给 WEB 编程初学者的一点建议(ASP 版)

Version: 1.0 Released at 2004-12-24(Merry X'Mas)

By Nick Ledson

引子：

首先说一下这篇建议的诞生原因：作为一名西北工业大学[学生之友]网站的普通程序员，我热爱着这个团体，因为我在这里快乐地写着代码，构建着自己的世界。网站的程序员前辈们(感谢铁匠&宝玉)给我树立了一个个榜样。无论是技术上还是对生活的态度上，我都从他们那里受益非浅。所以在这里我将自己在WEB编程学习过程中遇到的困惑和疑问整理成文，将前辈们的解答和自己的感悟写下来。对于网站的新同事们来说，这是一笔财富，我真诚希望新手们能通过这份扫盲性质的文档少走一些弯路，仅此而已。^_^

其次声明一下：我不是一个真正意义上的高手，顶多只是一个有了一定代码经验的 WEB 程序员而已。我之所以要这么做，只是想把自己学习 WEB 编程的经验同初学者们一道分享，如果我的建议对您有所帮助，那将是我莫大的荣幸！而且这这只是篇建议并不是教程，我在这里只不过提供了一些入门学习的经验和技术思想，并不会在语言层面上纠缠过深，因为各种教程已经是汗牛充栋了。

排版说明：红色字体表示关键字，蓝色字体表示关键句。

态度第一：

兴趣，天赋，态度都是做好一件事情的必备因素。无论做什么事情，兴趣能让你有深入其中一探究竟的冲动，天赋能让你迅速掌握到关键细节之所在，而态度则能将你的技术潜力发挥到极致，最终达到庖丁解牛游刃有余的境界！

所以从一开始，你就应该端正态度——是的，我是在学习一门极具挑战性的技术！

也许你曾经受过一些影响：WEB 编程不就是做网页吗？没什么大不了的嘛！

没错，从狭义的定义上来说，WEB 编程的根本目的就在于通过网页实现使用者和程序间的交互，从而将我们定义好的功能和信息呈现给使用者。

事实上，WEB 编程涵盖的方方面面是无法严格界定的。在你使用的软件比如 QQ 中，在操作系统的 Windows Update 功能中，你都能够找到 WEB 程序的踪迹。如果你能够上网搜索一下，你会惊讶地发现 WEB 编程现在正处于一个极度膨胀的发展阶段，新技术层出不穷。WEB 由于其分布式计算和瘦客户端等优势与 Web Service，网格计算等未来计算模式具有先天的兼容性！所以说无论是过去，现在还是将来，WEB 编程技术从来没有也将一点都不会停止创新的步伐，作为一名初学者，你只能尽自己的努力紧紧跟上技术的更新！记住一点，再简单的事情只要深入下去就不简单，任何容易的事情专业起来就不容易了！要明确一点，你不仅仅只是一个做网页的，你是一个真正的程序员！

过程第二：

WEB 编程的学习过程一点都没有比其他编程技术来得轻松！

一些说在前头的技巧：任何学习过程中，动手实践都是个很好的习惯，实践一遍比你看十遍理论的收获要大得多，而且在实践中更能够发现问题。在下面提到的每一步骤中，实践都是必须的，不再复述。学会自己查手册也是必备技能，因为不可能永远有一个老师跟在你身边，为你传道授业解惑。在大部分时间

里，你都要独立面对问题，那么技术手册就成了你摸索问题解决之道所可以依靠的唯一伙伴了！又因为当前 IE 浏览器具有绝对的市场占有率，而且本文针对的又是 ASP 技术，那么最好的技术手册当数 MSDN(Microsoft Developer NetWork 微软开发者网络)了。基本上所有微软技术的开发资源都包含在里面了。为了方便 E 文不好的哥们和更具针对性，在本文档附带的 chms 目录下有一些翻译好的从 MSDN 里分离出来的 chm 帮助文件，也许对你有所帮助！(这些文档都是从网上收集而来，版权归原作者所有)
它们分别是:

Behavior55.chm	默认行为中文手册
Css20.chm	样式表中文手册
Cssfilter.chm	样式表滤镜中文手册
DHTML 手册.chm	比较全面的一份 DHTML 文档，翻译得不完整
Dom10.chm	文档对象模型中文手册
Js55.chm	javascript 中文手册
Vbs55.chm	vbscript 中文手册
ASP SDK.chm	ASP 技术微软官方开发手册
ADO 中文帮助.chm	Microsoft® ActiveX® Data Objects (ADO)的官方开发手册

上网看一些技术文章和论坛帖子也是一种很便捷的学习手段，那是一座无所不包的经验宝库！
这里推荐几个比较不错的站点:

www.csdn.net

www.51js.com

www.aspsky.net

www.blueidea.com

同时，利用好搜索引擎也许也能给你一些意外惊喜！因为初学者犯的往往是相同的错误！【本文基于你已经拥有了配置好的开发环境，这个很容易，上网搜索能找到很多这方面的文章】如果自己实在想不到问题的解决方案而且确定没有人之前已经解决过相同的问题时，你可以尝试着在论坛上提问描述你的问题和自己的看法，这时侯你的问题往往已经比较有深度了，网上的高手们也比较关注这种问题，于是一个复杂的问题就在大家的讨论中慢慢被解决！

但是如果是一碰到问题无论大小就乱问一通是没有什么效果的。

还有就是看源码，学习优秀的源码能让你更加感性地认识到程序的编写方式以及一些技巧。文档附带的 sample 目录下有我写的两个很简单例子(注释比较详尽)其中包括了基本的数据库操作，也许对你会有帮助！

如果你觉得已经看得差不多了，决定开始编写代码时，从一开始就遵守良好的编码规范是个很棒的习惯！因为你的代码不可能都由你来维护，当需要别人来维护你的代码时，良好的编码习惯对大家都有好处。本文档附带的 standard 目录下有两篇 asp 和网页的编码规范，可供参考。

至于如何选择一本入门书籍，个人认为没有很大的必要去买一大堆市面上流行的所谓“精通”，“20 天全面掌握”之类的垃圾书来看，因为 ASP 入门容易而且技术上也相对过时了，建议从图书馆就可以找到一些不错的入门书。实际上，chms 目录下的那些 sdk 文档里往往包含着入门教程，那些是最好的技术指导了。还有就是看书不要贪多，选中一本比较适合自己学习方式的认真看完估计就已经差不多了。最重要的是，看书要批判着看，敢于发现书中的错误，始终记住一点：尽信书不如无书！

个人的学习习惯不同，有的人喜欢通过看源码学习，有的人又喜欢从一本理论书籍开始，不管采用什么方式，不要偏听偏信别人的观点，请选择自己最喜欢的那一种吧！（我喜欢混合的学习过程^O^）

说了这么一大通，那么现在那么多 WEB 编程技术：ASP，JSP，PHP，ASP.NET 该如何入手呢？

首先：你必须具备 HTML【注一】和 CSS【注二】的基础，能够通过阅读源代码看懂一个网页的布局结构

并能熟练地通过手写代码来创建一个网页，这是一名 WEB 程序员所应具备的基本技能。因为 WEB 程序的执行环境就是浏览器，它通过解析 HTML，CSS 代码将一个网页文件所要表达的信息呈现出来，再华丽再复杂的一个网页也是由这些基本代码构成的。而那些所谓的 ASP，JSP，PHP，ASP.NET 等技术的最终实质都是向浏览器输出包含有你想要传达信息的 HTML 和 CSS 代码，它们的差异只不过是你用来控制那些动态生成代码的不同手段而已。(就好像你放假回家可以坐火车也可以坐飞机一样)

HOW TO：现在有很多可视化的网页制作工具，比如 DreamWeaver 和 FrontPage，通过他们可以很容易地获得 WYSISWYG(what you see is what you get 所见即所得)的网页编辑效果，你可以先用它们制作出一些基本的效果比如表格(<table>)、图片(<img src="">)、换行(
)、分段(<p>)等等，然后看这些效果的实现源码；等熟悉了这些基本标签的写法之后，你可以开始尝试着看懂一些简单页面的布局和样式定义方法。接下来就是自己手写代码创建网页了。实践，查手册，提问，相信这并不会给你造成很大的难度。在此推荐两个很优秀的编辑器 EditPlus2 和 EmEditor，语法加亮功能让你阅读和编写代码时更加方便！

其次：有了一定的 HTML 和 CSS 上的积累后，你就可以开始学习客户端动态效果。也许你常常惊叹于网上一些看上去很神奇的网页特效，其实这就是一种客户端动态效果。实现这种效果的程序就叫做客户端脚本，它利用的完全是客户端的计算能力，不需要与服务端进行任何交互。编写这种脚本的语言则是 javascript(IE 还支持 vbscript 编写客户端脚本，但 javascript 是标准的 ECMAScript 实现，建议采用它以获得更好的兼容性)； javascript 采用的语法类似于 java 语言，但是没有 java 那么严格和功能强大。对于这个你可以参考 javascript 的帮助文档，上面有很详细的介绍，如果你有一定 C 语言编程经验的话这并不难。在对 JS 语法已经有了一定的了解基础后，你就必须开始有意识地培养一些编程思想和客户端脚本的执行机制了。因为就像会写字并不意味着一定会写好文章一样，光会 JS 语法而对浏览器的脚本执行机理一窍不通是写不出那些效果的，客户端脚本的实现精髓在于 DHTML。确切地说，DHTML 只是一种制作网页的概念，实际上没有一个组织或机构推出过所谓的 DHTML 标准或技术规范之类的。DHTML 不是一种技术、标准或规范，DHTML 只是一种将目前已有的网页技术、语言标准整和运用结合起来制作网页动态视觉效果的方式！简单来说，DHTML 其实是先前提到的 HTML，CSS，CSSL【注三】加上 DOM【注四】的混合体！理解这个编程模型需要有一定的 OO(Object Oriented 面向对象)思想为基础，但是没有基础也不要紧，下面我结合自己的理解简单说一下。DHTML 里面把每一个网页元素都当作一个对象(比如整个网页文档就是一个 document 对象，浏览器窗体则对应着 window 对象)，每个对象有自己的属性、方法、事件等，这些你都可以在 MSDN 的 DHTML Reference(不管是菜鸟还是高手都离不开它，在 chms 目录下的<DHTML 手册.chm>是它的不完整翻译版本)里面查到 IE 浏览器支持的每一个对象的所有属性、方法、事件等。为了方便进一步理解，让我来举个例子吧：例如我们可能经常看到的图片切换效果——鼠标移到图片上变成了另一张图片，移出后又变了回来。看似很神奇，其实用 DHTML 来实现很容易做到——从前面的基础学习中你应该已经了解到网页中图片的 HTML 代码写法是：<img src="图片路径">，如果我们把 img 看成一个对象，那么 src 就是这个对象的一个属性，我们改变这个 img 对象的 src 属性就可以让它显示不同的图片，而这个操作是我们在 onmouseover(鼠标经过)onmouseout(鼠标移出)事件中来动态处理的！下面的代码就实现了这种动态效果：

<img src="http://campus.nwpu.edu.cn/flea.jpg"
onmouseover="this.src='http://campus.nwpu.edu.cn/literature.jpg';"
onmouseout="this.src='http://campus.nwpu.edu.cn/flea.jpg'">简单举个例子而已，希望能对你了解 DOM 有所帮助。

HOW TO：学习客户端脚本的方法也是多看，多练。如果发现在网上别人用了一些很好的特效，可以把它的网页保存下来，然后用代码编辑器仔细地研究其中的 JS 代码，认真地领会里面的思想。加上前面学习过的 HTML 和 CSS 的知识，还有一些其它语言的学习功底，掌握它是一件不难的事情的。同时要注意积累，把一些平时看到的优秀代码和想法记下来，以后做东西是经常会用到它们的。更进一步，就要敢于创

新，改进别人的特效代码甚至实现一些别人想不到的效果，这才算得上是真正掌握了客户端脚本的精髓！当你熟练掌握了 JS 和 DOM 的思想后，实现那些神奇的效果肯定是轻车熟路了！

最后的关键：如果你已经熟练掌握了前两个阶段的基本技能，我们就开始进入 WEB 编程的核心环节——服务端脚本代码的实现！在讲到具体编码之前，我想先解释一下 Web Application 的执行机理。所谓的 Web Application 是基于 Browser/Server（浏览器/服务器，简称 B/S）架构的，它基于 HTTP【注五】协议由客户端通过浏览器发出请求(Request)，服务器响应(Response)请求并根据你编写的服务端脚本代码动态生成客户端需要的网页反馈到客户端的浏览器中显示出来。这是一个基于分布式计算的(大部分计算在服务端发生——执行服务端脚本，客户端的计算能力则用于执行客户端脚本)典型的离散式异步通讯结构体系。由于 HTTP 协议无状态，无过程只显示结果的性质，决定了服务端脚本代码响应客户端请求的行为不是实时同步的，也就是说，每个页面中的脚本程序在执行时并不会实时地将执行状态反馈给客户端，而是在整个页面的程序全部执行完毕后，再将处于结果状态的生成网页代码发送到客户端。而且这种结果状态是离散而不是连续的，也就是说对于每一次请求的响应都是各自互不相干的。除非用专门的代码人为地维护着状态，上一个页面执行的结果状态不会影响到下一个页面（这个页面也可以是同一个页面）的执行。讲了这么多，可能不好理解，但是服务端代码的执行机制是迟早必须领会的，而且还要充分理解其中包含的思想！因为无论对于 ASP 还是 JSP、ASP.NET、PHP 来说，Web Application 的执行原理在本质上都是一致的！这就需要在实践中不断地摸索，直到领悟！只有这样才能算得上是真正地掌握了 WEB 编程。下面就针对 ASP【注六】来说一下如何来构建一个 Web Application。

拿 ASP 开刀：ASP 服务端脚本代码的执行机制可以用页面生存周期的概念来理解，每个以.asp 后缀结尾的网页生存周期仅仅开始于某个特定请求，结束于服务器响应完成，不同的请求承载着各自独立的页面生存周期！而响应的结果只在页面生存周期结束时才一次性发送到客户端！举个例子：你不可能在服务端代码里头调用 MsgBox 这种客户端代码来显示一个对话框，因为程序并不是说执行到 MsgBox 语句时就会立刻在客户端弹出一个对话框的，如果要实现弹出对话框的效果则必须将服务端代码和客户端代码结合起来写。比如你可以使用 Response.Write “<script>alert('对话框的内容！');</script>”语句来在客户端弹出一个对话框，但是当这个对话框弹出时服务端页面生存周期已经结束了，这条语句的执行原理等于在服务端代码执行时往要生成的发往客户端的页面代码里写入<script>alert('对话框的内容！');</script>这条客户端脚本代码，当客户端浏览器接收到生成好的响应页面时再执行这条脚本语句弹出一个客户端对话框。ASP 的执行机制并不难理解，只要多多实践，很快就能掌握！

ASP 本身并不是一种脚本语言，它只是提供了一种使镶嵌在 HTML 页面中的脚本程序得以运行的环境。编写 ASP 的脚本语言受微软官方支持的有 Vbscript 和 Jscript 两种(chms 目录下有这两种语言的 SDK 文档)，这是两种解释执行的脚本语言，语法上很接近 basic 和 java，具有弱语法，弱数据类型等特点，在使用上十分的宽松方便，上手也是一件不难的事情。（虽然脚本语言门槛很低，有很多弱点，但是还是一门不错的语言，做为一名程序员，应该想的是怎么发挥一门语言的长处，而不是成天去讨论各种语言的好坏）然而宽松的语法和规则并不意味着你就可以随心所欲地编写代码，使用未定义的变量（虽然这是可行的，但是如果你不想让自己的代码被维护的后人骂或者是两个月后连自己都看不懂的话就不要冒险这样做），实际上我一直认为给代码加上 Option Explicit 的强制定义声明是很好的编程习惯。同时 Vbscript 和 Jscript 又都是一种基于对象（object based）的语言，可以尝试着将一些业务逻辑封装成类，使得整个程序的架构更加简洁。

Web Application 的动态效果往往被体现于在线数据库的操作，ASP 则提供了对数据库访问接口的强大支持。微软为整个 windows 平台的数据访问提供了一个统一的接口即 ADO【注七】，这是一种基于 COM(Compment Object Model 组件对象模型)的数据库访问技术，它封装了以往繁杂的通过 ODBC 或者 OLEBD 数据库驱动 API 来访问各种不同格式数据库的操作，为使用者提供了一个相同的调用方法！对于初学者来说，早期基本上进行的都是一些简单的数据库应用，就是增、删、改、查数据库里面的数据，

那么一个基本的 ADO 操作流程应该可以归结为：定义一个 Connection 及其数据库连接字串 → 打开这个 Connection → 定义一个 Recordset 及其对应 SQL 语句 → 将 Connection 对象作为参数将 Recordset 打开 → 对 Recordset 进行操作(update/delete/insert/显示等) → 关闭 Recordset 并释放资源 → 关闭 Connection 并释放资源。无论对于那种数据库的操作，这种调用方法都是一致的，也就是说只要更换 Connection 连接字串，再稍微调整一下其他代码，你的程序就能兼容不同的数据库！这就是 ADO 的强大之处！

用好 ADO 的根本在于充分了解 SQL【注八】语句的编写，因为 SQL 直接操作数据库，在效率上有着得天独厚的优势，所以将一些数据逻辑尽量放在精妙编写的 SQL 语句或者存储过程【注九】上能大大简化你的脚本代码量并且极大地提高系统执行效率！学习的时候，重点要学会把一个具体的事务处理抽象成为一个数据模型，比如文章表与文章分类表的操作可以使用 inner join 来进行多表关联查询等等。Chms 目录下有一份【SQL21 日自学通.pdf】文档，希望能引导你入门；另外，Access 的帮助文档中有一个名为《Microsoft Jet SQL 参考》的文档十分不错，包含了所有 ADO 支持的 SQL 语句的写法，而对于 SQL Server 或者其他类型的数据库，它们的帮助文档应该算得上是最好最全面的教程了，一定要好好利用。同时从这时开始，就应该有意识地培养自己独立架构一个系统的能力！在这方面我的经验是：首先明确系统的设计目标与实现功能，将具体逻辑抽象成编程可实现的模型，然后根据具体事务建立数据库及相互关系模型，接下来就是设计系统工作时的事务逻辑及各个功能模块的分工实现细节，最后才进入编码测试阶段。对于一个架构优良的系统来说，它的数据流也就是数据导向渠道应该是有始有终畅通无阻的！而且各个事务处理模块的职责独立明确，模块之间保持着松耦合的关系！同时，一个优秀的系统架构师应该对这个系统的整体框架结构和所能发挥的效能了如执掌，并且能够掌控全局，能够迅速而敏锐地定位到问题的关键细节所在！这些观念都是一个优秀的 ASP 甚至其他语言的程序员所应该具备的技能！

其实数据库应用只是 ASP 最常见的一种应用，ASP 通过 COM 组件可以实现很强大的功能的（实际上数据库操作也是通过 COM 组件来实现的，只是这个 MDAC 组件已经默认包含在 windows 操作系统里而已）你可以使用其他语言自己编写符合 ASP 调用规范的 COM 组件来扩展 ASP 的功能，比如大文件上传和图片加水印等等。总之，学习一门技术不光要看它已经能做什么，而且还要去发掘还可以利用它来做什么。只有通过不断地思考和实践，才能真正领悟到编程的思想和系统架构的意识。同时还不应该让自己局限于某一种语言或者平台，而是要博览群书、触类旁通、多方尝试、努力开拓自己的知识面，把其它语言的一些优点引入到 ASP 中来，这样才能做到融汇贯通、庖丁解牛游刃有余的境界。

HOWTO:师父领进门，修行在个人。相信你现在已经基本上了解了作为一名 WEB 程序员所应该具有的基本技能了。这一阶段学习的最好方法就是参与到实际项目中去，不光要想怎么实现，还要想为什么要这么实现，有没有什么更简洁的实现方法，各种实现方法孰优孰劣，那一种更符合实际要求等等.....从这时开始基本上已经没有什么条条框框来限制你该如何如何做了，因为你已经成为你所设计的系统的主宰，你已经可以开始思考自己将来的钻研方向了。至于应该怎样才能更好地提升自己，就不是现在的我力所能及的了，因为我也在摸索着前进！但我相信你一定能够找到一条适合自己的技术发展路线，那么，就让我们互相鼓励共同努力吧！

总结与提高：在熟悉了语言层面之后，你还有不少内容要学习，比如如何设计让用户和程序具有更好的互操作性，这就要求业务逻辑的设计十分人性化，用使用者的视角来设计是个不错的方法；还有就是记住一条原则：千万不要把未经处理的错误信息直接暴露在用户的浏览器上，对所有可能出错的代码都要人为进行处理，至少要给用户一个更友好的错误提示信息；此外，程序的安全性也是不容忽视的，Internet 会把你的信息暴露在整个世界面前，保证数据库的安全就成了一个程序员义不容辞的责任，所以你必须具有一些防止数据库被下载以及防止 SQL 注入【注十】的基本常识，这里还有一条安全原则：任何由客户端提交且未经验证的数据都是不安全的数据；

WEB 编程是一种高度的多种技术混用的过程，在学习的过程中要善于总结，充分理解各种技术、平台

的优劣，进而综合应用，实现项目的需求。而且，底层的编码工作只是一种基本的技能，一个真正的Programer(注意，这里相对的是 Coder，两者有天壤之别)应该对应用架构有着熟练的掌握，对编程思想有着深刻的认识，这就超出了语言和平台的范畴，达到了真正的高手境界。当然，如果你没有几万甚至几十万行代码经验就想谈一些深刻的架构思想的话，基本上可以肯定只是纸上谈兵而已。

这就好像老谋子的《英雄》里说的：第一阶段，手中有剑，心中无剑（你只能针对某一种语言来编写实现简单的功能）；第二阶段，手中有剑，心中亦有剑（恭喜恭喜，你已经开始思考着如何让程序更有效地工作并小有所成了，但还是局限于某种语言）；第三阶段，手中无剑，心中有剑（这时你已经具有了一定的编程思想，语言的差异已经对你影响甚小；不得不承认，你已经俨然成为一名高手，能够开始发表一些很 Cool 的见解了）；第四阶段，手中无剑，心中亦无剑（虽然我觉得这个比较胡扯，但是大师级的人物所考虑的应该就不仅仅局限于实际的层面，而是以超然的哲学观开创一种新的方法论了吧）。

能力的提高是没有止境的，如果你有志于在理论研究上能有所突破的话，努力一下也未尝不可哦！

Never Stop Learning: Web 编程技术的发展是没有止境的，这也就意味着对这方面技术的学习也是没有止境！如果你已经耐心看完了这篇扫盲性质的介绍文档，我想你应该已经对 WEB 编程领域有了一个大致的了解。然而正如我一开始提到的 Web 编程技术突飞猛进，我们光跟上时代的脚步就要费很大的功夫！无论自己已经达到了什么样的一个层次，谦逊、好学、对新技术有着敏锐的嗅觉都是一个优秀程序员所应该具备的基本素质！如果你象我一样有着一个梦想——努力成为自己研究领域的专家的话，那么你所能做得就是学习、学习、再学习！

下面列举了一些比较时髦的前沿技术术语，感兴趣可以用搜索引擎详细了解一下。其中绝大部分都是关于架构思想，从这里也可以侧面反映出技术的发展方向。

XML＋XSLT （传统 HTML 使用的是结构与内容混合的排版方式，而这一种全新的取代传统 HTML 将表示与内容分离的网页排版形式）

XHTML＋CSS（从传统 HTML 到 XML 的过渡表示形式）

OOP（Object Oriented Program 面向对象编程，现在已经成为流行编程语言的不二法宝了）

MVC 三层架构（Modle View Control， Web Application 中将模型、视图、控制分离的架构）

设计模式（经典设计理论，包含 23 个模式）

Web service（现在很火的一种 WEB 编程实现形式，公共计算的雏形，或者说是一种试验品吧——我的理解）

DNA 架构（MS 的 N tier 架构）

AOP（Aspect Oriented Program 面向方面编程）

SOAP（Service Oriented Architect Program 面向服务架构编程）

MDA（Model Drivened Architect 模型驱动架构）

附录一:

名词解释:

注一：【HTML】Hyper Text Markup Language超文本标记语言；一种定义网页文档结构与内容的描述语言；详细定义请访问：http://www.w3.org/MarkUp/

注二：【CSS】Cascading Styles Sheets层叠样式表；一种定义页面呈现样式的描述语言，主要就是用来帮助美化修饰网页，例如网页的字体、元素的边框、定位、显示隐藏等，同时可以结合javascript来实现一些动态的效果；详细定义请访问：http://www.w3.org/Style/CSS/

注三：【CSSL】注意！不是 CSS，是 CSSL，它是 Clent-Side Scripting Language 的缩写，译作“客户端脚本语言”，主要有 JavaScript(JS)，VBScript(VBS)，JScript。Netscape 主要支持 JS，IE 主要支持 JS，VBS 和 JScript。

注四：【DOM】Document Object Model文档对象模型；它采取一种非常直观且一致的方式将HTML文档进行模型化处理，并借此提供访问、导航和操作页面的简易编程接口；详细定义请访问：http://www.w3.org/DOM/

注五：【HTTP】Hypertext Transfer Protocol超文本传输协议；详细定义见：http://www.w3.org/Protocols/

注六：【ASP】Microsoft Active Server Page 一套微软开发的服务器端脚本环境，ASP 内含于 IIS 3.0 以上的版本中，通过 ASP 我们可以结合 HTML 网页、ASP 指令和 ActiveX 元件建立动态、交互且高效的 WEB 服务器应用程序。有了 ASP 你就不必担心客户的浏览器是否能运行你所编写的代码，因为所有的程序都将在服务器端执行，包括所有嵌在普通 HTML 中的脚本程序。当程序执行完毕后，服务器仅将执行的结果返回给客户浏览器，这样也就减轻了客户端浏览器的负担，大大提高了交互的速度。官方定义见：chms 目录下的【ASP SDK.chm】文件

注七：【ADO】Microsoft ActiveX Data Objects 使您能够编写应用程序，通过 OLE DB 提供者访问和操作数据库服务器中的数据。它的主要优点是易于使用，速度快，内存支出低，占用磁盘空间少。ADO 支持用于建立客户端/服务器和基于 Web 的应用程序的主要功能。官方定义见 chms 目录下【ADO 中文帮助.chm】文件

注八：【SQL】Struct Query Language:结构化查询语言，是所有关系型数据库的公共语言，提供对数据库数据访问的统一方法。

注九：【Store Procedure】存储过程是 SQL 语句和可选控制流语句的预编译集合，以一个名称存储并作为一个单元处理。存储过程存储在数据库内，可由应用程序通过一个调用执行，而且允许用户声明变量、有条件执行以及其它强大的编程功能。存储过程可包含程序流、逻辑以及对数据库的查询。它们可以接受参数、输出参数、返回单个或多个结果集以及返回值。具体参见：SQL Server 联机丛书

注十：【SQL Injection】程序员在编写代码的时候，如果没有对用户输入数据的合法性进行判断，就会使应用程序存在安全隐患。用户可以提交一段数据库查询代码，根据程序返回的结果，获得某些他想得知的数据，这就是所谓的SQL Injection，即ＳＱＬ注入。详情请搜索互联网或者参照如下网址：http://www.enet.com.cn/eschool/inforcenter/A20040508307433.html。

附录二：

这里是一些我认为开发人员需要具备的素质，当然这些只是我综合了一些自己和别人的看法总结出来的东西，希望大家看了去粗取精，批评指正。

- 比较宽的知识面，并且能够时时补充自己的知识和完善自己的知识结构
- 善于综合各方面信息情报，推断出结论或者解决方案
- 多和别人交流，无论他/她的水平如何。记住，如果你有一个苹果，我有一个苹果，交换后我们还是各自只有一个苹果；如果你有一个想法，我也有一个想法，交换后我们就都有了两个想法！
- 务必耐心，特别是在调试和学习阶段，世界上没有聪明的人只有刻苦的人。
- 做开发其实就两件事情学习和改错，在这两件事情中间的开发过程其实是比较简单的
- 永远记住：代码是给人看的而不是给机器运行的，计算机在执行程序时是不怕累的，而人在看代码时是非常累的
- 良好的编码习惯，至少保证自己能够随时看懂几个月前自己所写的代码
- 尽量让你的代码能够很容易的被别人理解，写代码时想想你会不会在两个月后被别人骂
- 不要容忍自己犯重复的错误和写重复的代码
- 知道如何做比得到代码更重要
- 尽量不要做已经有很多人已经做过的事情
- 学会尊重别人的开发成果，除非不得已不要下决定全面替换以前的系统
- 珍惜自己的时间与工作成果，尽量让别人分享自己的开发成果

- 避免假设，多思考极端情况与引起错误的可能性，努力减少低级错误
- 不要相信至少是不要轻信自己为自己所做的测试结果
- 懂得坚持自己的开发思路，并懂得理解与吸收别人的思想
- 学会与其他人保持一致，但不能放弃自己的开发特色
- 懂得向你周围的人学习，不论是开发能力上的还是开发经验上的
- 懂得软件结构的的重要性，分析代码前先分析代码的结构
- 学会总结，定期总结自己所学会的知识，看看自己前段时间因为各种原因所浪费的时间
- 学会计划，尽量不要认为通过延长自己的工作时间来完成任务
- 学会正确估计自己的能力和工作时间
- 多看些杂志，没事情的时候可以看看以前看过的参考书，书中很可能有你以前没有注意到的一些细节
- 不要太早去做一些与开发无关的事情，因为学习的黄金时期是很短暂的
- 学会分析别人的系统，多看看别人所开发的系统的先进的一面。这里包括别人的开发思路，实现时所采用的结构
- 时常感到自己将被淘汰，经常埋怨自己的不努力
- 知道自己缺少什么
- 强迫自己做一些自己不喜欢做的事情，比如说写文档
- 能够有勇气重写自己所开发的系统，但当你重做时你应该采取和前一次不同的工作方式和方法，否则重做就没有必要
- 将问题简单化而不是复杂化
- 将维护时会产生的一部分问题转移到开发时或是设计时来解决
- 有时候也不妨问问自己：我不做开发时去做什么
- 坚持为自己做每周的开发计划，在事情没有做完以前不要轻易转移自己的注意力
- 尽可能多的将你所做的教给别人，因为你有一天可能不再负责这个系统
- 做事一定要有始有终，除非是有不得已的理由否则不要在工作完成前提前离开
- 不要将你为公司做的东西带走，你因该带走的是开发经验和对开发的认识而不是代码和产品
- 感到累时就休息一下，给自己放个假也无妨，毕竟人不是机器

下面是一些我自己的怪癖了，也许并不对其他人适用：

- 对解决问题始终保持着激情，解决问题后能兴奋地大喊：Yes！It's amazing!The feeling is wanderful!
- 编程是一种严肃而充满艺术的活动！
- 每隔三分钟就傻笑一次，每当心情不错时就默默地说声 Lucky！
- 永远不满足现状，一心想着超越目前的水平，向往更高的层次！
- 快乐地工作，感染周围的人。——李开复说的，我很喜欢
- 虽然你翱翔在技术的天空里，你也许并不快乐，因为人与人的比较让你不堪重负，或者更直接一些是：钱！如果你一直在考虑如何赚更多钱的话，那么，请把技术工作留给别人(比如我了^O^)吧！
- 我一直认为人是一种容器(Container)，是容纳你的魅力、能力、潜力还有灵魂的容器，这个容器的大小是可以由自己掌握的，当你意识到自己容器的大小并有意识地去扩充它，你就会变得更加有魅力、更加宽容、更加理性、同时具备常人所艳羡的能力。其实这一切谁都可以做到，但是你要将目光放在远处，追逐着一个远在地平线上的目标，还要忍受不被理解的孤独感以及令人沮丧的嘲笑！最终,当你超越了所有的一切时，你就会觉得自己的所有付出都是值得的，而你的器量和胸怀也就达到了无限的层次！

附录三：

铁匠的学习经验和观点(感谢他对我的指导和帮助)：

在学习的过程中（其它的学习事实上也是一样的）要注意的是技术的积累，每个人都是通过一步一步的积累成长起来的。同时在练习的时候，要注意转换思维，那就是把很多现实中的模型转化成为程序中可

以实现的模型，当很熟悉这些过程的时候，就可以在拿到一个项目的时候，很快地把它转化成为对应的模型，理出里面的技术难点，通过研究或是转换思路的方式来攻克它。在这里我觉得对一名程序员来说，转换思路是很重要的，经常可以看到一些问题从正面想，那个算法的复杂程度是超过人的想象能力了，但是换个角度去解决的话，经常只是几行代码就搞定的事情。

学习 WEB 编程就像《笑傲江湖》中华山派的剑宗和气宗一样的。如果一开始就用 Dreamweaver,FrontPage 来做，那是会很快就会做出漂亮又实用的页面出来，但是经过几年的实践后，水平基本上不会有很大的提高的；如果一开始就去想一些比较深的问题，也许没有别人做得快，但是假以时日，是会有很大的进步的。同时也要注意学习一些软件工程的知识，这些东西对编程也是很有帮助的。

附录四：

一些比较积极的观点，总比那些一天到晚无病呻吟叫唤着累啊不行啦薪水太少啦之类的伪程序员要好得多：

“程序员是最好的职业选择！每两年，微处理器的速度要加倍，内存容量是原来的四倍，但是我们的大脑还是原来的大小，很显然，将需要更多的程序员来补充这些。”——Anders Hejlsberg（我的启蒙 IDE－Turbo Pascal 的编写者，Delphi 的发明人，Borland 公司创始人，现任微软.NET 与 C＃语言首席架构师）

“优秀程序员不在他们的代码中放入错误。一个错误进入基本代码中一个月后，要进行修复的代价多达十倍，如果错误出现在已完成的产品中，代价将是 100 倍。”——Brad Silerbery (微软技术副总裁，曾领导开发 Windows95)

“软件开发是小组成员协调努力的结果。”——Ike Nassi(Apple 公司技术副总裁)

“优秀程序员都是渴望学习的人。”——Enrique Salem(Norton Utility 首席设计师)

“当在计算机前工作时间越来越多时,你必将成为更好的程序员!”——John Karmark（3D 引擎的顶级权威开发人员，DOOM，Quake，Wolf Cast 系列游戏的编写者）

“程序员的三大优良品质：偷懒，没有耐性和骄傲自大。”——Larry Wall(Perl 语言发明人)
偷懒和没有耐性会促使优秀的程序员无法忍受重复做同样的事情。骄傲自大定义为“从过度自豪或激情中产生的不现实的狂傲情绪”。

“每一个程序都可以精简至少一条指令，每个程序都至少包含一个错误（bug），因此可以归纳得出：每个程序都可以被精简至一条无法工作的指令。”——Ken Amold(Unix Curses 和 rogue 的发明人，著名游戏 Dungeons&Dragons（龙与地下城－RPG 游戏的开山鼻祖）编写者

参考文档：

Web编程学习经验——by 铁匠
给新同事的一封信——by 宝玉

感谢：
Tlxxp,铁匠