赋值运算符是使用最多的运算符，常见的赋值运算符就是“=”，它将右边的值赋与等号左边的变量。

逻辑运算符

逻辑运算符用来判断操作数之间的逻辑关系，返回值为 true 和 false。javascript 支持一下三种逻辑运算符（见5-2）。

表 5-2　逻辑运算符

运算符	说明	例子	结果
&&	逻辑与	5>3 && 4>3	TRUE
||	逻辑或	5>3 || 3>5	FALSE
!	逻辑非	!(3>5)	TRUE

连接运算符

连接运算符“+”能将字符串连接起来构成一个新的字符串。如：

```
var txt1 = "JavaScript";
var txt2 = "基础教程";
var txt3 = txt1 + txt2;
```

变量 txt3 的结果是“JavaScript 基础教程”。

5.2.5　流程控制

和其他语言一样，JavaScript 也是由上到下逐行执行代码，还可以通过分支、循环改变执行流程。

语句块

语句块用来组织多条语句。用一对花括号界定语句块。语法如下：

```
{
  statement_1;
  statement_2;
  ...
  statement_n;
}
```

注意语句块不是以分号结尾。语句块常在流程控制语句中使用。

if...else

根据设定条件，选择性地执行语句，当指定条件为 true 时会执行一条语句，如果该条件为 false，则执行另一条语句。

```
if (condition) {
   statements1
} else {
   statements2
}
```

switch

switch 语句对一个表达式求值，将结果与 case 子语句比较，如果匹配，则从 case 处的语句向下执行。

```
switch (expression) {
  case value1:
    // 当 expression 的结果与 value1 匹配时，从此处开始执行
    statements1;
    [break;]
  case value2:
    // 当 expression 的结果与 value2 匹配时，从此处开始执行
    statements2;
    [break;]
  ...
  case valueN:
```

```
    // 当 expression 的结果与 valueN 匹配时，从此处开始执行
    statementsN;
    [break;]
  default:
    // 如果 expression 与上面的 value 值都不匹配时，执行此处的语句
    statements_def;
    [break;]
}
```

switch 语句中的 break 语句，能保证程序执行到该语句时，不再执行后续语句，而是跳出 switch 语句块。

for

for 语句创建一个包含初始条件、终止条件和步进规则的循环体。例如下面的代码将循环 10 次（从 0-9），然后在控制台中打印出 0-9 的数字：

```
for (var i = 0; i < 9; i++) {
   console.log(i);
   // more statements
}
```

while

while 语句可以在某个条件表达式为真的前提下，循环执行指定的一段代码，直到那个表达式不为真时结束循环。语法如下：

```
while (condition) {
  statement
}
```

do...while

do...while 语句创建了一个循环，该循环执行一个指定的语句直到 condition 的值为 false。condition 在执行 statement 后才会被赋值，statement 至少执行一次。语法如下：

```
do {
   statement;
   ...
 }
while (condition);
```

5.2.6 函数

使用关键字 function 声明一个函数。函数还可有参数和返回值。

```
function name([param,[, param,[..., param]]]) {
   [statements]
}
```

参数可以是常量、变量或表达式，多个参数之间用逗号隔开。若函数需要返回值，使用 return 语句。默认情况下，函数是返回 undefined 类型。想要返回一个其他的值，函数必须通过一个 return 语句指定返回值。

定义函数的时候，并不会执行构成该函数的任何语句。只有当调用该函数时，函数里面的语句才会执行，直到碰到 return 语句，return 语句之后的内容不会被执行。

回调函数

回调函数

5.2.7 对象

Javascript 中对象 (object) 的概念可以比照着现实生活中实实在在的物体来理解。

在 JavaScript 中，一个对象可以是一个单独的拥有属性和类型的实体。我们拿它和一个杯子做下类比。一个杯子是一个对象 (物体)，拥有属性。杯子有颜色，图案，重量，由什么材质构成等等。同样，javascript 对象也有属性来定义它的特征。

一个对象就是一系列属性的集合，一个属性包含一个名字和一个值。一个属性的值可以是函数，这种情况下属性也被称为方法。

在 JavaScript 中，几乎所有的东西都是对象。所有的原生类型除了 null 与 undefined 之外都被当作对象。它们可以被赋予属性，并且它们都有对象的全部特征。

对象的创建

JavaScript 拥有一系列预定义的对象。另外，我们可以通过对象初始化器（Object Initializer）创建自己的对象。或者通过创建一个构造函数并使用该函数和 new 操作符初始化对象。

通过对象初始化器创建对象的方法如下：

```
var obj = { property_1:   value_1,   // property_# may be an
   identifier...
            property_2:   value_2,   // or a number...
            // ...,
            "property n": value_n }; // or a string
```

这里 obj 是新对象实例的名称，每一个 property_i 是一个标识符（可以是一个名称、数字或字符串字面量），并且每个 value_i 是一个其值将被赋予 property_i 的表达式。属性的集合用标记，通过点符号来访问一个对象的属性。对象有时也被叫作关联数组，因为每个属性都有一个用于访问它的字符串值。如：

```
obj['property_1'] = value1;
obj[0] = value1;
```

但显然，用方括号访问对象属性的方式不如使用点符号的方式简洁。

通过 new 运算符创建对象的示例如下：

```
var author = new Object();
author.words = '绿叶恋爱时便成了花';
console.log(author.words);
```

new 运算符的作用是创建一个对象实例。单纯的使用 new 运算符创建对象实例的方式不够灵活，尤其是要创建对象的多个实例时，需要逐一完成对象属性的赋值工作。因此，常通过构造函数和 new 运算符的结合创造对象，这种对象创造方法能在创造对象的同时完成对象属性的赋值。

构造函数，是一种特殊的方法。主要用来在创建对象时初始化对象，即为对象成员变量赋初始值，与 new 运算符一起使用在创建对象的语句中。

通过构造函数和 new 运算符创建对象的例子：

```
function student(sName, sex, major) {
        this.sName = sName;
        this.sex   = sex;
        this.major = major;
    }
var student1 = new student('小明', '男','新媒体');
```

其中的 this 关键字指代对象本身。

对象属性的枚举

在有些情况下，开发人员并不了解对象的内部属性，这时可以使用 for…in 语句枚举对象的所有属性。如下例所示：

```
for(var i in student1) {
  console.log(i + "=" + student1[i]);
}
```

5.3　JavaScript 变量类型

最新的 ECMAScript 标准定义了 7 种数据类型:Boolean、Null、Undefined、Number、String、Symbol[①] 和 Object。

5.3.1　Boolean 布尔型

布尔型数据的取值只有两个：true 和 false。布尔型数据不能用引号括起来，否则就变成字符串了。

```
var a = false; //boolean 类型
var b = 'false'; //字符串类型
```

5.3.2　Null 空类型

值 null 是一个 JavaScript 字面量，表示空值（null or an "empty" value），即没有对象被呈现（no object value is present）。

5.3.3　Undefined 未定义类型

一个未初始化的变量的值为 undefined, 使用严格相等运算符 (===) 来判断一个值是否是 undefined:

```
var x;
alert(typeof x);
```

5.3.4　Number 数值型

在 Javascript 中，数值型数据不区分整数和小数，数值型和字符型数据的区别是数值型数据不用引号括起来。例如：

```
var num1 = 54;
var num2 = 5.4;
```

Number 类型内置了一些方法，其中 toString() 方法返回指定 Number 对象的字符串表示形式。

```
var count = 10;

print( count.toString() );    // 输出 "10"
print( (17).toString() );     // 输出 "17"
```

[①] symbol 类型是 ECMAScript 6 新增的数据类型，是一种唯一、不可变的变量。

```
var x = 6;

print( x.toString(2) );       // 输出 "110"
print( (254).toString(16) ); // 输出 "fe"


print( (-10).toString(2) ); // 输出 "-1010"
print( (-0xff).toString() ); // 输出 "-11111111"
```

5.3.5　String 字符串

字符串由多个字符构成，字符可以是字母、数字、标点符号或空格。字符串必须放在单引号或者双引号中。例如：

```
var txt1 = "JavaScript";
```

5.3.6　Symbol 符号类型

符号类型在 ECMAScript 第 6 版中被引入 Javascript。符号类型是唯一的并且是不可修改的。

5.3.7　Object 对象类型

在 Javascript 里，对象是非常重要的一种数据类型，对象可以被看作是由一些彼此相关的属性和方法集合在一起构成的数据实体。JavaScript 中内置了一些对象，如 array（数组）、date（日期）等等。

数组

字符串、数值型和布尔型变量在任意时刻只能存储一个值。如果要用一个变量存储多个值，就需要使用数组。数组是由名称相同的多个值构成的一个集合，集合中的值是数组的元素（element）。例如可以将一个班级的所有成员姓名存储在数组中。

在 Javascript 中，使用 array 关键字来定义数组，在定义中，可以声明数组的大小，也可以不声明数组中元素的个数。例如：

```
var country = new array(180);    //定义了数组的大小
var mybooks = new array();       //没有定义数组的大小
mybooks[0] = “HTML”;             //为数组mybooks的第一个元素赋值
mybooks[1] = “JavaScript”;       //为数组mybooks的第二个元素赋值
```

数组中元素的索引默认从 0 开始，上例中的“mybooks[0]”表示数组中的第一个元素。

除了使用 array 关键字定义数组外，还可以直接使用方括号定义数组，数组中的元素要用逗号隔开，最后一个元素后没有逗号。如：

```
var site = [”腾讯”,”新浪”,”搜狐”,”网易”];
```

5.3.8　查看变量类型

在 JavaScript 中，typeof 操作符能返回一个字符串, 表示未经求值的操作数 (unevaluated operand) 的类型。

```
var a = 1;
alert(typeof a);
```

5.3.9　变量类型的转换

JavaScript 是一种弱类型或者说动态语言。开发者不用提前声明变量的类型，在程序运行过程中，类型会被自动确定，这也意味着我们可以使用同一个变量保存不同类型的数据：

```
var foo = 42;    // foo is a Number now
var foo = "bar"; // foo is a String now
var foo = true;  // foo is a Boolean now
```

除了让程序自动确定类型外，有时，我们也可以使用以下函数强制转换数据类型：

Number()　强制内容转为数字；
Boolean()　强制内容转为布尔值；
String()　强制内容转为字符串；

如：

```
var  t1  =  Boolean("");    //返回false,空字符串
var  t2  =  Boolean("s"); //返回true，非空字符串
```

5.4　浏览器对象模型 BOM

由于网页是在浏览器中运行的，因此 JavaScript 提供了一系列对象以便同浏览器窗口进行交互。这些对象有 window、navigator、location、document、screen 和 history 等，这些对象通称为浏览器对象模型（Browser Object Model），简称 BOM。

BOM 提供了独立于页面内容而与浏览器窗口进行交互的对象，其中 window 对象是整个 BOM 的核心，所有对象和集合都以某种方式与 window 对象关联。

navigator、location、document、screen 和 history 是 windows 对象的子集，document 对象又包含若干子对象（location、forms、anchors、images、links、embeds、applets），BOM 中的对象关系如图5-1所示：

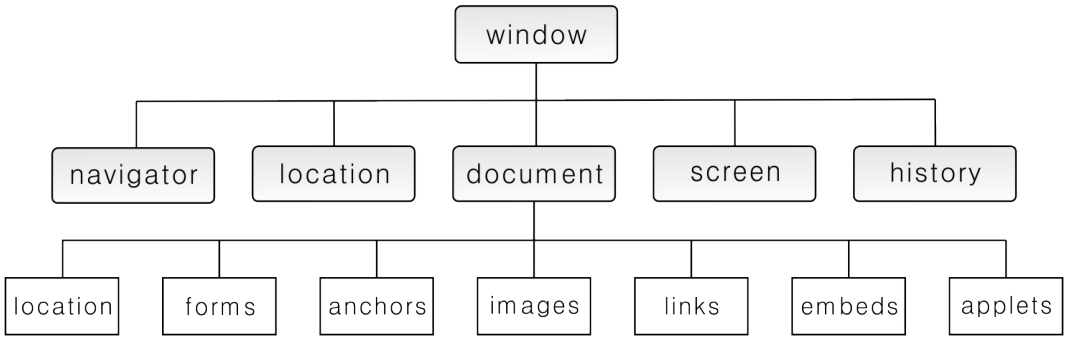

图 5-1　BOM 对象关系图

5.4.1　window 对象

JavaScript 在一定的环境中运行，这个运行环境本身也是对象，称为“顶层对象”。这就是说，JavaScript 的所有对象，都是“顶层对象”的下属。不同的运行环境有不同的“顶层对象”，在浏览器环境中，这个顶层对象就是 window 对象。

Window 对象表示整个浏览器窗口，但不包括其中的页面内容，使用 window 对象可以直接对浏览器窗口进行操作。window 对象提供可以主要功能有：调整窗口大小和位置；打开新窗口和关闭窗口；系统提示框；状态栏控制；定时操作。由于 window 对象的属性和方法众多，我们就不一一作介绍，如想深入了解，请访问`https://developer.mozilla.org/zh-CN/docs/Web/API/Window`。

常用属性

Window 对象拥有众多属性，常用的有：

innerHeight　浏览器显示区域高度
innerWidth　浏览器显示区域宽度
document　window 对象的子对象 document 是 javascript 的核心对象，HTML 文档中包含 BODY 元素时就会创建一个 document 实例。
location　给出当前窗口的 URL 信息或指定打开窗口的 URL。

常用方法

close()　关闭页面。
open()　打开页面。
alert()　弹出警告对话框。

print()　模拟用户点击浏览器上的“打印”按钮，通知浏览器打开打印对话框打印当前页。

confirm()　显示需要用户确认信息的对话框。

scrollTo()　将窗口中的内容滚动到指定位置。

scrollByPages()　滚动当前文档到指定页。

setTimeout()　在指定的延迟时间之后调用一个函数或执行一个代码片段。

setInterval()　周期性地调用一个函数 (function) 或者执行一段代码。

window 对象的属性和方法在调用时，可以省略其对象名称，即 window.innerHeight 可简写为 innerHeight。

5.4.2　navigator 对象

navigator 对象主要用来检查客户端浏览器信息、浏览器版本、操作系统的类型等等。navigator 对象最常用的属性是 userAgent，该属性能读取浏览器及操作系统的信息。

5.4.3　location 对象

location 对象的主要作用是分析和设置页面的 URL 地址，其中 location.herf 属性用于获得和设置窗口的 URL。

5.4.4　screen 对象

screen 对象主要用来获取用户计算机的屏幕信息，包括屏幕的分辨率、屏幕的颜色深度等信息。

5.4.5　history 对象

history 对象主要用来控制浏览器的后退和前进。它可以访问历史页面，但不能获取历史页面的 URL。

5.4.6　document 对象

从 BOM 的角度看，document 对象是一系列集合构成，这些集合可以访问文档的各个部分，并提供页面自身的信息。由于 BOM 没有统一标准，各种浏览器中的 document 对象特性并不完全相同，因此要使用各类浏览器都支持的通用属性和方法。常用的属性和方法有 title、lastModified、write 等等。

5.5　DOM

文档对象模型 DOM（Document Object Module）是由 W3C 定义的提供与任何 HTML 或 XML 文档进行交互的 API（编程接口），可以说是 HTML 之后的又一伟大创新。它使得用户可以通过 javaScript 访问 HTML 文档中的任意元素和内容，DOM 提供的接口可以操作 HTML 文件中的节点。当浏览器加载一个网页后，这个网页就可以看作是文档树，由多个节点构成。所谓 DOM，就是将 HTML 文档中各个元素按照从属关系建立起的模型。总的来说，我们可以利用 DOM 完成以下应用：

- 访问指定节点；
- 访问相关节点；
- 访问节点属性；
- 检查节点类型；
- 创建节点；
- 为节点添加事件；
- 操作节点。

5.5.1　HTML 文档与 DOM

我们先看下面这个简单的 HTML 文档：

```
<!DOCTYPE html>
<html lang="zh">
<head>
  <meta charset="utf-8">
  <title>一个简单的HTML文档</title>
</head>
<body>
  <h1>这是一个简单的HTML文档</h1>
  <p>文档中包含着段落和<a title="link" href="#">超级链接</a></p>
</body>
</html>
```

这个 HTML 文档的 DOM 可示例如下图5-2：

在上面的 DOM 中，html 位于最顶端，是 DOM 的跟节点，head 和 body 是 html 的子节点，它们属于同一层，并不互相包含，是兄弟关系，h1 和 p 是兄弟元素，其父元素是 body，p 的子元素是 a 元素。

5.5.2　节点

节点（node）的概念来源于计算机网络，它代表网络中的一个连接点。在 DOM 中，文档也是由节点构成的集合。DOM 定义了多种节点，常用的是元素节点、文本节点、和属性节点，分别对应元素、元素中包含的文字内容和元素的属性。

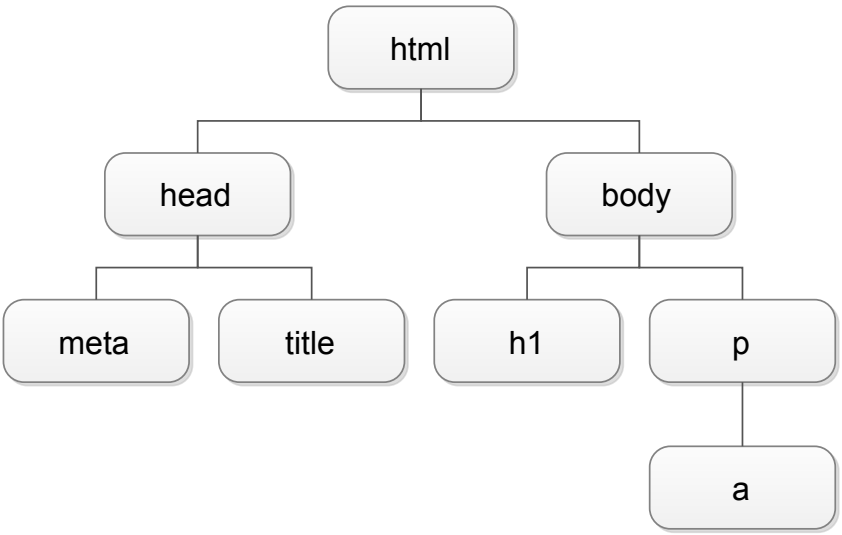

图 5-2　DOM 模型

例如上面的代码中：

```
<a title="link" href="#">超级链接</a>
```

a 元素的 title 和 href 属性就是 a 元素节点的属性节点，a 元素所包含的内容“超级链接”就是文本节点。如下图5-3所示：

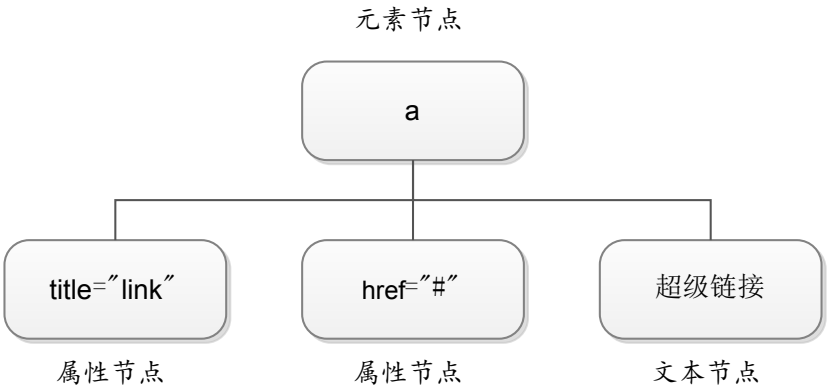

图 5-3　各种节点及关系

5.5.3　使用 DOM

DOM 提供了一种访问文档内容的机制，我们可以使用 DOM 处理 HTML 文档的内容。DOM 由节点（node）构成，每一个节点，都有一系列的属性、方法可以使用，常用属性、方法见下表5-3：

表 5-3　node 常用属性和方法

属性/方法	类型/返回类型	说明
nodeName	String	节点名称
nodeValue	String	节点的值
nodeType	Number	节点类型
firstChild	Node	childNodes 列表的第一个节点
lastChild	Node	childNodes 列表的最后一个节点
childNodes	NodeList	所有子节点列表，方法 item(i) 可以访问其中节点
parentNode	Node	父节点
previousSibling	Node	指向前一个兄弟节点
nextSibling	Node	指向后一个兄弟节点
hasChildNodes()	Boolean	是否包含子节点
attributes	NameNodeMap	包含一个元素特性的 Attr 对象
appendChild(node)	Node	将 node 节点添加到 childNodes 末尾
removeChild(node)	Node	从 childNOdes 中删除 node 节点
repaceChild(new,old)	Node	将 childNodes 中的 oldnode 节点替换为 newnode 节点
insertBefore(new,ref)	Node	在 childNOdes 中的 refnode 节点之前插入 newnode 节点
innerHTML	String	读取或者设置某个标记之间的所有内容
className	String	读取或者设置节点的 CSS 类别

访问节点

DOM 提供了一些很便捷的方法来访问文档的节点，最常用的是 getElementsByTagName() 和 getElementById()，此外比较常用的是 getElementsByName()、getElementsByClassName();

```
……
    var oLi = document.getElementsByTagName('li');
    for(var i = 0 in oLi) {
        console.log(i + '=' + oLi[i]);
    }
    console.log(oLi);
    console.log(oLi.length);
    console.log(oLi[0].childNodes[0].nodeValue);
……
```

将访问 oLi 子节点列表中的第一个节点的值。

检测节点类型

通过节点的 nodeType 属性可以检测出节点的类型，DOM 定义了 12 种类型，大多数情况下，我们用到的是以下类型：

1. ELEMENT_NODE 元素节点 nodeType 的值为 1；
2. ATTRIBUTE_NODE 属性节点 nodeType 的值为 2；
3. TEXT_NODE 文本节点 nodeType 的值为 3；

利用父子兄关系查找节点

在获取了某个节点之后，可以通过父子关系，利用 hasChildNodes() 方法和 childNodes 属性获取该节点所包含的所有子节点。例如：

```
……
    var oLi = document.getElementById('myList');
    var DOMString = "";
    if(oLi.hasChildNodes()){
      var oChild = oLi.childNodes;
      for(var i=0; i<oChild.length;i++){
        DOMString += oChild[i].nodeName + "\n";
      }
    }
    console.log(DOMString);
……
```

从运行结果我们可以看出，不光有元素节点，连它们之间的空格也被当成了子节点。

通过 parentNode 属性，可以获取父元素节点。如：

```
……
    var myItem = document.getElementById('active');
    console.log(myItem.parentNode.tagName);
……
```

DOM 还提供了处理兄弟之间关系的属性和方法，如 nextSibling、previousSibling。

设置节点属性

找到节点后，可以通过 getAttribute()、setAttitude() 方法取得或者设定节点的属性。

```
……
var myItem = document.getElementById('active');
console.log(myItem.parentNode.tagName);
console.log(myItem.getAttribute("title"));
myItem.setAttribute("title","主要用以和用户交互");
console.log(myItem.getAttribute("title"));
……
```

创建和添加节点

创建元素节点采用 creatElment(), 创建文本节点采用 creatTextNode(), 创建文档碎片节点采用 creatDocumentFragment() 等等。

在插入元素之前，先将元素内容添加到元素节点，再将元素节点及其包含的文本节点添加到指定节点。

```
……
var oP = document.createElement("p");
var oText = document.createTextNode("这是一个使用DOM生成的段落");
oP.appendChild(oText);
document.body.appendChild(oP);
……
```

删除节点

删除节点使用 removeChild() 方法，通常先找到要删除的节点，然后利用 parentNode 属性找到父节点，然后使用父节点的 removeChild 方法。

```
……
<body onload="removeLi()">
  <ul id="myList">
    <li>HTML</li>
    <li>CSS</li>
    <li id="active" title="JavaScript是脚本编程语言">JavaScript</li>
  </ul>
  <script>
  function removeLi() {
    var oLi = document.getElementById('active');
        oLi.parentNode.removeChild(oLi);
  }
  </script>
</body>
……
```

替换节点

DOM 提供 replaceChild() 方法来替换节点，具体过程和删除节点类似，先找到想要替换的节点，再创建新节点，然后使用父节点替换方法。

```
……
<body onload="replaceLi()">
  <ul id="myList">
    <li>HTML</li>
    <li>CSS</li>
    <li id="active">JavaScript</li>
  </ul>
  <script>
  function replaceLi() {
    var oOldLi = document.getElementById('active');
        var oNewLi = document.createElement('li');
        var oText = document.createTextNode('ECMAScript');
        oNewLi.appendChild(oText);
        oOldLi.parentNode.replaceChild(oNewLi,oOldLi);
  }
  </script>
</body>
……
```

innerHTML 属性

innerHTML 属性虽然不是 W3C DOM 的组成部分，但它得到了目前主流浏览器的支持。该属性表示某个标记之间的所有内容，包括代码本身。

```
var oLi = document.getElementById('active');
console.log(oLi.innerHTML);
oLi.innerHTML = "<a href='http://www.baidu.com'>Baidu</a>";
```

className 属性

className 属性是一个非常实用的属性，可以修改一个节点的 CSS 类别。

```
var oLi = document.getElementById('active');
console.log(oLi.className);
oLi.className += ' jsdemo';
console.log(oLi.className);
```

使用 className 属性如果要追加 CSS 样式，而不是覆盖 CSS 样式的话，使用“+=”连接运算符，另外注意不同的样式名称使用空格分割。

5.6　事件

事件是 JavaScript 和 DOM 之间进行交互的桥梁，当某个事件发生时，通过它的处理函数执行相应的 JavaScript 代码。对于用户而言，常用的事件有鼠标事件、HTML 事件和键盘事件。常见事件见表5-4。

表 5-4　常见事件列表

事件名	描述
onclick	点击鼠标左键时触发
onmouseover	鼠标指针移动到元素上时触发
onmouseout	鼠标指针移出元素时触发
onload	页面完全加载后触发
onblue	元素或者窗口失去焦点时触发
onfocus	元素或者窗口获得焦点时触发
onsubmit	提交表单时触发
keydown	按下键盘某个按键时触发
keyup	释放按键时触发

5.6.1　事件监听方法

通用事件监听方法

页面中的事件需要一个函数来响应，这类函数被成为事件监听函数（event listener），这些函数也被称为事件处理函数（event handler）。

可以直接在 HTML 标签中分配事件处理函数。几乎所有的 HTML 标签都支持 onclick 方法，例如：

```
<p onclick="alert('我被点击了。');">Click me.</p>
```

这种方法在主流的浏览器中兼容性很强，但并不符合 Web 标准的原则——内容、表现和行为相互分离。因此，通常采用如下例所示的方法，即在脚本中设置事件监听函数。

```
<!DOCTYPE html>
<html lang="zh">

<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>事件监听函数</title>
    <style type="text/css" media="screen">
    body {
        width: 960px;
        margin: 0 auto;
        font: 18px/1.8em '微软雅黑';
    }

    p {
        font-size: 20;
        padding: 5px 0;
        font-weight: bolder;
    }
    </style>
    <script>
    window.onload = function() {
        //获取具体的对象
        var oP = document.getElementById('myP');
        // 设置事件监听函数
        oP.onclick = function() {
            alert('我被点击了。');
        }
    }
    </script>
```

```
</head>

<body>
    <div>
        <p id="myP">请点击我。</p>
    </div>
</body>

</html>
```

对于同一个事件，这两种方法都只能添加一个函数。而 DOM 则规定了另外一种方法，可以添加多个事件监听函数。

DOM 监听事件的方法

DOM 定义了两个方法分别用以添加和删除监听函数，即 addEventListener() 和 removeEventListener()。这两个函数的语法如下：

```
[object].addEventListener("事件名称",监听函数名称,是否用于捕获阶段);
[object]removeEventListener("事件名称",监听函数名称,是否用于捕获阶段)
   ;
```

利用这两个函数，可以为 DOM 中的节点添加或者删除多个事件监听函数，如：

```
<!DOCTYPE html>
<html lang="zh">

<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>DOM事件监听函数</title>
    <style type="text/css" media="screen">
    body {
        width: 960px;
        margin: 0 auto;
        font: 18px/1.8em '微软雅黑';
    }

    p {
        font-size: 20;
        padding: 5px 0;
        font-weight: bolder;
    }
    </style>
    <script>
```

```
    // 定义全局变量
    var oP;
    window.onload = function() {
        //获取具体的对象
        oP = document.getElementById('myP');
        // 添加监听函数
        oP.addEventListener("click",fnClick1,false);
        // 可以为某个事件添加多个监听函数
        oP.addEventListener("click",fnClick2,false);
    }
    function fnClick1 () {
        alert('这是fnClick1函数');
        // oP.removeEventListener("click",fnClick2,false);
    }
    function fnClick2 () {
        alert('这是fnClick2函数');
    }
    </script>
</head>

<body>
    <div>
        <p id="myP">请点击我。</p>
    </div>
</body>

</html>
```

这两个函数中的事件名称是类似与”click”、”load” 的事件名称，而不是”onclick”、”onload” 的事件名称。第三个参数如果是冒泡阶段，则为 false，否则就是用于捕获阶段。[①]

5.6.2　事件对象

事件对象只在发生事件时才被创建，且只有事件处理函数才能访问。所有事件处理函数执行完毕后，事件对象就被销毁。事件对象，包含的信息如下：引起事件的对象；事件发生时鼠标的信息；事件发生时键盘的信息等等。

让开发人员郁闷的是，不同的浏览器，其事件对象的属性和方法不尽相同，有着较大的差异。

如何在这种添加事件方式下获取到事件对象？IE 中 event 是作为全局对象的，所

[①]浏览器中的事件模型有两种，捕获型事件和冒泡型事件。冒泡型事件指的是事件从最特定的事件目标到最不特定的事件目标逐一触发。简单地说就是从 DOM 模型的底层向顶层逐层触发。而捕获型事件刚好相反，即从不确定的目标到最精确的目标逐一触发。IE 浏览器不支持捕获型事件。

以直接使用 event 即可，如下：

```
window.event
```

而 W3C 定义的 DOM 中，是把事件对象作为事件处理函数的第一个参数传入进去，如下：

```
document.onclick = function (evt) {//事件对象只能在对应的事件处理函数内部可以访问到
alert(evt);
};
```

下面的案例，在不同的浏览器中，有着不同的输出：

```
<!DOCTYPE HTML>
<html lang="zh">

<head>
    <meta charset="utf-8">
    <title>event对象</title>
    <style type="text/css" media="screen">
    body {
        width: 960px;
        margin: 0 auto;
        font: 18px/1.8em '微软雅黑';
    }

    h1 {
        color: green;
    }

    p {
        font-size: 20;
        padding: 5px 0;
        font-weight: bolder;
    }
    </style>
</head>

<body>
    <h1>event对象</h1>
    <p>事件对象在不同的浏览器中访问的方法、其内置的属性和方法都不尽相同。比如以下代码在IE中无效：
    </p>
    <pre>
```

```
    document.onclick = function(evt) {
        for (var i = 0 in evt) {
            console.log(i + "=" + evt[i]);
        }
    };
    </pre>
    <script>
    document.onclick = function(evt) { //事件对象只能在对应的事件处理
        函数内部可以访问到
        for (var i = 0 in evt) {
            console.log(i + "=" + evt[i]);
        }
    };
    </script>
    <p>IE将event当作全局变量，当不过只有当事件发生时，event对象才有内
        容。</p>
    <pre>
    console.log(window.event); //null
    window.onload = function() {
        console.log(window.event);
    };</pre>
    <script>
    console.log(window.event); //null
    window.onload = function() {
        console.log(window.event);
    };
    </script>
</body>

</html>
```

参考文献

[1] The Mozilla Foundation. JavaScript | MDN[EB/OL]. 2015 [2015-5-26]. https://developer.mozilla.org/zh-CN/docs/Web/JavaScript.

[2] RESIG J. 精通 JavaScript[M]. 江疆, 陈贤安, 译. 北京 : 人民邮电出版社, 2008.

> 创造优美事物的方式往往不是从头做起，而是在现有成果的基础上做一些小小的调整，或者将已有的观点用比较新的方式组合起来。
>
> 节选自保罗·格雷厄姆《黑客与画家》

第 6 章　jQuery

即便掌握了 JavaScript 脚本的基本知识，用户在使用 JavaScript 开发时，需要面临一个重大挑战，那就是不同浏览器对 Web 标准的支持不尽一致，往往大部分时间和精力花在解决浏览器兼容性的问题上。另外，常见的 JavaScript 应用如果在每次内容制作中都要重写的话，工作效率将是非常低下的。为此，很多开发者提供了“库”来解决上述问题。常见的 JavaScript 库有 jQuery、Prototype、Mootools 等等，经过若干年的发展和竞争，jQuery 脱颖而出，成为使用最为广泛的 JavaScript 库。可以毫不夸张地说，jQuery 的出现改变了开发者使用 JavaScript 的习惯。

6.1　jQuery 简介

jQuery 是一个开源免费的优秀 JavaScript 库，它能使用户更加方便地处理 HTML 文档、事件等等。jQuery 由约翰·雷西格（John Resig）于 2006 年创建[1]，从最初的增强 CSS 的选择器功能，发展到现在，主要提供以下功能：

1. jQuery 大大简化了选取 DOM 局部内容的步骤，可以灵活高效地选择页面内容。
2. 引入 jQuery 后，开发人员不再需要考虑复杂的浏览器兼容问题，可以更加轻松地处理事件，提高脚本的稳定性。
3. jQuery 内置了可自定义参数的动画效果，简化了应用动画效果的过程。
4. jQuery 还提供了很多附加的功能简化了常用的 JavaScript 操作。
5. jQuery 使用最新的 CSS3 标准，即便不支持 CSS3 的浏览器，也能通过脚本达到 CSS3 的效果，极大地丰富网页的表现形式。
6. jQuery 提供了一整套 Ajax 相关操作的方法，大大方便了异步交互的开发和使用。

此外，jQuery 是一个开源软件，围绕着 jQuery 已经建立了良好的“生态系统”，用户众多，文档齐全，学习成本较低。同时还有许多成熟的插件可供选择。

学习 jQuery，最好的资源还是官方文档。此外，还有非官方的中文社区。

6.2　jQuery 的基础概念

在 jQuery 官方网站下载最新版[①]的 jQuery，将其放置在合适的目录中，然后在 HTML 文档中使用 script 元素引用 jQuery 库，即可使用 jQuery 提供的强大方便的功能。

```
<script src="js/jquery-1.11.3.min.js"></script>
```

6.2.1　jQuery 中的“$”

在 jQuery 中，最频繁使用的莫过于美元符号“$”，在 jQuery 中，美元符号“$”等同于“jQuery”，表示 jQuery 对象。jQuery 对象就是通过 jQuery 包装 DOM 对象之后产生的对象。为了方便代码的编写，通常采用“$”代替“jQuery”。“$”提供了丰富的功能，包括选择页面中的元素、作为功能函数的前缀、window.onload 的改进、创建 DOM 等等。一定要注意，在 jQuery 对象中无法直接使用原生 JavaScript 中 DOM 对象的方法和属性，但可使用 get(index) 方法将 jQuery 对象转化为 DOM 对象。

$(document).ready()

浏览器载入 HTML 文档并不是一次性全部载入的，而是从上到下依次载入。但我们的脚本可能就在文档的顶端或者中间，当脚本已经载入，但 HTML 文档还没有完全载入完毕时，脚本去处理还没有载入的 DOM 时会产生无效或者错误结果。为了解决这一问题，人们想出了一些方案：

将脚本放在页面最下面　这种方案在大多数时候是有效的，但是有的时候，创作者并没有能将脚本放在页面底部的权限或者可能性。[②]

使用 window.onload 事件　window.onload 会在页面所有元素都载入后触发，当页面中有大量图片或者视频时，触发的时机有些过晚。还有，window.onload 只能使用一次，当页面中的多个脚本调用 window.onload 时会产生冲突。

jQuery 中的 ready() 方法很好地解决了上述问题，它能够自动将其中的函数在页面加载（实际上是 DOM 准备完毕后）完成后运行。并且同一个页面可以使用多个 ready() 方法，而且互不冲突。$(document).ready() 可简写为 $()。

```
<!DOCTYPE html>
<html lang="zh">
```

[①] jQuery 的版本有两个分支，一个是支持低版本 IE 浏览器的 1.xx，另一个是不支持低版本 IE 浏览器的 2.xx。鉴于目前仍有相当可观数量的低版本 IE 用户，在大多数情况下我们选择 1.XX 的最新版。

[②] 例如，在大多数 CMS 中，普通编辑人员只能就文档的局部内容进行修改，整体文档的生成则是不同部门的合作结果。想象一下淘宝、天猫的首页、门户网站的首页等等皆是如此。

```
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>jQuery中的ready方法</title>
    <script src="js/jquery-1.11.3.min.js"></script>
    <style type="text/css" media="screen">
    body {
        width: 960px;
        margin: 0 auto;
        font: 18px/1.8em '微软雅黑';
    }

    h1 {
        color: green;
    }

    p {
        font-size: 20;
        padding: 5px 0;
    }
    </style>
    <script>
    $(document).ready(function() {
        console.log("document loaded");
    });

    window.onload = function() {
        console.log("window onload 事件触发");
    };
    // jQuery中的ready方法可以多次调用，互不冲突
    $(document).ready(function() {
        console.log("another ready");
    });
    // window.onload事件多次调用会产生冲突
    window.onload = function() {
        console.log("window onload 又一次事件触发");
    };
    // $( document ).ready()可简写为$()。
    $(function(){
      console.log("$( document ).ready()可简写为$()。");
    });
    </script>
</head>
```

```
<body>
    <iframe src="http://www.baidu.com"></iframe>
    <!-- <script src="js/jquery-1.11.3.min.js"></script> -->
    <!-- jQuery库引用位置如果放在这里，则之前的jQuery代码出错 -->
</body>

</html>
```

上面的页面运行后，我们在浏览器控制台中可以看到 $(document).ready() 方法运行的三个结果，而 window.onload 事件虽然脚本中设置了两次，但输出的结果只有一个。

选择符

在 CSS 中，选择符的作用是选择页面中的某些元素，例如，在 CSS 中，我们可以使用 id 选择符选择特定的元素，然后为其添加样式：

```
#demo {
  color:red;
}
```

虽然 CSS3 提供非常丰富的选择器，但是由于某些选择器被各个浏览器支持的情况不一样，所以很多选择器在实际 CSS 开发中很少用到。而在 jQuery 中，“$”也能使用 CSS 中的选择符，达到选择特定元素的目的。而且，jQuery 中的选择符已经为用户处理了繁琐复杂的浏览器兼容性问题。

```
<!DOCTYPE html>
<html lang="zh">

<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>jQuery中CSS选择符的用法</title>
    <script src="js/jquery-1.11.3.min.js"></script>
    <style type="text/css" media="screen">
    body {
        width: 960px;
        margin: 0 auto;
        font: 18px/1.8em '微软雅黑';
    }

    h1 {
        color: green;
    }
```

```
    .red {
        color: red;
    }

    .center{
        text-align: center;
    }
    .even {
        background-color: #eee;
    }
    </style>
    <script>
    jQuery(document).ready(function() {
        // jQuery能使用CSS选择符选择特定的元素
        $("#demo").addClass('red');
        // :contains()选择符
        $("p:contains('.contains')").addClass('red');
        // :eq()选择符
        $('p:eq(1)').addClass('center');
        $('li:gt(0)').css('backgroundColor','yellow');
        $('li:lt(1)').css('backgroundColor','yellow');
        $('div:has("p")').addClass('red');
        $(":header").css('backgroundColor','yellow');
        $("tr:even").addClass('even');
    });
    </script>
</head>

<body>
    <h1 id="demo" class="center">jQuery使用CSS选择符选择特定节点</h1>
    <p>jQuery使用CSS中的选择符，比如ID选择符，选择特定节点。</p>
    <p>jQuery还可以使用.contains()选择含有特定内容的元素。</p>
    <ul>
        <li>:eq()</li>
        <li>:gt()</li>
        <li>:lt()</li>
    </ul>
    <div>
        <p>jQuery中新增了:has()选择符</p>
    </div>
    <h2>jQuery可通过:header选择所有的标题元素</h2>
    <table>
        <tr>
            <td>单元格</td>
```

```
            <td>单元格</td>
            <td>单元格</td>
            <td>单元格</td>
            <td>单元格</td>
        </tr>
        <tr>
            <td>单元格</td>
            <td>单元格</td>
            <td>单元格</td>
            <td>单元格</td>
            <td>单元格</td>
        </tr>
        <tr>
            <td>单元格</td>
            <td>单元格</td>
            <td>单元格</td>
            <td>单元格</td>
            <td>单元格</td>
        </tr>
        <tr>
            <td>单元格</td>
            <td>单元格</td>
            <td>单元格</td>
            <td>单元格</td>
            <td>单元格</td>
        </tr>
        <tr>
            <td>单元格</td>
            <td>单元格</td>
            <td>单元格</td>
            <td>单元格</td>
            <td>单元格</td>
        </tr>
    </table>
</body>

</html>
```

从上例的运行结果我们可以看出，CSS 选择符确实可以在 jQuery 中使用。事实上 jQuery 通过预先的编程，提供了几乎所有 CSS3 标准下的选择器。

6.3　jQuery 的扩展选择符

jQuery 作为 JavaScript 库，是用来处理网页中的内容，实现特定效果的。要改变网页的内容，首先得选择网页的内容，原生 JavaScript 对 DOM 进行操作前，也得通过类似于 getElementsByTagName() 的方法得到 DOM 的某个部分。而 jQuery 在 CSS 选择符（详见4.3）基础上，又添加了一些选择符，使得 jQuery 的选择符功能非常强大。下面就介绍 CSS 选择符之外，jQuery 新增的选择符：

:animated 选择所有正在执行动画效果的元素。

[name!="value"] 选择不存在指定属性，或者指定的属性值不等于给定值的元素。

:button 选择所有按钮元素和类型为按钮的元素。

:checkbox 选择所有类型为复选框的元素。

:contains() 选择包含特定文本内容的所有元素。

:eq() 选择节点列表中的第 n 个元素（起始数字为 0），如果 n 为负值，表示从后往前。:eq() 选择符相当于 CSS 选择符中的:nth-child()、:nth-last-child()。

:even 选择所引值为偶数的元素，从 0 开始计数。

:file 选择所有类型为文件（file）的元素。

:first 选择第一个匹配的元素。

:gt() 选择节点列表中的第 n 个元素以上的所有元素，如果 n 为负值，表示从后往前。

:has() 选择至少含有一个指定元素的所有父元素。

:header 选择所有的标题元素，比如 h1、h2 等等。

:hidden 选择所有隐藏的元素。

:input 选择所有 input, textarea, select 和 button 元素.

:last 选择最后一个匹配的元素。

:lt() 选择节点列表中的第 n 个元素以上的所有元素，如果 n 为负值，表示从后往前。

:odd 选择索引值为奇数元素，从 0 开始计数。

:parent 选择所有含有子元素或者文本的父级元素。

:password 选择所有类型为密码的元素。

:radio 选择所有类型为单选框的元素。

:reset 选择所有类型为重置的元素。

:selected 获取 select 元素中所有被选中的元素。

:submit 选择所有类型为提交的元素。

:text 选择所有类型为文本的元素。

:visible 选择所有可见的元素。

除了 CSS 选择符及 jQuery 扩展的选择符之外，jQuery 还提供了一个功能强大的 filter() 方法，该方法的参数既可以是选择符，还可以使一个函数。作为 filter() 参数的函数，其返回值必须是布尔值，filter 方法会逐一在选择结果中执行函数，最终对结果集进行过滤。

```
<!DOCTYPE html>
<html lang="zh">

<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>jQuery中filter方法</title>
    <script src="js/jquery-1.11.3.min.js"></script>
    <style type="text/css" media="screen">
    body {
        width: 960px;
        margin: 0 auto;
        font: 18px/1.8em '微软雅黑';
    }

    h1 {
        color: green;
    }
    .info {
      background-color: yellow;
    }
    </style>
    <script>
    jQuery(document).ready(function() {
        // 选择指定列
      $('td:nth-child(5n+3)')
        // 将函数作为参数传给filter方法
        .filter(function() {
            console.log($(this).text());
        if($(this).text() < 21) {
                return true;
            } else {
                return false;
            }
      }).addClass('info');
    });
    </script>
</head>
```

```
<body>
    <h1>jQuery 中的 filter 方法</h1>

    <table>
        <tr>
            <th>序号</th>
            <th>姓名</th>
            <th>年龄</th>
            <th>性别</th>
            <th>专业</th>
        </tr>
        <tr>
            <td>1</td>
            <td>小明</td>
            <td>21</td>
            <td>男</td>
            <td>新闻</td>
        </tr>
        <tr>
            <td>2</td>
            <td>小冷</td>
            <td>20</td>
            <td>女</td>
            <td>广告</td>
        </tr>
        <tr>
            <td>3</td>
            <td>小宝</td>
            <td>22</td>
            <td>女</td>
            <td>数学</td>
        </tr>
        <tr>
            <td>4</td>
            <td>老盖</td>
            <td>24</td>
            <td>男</td>
            <td>软件工程  </td>
        </tr>
    </table>
</body>

</html>
```

在这个例子中，我们将一个函数作为参数，传递给 filter 方法，最终实现了将年

龄小于 21 岁的单元格标记出来的功能。

6.4　jQuery 事件处理

JavaScript 内置了很多处理用户交互行为的其他事件的方法，jQuery 增强和扩展了其事件处理的机制，并使用更加优美的语法来处理事件。

6.4.1　处理简单事件

jQuery 提供的 on() 方法，能够将事件处理函数绑定到选中元素的一个或多个事件上。要移除绑定的事件，使用.off() 方法。

on() 方法的参数如下：

events　事件名称，必需参数。（多个事件名称用空格隔开）

selector　选择符，可选参数，作用是将事件绑定到选择结果中的后代选择符中。

data　数据，可选参数，当事件触发后，将数据传递以 event.data 的形式传递到事件处理函数。

handler　处理函数，必需参数，当事件触发时执行的函数。

on() 方法相当于 DOM 中的 addEventListener()，但要比后者功能丰富。

另外，大多数事件类型可以直接利用事件名称作为绑定函数，接受参数为监听函数，例如：

```
$(#demo).click(function(){
  //事件处理
  });
```

6.4.2　事件对象

在 jQuery 中，事件对象是通过唯一的参数传递给事件监听函数的。

```
function(e){
  e.preventDefault()
}
```

对于事件对象，jQuery 最重要的工作就是替开发者解决了浏览器兼容性问题。常用的属性和方法有：

target　引起事件的元素、对象。

type　事件的名称。

preventDefault()　阻止事件的默认行为。

6.5 jQuery 与 CSS

jQuery 提供了一系列处理 CSS 样式的方法，在之前的章节中，我们已经使用过 css() 和 addClass() 方法，下面我们介绍 jQuery 中和 CSS 有关的常用方法。

addClass() 为所有选中的元素添加指定的 class 名称。如：

```
$( "p:last" ).addClass( "selected highlight" );
```

css() 获取选中的元素的指定样式值，或者设置样式。

```
var color = $( this ).css( "background-color" );
$( this ).css( "color", "red" );
```

用 css() 方法设置多个样式值时，将多个属性及其值使用对象字面量包含起来，属性和值之间用冒号":"，多个属性及其值之间用逗号"," 隔开。

```
$( this ).css({
    "background-color": "yellow",
    "font-weight": "bolder"
  });
```

hasClass() 在所有选中的元素中进行判断，看是否存在包含有指定内容的 class 的元素。例如：

```
$( "#mydiv" ).hasClass( "bar" )
```

scrollTop() 得到首个选中元素的垂直滚动条位置，或者为所有选中元素设置垂直滚动条位置。这个方法常用来设置类似"返回页首"的功能。例如：

```
$(window).scroll(function() {
 if ($(this).scrollTop() != 0) {
    $('#toTop').fadeIn();
    } else {
    $('#toTop').fadeOut();
  }
 });
```

toggleclass() 所有匹配的元素添加或者移除一个或者多个 class 类，增加、移除操作取决于当前 class 的值。例如：

```
$( this ).toggleClass( "highlight" );
```

6.6　jQuery 特效

jQuery 提供了一些为页面添加动画特效的方法，从常用的淡入淡出到复杂的动画，jQuery 都可应对。

6.6.1　显示隐藏

hide() 和 show() 方法能隐藏、显示选中的元素，相当与在这些元素的起始标签中加入了 display 样式。例如：

```
$("p:first").show();
```

没有参数的 show()、hide() 方法，会立即显示、隐藏选中的元素。hide()、show() 方法的常用参数如下：

duration　一个字符串或者数字决定动画将运行多久。字符串有"slow"、"fast"，持续时间是以毫秒为单位的，数值越大，动画越慢，不是越快。字符串"fast"' 和"slow"分别代表 200 和 600 毫秒的延时。

easing　用于确定使用的缓动函数。一个缓动函数指定用于动画进行中在不同点位的速度。在 jQuery 库中 easing 默认的是调用 swing, 如果想要在一个恒定的速度进行动画，那么调用 linear。

complete　在动画完成时执行的函数，可选参数。

toggle() 方法能自动识别状态，实现显示、隐藏效果的切换，其使用方法和 show()、hide() 方法类似。

6.6.2　淡入淡出

fadeIn() 和 fadeOut()、以及 fadeTo() 方法通过改变匹配元素的不透明度做动画效果，fadeTo() 方法不会隐藏的元素并可以指定最后的透明度值。fadeToggle() 方法能在淡入、淡出效果之间切换。

这几个方法的常用参数和 show() 方法类似：

duration　一个字符串或者数字决定动画将运行多久。字符串有"slow"、"fast"，持续时间是以毫秒为单位的，数值越大，动画越慢，不是越快。字符串"fast"' 和"slow"分别代表 200 和 600 毫秒的延时。

easing　用于确定使用的缓动函数。一个缓动函数指定用于动画进行中在不同点位的速度。在 jQuery 库中 easing 默认的是调用 swing, 如果想要在一个恒定的速度进行动画，那么调用 linear。

complete　在动画完成时执行的函数，可选参数。

例如下面的代码将使 div 在 3000 毫秒时长内淡入，并在动画结束后，使 span 元素在 100 毫秒淡入：

```
$("div").fadeIn(3000, function() {
    $("span").fadeIn(100);
});
```

6.6.3 上下滑动效果

slideUp()、slideDown() 和 slideToggle() 方法用来实现上下滑动效果，其用法和淡入淡出、显示隐藏的方法类似。

6.6.4 自定义动画效果

.animate() 方法能根据一组 css 属性，执行自定义动画效果。这个方法的参数如下：

properties 一个 CSS 属性和值的对象, 动画将根据这组对象移动。所有用于动画的属性必须是数字的，除非另有说明；这些属性如果不是数字的将不能使用基本的 jQuery 功能。。

duration 一个字符串或者数字决定动画将运行多久。字符串有"slow"、"fast"，持续时间是以毫秒为单位的，数值越大，动画越慢，不是越快。字符串"fast" 和"slow"分别代表 200 和 600 毫秒的延时。

easing 用于确定使用的缓动函数。一个缓动函数指定用于动画进行中在不同点位的速度。在 jQuery 库中 easing 默认的是调用 swing, 如果想要在一个恒定的速度进行动画，那么调用 linear。

complete 在动画完成时执行的函数，可选参数。

例如：

```
.animate({
          width: '50%'
      }, 'slow')
      .animate({
          left: switcherWidth - paraWidth
      }, 'slow')
      .animate({
          height: '+=20px'
      }, 'slow')
      .animate({
          borderWidth: '5px'
      }, 'slow');
```

6.7 AJAX

AJAX（Asynchronous Javascript And XML），是一种创建交互式网页应用的网页开发技术[2]。通过在后台与服务器进行少量数据交换，AJAX 可以使网页实现异步更新，这意味着可以在不重新加载整个网页的情况下，对网页的某部分内容进行更新。Ajax 的核心是 JavaScript 对象 XMLHttpRequest。该对象在 Internet Explorer 5 中首次引入，它是一种支持异步请求的技术。简而言之，XMLHttpRequest 可以使用 JavaScript 向服务器提出请求并处理响应，而不用刷新页面，从而极大地提高了用户体验。

Ajax 并非一项全新的技术，它实际上是几项成熟技术的组合：

- 使用 HTML 和 CSS 标记及显示内容；
- 使用 DOM 动态显示和交互；
- 使用 XML 和 XSLT 交换和控制数据；
- 使用 XMLHttpRequest 对象异步取回数据；
- 使用 JavaScript 将上面几项内容粘合在一起。

6.7.1 获取异步数据

6.8 jQuery 案例

6.8.1 回到顶部

原理：scroll() 事件触发后，判断位置，根据位置显示或隐藏按钮，点击后通过设置 scrollTop() 方法达到返回顶部的效果。

```
$(function() {
    var b = $('#toTop');
    $(window).scroll(function() {
        100 < $(this).scrollTop() ? b.fadeIn() : b.fadeOut()
    });
    b.click(function(e) {
            e.preventDefault();
            $("body, html").animate({
                scrollTop: 0
            }, 1000);
        })
});
```

其中用到了 window 对象的 scroll 事件，另外有一个小细节，当图片被点击后，使用事件对象的 preventDefault() 方法阻止了超级链接的默认行为。

6.8.2　图片轮播

原理：先将图片通过改变定位方式隐藏起来，然后通过 setInterval() 方法改变 z-index 属性，使之显示出来。

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="utf-8">
    <title>使用jQuery实现图片切换</title>
    <script src="js/jquery-1.11.3.min.js"></script>
    <style>
    body {
        width: 960px;
        margin: 0 auto;
        font: 18px/1.8em '微软雅黑';
    }

    h1 {
        color: green;
    }

    #slide {
        height: 300px;
        width: 960px;
    }

    #slide div {
        position: absolute;
        z-index: 0;
    }

    #slide div.previous {
        z-index: 1;
    }

    #slide div.current {
        z-index: 2;
    }
    </style>
</head>

<body>
    <h1>使用jQuery实现图片切换</h1>
```

```
    <p>
        原理：使用setIntervel()方法每隔一段时间，改变图片的z-index属
            性，从而实现图片的轮播。
    </p>
    <div id="slide">
        <div class="current"><img src="images/Paris.jpg"></div>
        <div><img src="images/Paris-2.jpg"></div>
        <div><img src="images/Hong-Kong.jpg"></div>
        <div><img src="images/Hong-Kong-2.jpg"></div>
        <div><img src="images/London.jpg"></div>
        <div><img src="images/London-2.jpg"></div>
        <div><img src="images/San-Francisco.jpg"></div>
        <div><img src="images/San-Francisco-2.jpg"></div>
    </div>
    <script>
    $(function() {
        setInterval(rotateImages, 3000);
    });

    function rotateImages() {
        var oCurPhoto = $('#slide div.current');
        var oNxtPhoto = oCurPhoto.next();
        if (oNxtPhoto.length == 0)
            oNxtPhoto = $('#slide div:first');

        oCurPhoto.removeClass('current').addClass('previous');
        oNxtPhoto.css({
                opacity: 0.0
            }).addClass('current')
            .animate({
                    opacity: 1.0
                }, 1000,
                function() {
                    oCurPhoto.removeClass('previous');
                });
    };
    </script>
</body>

</html>
```

6.8.3　全选与取消全选

在某些情况下，我们需要快速选择所有条目或者取消选择，利用 jQuery 提供的 each() 方法和 prop() 方法，可以方便得达到以上目的，先利用 each() 方法为符合条件的元素快速逐一增添方法，而利用 prop() 方法或者 attr() 方法获取或设置元素的属性。

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="utf-8">
    <title>使用jQuery实现全选、反选</title>
    <script src="js/jquery-1.11.3.min.js"></script>
    <style>
    body {
        width: 960px;
        margin: 0 auto;
        font: 18px/1.8em '微软雅黑';
    }

    h1 {
        color: green;
    }
    </style>
</head>

<body>
    <h1>使用jQuery实现全部选择与反选</h1>
    <p>
        原理：使用each()方法为众多同类型元素添加方法，使用prop()方法
            获取并设置属性。
    </p>
    <p>
        <input type="checkbox" name="gid[]" value="1" />1
        <input type="checkbox" name="gid[]" value="2" />2
        <input type="checkbox" name="gid[]" value="3" />3
        <input type="checkbox" name="gid[]" value="4" />4
        <input type="checkbox" name="gid[]" value="5" />5
    </p>
    <input type="checkbox" name="chk_all" id="chk_all" />全选/取消全
        选
    <script>
    $(function() {
        // 为按钮添加单击事件处理函数
```

```
        $('#chk_all').click(function() {
            // 为每个input添加函数
            $('p input').each(function() {
                // 将按钮的值传递给checkbox
                $(this).prop('checked', $('#chk_all').prop("checked")
                    );
            });
        });
    });
    </script>
</body>

</html>
```

在这个案例中，如果我们使用 attr() 方法，会发现脚本运行结果与预想的不一致，为了理解运行结果，我们有必要深入了解 attr() 方法和 prop() 方法之间的区别。[3]attributes 和 properties 之间的差异在特定情况下是很重要。从 jQuery 1.6 开始，.prop() 方法方法返回 property 的值，而.attr() 方法返回 attributes 的值。

例如 HTML 标记中定义的以下 DOM 元素

```
<input type="checkbox" checked="checked" />
```

假设它是一个 JavaScript 变量命名的 elem，input 元素的 checked 特性（attribute）值不会因为复选框的状态而改变，而 checked 属性（property）会因为复选框的状态而改变。自定义的属性对于 attribute 和 property 是互不干扰的，但是对于 DOM 自带的属性就共享了，因此，跨浏览器兼容的方法来确定一个复选框是否被选中，是使用该属性（property）。

表 6-1 .prop() 和.attr() 运行结果对比

elem.checked	true (Boolean) 将随着复选框状态的改变而改变
$(elem).prop("checked")	true (Boolean) 将随着复选框状态的改变而改变
elem.getAttribute("checked")	"checked" (String) 复选框的初始状态; 不会改变
$(elem).attr("checked") (1.6)	"checked" (String) 复选框的初始状态; 不会改变

参考文献

[1] The jQuery Foundation. History | jQuery Foundation[EB/OL]. 2015 [2015-6-7]. https://jquery.org/history/.

[2] Jesse James Garrett. Ajax: A New Approach to Web Applications[EB/OL]. 2005 [2015/11/23]. https://courses.cs.washington.edu/courses/cse490h/07sp/readings/ajax_adaptive_path.pdf.

[3] The jQuery Foundation. .prop() | jQuery API Documentation[EB/OL]. 2015 [2015-11-15]. http://api.jquery.com/prop/.

第 7 章　Bootstrap 框架

7.1　bootstrap 框架简介

参考文献

附录 A　Git 简明教程

本教程的首要目的，是使读者能用 Git 这个最流行的分布式版本控制系统管理自己的项目。本教程的另外一个目的，是让选修《PHP 网站开发》课程的同学能通过 Git 工具轻松获取课程的相关文件（代码、教案等）。

本教程并不能使你成为 Git 工具的专家，很多高级命令并不涉及，如果想进一步学习，请阅读《Pro Git》[1]。

A.1　为什么要用 Git

A.1.1　什么是版本控制

什么是版本控制？我为什么要关心它呢？版本控制是一种记录一个或若干文件内容变化，以便将来查阅特定版本修订情况的系统。在本书所展示的例子中，我们仅对保存着软件源代码的文本文件作版本控制管理，但实际上，你可以对任何类型的文件进行版本控制。

如果你是位图形或网页设计师，可能会需要保存某一幅图片或页面布局文件的所有修订版本（这或许是你非常渴望拥有的功能）。采用版本控制系统（VCS）是个明智的选择。有了它你就可以将某个文件回溯到之前的状态，甚至将整个项目都回退到过去某个时间点的状态。你可以比较文件的变化细节，查出最后是谁修改了哪个地方，从而找出导致怪异问题出现的原因，又是谁在何时报告了某个功能缺陷等等。

使用版本控制系统通常还意味着，就算你乱来一气把整个项目中的文件改的改删的删，你也照样可以轻松恢复到原先的样子。但额外增加的工作量却微乎其微。

A.1.2　三种类型的版本控制系统

本地版本控制系统

为了解决上面提到的问题，人们很久以前就开发了许多种本地版本控制系统，大多都是采用某种简单的数据库来记录文件的历次更新差异。

其中最流行的一种叫做 rcs，现今许多计算机系统上都还看得到它的踪影。甚至在流行的 Mac OS X 系统上安装了开发者工具包之后，也可以使用 rcs 命令。它的工作原理基本上就是保存并管理文件补丁（patch）。文件补丁是一种特定格式的文本文

件，记录着对应文件修订前后的内容变化。所以，根据每次修订后的补丁，rcs 可以通过不断打补丁，计算出各个版本的文件内容。

集中化的版本控制系统

接下来人们又遇到一个问题，如何让在不同系统上的开发者协同工作？于是，集中化的版本控制系统（Centralized Version Control Systems，简称 CVCS）应运而生。这类系统，诸如 CVS，Subversion 以及 Perforce 等，都有一个单一的集中管理的服务器，保存所有文件的修订版本，而协同工作的人们都通过客户端连到这台服务器，取出最新的文件或者提交更新。多年以来，这已成为版本控制系统的标准做法。

事分两面，有好有坏。这么做最显而易见的缺点是中央服务器的单点故障。如果宕机一小时，那么在这一小时内，谁都无法提交更新，也就无法协同工作。要是中央服务器的磁盘发生故障，碰巧没做备份，或者备份不够及时，就会有丢失数据的风险。最坏的情况是彻底丢失整个项目的所有历史更改记录，而被客户端偶然提取出来的保存在本地的某些快照数据就成了恢复数据的希望。但这样的话依然是个问题，你不能保证所有的数据都已经有人事先完整提取出来过。本地版本控制系统也存在类似问题，只要整个项目的历史记录被保存在单一位置，就有丢失所有历史更新记录的风险。

分布式版本控制系统

于是分布式版本控制系统（Distributed Version Control System，简称 DVCS）面世了。在这类系统中，像 Git，Mercurial，Bazaar 以及 Darcs 等，客户端并不只提取最新版本的文件快照，而是把代码仓库完整地镜像下来。这么一来，任何一处协同工作用的服务器发生故障，事后都可以用任何一个镜像出来的本地仓库恢复。因为每一次的提取操作，实际上都是一次对代码仓库的完整备份。

A.1.3　Git 简史

同生活中的许多伟大事件一样，Git 诞生于一个极富纷争大举创新的年代。Linux 内核开源项目有着为数众广的参与者。绝大多数的 Linux 内核维护工作都花在了提交补丁和保存归档的繁琐事务上（1991 − 2002 年间）。到 2002 年，整个项目组开始启用分布式版本控制系统 BitKeeper 来管理和维护代码。

到了 2005 年，开发 BitKeeper 的商业公司同 Linux 内核开源社区的合作关系结束，他们收回了免费使用 BitKeeper 的权力。这就迫使 Linux 开源社区（特别是 Linux 的缔造者 Linus Torvalds）不得不吸取教训，只有开发一套属于自己的版本控制系统才不至于重蹈覆辙。他们对新的系统制订了若干目标：

- 速度
- 简单的设计
- 对非线性开发模式的强力支持（允许上千个并行开发的分支）

- 完全分布式
- 有能力高效管理类似 Linux 内核一样的超大规模项目（速度和数据量）

于是 Linus 花了两周时间自己用 C 写了一个分布式版本控制系统，这就是 Git。一个月之内，Linux 系统的源码已经由 Git 管理了！什么是大牛？大家可以体会一下。[2]

自诞生于 2005 年以来，Git 日臻成熟完善，在高度易用的同时，仍然保留着初期设定的目标。它的速度飞快，极其适合管理大项目，它还有着令人难以置信的非线性分支管理系统，可以应付各种复杂的项目开发需求。

Git 迅速成为最流行的分布式版本控制系统，尤其是 2008 年，GitHub 网站上线了，它为开源项目免费提供 Git 存储，无数开源项目开始迁移至 GitHub，包括 Linux[①]，jQuery，PHP，Ruby 等等。

历史就是这么偶然，如果不是当年 BitMover 公司威胁 Linux 社区，可能现在我们就没有免费而超级好用的 Git 了。

A.2　安装 Git

Mac OS 最近的版本中，已经内置了 git 工具，无需安装。

Windows 上安装 Git 非常简单，可以到 GitHub 的 msysGit 项目下载安装文件。完成安装之后，就可以使用命令行的 Git 工具（已经自带了 ssh 客户端）了，另外还有一个图形界面的 Git 项目管理工具。

A.3　用 Git 获取代码

获取已有项目的内容，分为两种情况，即首次获取和更新数据。

A.3.1　首次获取代码

第一次获取已有项目数据的并不复杂，首先在想要保存项目代码的文件夹上按右键，选择 Git Bash，进入 Git 命令行。见图A-1：

然后在命令行中执行以下命令，即可获得托管在远程网站中的项目文件（包括代码和版本仓库），如：

```
git clone https://git.oschina.net/yangjh/LearningPHP.git
```

其中第二个参数（https://git.oschina.net/yangjh/LearningPHP.git）就是公布在托管网站中的项目地址。上述命令执行完后，用户将得到《PHP 网站开发》公选课的所有项目文件。

[①] https://github.com/torvalds/linux，由大神 Linux Torvalds 管理。

图 A-1　利用右键进入 Git Bash

A.3.2　获取远程仓库的更新代码

在完成了对项目仓库的首次克隆后，想要获取项目的最新代码，只需在 Git Bash 命令行中执行如下操作：

```
git pull
```

A.4　用 Git 管理自己的项目代码

A.4.1　设置 Git

使用 Git 工作之前，我们需要做个一次性的配置，这样 Git 就能跟踪到谁做了修改：

```
git config --global user.name "your_username"
git config --global user.email "your_email@domain.com"
```

还需要设定推送（push）的默认值：

```
git config --global push.default simple
```

A.4.2　首次建立 Git 仓库

使用 Git 管理自己的项目代码，也分为两种情况，首次建立仓库和提交更新内容。

1. 初始化本地项目。在相应项目的文件夹上按右键，选择 Git Init here。或者进入 Git Bash 命令行中执行

   ```
   git init
   ```

2. 在项目文件夹上按右键，选择 Git add all files now，将文件夹中的所有文件添加到本地仓库中。或者进入 Git Bash 命令行中执行

```
git add .
```

3. 在项目文件夹上按右键，选择 Git commit tool，在随后弹出的对话框中输入更新说明，提交。或者进入 Git Bash 命令行中执行

```
git commit -m ".....更新说明"
```

4. 如果想对本地仓库建立远程的镜像，即将本地仓库托管到 github 之类的站点，应先到此类站点注册帐号，建立远程仓库，然后执行如下命令建立本地仓库和远程仓库的对应关系：

```
git add remote origin https://仓库地址
```

5. 推送本地仓库数据到远程仓库，执行如下命令：

```
git push -u origin master
```

然后输入用户名和密码即可完成推送。

A.4.3　提交更新内容

完成上述操作后，以后的工作流程就更简洁了：

1. 编写或者修改代码，保存文件。
2. 添加更新文件到本地仓库，选择 Git add all files now 即可。或者进入 Git Bash 命令行中执行

```
git add .
```

3. 选择 Git commit tool 为这些更新添加说明注释。或者进入 Git Bash 命令行中执行

```
git commit -m ".....更新说明"
```

4. 执行 Git push 将本地仓库同步到远程仓库。

至此，我们已经能用 Git 工具来管理和追踪项目的变化了，enjoy Git!

A.5　使用 GitHub Pages 建立个人站点

GitHub 不仅能托管代码，还可以通过 GitHub Pages 工具免费建立静态站点，非常适合前端开发人员学习、展示作品。具体步骤如下：

1. 在 Github 站点注册帐号，邮箱验证激活。

2. 创建仓库。在GitHub注册登陆后，创建以自己用户名开头的“username.github.io”的公开仓库，其中 username 必须和 GitHub 注册时的用户名一致，否则无法使用 Github page 服务。
3. 克隆仓库。进入到想要存储项目的文件夹，执行如下命令克隆仓库：

```
git clone https://github.com/username/username.github.io.git
```

   输入用户名、密码，完成克隆操作后，应该在文件夹生成“username.github.io”的子文件夹，之后个人站点的文件和操作都在该文件夹中完成。
4. 创建首页。进入本地“username.github.io”文件夹，使用编辑器创建 index.html，这个文件将是个人站点的首页。
5. 提交代码到 GitHub。完成页面编辑后，就可以发布代码到 GitHub：

```
git add --all
git commit -m "Initial commit"
git push -u origin master
```

   如果是首次运行 git 工具，还要进行全局性用户名和邮箱的声明：

```
git config --global user.email "username@mail.com"
git config --global user.name "username"
```

   上述操作完成后，本地仓库中的代码将推送到 GitHub 远程仓库。
6. 浏览站点。启动浏览器，访问如下地址，即可浏览站点：

```
http://username.github.io
```

A.6　Git 进阶

初学者可先不阅读这一节内容，在有需求的时候再深入学习。

A.6.1　忽略项目中的特定文件

在项目中，总会有一些特定的文件不想采用 Git 工具进行版本的控制，如临时文件、编译时产生的过渡文件或包含帐号信息的文件，对于这类文件，Git 提供了一个非常高效灵活的方式进行屏蔽，即创建一个.gitignore 文件。

在这个文件中，项目拥有者只需将不想进入版本仓库的文件列举出来即可，支持通配符。例如：

```
*.sublime-project
*.sublime-workspace
*.bak
*.dump
```

```
.gz(busy)
test/*
```

需要提醒的是，当.gitignore 文件更改后，并不能立即起效，需要进行如下操作：

1. 清除缓存。命令为：

```
git rm -r --cached .
```

2. 添加文件到仓库。

```
git add .
```

之后就可添加说明、推送。

参考文献

[1] SCOTT C. Pro Git[M/OL]. 1 edition. Berkeley, CA : New York : Apress, 2009. http://git-scm.com/book/zh.

[2] 廖雪峰. Git 教程 [EB/OL]. 2014 [2014-10-11]. http://www.liaoxuefeng.com/wiki/001373951630592960dd18361248578c67b8067c8c017b000.

工欲善其事，必先利其器。

节选自——孔子《论语·卫灵公》

附录 B　Sublime Text 编辑器

B.1　为什么选择 Sublime Text 编辑器

作为开发人员，选择一款理想的文本编辑器，有助于工作效率的提高。实际上，有很多出色的文本编辑器供我们选择，比如 Vim、Emacs、WebStorm、Dreamweaver、Notepad++、Sublime Text 等等。我们之所以选择 Sublime Text 编辑器作为开发工具，主要是基于如下优点：

学习成本低 Vim 和 Emacs 虽然功能强大，但其具有较为陡峭的学习曲线，而 Sublime Text 作为一款现代编辑器，更注重用户体验，基本做到了“开箱即用”。

跨平台 Sublime Text 均为跨平台编辑器（在 Linux、OS X 和 Windows 下均可使用）。为了减少重复学习，使用一个跨平台的编辑器是很有必要的。[1]

界面优美 与大多数编辑器只注重满足功能性需求不同，Sublime Text 编辑器的界面非常简洁，尤其是选择免打扰模式后，开发人员可集中注意力于工作本身，不被系统中的其他软件信息干扰。[2] 还有，其特有的文件迷你地图可让编辑人员直观地了解到自己在文本中的位置。

免费 与商业软件 WebStorm、Dreamweaver 相比，Sublime Text 编辑器可免费使用。Sublime Text 虽然不是开源免费软件，但可免费使用全部功能，只是偶尔会在保存文件时提示购买信息。

轻便快速 Sublime Text 编辑器安装包非常小，只有区区几兆（而其它的商业开发工具往往在 1G 左右，非常笨重），甚至可以放在 U 盘中使用；还有，在 Sublime Text 中，可使用“转到任何”功能，快捷键为 Ctrl + P 实现文件的快速打开和跳转，提高工作效率；另外，Sublime Text 编辑器打开文件时速度很快，尤其是文件比较大时。

扩展性良好 Sublime Text 编辑器吸收了其它编辑器的优点，用户可对其进行定制和扩展，因此，涌现出能各种需求的扩展程序包，正是这些扩展程序包，使得 Sublime Text 编辑器和其它编辑器相比有了更多的吸引力。除由他人开发的控制包外，开发人员还可以自定义代码段，能够按照自己的需求自动完成代码，可大大提高编写效率。

生态系统完善 不同于其他编辑器需要用户自行寻找、自行安装扩展包的方式，

Sublime Text 编辑器的扩展包都统一在一个入口中，这使得用户能够非常方便地获取和更新扩展包，围绕着特定功能往往有多个扩展包可供选择，在https://packagecontrol.io/网站，用户可以通过对众多用户行为（使用、安装、卸载）的统计来对扩展包做出选择，借助于统一公开的插件入口和大多数人做出的选择，使得扩展包的开发存在着有序竞争，保障了 Sublime Text 编辑器的活力。

B.2　安装与配置

从 Sublime Text 官方网站http://www.sublimetext.com/3选择合适的安装版本，安装后需要进行必要的配置才能更高效地使用 Sublime Text。

B.2.1　设置字体

字体的选择，看起来无关紧要，但如果字体不合适，会给开发工作带来不必要的麻烦。字体的选择，第一个原则是近似字符的区分要清晰，比如 K 和 k，‘和’。字体选择的另外一个因素就是中文支持。结合这两点，建议选择 Consolas 或者 Ubuntu Mono 字体。选择菜单“首选项 设置-用户”，打开用户配置文件，加入字体配置：

```
"font_face": "Consolas",
```

B.2.2　安装 Package Control

Package Control 是一个非常方便的扩展包管理扩展，安装方法如下：

1. 进入https://packagecontrol.io/installation#st3，复制其中的如下内容：

```
import urllib.request,os,hashlib; h = '
   eb2297e1a458f27d836c04bb0cbaf282' + '
   d0e7a3098092775ccb37ca9d6b2e4b7d'; pf = 'Package Control.
   sublime-package'; ipp = sublime.installed_packages_path();
    urllib.request.install_opener( urllib.request.
   build_opener( urllib.request.ProxyHandler()) ); by =
   urllib.request.urlopen( 'http://packagecontrol.io/' + pf.
   replace(' ', '%20')).read(); dh = hashlib.sha256(by).
   hexdigest(); print('Error validating download (got %s
   instead of %s), please try manual install' % (dh, h)) if
   dh != h else open(os.path.join( ipp, pf), 'wb' ).write(by)
```

2. 启动 Sublime Text 编辑器后，选择“View Show Console”，打开 Sublime Text 控制台，将上述内容粘贴到控制台，回车运行；

3. 重启 Sublime Text 编辑器，完成 Package Control 的安装。

B.2.3　安装中文语言包

安装好 Package Control 后，就可以方便的安装、卸载、浏览 Sublime Text 的扩展包，我们先安装 Sublime Text 的中文语言包，具体步骤如下：

1. 选择“Preference→Package Control→Install Package”，在对话框中键入 Chinese，选择 Chinese Localizition 扩展包；
2. Package Control 会自动从网站下载并安装 Chinese Localizition 扩展包，安装完成后即可看到中文菜单。

B.3　常用组件

以下介绍一些常用、必要的扩展包：

IMESupport　解决 Sublime Text 中使用中文输入法时的光标跟随问题。

git　在 Sublime Text 中使用 git 工具。

Alignment　按照“=”对齐多行，使代码显得更加整洁、可读性更强。安装后使用的快捷键为 Ctrl + Alt + A 。

CodeIgnite Snippets　PHP 框架 CodeIgnite 的代码段，方便 CodeIgnite 代码的输入。

DocBlockr　非常便利的函数功能注释生成扩展，支持 Javascript, PHP, Coffee-Script, Actionscript, C , C++ 等等。

Emmet　快速生成 HTML、CSS 代码的扩展包。该扩展包在书写 HTML、CSS 代码时可通过缩写提高效率，具体使用方法参见`http://docs.emmet.io/`。

FTPsync　使用 ftp 方式同步本地代码与远程代码。

HTML-CSS-JS Prettify　使用 node.js（需要单独安装 node.js）美化 HTML, CSS, JavaScript 和 JSON 代码。

B.4　常用快捷键

以下为 Windows 版本常用功能对应的快捷键：

以下为 Mac OS 版本常用功能对应的快捷键：

全部保存　Ctrl+Alt+S

表 B-1　Sublime Text for windows 常用快捷键

快捷键	说明	快捷键	说明
Ctrl+Shift+D	重复行	Ctrl+Shift+K	删除行
Ctrl+K，Ctrl+K	删除到行末	Ctrl+/	添加或取消注释
Ctrl+Enter	插入空行到下一行	Ctrl+Shift+Enter	插入空行到上一行
Alt+-	返回上次编辑点	Alt+Shift+-	跳转到编辑点
Ctrl+Shift+G	为当前内容添加起始和结束标签	5>8	FALSE

B.5　常见问题

B.5.1　如何使用快速跳转功能

当项目文件数量众多时，查找文件的特定部分就变得低效且繁琐，针对这一问题，Sublime Text 提供了快速跳转命令（Go Anything，快捷键为 Ctrl + P）解决方案[①]。如：

使用 Ctrl + P 打开快速查找对话框后，输入 wista/index.html，就会打开 wista 文件夹中的 index.html 文件。

使用 Ctrl + P 打开快速查找对话框后，输入 @body，可以快速定位到当前文件的 body 标签。

也可以把上面两个操作合二为一，如 wista/index.css@body 可以快速打开 wista 目录中的 index.css 文件并找到 body 标签。

B.5.2　如何为特定功能设置快捷键

Sublime Text 中可为任何命令绑定快捷键，这也是它能吸引用户的特性之一。

1. 使用快捷键 Ctrl + ‘ 打开控制台，输入如下内容，以便查看命令名称（当然，如果你知道准确的命令名称的话，就可跳过这一步骤）：

```
sublime.log_commands(True)
```

比如，我们要为切换状态栏显示状态设定快捷键，可通过刚才的设定，看到其命令为：toggle_status_bar。

2. 点击菜单“首选项 ▯ 按键绑定-用户”，打开快捷键配置文件，将控制台反馈的命令按照自己的习惯设定快捷键即可：

```
[
  { "keys": ["ctrl+k", "ctrl+b"], "command": "
     toggle_status_bar" },
]
```

[①]该功能在打开文件夹的情况下，才能正常使用，否则，只能在已经打开的文件中进行跳转。

通过刚才的设定，当我们按下快捷键 Ctrl + K 后再按 Ctrl + B 就可以显示或隐藏状态栏。

B.6 学习资源

1. Sublime Text 官方论坛
2. Sublime Text 官方文档
3. 慕课网前端开发工具技巧介绍—Sublime 篇
4. 慕课网快乐的 sublime 编辑器

参考文献

[1] 巩朋. Sublime Text 全程指南 - Lucida[EB/OL]. 2016 [2016-1-25]. http://lucida.me/blog/sublime-text-complete-guide/.

[2] HQ Pty Ltd. Sublime Text 3 Documentation[EB/OL]. 2015 [2015-6-6]. http://www.sublimetext.com/docs/3/.

附录 C　JavaScript 保留字

以下这些是保留字，不能用于变量、函数名、过程、和对象名。[1]

C.1　JavaScript 关键字

break、case、catch、continue、default、delete、do、else、finally、for、function、if、in、instanceof、new、return、switch、this、throw、try、typeof、var、void、while、with。

C.2　ECMAScript 特性关键字

abstract、boolean、byte、char、class、const、debugger、double、enum、export、extends、final、float、goto、implements、import、int、interface、long、native、package、private、protected、public、short、static、super、synchronized、throws、transient、volatile。

C.3　Mozilla 已使用关键词

const、export 和 import。

参考文献

[1] Mozilla Foundation. Reserved Words - JavaScript | MDN[EB/OL]. 2015 [2015-5-11]. https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Reserved_words.

参考文献

[1] 丹尼尔·平克. 驱动力 [M]. 龚怡屏, 译. 北京 : 中国人民大学出版社, 2012.

[2] 李纳斯·托沃兹, 大卫·戴蒙. 乐者为王 [M]. 王秋海, 译. 北京 : 中国青年出版社, 2001.

[3] 费曼. 别闹了，费曼先生 [M]. 吴程远, 译. 北京 : 生活·读书·新知三联书店, 2005.

[4] 萧井陌. 编程入门指南 v1.4 - 萧井陌的专栏 - 知乎专栏 [EB/OL]. 2015 [2015-12-25]. http://zhuanlan.zhihu.com/xiao-jing-mo/19959253.

[5] W3C. Standards - W3C[EB/OL]. 2014 [2014-10-27]. http://www.w3.org/standards/.

[6] JEFFREY Z. 网站重构——应用 Web 标准进行设计 [M]. 傅捷, 王宗义, 祝军, 译. 北京 : 电子工业出版社, 2005.

[7] W3C. HTML5[EB/OL]. 2014 [2015-5-17]. http://www.w3.org/TR/html5/.

[8] 新华社. “万维网之父”回顾设计留遗憾 [EB/OL]. 2009 [2014-04-21]. http://www.yznews.com.cn/yzwb/html/2009-03/15/content_502176.htm.

[9] W3C. HTML5[EB/OL]. 2015 [2015-4-25]. http://www.w3.org/TR/2014/REC-html5-20141028/single-page.html.

[10] BOAG P. Semantic code: What? Why? How?[EB/OL]. 2005 [2016-04-21]. https://boagworld.com/dev/semantic-code-what-why-how/.

[11] W3C. Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification[EB/OL]. 2016 [2016-3-17]. https://www.w3.org/TR/2011/REC-CSS2-20110607/.

[12] W3C. Selectors Level 3[EB/OL]. 2011 [2015-5-8]. http://www.w3.org/TR/2011/REC-css3-selectors-20110929/.

[13] CHANG D. 他真的不是我兄弟：像素跟點大不同 [EB/OL]. 2014 [2016-04-22]. http://blog.fourdesire.com/2014/12/11/ta-zhen-de-bu-shi-wo-xiong-di-xiang-su-gen-dian-da-bu-tong/.

[14] W3C. CSS Color Module Level 3[EB/OL]. 2011 [216-04-22]. https://www.w3.org/TR/css3-color/.

[15] W3C. CSS Image Values and Replaced Content Module Level 3[EB/OL]. 2012. https://www.w3.org/TR/css3-images/.

[16] BOS B, ETEMAD E J, KEMPER B. CSS Backgrounds and Borders Module Level 3[EB/OL]. 2014 [2016-4-19]. https://www.w3.org/TR/css3-background/.

[17] The Mozilla Foundation. JavaScript | MDN[EB/OL]. 2015 [2015-5-26]. https://developer.mozilla.org/zh-CN/docs/Web/JavaScript.

[18] RESIG J. 精通 JavaScript[M]. 江疆, 陈贤安, 译. 北京 : 人民邮电出版社, 2008.

[19] The jQuery Foundation. History | jQuery Foundation[EB/OL]. 2015 [2015-6-7]. https://jquery.org/history/.

[20] Jesse James Garrett. Ajax: A New Approach to Web Applications[EB/OL]. 2005 [2015/11/23]. https://courses.cs.washington.edu/courses/cse490h/07sp/readings/ajax_adaptive_path.pdf.

[21] The jQuery Foundation. .prop() | jQuery API Documentation[EB/OL]. 2015 [2015-11-15]. http://api.jquery.com/prop/.

[22] SCOTT C. Pro Git[M/OL]. 1 edition. Berkeley, CA : New York : Apress, 2009. http://git-scm.com/book/zh.

[23] 廖雪峰. Git 教程 [EB/OL]. 2014 [2014-10-11]. http://www.liaoxuefeng.com/wiki/001373951630592960dd18361248578c67b8067c8c017b000.

[24] 巩朋. Sublime Text 全程指南 - Lucida[EB/OL]. 2016 [2016-1-25]. http://lucida.me/blog/sublime-text-complete-guide/.

[25] HQ Pty Ltd. Sublime Text 3 Documentation[EB/OL]. 2015 [2015-6-6]. http://www.sublimetext.com/docs/3/.

[26] Mozilla Foundation. Reserved Words - JavaScript | MDN[EB/OL]. 2015 [2015-5-11]. https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Reserved_words.