

写在前面

突然想把自己这几年——从让人既爱又恨的象牙塔，到后来的几年软件开发中的所看所学，教授给刚刚入门计算机的朋友们。每次和陌生人聊天，总是沉默寡言，听着一个个来自天南地北的人，神侃着他们姘妍美丑的经历，品味着各种各样的人生。席间有人敏感地察觉到了我的矜持，便关怀起我的行业来。把详情报出，一般会引来一阵叫好声，或者是客气的赞美，或者是由衷的钦佩。如果恰巧碰上一个同行，我们立刻能够以他人完全不懂得语言交流起来，尽管这些话听上去，好像也就是汉字和数字的简单组合。

如果让我在计算机编程技术入门和修理抽水马桶在难易程度上做个比较，我会十分为难。修理抽水马桶以其高度的问题的不可解决性，能够让我这二十多岁的大活人，抓耳挠腮，坐立不安长达一个星期之久。而再复杂的编程技术，只要下功夫，绝对可以在七天内入门。为了证实我所言不虚，拟将此本书，作为一个七日教材，旨在帮助即使是意志薄弱到像在下这样的人，也能轻松愉快的和计算机有一次亲密的接触。当然，根据你对计算机的认识不同，这七日的生活也许会让你颇为为难。有那么几天累到吐血，并产生焚书坑儒的冲动，而另外的几天则喝着咖啡，像小说一样浏览。在下尽力将它写到像武侠小说那样生动，却无法正确的判断每个读者的当前水平。不足之处。还请读者海涵。

前言的最后，一般是感谢父老乡亲 and 欢迎读者纠错的部分。谦虚话似乎上面已经说过了，此处如果在唠叨，就会落得和其他书籍一样俗套，请读者允许在下在此就小小的不谦虚一回。而感谢的人，却不敢不写上来。严厉的父亲和慈祥的母亲自不必说，他们在我生命中的每一刻，都给予我最大的关怀和帮助，我更是准备把此书，献给我将来的老婆大人，宋羽宋司令同志。

如果看了本书，您对我相当崇拜，那么我也厚着脸皮提供以下联系方式给您：

QQ： 1351013 **非诚勿扰** **MSN：** sandycsmenu@hotmail.com **从来不上**

手机： 13XXXXXXXXX **只向有缘者发布** **开心网加盟链接：**

<http://www.kaixin001.com/reg/?uid=1796533&usercode=e93a11fbef3cafe90368952cd373663a>

[1796533](#) **绝对欢迎骚扰，请随便骚扰**

目录

写在前面.....	1
准备工作.....	5
第一日：御剑术.....	7
1.1 计算机语言的学习步骤.....	7
1.2 那我为什么要学 ASP	9
1.2.1 话说回来，ASP 究竟是什么东西.....	10
1.2.2 服务器端和客户端	11
1.2.3 我的系统，需要做什么配置吗	13
1.2.4 第一个简单的 ASP 页面	15
第二日：万剑诀.....	20
2.1 简单的 HTML	20
2.1.1 Freedom	20
2.1.2 元素和属性	25
2.2 让页面漂亮起来.....	43
2.2.1 字体和图片	43
第三日：仙风云体术.....	49
3.1 JavaScript 是一门简单的脚本语言	49
3.1.1 输入与输出，变量与常量	50
3.1.2 判断语句与循环结构	59
3.2 JavaScript 的高级部分	69
3.2.1 基于对象的 JavaScript 语言	69

3.2.2	JavaScript 中的数组	74
3.2.3	附加内容：算法	78
第四日：醉仙望月步.....		90
4.1	粉墨登场的 VBScript	90
4.1.1	网页数据的传递	91
4.1.2	VBScript 的输入和输出	93
4.1.3	一个 VBScript 的例子.....	95
4.2	JavaScript 和 VBScript	106
4.2.1	判断语句与循环结构	107
4.2.2	数组与高级函数	108
第五日：天剑.....		111
5.1	数据库简介.....	111
5.1.1	数据库中的表	112
5.1.2	插入，选择，删除和更新	116
5.2	网页中的数据库连接.....	128
5.2.1	数据库连接	128
5.2.2	SQL 代码的执行.....	132
第六日：剑圣.....		136
6.1	网页中的计数器.....	136
6.1.1	Application 对象	136
6.1.2	Session 对象	141
6.2	高级的 ASP 语法.....	142

6.2.1 Cookies 对象	142
6.2.2 Server 端对象及 Server 端相关信息	144
第七日：剑神	148
7.1 页面刷新的问题	148
7.2 XML 的启示	153
7.3 出错处理及其应用	160
后记	166

我不是个随便的人，我随便起来不是人，谢谢大家！

准备工作

就这么开始修炼，显然是有些仓促了。有道是工欲善其事，必先利其器。就是闻名遐迩的九阴真经，也绝对在介绍招式前，有那么几页筑基功夫。我说的准备工作，其实相当的简单，弄得太复杂了，让人怀疑有欺骗读者之嫌（明明是八天，非得打肿了脸，说成七天）。要读者作的所有事情，就是回答几个问题。

1 之前是否接触过电脑，对于鼠标键盘的操作是否熟练呢。

这一点相当重要。本书面对的读者，也许有六指琴魔那种牛 B 人士，也有刚刚接触电脑的小菜鸟。当然无法一视同仁。如果你果真是除了上网 QQ 和打开浏览器看 SINA 以外，对电脑几乎一无所知，那么在下可以荣幸的通知你，你这七日之行，必将呕心沥血，来不得半点偷懒。你也必须，特别注意笔者在每个章节后的注释，它们应该能解决你大部分的疑惑。

2 你对计算机语言是否有所接触。以前接触的都是什么语言，是否明白算法是怎么一回事呢

这本来都是大学计算机系的基本内容。如果读者有幸，完全没有接触过的话，那么，这七日的学习中，做一些简单的程序练习时必要的。也就是所谓的课后作业。如果您打定主意偷懒不做，在下自然也不能对您威逼利诱，只是学习的效果，必然会大打折扣。而这本书的钱，也必然白掏了。请读者务必看在自己血汗钱的份上，全力支持在下的提议：)

3 对于 HTML 语言是否了解。网页编程一般采取什么模式呢

如果读者前两个问题已经轻松跳过，在下就有些心虚了。读者的水平太高，在下就愈发不安，面对着爱因斯坦讲解相对论，和面对小学生将微积分，两者的心情自然是不太相同。

许多做网站的人，都是采用 Dreamweaver 这个工具进行的。（没有听说过这个工具的读者不要紧，这和本书的内容无太多联系）使用 Dreamweaver 开发，多半是看重它良好的页面编辑功能，随便几个东西一拖一拉，简单的网页就立刻呈现在大家眼前。如果你不曾亲自写过 HTML 代码的话（无论是否做过网页），那么，这本书所说的 ASP，可能一开始会让你感觉到恶心和反感，那么请在家长的陪同下观看本书。

4 对于 ASP 有多少的了解？是否曾经开发过 ASP 页面

如果您仍然肯定的回答在下。则在下要考虑您是不是有砸场子的嫌疑了。不过本书仍然值得你看一看，在下所提到的内容，很多是自己的亲身体会，2 年时间编写 ASP 所总结出来的经验和技巧，它们对于您，或多或少，都会有些帮助。当然，不当之处，也需要读者批评和指正。

我不是个随便的人，我随便起来不是人，谢谢大家！

第一日：御剑术

终于扯到正轨上来。今天就开始所谓的 ASP 教学。基础是相当重要的。学习 ASP，必然要具备基本的脚本语言，HTML 语言和简单的算法水平。鉴于刚刚开始，如果就把困难的东西讲掉，读者也许会把此书放回书架，并咒骂在下的不知轻重。所以，第一日，我们会从学习简单的脚本语言和算法开始。HTML 语言则放在后面来说，因为 HTML 的语言，像规范多过像语言，它也许会混淆读者在学习脚本语言时所掌握的知识，请读者不要着急。

1.1 计算机语言的学习步骤

读者随便翻开，任何一般计算机语言的教学书。都可以发现，章节的安排次序，一定是这样的

1. 输入和输出
2. 变量和常量
3. 判断语句
4. 循环结构
5. 高级技巧

即使是在下本人认为最复杂的 C++，也逃脱不了这个必然规律。如此安排，却到底是为了什么呢。我们来看个例子

说有两军对垒，双方的将军，都是那种意气用事型，不讲究任何策略与战术，选择了辽阔的大草原进行决斗，没有兵种相克，单兵在作战能力相同。在此理想状况下，**A 组 500 人 vs B 组 400 人**，打得昏天黑地，血肉模糊。战斗结果出来，读者也猜得到，胜利自然是属于人数众多的 A 组，不会有什么以少胜多的情况出现。既然情况如此理想，A 组剩下多少人，就引起了某些无聊人士的兴趣。于是，他们本着将无聊进行到底的态度，进行了周

密的分析和计算，居然真的总结出一个公式来。当其他情况相同的时候，A 组剩下的人数，将等于----A 组人数的平方减去 B 组人数的平方，再开根号。换句话说，按照这道题目的例子，A 组将会剩下 300 人左右。这么划算的战役。A 组显然愿意一天打上几次，B 组却耗不起。

先不去管这个公式到底对还是错。 现在的要求如下，我告诉电脑，A 组的人数和 B 组的人数，要求按照刚才的公式，电脑必须告诉我，哪边能够获胜，获胜方剩下多少人。

超级简单的题目，如果把这个公式交给初中学生。解答这种类型的题目，实在是易如反掌。而现在的难题是，我们面对的，不是一个活蹦乱跳的初中学生，确是一台长相比较丑陋的电脑，

把问题拆解开来。就是 3 个步骤。

- 1 告诉电脑两组的人数
- 2 电脑计算
- 3 电脑告诉我们答案

哈，聪明的读者一定看到，前面提到的输入和输出，就是这 3 个步骤中的 1 和 2。随便拖出一个问题，都基本会涉及到输入和输出这一块，因此，必须把它放到计算机语言的最开始来说。我们再深入剖析一下，电脑计算这个步骤。

电脑处理问题，是基于人脑的结构进行设计的。因此，只要思索清楚我们会怎么做，则电脑的做法，就不言而喻。看到这种简单且弱智的题目，我们直接会在大脑里，分析这两个输入的数字（比如 500 和 400），先进行比较，找出大的数。然后大的平方减去小的平方，最后开根号，得出结果。

这个过程中。我们不断在使用的一项技能，与生俱来的，就是记忆。如果所有过程都靠心算，则我们一共需要记住，500，400，250000,160000,90000,300 这几项数字。这个记忆，就是我们能够将心算进行下去的必要前提。

电脑显然没有这么土，我们把电脑语言中，记忆这些数字的形式，叫做**用变量存取**。

这就用到了前面提到的**常量和变量**。对比 500 和 400，找出较大值放到前面作为被减数，这

就是前面说的**判断语句**。而如果你希望电脑，算完一组，再算一组，不准备让它歇着，就构成了**循环**。

一个没有什么技术含量的问题，居然牵扯到一本计算机语言书所介绍的大部分。这不能不让大多数读者有打退堂鼓的想法。不过好在在下考虑到此事，这些繁琐的过程，在下也会义无反顾地，在一天之内，全部给大家灌输完。填鸭式教育，一向是在下的最爱^_^
不过不是今天，今天的任务，主要是消除读者们对计算机的恐惧心理，程序这东西，应该像玩网络游戏那样畅快淋漓，才素王道。

1.2 那我为什么要学 ASP

是啊.....为什么要学 ASP 呢？

最简单的原因，当然是艺多不压身，多几门技术，养家糊口。

其实计算机行业，和其他所有行业一样，喜欢炒作。新技术一出来，立刻会有诸如我之流的拜金作家，开始忙乎关于某某新技术的"指南"和"入门"了。但作为读者，一定曾经迷茫过，到底学些什么东西好。每本书都是入门，那么多门可入，且每次被作者形容的相当轻松的入门，似乎都要经历九死一生，脱胎换骨的痛苦旅程，选择"可入的门"就显得比较重要了。

可以明白的说，如今网页编程，已经有愈演愈烈的趋势。只看微软不遗余力的在推出这个方面的开发工具和互联网近几年来几乎疯狂的发展，就知道此行业，起码在 5 年内，都会吃得很开。而网页的发展，也经历了从静态的 HTML 发展到动态的 ASP,ASP.NET,JSP,PHP 等等这百家争鸣的时代。就像学习生物，必须从单细胞的草履虫（你不要告诉我你不知道草履虫，你初中的生物会考怎么过的）开始一样，学习任何东西，都得有浅入深。从 HTML 起步，将是一个严谨的学习步骤。而如果直接跳跃到 ASP.NET,那么无论再怎么"入门"，"指南"，读者都会有些摸不着头的感觉。在下选择 ASP 作为话题，有这么几点考虑

- 1 HTML 的部分，其实是个油翁卖油的问题（详情请见中学教材"卖油翁"）--唯手熟耳。在下一定会较为详细的介绍，HTML 最为基本和关键的部分。然后再将读者带入 ASP 的殿堂。
- 2 ASP 是动态网页的入门初步，撇开技术层面不谈，学习它以后会对整个网页的

运行的机制比较了解。如果进一步学习其他更高层次的开发技术，也能轻松上手。

3 ASP 的功能已然十分强大，其他动态网页能实现的功能它也基本都可以实现。

4 本人不了解 JSP 或者 PHP 这些其他网页形式，赫赫。这应该是最大原因之一吧☺

综上几点，不知道算不算坚定了读者跟在下共游 ASP 殿堂的决心。如果没有坚定.....读者也不妨当休闲小说看下去，万一学到个一星半点，作为作者的在下也会很高兴。

1.2.1 话说回来，ASP 究竟是什么东西

这还用问.....当然是 Active Server Page 的缩写。

在下翻阅了好多入门教材，似乎都是这么解释的。不过个人认为，似乎和没解释一样。这就像有人问，太阳是什么东西，你回答说，“太阳？太阳就是 SUN”啊一样。属于绝对没错，但是完全不能提供任何有用信息的答复。笔者只好担起这部分重任，为迷途的羔羊们指引一条通往狼穴的道路。

ASP，实际是一种页面类型的名称，和 HTML 一样。我们在上网的时候，经常会把一个叫 Internet Explorer(简称 IE)的东西打开，那个玩意，其实是[浏览器](#) ①。聪明而善良的外国人，把上网这个过程，说成是像翻书一样的动作。因此总是听到这个页面，那个页面的说法。浏览器呢，则是一个翻译，他把由程序员编写的，枯燥而乏味的代码，翻译成生动活泼的样子展现在大家面前。为了避免出现，像现在世界上这样群雄并起，N 多语言并存的情况，人们特地制定了一些标准出来。让程序员都说一样的话，这样那个翻译，担子就会轻松很多。最近听说又来了个叫 firefox 的翻译，准备和 IE 抢饭碗。因为既然语言固定住了，翻译这种行业，当然可以人人做。HTML，就是这种较早的语言标准之一。

原来的 HTML 很简单。它的作用，其实只是为了定义一些显示的样式，比如，调用图片，改变字体颜色，调整页面布局等等。人们也非常爽快地使用着，爱好者们将页面作的花

里胡哨，都是一幅“页”不惊人死不休的样子，这就是我们所说的---静态页面。

可是，读者中也许已经有人要发飙了，截止目前，我都没有讲述，静态和动态到底说的
是什么。那么请耐心听下去。

当爱好者将页面改的夸张到无以附加之时，终于有人冷静下来，他们提出，浏览网页的
人，只能获取信息，却不能提供信息。就像一张报纸，看完就扔掉。爱好者们觉得很受伤，
他们想知道，到底谁看过这份报纸，到底谁曾经支持过他们的网站，即使留下个“**到此一
游”也好。同样的，看报纸的人们，有时也是一腔热情，想在网页上留点什么，却无从下手。
换句话说.....大家都有一个共同的想法，就是这网页缺少一个最为重要的功能---交互。

有了交互的世界，肯定是美好的。留言板这种东西。就是动态网页的拳头产品了。所以，
区分动态网页和静态网页的最简单的办法，就是这个页面里是否需要浏览页面的用户传递信
息给放置页面的服务器。当然，ASP 本身就是 HTML 发展过来的，如果你在 ASP 上一点动
态部分的代码都不写，它也就成为一个 HTML。而 HTML 上写 ASP 的动态部分代码，则完
全不会正确运行。如果仍有不明白的地方，且听下回分解。

1.2.2 服务器端和客户端

在本书中第一次出现用术语做标题，实在是为了吸引眼球的无奈之举。在下怕如果不再
适当地时候露两手，会被人认为在招摇撞骗。

服务器端和客户端，是学习 ASP 之间，一定要搞明白的观念，其影响之深远，已经到
达了骇人听闻的地步。甚至很多资深 ASP 玩家，都会在某些时候，一时大脑转不过弯来，
因为这个犯一些低级的错误出来。

这恐怕得从盘古开天辟地说起了。每次我们打开浏览器，输入 www.sina.com.cn 的时
候，浏览器就发送了一个请求给 sina 的服务器。请求里写道，“亲爱的 sina,我是来自福建
省福州市的一名电信用户，本人品行端正，无不良嗜好，喜欢美女，但并非随便的人，请你
把你的首页给我看一看，谢谢咯”。sina 接收到请求后，就把主页发送过来，用户的浏览器

接收到以后，翻译成网页，显示给用户看。

sina 发过来的信息，可能有两种。就像我们上面所说的。一种是不需要用户作任何回馈，纯粹灌输给你---即 HTML，另一种则是，它希望用户和他进行交互，用户可以注册，可以留言，---这就是动态网页。需要记住的是，ASP 是动态网页，但并非所有的动态网页都是 ASP，因为还有 JSP，PHP 那些也依然在世界舞台上努力的表演者。在交互的过程中，用户，就是所谓的客户端，而 sina 就是服务器端。

每个 ASP，如果想要实现动态功能的话，就会有**服务器端代码**。顾名思义，它当然是在服务器端运行的代码。客户端显然不能随便的控制服务器端做它不想做的事（不过有些牛 X 的客户端做到了，他们叫做黑客，就是长相比较黑的客户端）。既然客户端不能随意的下命令，而服务器端又希望接收到客户端那些有用的命令怎么办。

就像你是一个被雇佣的保姆（服务器端），主人（客户端）现在说，喂，我想见识下自由落体，你从 48 楼跳下去我观察下。我们当然宁死不从，跳下去摔死事小，满足主人如此变态的想法，违反了保姆的原则。所以，签订雇佣合同的时候，我们必须说"除了做饭，扫地，洗衣服"其他的命令，一概无视。这样既保障了自身的权利，又可以去完成主人合理的命令。没错，服务器端也是这么做的，当客户填写用户名密码提交的时候，相当于客户端发出了一个命令，处理这个命令的，就是所谓的"服务器端代码"。只有当命令在我们服务器端许可的范围里，才会发生他的作用。

需要注意的是，客户端，并不知道，他的命令发出去，你会做些什么，他也不关心。每个客户点下一个按钮后，仅仅是提交了他自己的一些信息，比如用户名或者密码，然后提交了一个"按钮点击"的命令。得到这些信息，服务器端可以想怎么样就怎么样。它甚至可以弹出"喂，你叫我注册我就让你注册，那我多没面子"的提示框。这全部取决于你的"服务器端代码"。

相应的，就有所谓的"客户端代码"。客户端代码，它也是被 sina 的程序员写好的，但它在运行的时候，就不会跟服务器端发生任何联系。简单的例子是，有的时候注册，你的用户名字母超过 10 个，就会弹出框来提示你所短用户名，那些一般都是用客户端代码写的。当然也不排除某些极端分子会有把用户名传给服务器端那边，让服务器对它进行判断的愚昧

行为。一个好的网页，应该是除非必要，否则尽量使用客户端代码，因为服务器虽然强悍，但始终资源是有限的，它不能无休止的和所有的客户端交互，它也有自己的家庭生活。为什么论坛人多的话，速度会慢，就是这个简单的道理。如果所有的代码都是**服务器端**的，则每个人的机器，就几乎没有负担，系统地开销就是一个 IE 浏览器。但是服务器那边就受不鸟了，每个用户的访问，事无巨细，都要消耗他的资源，而客户端那边都还闲着哪。仿佛一个老板（服务器端）招了一堆员工（客户端），结果老板拼了老命干活，员工都翘着腿聊天。沉着而阴险的老板，终于无法忍受发钱白养人这种近乎慈善事业的行为，于是，他责令员工（客户端）多干活，当员工为了生计都只好卖命苦干的时候，而那个榨取剩余价值的邪恶资本家，尽管仍然需要协调组织内部关系，但显然要轻松愉快很多，而整个公司的效益也较之原来提高了不少。所以，在下在此处特地强调，尽管 ASP 的服务器端代码，写起来会越来越过瘾（这几乎是必然的，因为与客户交互的乐趣，是常人无法想象的，很多人为了体现自己网页的人性化，甚至出现访问者随便动个鼠标，敲一下键盘就访问服务器数据库这种浪费资源的情况），需要时刻谨记，你**现在是那个埋头苦干的老板**，你让服务器做如此之多的事，只会让你的网站访问速度下降，用户慢慢会离你而去，到时候众叛亲离，江湖甚至放出你**命犯天煞孤星**的传言，那莫怪在下没有事先提醒你咯。

1.2.3 我的系统，需要做什么配置吗

好的，既然理解了客户端和服务端，有个浮出水面的问题，就不得不好好说一下了。都已经知道用浏览器来访问网站，上 sina,google,sohu,这几乎是一个上网者的必备素质。但现在我们不是上网索取信息这么简单，而是已经有加入茫茫的做网站大军的觉悟。那么，作为那个发布信息的服务器，我们总得做些准备工具吧。俗话说，工欲善其事，必先利其器。我们这“器”，还没有认真地介绍过呢。且听在下细细道来。

简单来说，做 ASP 页面，只要两个简单的东西-----IIS，系统的记事本程序。没有听说过记事本程序的读者，本人只能表示遗憾，并将寻找记事本程序，作为您今天的家庭作业，

希望认真完成，明天上交作业的时候，要求家长签字。这里主要是要介绍 IIS 这个可能许多读者连名字都没有见过的东东。

按照国际惯例，把 IIS 的全称写出来作为它最简单的一种解释。IIS 是 Internet Information Service。中文版的 Windows 里，它应该叫 Internet 信息服务。智慧的读者们已经注意到，所谓的国际惯例对于解释 IIS 是什么东西，几乎起不到任何帮助^③而深明大义的在下，当然应该肩负起答疑解惑的重任。

前面咋呼了半天服务器端，客户端。那么，是否想过，这两部分的代码，都由谁来负责解释呢？不要认为你随便敲一段符合 ASP 语言规范的代码，计算机都认得，计算机从来只认识二极管晶体管那些东西。你们之间想交流，需要找个良好的翻译。而在下在前面已经提过，浏览器，就是这么一个翻译，它能把代码编写者编写的代码，翻译成花哨的样子展现在用户面前。但是注意，服务器端代码是在[服务器那边](#)执行的，而浏览器，则是在访问网站的上网者这边，它——是属于客户端的。也就是说，我们还需要在服务器端那边找个翻译，它负责翻译服务器端代码，并执行它们。而这个身负重任的翻译，就是 IIS。

那么，IIS 既然要安装在服务器端那边，现在没有服务器怎么办，难道就如此放弃伟大的 ASP 直到有钱租用服务器为止么？在下请读者放心，服务器三个字远没有你想象的那么恐怖，从来没有人规定，装上 4G 内存，插上 16 个 CPU 的那种电脑，才可以命名为服务器。任何一台电脑，哪怕你能把你尘封已久的奔腾 100 抱出来，都可以作为服务器。目前所有的 ASP 代码编写者，都是将自己的机器，既做服务器端，又做客户端，反复调试自己的程序，直到满意后，才把他上传到真正需要发布给大众的那个服务器上去。因此，幸运的您，完全可以在自己的机器上安装 IIS，让自己那台也许高档也许配置较差的机器开始认识 ASP 这三个字的真正含义。^②

一般来说，安装好 Windows，都不会带有 IIS。请读者按照以下步骤操作，安装 IIS：

- 1 进入我的电脑----控制面板
- 2 选择添加删除程序
- 3 选择添加/删除 Windows 组件
- 4 在 Internet 信息服务(IIS)前打勾(英文版的应该是 Internet Information Service)

5 点击下一步安装

期间可能要求读者放入 Windows 的安装光盘。以上的办法适用于所有的 Windows 的系统。如果读者中有适用 Mac OS 或者 Linux 的，抱歉，在下对此二者连最起码的常识都欠奉。读者只能请周围的专业人士帮忙解决，对此在下深感抱歉。

那么好,安装完成后，今天的工作，将会以一个简单的 ASP 页面作为终结，原来准备头悬梁，锥刺股的读者们，可以松下一口气了☺

1.2.4 第一个简单的 ASP 页面

一. 虚拟目录

一个成熟的网站，其实看上去，就像一个磁盘分区一样，拥有自己的目录结构。理论上和我们在我的电脑里点进 C 盘后看到的那些文件夹概念上差不多。

好的网站设计者，会仔细设计并安排这个目录结构。比如 download 目录下，放的是可供下载的东西，而 database 目录则存放数据库文件等等。而这个就是**虚拟目录**这个名词的由来。我们看到 www.sina.com.cn 这个名字，并开始访问的时候，实际上，访问的应该是 SINA 服务器上的一个 ASP 文件(或者其他类型的网页文件)这个被访问到的文件路径，是由服务器的 IIS 预先设置好了-----即把硬盘上的某个真实的目录，作为 sina 这个网站的根目录，这样当用户访问 sina 的时候，就相当于去服务器硬盘上这个真实目录下访问文件。多说无益，我们实际操作一把。

- 1 打开我的电脑
- 2 进入 C 盘
- 3 新建文件夹，命名为 ABC (如果有重复的名称，请自行调整)
- 4 打开控制面板->管理工具->Internet 信息服务
- 5 见到了以下画面



其中 MENUCHEN 就是在本人的电脑名称。而读者打开的当然显示的是读者的名字。

用右键点击默认网站，选择**新建→虚拟目录**

- 6 点击下一步，为你即将建立的第一个网站目录起一个名字，(随便什么名字都好，但尽量避免使用中文，中文的问题，我会在后面的章节详细阐述)，我们先用"Test"作为例子。然后下一步
- 7 选择刚才新建的文件夹---即 C 盘上的 ABC 目录，点击下一步
- 8 不用做任何改变，一直点击下一步直到完成为止。
- 9 第一个虚拟目录就建立完成了。

可以看到，真实地目录 ABC，已经被我们命名为 Test.换句话说，如果我们通过浏览器访问自己做的网站，将不会认识"ABC"，而只知道 Test. IIS 将会将用户对 Test 的访问，转到 ABC 上。而用户对此将一无所知。

二. 第一个 ASP 页面

- 1 进入刚才新建的那个目录 ABC (你可以关闭 IIS 了, 它暂时失去了利用价值)
- 2 新建一个文本文档, 并将它命名为 Myfirstpage.asp, 记住, 扩展名③一定要改成 ASP。

- 3 将 Myfirstpage.asp 用记事本打开。

在空白的记事本里写上:

```
<%  
  
    Response.write"It' s my first page"  
  
    Response.end  
  
%>
```

注意不要漏掉<%和%>这两个东西

- 4 储存文件
- 5 打开浏览器, 在地址栏里输入 <http://localhost/test/Myfirstpage.asp>, 大小写随意, 没有关系
- 6 浏览器显示出 " It' s my first page"
- 7 至此, 第一个简单的 ASP 页面就完成了。

其中 localhost 是代表本计算机的意思, 你可以用 localhost, 也可以使用本电脑的名称来代替, 就仿佛我可以写 <http://menuchen> 一样。理论上说, 访问 <http://localhost> 和 <http://menuchen> 效果是一样的。



请用户养成输入 `http://` 的习惯, 尽管很多时候不用输入它也可以正确的访问。但岛主确实遇到过因此而产生的问题。

有些用户可能开始有点乱了，那么在下再多嘴一下帮助大家理清思路。

刚才，我们拿台可爱的电脑，同时作了服务器和客户机。当我们在浏览器里敲入以下地址：<http://localhost/test/Myfirstpage.asp> 的时候，我们就是一个普通的访问者，客户端浏览器（就是自己这台电脑的浏览器）向服务器端（还是自己这台电脑）的 IIS 发送访问的请求。服务器端接收到请求信息，从 Test 这个虚拟目录找到了服务器上的真实目录 C:\ABC，并顺藤摸瓜的寻找到了 MyFirstPage.asp 这个文件，服务器老实的将该文件翻译了过来，并把翻译结果，传回给客户端（还是自己），客户端的浏览器接收到以后，将 It's my first page 这句话显示给我们看。由于这一系列操作，其实都是在自己的机器上完成，所谓雌雄同体。这样看起来，客户端和服务端并非那么壁垒分明，似乎很容易混淆。我们需要注意到以下几件事情

- 1 客户端看到的目录是虚拟目录，它和服务端的真实目录，名字可能是不一样的。
这也就是虚拟这两个字的由来
- 2 服务器端通过 IIS 负责翻译服务器端代码，
- 3 客户端浏览器负责解析服务器端回传过来的翻译过后的代码

至此为止，第一个 ASP 页面也宣告完成。今天的课也讲到这里。在下做人厚道，暂时不留下什么课后作业，大家主要把上面这段话看明白，弄清楚服务器端和客户端两者的区别，就是这一章最重要的部分了。

① 浏览器：我们上网用的最多的，就是浏览器。经常听到的 IE，就是浏览器的一种。全称是 Internet Explorer。IE 是微软公司提供的浏览器。当然市面上也并非就只有这么一种产品，其他的例子有最近在我们这些老网虫中火热流行开来的 Firefox，以及以前红火过一阵的 Netscape，不愿意一直被微软牵着鼻子走的读者可以弄来把玩把玩。

② 一般来说，将来你做好了网站，就需要租用域名和空间。域名就是我们经常敲入的什么 www.sina.com.cn 这种东西，空间则是指用于存放你网站的服务器，租来以后，你可以通过 FTP 将自己的网站传到服务器上去，而此时的 IIS 配置工作，就不需要你来操心了。

③ 在很早的时期。文件名被定义为拥有两个部分，第一个是文件的主名字，后一个则是文件的扩展名。我们经常听到的"mp3"，就是文件扩展名的一种。一个 mp3 文件通常都是这种形式 "流着泪的你的脸.mp3"，ASP 文件的扩展名，也就是"ASP"

我不是个随便的人，我随便起来不是人，谢谢大家！

第二日：万剑诀

对于第一日的学习，不知道读者作何感受。几家欢喜几家愁似乎是无法避免的。我把第一章的初稿发给一个不太懂电脑的 MM 看时，她表示基本看不明白。在下一边吐血，一边仔细研究原因。最后的结论是，如果你毫无电脑基础，看起来吃力，几乎是肯定的，建议你先找一些网页设计的初级教程来看一看。在下这个虽然也是初级教程，但基调毕竟是已经向程序开发的方向靠拢，晦涩难懂的字眼经常出现自不必说，有些基本的电脑操作，在下也没有办法一一赘述。还请读者见谅。

今天我们来接触一下 HTML 和一些简单的 ASP 控件。从现在开始，如果不特别说明，文件都将用 ASP 作为扩展名，尽管 ASP 里面如果不写上服务器端代码，则跟静态的 HTML 是一样的，不同仅在扩展名（因为此时 IIS 根本不需要进行任何的翻译工作）。

2.1 简单的 HTML

先复习一下昨天的内容。读者按如下的步骤做：

- 1 建立虚拟目录，名称自拟，记住尽量使用英文。如果你一定要使用什么“愤怒的葡萄”这种名字，也请翻译成“AngryGrape”（**下文中将采用 test 作为目录**）
- 2 在该目录下建立文件 MyFirstPage.asp(注意此处的扩展名是 html,读者如果嫌麻烦。可以使用其它名称，本章后面都会使用 MyFirstpage 这个名称来讲述，只要能对号入座就好)
- 3 通过 <http://localhost/Myfirstpage.asp> 来观看页面。

2.1.1 Freedom

还记得有个片子，勇敢的心（港译惊世未了缘），梅尔吉普森在临死前，声嘶力竭的喊

出"Freedom",让我这个身处事外的观众也为之动容。现在的程序员,也开始渐渐脱离以前的老路子,开始玩起所谓的"自由"来。为什么说 HTML 是一种自由的语言呢?且听我慢慢道来。

首先声明的是 如果有可能的话 请安装 Ultraedit 或者 EditPlus 之类的文本编辑工具。它们会让你写代码的效率成倍提升。如果你坚持使用记事本。在下建议你不要选择记事本中的"自动换行",这样代码看起来会比较整齐和便于调试。不要问我这些工具到哪里下载,强大的工具,从来都是要收费的,"下载"这种免费的事情。最好自己想办法.....

打开 MyFristPage.asp。输入以下代码

```
<html>

<head>

<title>my first page</title>

<style>

.normal {font-family:verdana;font-size:9pt}

</style>

</head>

<body>

<table border="0" cellspacing="1"

cellpadding="1" bgcolor="#CCCCCC" class="normal">

<tr bgcolor="white">

<td colspan="3">飞行岛日用品购买列表</td>

</tr>

<tr bgcolor="#E8EEF7">

<td>牙刷</td>

<td>12 元</td>

</tr>

<tr bgcolor="white">

<td>牙膏</td>
```

```
<td>1 元</td>

</tr>

<tr bgcolor="#E8EEF7">

<td>防恐龙喷雾剂</td>

<td>1.5 元</td>

</tr>

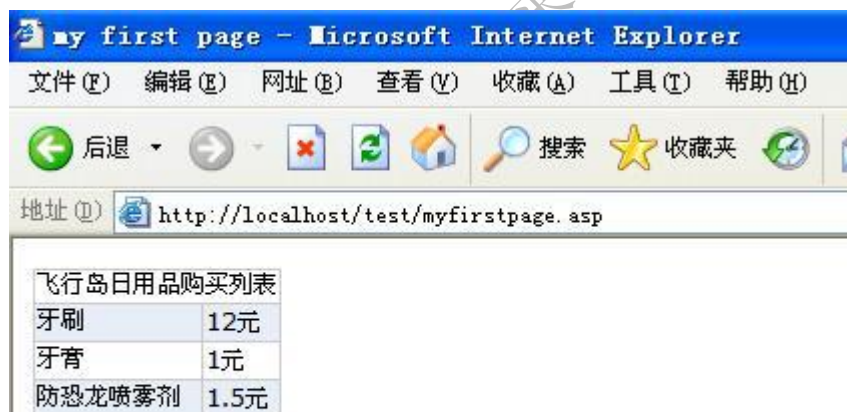
</table>

</body>

</html>
```

打开浏览器

输入//localhost/test/myfirstpage.asp,画面如下所示:



下面将详细讲述一下以上的代码。

我们从头看起：

一个标准的 HTML 文件，都会以<html>作为它的开头。这是一种约定俗成。

细心的读者可以看到，这个文件里，充斥着一种符号：<>。这个我们叫它“尖括号”，号如其名。我不准备用那些类似于定义的说明来限制读者，但有些必要的叫法，还是要用起

来，下面说明会方便很多。

再细看一步，你会注意到一个现象。每个<>中的东西，都有两个部分。比如<html>，你就可以相应的找到</html>，而<head>你也可以找到</head>，这种有始有终的做法，网友们调侃它为“开门，关门”。其中的 html,head,table,tr,td 等等，我们不妨叫它为“元素”。大部分的元素，都会用一下的格式书写：

<元素></元素>

网友们把这种现象，戏称为“开门，关门”。不过也有一些例外。某些元素，它仅仅出现一次。比如 br,input 等等。它们的书写规则更为简单

<元素/>

这种元素，我们叫它“随手关门元素”。



岛主用到的语言，诸如“随手关门”元素等等，岛主是自创或者参照其他网友的一些口语化得文章，如果读者觉得好，可以尽情的在自己的文章中使用它们，但注意，正式的文献中，请使用标准的说法。

有些读者想了解，到底哪些元素，是随手关门的，哪些是自己开门后需要在后面关门的呢？首先要声明的是，本书并非 HTML 字典，不可能将所有的元素都列出来，一一说明，那样读者也不一定能良好的记忆下来。在下这里提供一种简单的方法，因为在下自己都是如此判断的。

写成<元素></元素>这种形式的，一般是开门和关门之间，还有很多其他代码要写，比如需要显示出来的文本（上面代码中的 html,title 等），或者它会像一个容器一样，还包括着其他的元素（上面代码中的 table 就是一个这样的容器）。而以<元素/>这种形式存在的元素，则是它不需要任何额外的信息，它也不作为任何形式的容器。典型的如 br，这个

元素的功能，就是在网页中换行（功能类似于你在记事本里写东西的时候敲回车）。这样一个简单的功能，当然简单的随手关门就好，不必多余的写成 `
</br>` 这样浪费你我的时间。

为什么说，HTML 是自由的呢？自由这两个字，总是具有相对性，比如我们现在，生活在一个自由的国度里，这并不意味着，你可以烧杀抢掠，无恶不作。HTML 的自由，其实是相对于原来准备给它制定的规范而言。现在读者可以试验一下，把所有的关门都去掉（记住是所有的关门），并重新打开页面观察效果。

如果我料没错，页面应该是空空如也。这是否代表着出错呢？

并非如此。HTML 可以容忍这种现象出现。或者严格的说，IE 可以容许这种现象。要知道，IE 就是一个翻译 HTML 语言的翻译，当需要翻译的语言，本身已经不按常理出牌的时候，作为一个 Professional 的翻译，也只能去容忍它，并努力尝试着翻译它，当然结果也许就会和原来的期望值产生差距。

撰写 HTML 的时候，甚至自由到，你在 Myfirstpage.asp 里面仅仅写上 abc，然后马上察看效果，你也会看到浏览器里显示出正确的 abc 来，什么 `<html>`，什么 `<head>`，什么 `<body>`，还有等等等等的页面控制符号，都可以不要，唯一的规矩，就是没有规矩。

只有当你对页面的显示要求很高的时候，希望它确确实实的按照你想要的方式显示时，你才需要把这些规矩给规范起来。但是作为一个初学者，在下建议应该严格的按照它的规则去撰写，虽然这很痛苦，但他确实让你把 HTML 搞清楚的必经之路，等你已经完全熟悉后，是否仍然按照这些规则，则看当时的需要了。



有兴趣的读者，可以学习一下 XML，它对格式的要求非常严格，而它也是将来技术发展所需要掌握的知识。岛主推荐一本《无废话 XML》，读者可以到网络上搜寻。

2.1.2 元素和属性

一、常用的 HTML 元素

我们回到刚才的那个页面来。

页面中使用到的元素有 html,head,body,table,tr,td,Style。我们将一个一个的把它介绍一下。

html 刚才说过了，是一个约定俗成的规范。一个标准的 html 页面，需要以<html>来开头，然后</html>来结尾。当然，刚才说过了，不写也可以，而且页面也完全正常。不过千万记住，一旦开门，即写了<html>，一定要记得关门，即写</html>。否则可能会出现一些无法预料的问题。我们如果发现有个页面显示不正常，第一步就是要把这些开门和关门的部分检查清楚，看是否每个元素都正确的关闭。

head 则是算是一个预先说明的部分。就像一本书的前言，或者一篇文章开头的"写在前面的话"之类。在 head 中经常会定义页面中用到的样式文件，还有 js 文件等。Js 文件就是著名的 JavaScript，而样式文件，则是记录页面中需要用到的，包括按钮的大小阿，字体颜色等等信息，它们往往用<style>包含起来，或者另外在一个文件中存放，这在后面将会详细的说到。

读者可以注意到，head 里面还有一个叫做 **title** 的元素。随便打开一个网站，都会看到浏览器的正上方那个蓝色的部分，会显示一行字，那就是所谓的 title 了。它是这个页面的标题。

Body，则相当于一篇文章的正文部分。从这里开始，就是真正的页面内容了。除了前面说到的 html,head 或者样式定义用的 style 以外，其他的元素，基本都被装在 body 这个大容器中。

着重说一下 **table**。

网页并非一个空白的画布。我们画画的时候，除非是有点美术功底，否则基本就是喜欢东一笔，西一笔的作画，在下也是如此。所以刚开始做网页的时候，使用 Dreamweaver，

非常的不适应。因为即使想在网页的正中显示一行字这种简单的要求，实现起来都很困难。

网页设计，是需要预先排版的。Table 就是这么一个典型的排版工具。我们来看看上面用过的这段代码：

```
<table border="0" cellspacing="1"
        cellpadding="1" bgcolor="#CCCCCC" class="normal">
  <tr bgcolor="white">
    <td colspan="2">飞行岛日用品购买列表</td>
  </tr>
  <tr  bgcolor="#E8EEF7">
    <td>牙刷</td>
    <td>12 元</td>
  </tr>
  <tr  bgcolor="white">
    <td>牙膏</td>
    <td>1 元</td>
  </tr>
  <tr  bgcolor="#E8EEF7">
    <td>防恐龙喷雾剂</td>
    <td>1.5 元</td>
  </tr>
</table>
```

前面两行我们先不用考虑

从第一个 tr 开始看。所谓的 tr,是一个行的概念。一个符合规范的 table 内部（意思是 tr,td 都有正确的关门），有多少个 tr,就代表你即将画出来的表格有多少行。比如从上面这段

代码,可以看出表格一共有 4 行。而包在 tr 中的 td 则代表这一行有多少个列。

同样可以清楚地知道,代码里所写的,除了第一行外,其他每行都被分成两列。至于第一行,其中有一个重要的代码 `colspan="2"`,它的意思,就是该行的前两列,将会合并起来。读者就奇怪了。什么叫,前两列合并起来,仿佛这行已经存在了两列似的,这不是 tr 刚开始就写 td 了嘛。

没错,有个恶心的地方在于,读者如果想玩点花哨的东西,比如其中一行的两列是三七开,而另外一行的两列则四六开。那么很抱歉的通知你,如果你就写上两个 td,这是无法做到的。因为一旦定义了其中一行的列,则另外的行也会默认拥有这个列。所以。列的宽度,必然是相同的。这也就是第一行特别写一句 colspan 的原因。因为它默认拥有两个列了,尽管这两个列在下面的 tr 才被定义到,而 colspan 就是为了实现列合并而诞生的东西。因此, `colspan="1"`,就没有任何意义了。

那么,有的读者就想问了。难道想一行三七开,一行四六开,就怎么也不行了吗?放心,如果如此的死板,那绚丽多姿的网页又从何而来。有两种方法可以做到

- 1 将每一行,都分成 10 列,然后其中一行写上

```
<td colspan="4">牙刷</td>
```

```
<td colspan="6">12 元</td>
```

另外一行则是

```
<td colspan="3">牙膏</td>
```

```
<td colspan="7">1 元</td>
```

就解决问题了。

- 2 这种方法则更加灵活一些

就是一行里只放一列,但是在这一列里,再放上一个 table,代码如下

```
<tr bgcolor="#E8EEF7">
```

```
<td>
```

```
<table>
```

```
<tr>
```

```
<td width="40%">牙刷</td>
<td width="60%">12 元</td>
</tr>
</table>
</td>
</tr>
<tr bgcolor="white">
<td>
<table>
<tr>
<td width="30%">牙膏</td>
<td width="70%">1 元</td>
</tr>
</table>
</td>
</tr>
```

代码中的 width 是 td 的属性，代表宽度的意思，这在稍后马上会说到。

既然可以嵌套 table 进去，当然比第一种方法要灵活的多，而且每次改动其中的一行，不用去操心另外一行是否会受到影响。不过尽量减少 table 的数量，因为页面一旦大起来，维护就会很不方便。检查关门的时候也很麻烦，所以请读者慎重。

那么。一个网页，如果规划好了，就是几个 table 的嵌套。行的高度，列的宽度可以调节，则你想在页面某处显示的要求自然可以实现。等马上将属性讲完后，就要出现本书第一次的 homework 咯。

二. 还是常用的 HTML 元素

说了这么多,读者似乎发现。这么搞下去,一个网页只会有一些单调的文字,既无按钮,又没有图片,可谓家徒四壁。在下作为一个有良知的中国人,显然不能容忍这种耽误读者的现象发生,不过,在介绍这些元素之前,先要把属性这个咚咚说明白。

我们注意一下这句话

```
<table border="0" cellspacing="1" cellpadding="1" bgcolor="#CCCCCC"
class="normal">
```

在 table 元素的尖括号里,多了诸如 border,cellspaceing 这些东西。这些跟随在元素后面的,格式为 XXX="YYY"的东西,我们很荣幸的叫它做——属性。

属性的由来,意义是显而易见的。光光写个 table,程序并不知道你要的这个 table,有多大,需不需要边框,以及采取什么样式等等。但是 HTML 是自由的,你如果未定义任何一种属性,table 一样可以顺利的被表现出来,只是所有属性都是默认的。因此。属性这个咚咚,完全是为了个性化而存在,每个元素的属性也不尽相同。如 cellspacing 这个属性,就无法作用于 tr,td(当然你写了也没有任何关系,它仅仅是不起作用而已)

前面已经提到过,在下的小说,不能作为字典。当然,在下还是要负责任的,起码要为大家指出,找到字典的办法。

作为一个程序员,都不得不承认,其实一个语言的帮助系统,就是一本最好的字典。而作为 ASP 的帮助系统,MSDN 就是不遑多让的选择。它里面有最为详细的资料 and 介绍信息供程序员查询。但是由于 MSDN 太庞大,就像我们上小学就是用辞海来查询一个语文课本上的生字一样——虽然一定可以查到,却费时费力。有好事者将关于网页制作的部分,整理了出来,以方便你我的使用。拜谢该好事者的同时,在下特地在此提供下载地址:

DHTML 使用说明书

在该帮助字典里,你可以搜索到所有元素的所有属性,体贴处是它甚至标注出了,哪些属性适合那些版本的 IE,虽然现在大多数人基本都在使用 IE6 了。在下不会——介绍元素的所有属性,但是每次运用过的属性,都会详细的讲解,所以请读者放心。

我们再回头看看上面那段代码,代码的显示效果相信大多数读者都看过了。就是隔行显示。但这个简单的隔行显示,实际上却用到了一些小小的技巧。

请读者尝试把 table 后的这段语句 `cellspacing="1" cellpadding="1" bgcolor="#CCCCCC"` 去除，观察显示结果。

对比结果如下图：

飞行岛日用品购买列表		飞行岛日用品购买列表	
牙刷	12元	牙刷	12元
牙膏	1元	牙膏	1元
防恐龙喷雾剂	1.5元	防恐龙喷雾剂	1.5元

可以注意到，前者拥有一个灰色的边框。

暂时不讨论为什么会出现如此的区别，我们先看看隔行显示如何实现。

细心的读者马上就注意到，写颜色的部分，一共就只有那么几个地方。英语单词 color 把它们的底给漏了，很明显，`bgcolor` 就是 tr 的一个属性，它定义了这行的背景颜色。白色用的是 `bgcolor="white"` 淡蓝色，则是使用 `bgcolor="#E8EEF7"`，其中，E8EEF7 这种颜色，我是通过 Photoshop 调出来的，一般初学者可以简单的用 white,blue,red,brown,green 这些耳熟能详的颜色进行程序的调试，虽然界面可能看起来突兀，但如果仅限于练习使用，不用考虑那么多。

大家也还记得，刚才说到用于调节宽度的 `width="30%"` 也是属性之一，宽度的写法有两种，一种是使用百分比，如 30%，一种则是使用具体数字，如 500px。（px 代表像素的意思，我们经常听到的 1024*768，单位就都是像素。意思就是屏幕长 1024 个像素，宽 768 个像素）。而百分比则是相对于 table 的总宽度而言。

在下当年是一个喜欢胡思乱想的人，我们假设一个 tr 里面包含了两个 td，就像上面所说，一个 30%，一个 70%。在下曾经想，如果两者加起来，并非 100%，怎么办。比如，恶作剧的将第一个 td 的宽度设定为 30%，第二个设置为 90%，看看 IE 怎么办。实验结果非常奇妙。第一个 30%起了作用，第二个 td 的长度被强制变成 70%。事实证明，这里的规律如此之复杂，以致于鄙人至今都没有准备把它弄清楚。但是可以肯定的一点是，一个

tr 中的 td,如果已经设置了各个 td 的宽度,而它们的总和并不等于该行的总长(有的时候使用像素,有的时候使用百分比),那么实际显示出来的 table 效果就有可能并不如你所愿。甚至比例完全打乱的情况都会发生。但是每次都去计算总和,非常麻烦,特别是有的时候一个 tr 里面包括了 4-5 个 td,每个 td 的宽度又是 17%,26%等等不太容易速算的数字,这样每次都要拿出计算器计算最后一个 td 的长度(要用 100 来减)。在下后来的办法是,将最后一个 td 的宽度留出来不写,让"自由"的 HTML 自己去计算该留多长。比如上面的这段代码

```
<tr>
    <td width="30%">牙膏</td>
    <td width="70%">1 元</td>
</tr>
```

干脆就变成

```
<tr>
    <td width="30%">牙膏</td>
    <td>1 元</td>
</tr>
```

70%是会被自动计算出来的。这样既省去了麻烦的减法动作(这项动作相信众位读者在小学期间已经练习得够多了),又能保证比例不会出错,所以把此方法也推荐给读者。

再多嘴几句。某些时候,我们对于宽度的判定是不准确的。比如,将 td 的宽度设置为 100px,里面应该放置多少个汉字或英文字母妥当呢。如果放置的汉字很多,宽度又定死,又会出现什么情形?这种情况实际上是经常发生的,周密的部署就这么被一两个超长的英文字母给破坏掉。于是,我们就有了两种解决方法。

- 1 更改已经被定死的长度。不过所谓牵一发而动全身。如果你当时每一个 td 都是经过变态的计算后确认好的宽度,那么为了重新达到整体的美观,只能从头到尾再安排一次,过程将会痛苦而漫长。

- 2 让它们自动换行！将一段长长的词组或单词，分成两行显示，这并不是件丢脸的事情。估计已经有读者开始抱怨，说了两种方法，和没说一样。那么。亲爱的读者，须知世间的事，并非都像想象中的那么简单。换行的规则，我们还未说明。所谓换行的规则，其实是指是否将单词截断。把 Love 这样一个美丽的单词，拆分为第一行结尾显示 Lo,第二行的开头显示 ve,想必不是大家愿意看到的事情。这里，我们就引用到一个属性：word-break。

[点此查看](#)

说明：word-break 一共可以被赋予 3 个值：

normal：依照亚洲语言和非亚洲语言的文本规则，允许在字内换行

break-all：该行为与亚洲语言的 normal 相同。也允许非亚洲语言文本行的任意字内断开。该值适合包含一些非亚洲文本的亚洲文本

keep-all：与所有非亚洲语言的 normal 相同。对于中文，韩文，日文，不允许字断开。适合包含少量亚洲文本的非亚洲文本

如何使用 word-break，我将会在后面讲述 style 的时候提到。



学习 HTML 最快的手段，就是看看别人怎么写的，这样进步无疑是非常明显的。右键点击刚才列出的示例页面，并选择查看源代码，就可以看到人家是如何写出这种效果，我们甚至可以完全抄袭过来，只要你能消化为自己的东西。

言规正传。我们介绍了属性的概念，可以开始接触一些 HTML 中其他的常用元素了。页面元素使用最多的，莫过于注册页面。

不错，这种几乎充斥于我们整个网络生活的页面，其实也是最没有什么技术含量部分，属于火车上“打劫”的（出自电影天下无贼），但是它几乎应用了所有最为常用的 HTML 元素。

所以注册页面对与初学者来说，是学 HTML 以及 ASP 的一个极好的教材。

一个没有人品问题的注册页面，一般是像如下显示：

新用户注册		
用户名	<input type="text"/>	用户名可以由英文或汉字组成，最多可以输入20个英文字母或10个汉字
密码	<input type="password"/>	密码最好使用数字和字母的组合
确认密码	<input type="password"/>	请重新输入密码进行确认
密码提示问题	<input type="text"/>	密码提示问题和答案有助于您找回您的密码
密码提示问题的答案	<input type="text"/>	密码提示问题和答案有助于您找回您的密码
邮件地址	<input type="text"/>	请输入有效的邮件地址
真实姓名	<input type="text"/>	真实姓名有助于您的客户或者供应商更加方便的联系您
所在地区	<input type="text" value="[请选择一个]"/>	
详细地址	<input type="text"/>	输入您的详细地址
联系电话	<input type="text"/>	提供您的电话
是否公开资料	<input type="radio"/> 是 <input checked="" type="radio"/> 否	
<input type="button" value="提交"/>		

这个是在下自己制作的一个网站的注册页面，现在拿出来作为教材现身说法。

除了大家熟悉的 table 以外（请注意，我说的这一切，都是这第二天需要掌握的内容。如果你开始觉得记忆有点吃力了，不妨多看几遍），还多了不少其他的 HTML 元素。让我们直接从设计开始，完成该页面吧。

首先当然是，先在大脑中构思，一个注册页面，大概是个什么长相。这个步骤决不能省略。做每一个页面的时候，都得先有它的构架，构架搭好了，整整制作起来是非常方便的。我当时做此页面的时候，映入脑海的，就是上面的那个截图——一个大 table,总体来说分为 3 列，第一列显示"姓名" "密码"等等内容，第二列则是输入部分，第三列则是对该行内容的一些备注。很简单的一个构架，我们可以马上着手实现它。

```
<table border="0" cellpadding="3" cellspacing="1" width="100%"
bgcolor="#CCCCCC">

<tr bgcolor="#FFFFFF">
```

```
<td width="20%">用户名</td>
```

```
<td width="35%"></td>
```

```
<td align="left">用户名可以由英文或汉字组成，最多可以输入 20 个英文字母  
或 10 个汉字</td>
```

```
</tr>
```

```
<tr bgcolor="#FFFFFF">
```

```
<td width="20%">密码</td>
```

```
<td width="35%"></td>
```

```
<td align="left">密码最好使用数字和字母的组合</td>
```

```
</tr>
```

```
<tr bgcolor="#FFFFFF">
```

```
<td width="20%">确认密码</td>
```

```
<td width="35%"></td>
```

```
<td align="left">请重新输入密码进行确认</td>
```

```
</tr>
```

```
<tr bgcolor="#FFFFFF">
```

```
<td width="20%">密码提示问题</td>
```

```
<td width="35%"></td>
```

```
<td align="left">密码提示问题和答案有助于您找回您的密码</td>
```

```
</tr>
```

```
<tr bgcolor="#FFFFFF">
```

```
<td width="20%">密码提示问题的答案</td>
```

```
<td width="35%"></td>
```

```
<td align="left">密码提示问题和答案有助于您找回您的密码</td>
```

```
</tr>
```

```
<tr bgcolor="#FFFFFF">
```

```
<td width="20%">邮件地址</td>
```

```
<td width="35%"></td>

<td align="left">请输入有效的邮件地址</td>

</tr>

<tr bgcolor="#FFFFFF">

<td width="20%">真实姓名</td>

<td width="35%"></td>

<td align="left">真实姓名有助于您的客户或者供应商更加方便的联系您</td>

</tr>

<tr bgcolor="#FFFFFF">

<td width="20%">所在地区</td>

<td width="35%"></td>

<td align="left">&nbsp;</td>

</tr>

<tr bgcolor="#FFFFFF">

<td width="20%">详细地址</td>

<td width="35%"></td>

<td align="left">输入您的详细地址</td>

</tr>

<tr bgcolor="#FFFFFF">

<td width="20%">联系电话</td>

<td width="35%"></td>

<td align="left">提供您的电话</td>

</tr>

<tr bgcolor="#FFFFFF">

<td width="20%">是否公开资料</td>

<td width="35%">

<td align="left">&nbsp;</td>
```

```
</tr>  
</table>
```

读者一看就倒了，这么多行.....我先再这里声明一下，看我这本书的读者，会变成一个代码编写者。代码编写者这个名号，是针对“拖拖拉拉者”而命名的。如果读者有幸学过 Dreamweaver 这个工具，一定会认为在下在折磨网页初学者以实现某些不可告人的目的。所谓“拖拖拉拉者”，其实就是使用工具，很轻松的建立 table,放上各种网页元素，而且可以对 HTML 一点都不用去了解。在下并非不赞同读者采取这种方式，而是需要将自己的网页技术切实提高，从 HTML 入手，是最为妥当的一种方式。当你把 HTML 玩的很熟的时候，学什么 Dreamweaver,上手最多只是一两天的事情。

且把闲话休提，看看上面这段代码。经过上面的 table 和 tr 的介绍，代码的大部分应该并不难懂。唯一有区别的地方，我把它列出来介绍给大家听：

Table 的属性：

Border="0" 代表该 table 的边框宽度为 0，读者可以自行设置为 1，查看效果。

Cellspacing="1" 该值用于设定 table 中每一个单元格（行和列组成单元格）的间距。注意，这个值得存在，说明每一行和每一列，并非紧紧相依，除非你把它强制设成 0

Cellpadding="1" 该值用于设定，每个单元格的边框与单元格内容的间距，如果设定为 0，则会出现内容紧贴着边框那种难看的现象。

Width="100%" 虽然已经介绍过，这是宽度的概念。再次于此处声明，每一个页面元素的宽度，如果是百分比，则这个百分比，对象都是包含它的元素。比如这个 table 如果没有包含在其他页面元素里，那么设置成 100%意味着该 table 的宽度等于打开 IE 浏览器的那个宽度）

TD 的属性

Align="Left" 这个值相当简单。就是表示该单元格的内容，水平方向上是靠哪边对齐（所谓对齐，其实就是往那个方向靠拢）。一共可以设置 **Left,Right,Center** 三种。

它们分别代表左，右，中。与此对应的，还有一个

vAlign="Top" 这是垂直方向的对齐方式，**Top, Bottom, Middle** 分别代表向上对齐，向下对齐，和置于中间。

有了上面的解释，想来看懂那段长长的代码，对应上它显示出来的效果，应该不是件困难的事情。另外，严肃的说，看懂代码，只是爬山的第一步。想要真正的消化和吸收，非得锻炼到自己写出一段来不可，这个道理相信大家都明白的。



HOMEWORK: 请读者自行用以上介绍过的属性，加上对 **Table** 的了解，画出一个简单的商店物品单价表来。行和列可以随便设计，但是信息要求完整，并让消费者对所列出的物品有所了解。

如此自由度的 Homework，不知道读者是否有无所适从的感觉呢。诚然，编写网页的时候，不一定所有的属性都会被使用上，往往我们都只是使用到其中的一小部分。那么，在这里公布一条原则，做 Homework 的时候，不要管什么人性化和界面美观，力求将学习到的属性，一一用上，所谓有困难要上，没有困难制造困难也要上。还是那句话，终有一天，你会得心应手，那个时候再开始自由发挥不迟。

OK，现在页面已经初具形状，但仍然不满足我们的要求。页面上仅仅有个框架，用户根本无法填写——我们需要提供给用户一个输入框。

让我们再对上面的代码进行小小的变动。（不再把所有代码写出，仅仅取出其中的一部分进行修改，其他的大家可以类推）

```
<tr bgcolor="#FFFFFF">  
  <td width="20%">用户名</td>  
  <td width="35%">
```

```
<input name="txtUserName" type="text" value="">

</td>

<td align="left">用户名可以由英文或汉字组成，最多可以输入 20 个英文字母或 10
个汉字</td>

</tr>

<tr bgcolor="#FFFFFF">

<td width="20%">密码</td>

<td width="35%">

<input name="txtPassword" type="password" value="">

</td>

<td align="left">密码最好使用数字和字母的组合</td>

</tr>

<tr bgcolor="#FFFFFF">

<td width="20%">确认密码</td>

<td width="35%">

<input name="txtConfirmPassword" type="password" value="">

</td>

<td align="left">请重新输入密码进行确认</td>

</tr>
```

不错，我们先接触到的，就是这个 input.

在 ASP 页面中，一般 input 有这么几种使用方式

- 1 作为文本输入框: `type="text"`
- 2 作为密码输入框: `type="password"`
- 3 作为页面上的按钮: `type="button"`
- 4 作为复选框: `type="checkbox"`
- 5 作为单选按钮控件: `type="radio"`

6 作为提交按钮: `type="submit"`

7 作为隐藏页面空间: `type="hidden"`

在下不可能一个个现在就介绍过去,否则马上有人会睡着。作为一个成功的讲师,让台下的人睡着,是非常失败的一件事情(尽管已经有读者反应,已经一觉睡去,然后醒来了)

我们只能接触过什么,就介绍什么,不过大家放心,上面说的所有使用方式,最终都会被使用到。因为它们都是最为常用的页面控件。

最为普通的输入框,我们都是使用 `input type="text"` 这种形式。可以用作输入的有,用户名,密码提示问题啊等等的部分。因为它们的要求很简单,柴米油盐酱醋茶,平平淡淡的过掉下半生就好,其他的一概不想考虑。请注意。那些对内容的长度或者输入格式的控制,是属于 JavaScript 的编程部分,而网页代码仍然就是 `type="text"` 这种简单的形式。我们再看一下这句话

```
<input name="txtUserName" type="text" value="">
```

`Type="text"` 已经解释过了,还有两个不太明了的地方。一个是 `name`,一个是 `value`。该两个属性一定要好好认识一下,它们可谓阴魂不散,绕梁不绝。它们在很多不同的元素中出现。

`Name`,顾名思义。就是一个页面元素的名称。我们在后面的学习中,经常需要通过程序来控制页面元素(比如获取或设置页面元素的长度高度等等),这个时候,就像一个生了十多个小孩的母亲一样,不能整天"喂,喂"的来呼唤孩子们,给他们取个名字会是一个不错的选择。注意,不同元素的名字不要取的相同。如果你坚持要用相同的名字,结果也会相当明了的,喊一句"张三洗澡了",两个同名的孩子就要吵半天的架。奇妙的是,HTML 并不会指出这种名字相同的错误,页面浏览的时候也不会报错。HTML 的做法就是自动将你的名字扩展,让控件以一种集合的形式存在,有时候,我们甚至要利用到这一种属性。还用小孩洗澡的例子来说,有一天,当母亲的终于厌恶了每次交大家睡觉的时候,要从"张三睡觉了"一直喊到"马六睡觉"为止,母亲干脆把他们取成一样的名字,比如都叫"二狗子",时间一到,直接一句"二狗子全部滚到床上去",世界就清静了.....而 HTML 会自动把所有的二

狗子编上号，你要单独教训其中一个二狗子时，就需要使用"二狗子*号"这样的名字来精确的定位。具体的步骤和过程，将在后面的 JavaScript 课程中讲到。

再来收拾一下 value,每一个输入框，我们都会给它一个值。比如你在普通的文本输入框里写"abc123"这样的字符时，其实正在进行一个赋值得步骤。大家可以做个小实验，把这句话：

```
<input name="txtUserName" type="text" value="">
```

改成：

```
<input name="txtUserName" type="text" value="abc123">
```

并观察页面上的效果。你就会看到，原本空白的输入框里，现在显示着"abc123"。在 type="Text"这种普通的文本框元素里，value 就代表了框中的内容。就像唐伯虎刚进入华府的时候化名为"华安"，编号是 9527，相应的我们可以理解为，他的 name="华安",而 value="9527"。不同类别的元素，value 所代表的含义也不同，并不能简单的认为 value 就是显示在页面上的部分。后面介绍其他元素的时候，都会——介绍他们的 value 含义。

弄懂了上面三段对 type,name 和 value 的介绍。后面的过程，可以说驾轻就熟了。

再看这个

```
<input name="txtPassword" type="password" value="">
```

Type="password"前面说过，是密码输入框。它和普通的文本输入框的唯一区别，就在于，输入密码的时候，先是出来的是"*"这样的字符（相信经常在各个论坛摸爬滚打的读者们，对这种输入框是不会太陌生的了）当然，显示归显示，密码输入框的 value 属性，仍然是如假包换的咚咚（也就是说，你在输入框里输入 123456,屏幕上啪啪啪的显示 6 个星号出来，但 value 依旧老老实实的是 123456。这样方便将来的代码在后台进行操作。

瞬间解决了名字输入和密码的问题，我们接着往下更改代码。刚才的注册页面中。当你了解到普通输入框的用法后，基本已经可以将它完成大半。这里将会特别提到几个有可能出现新元素的地方。

```
<tr bgcolor="#FFFFFF">
```

```
<td width="20%">所在地区</td>
```



```
<td width="35%"></td>

<td align="left">&nbsp;</td>

</tr>

<tr bgcolor="#FFFFFF">

<td width="20%">是否公开资料</td>

<td width="35%">

<td align="left">&nbsp;</td>

</tr>
```

我们希望,在所在地区的填写时,不要让用户自己来写,而是给出全国所有的省,直辖市,自治区让用户选择,这样会显得更加人性化一点。而下面的是否公开资料,也当然应该给出两个选项(是或否)供用户选择。

因此,下拉式菜单和单选钮就被提到台面上来讨论了(他们的样子可以参照上面的贴图)

首先当然是下拉式菜单的出场。

下拉式菜单并不是 input 类元素的一种。它使用的是以下形式存在

```
<select name="cars">

<option value="1" selected>宝马

<option value="2">保时捷

<option value="3">奔驰

</select>
```

一个下拉式菜单。就是一组的 Select 元素,和文本输入框一样,我们给它命名为 cars。select 之间包括的,就是用户看到的"选项"。以上面的代码为例,用户将会看到"宝马","保时捷","奔驰"可供选择,且开始默认选中的是"宝马"。(这就是第一个 option 里多了一个 selected 的原因,它表示该项目被默认选中)

每一个选项。或者说每一个单位,都是一个 option,可以看到,他们也有一个 value 值。这里的 value,就不会被显示在屏幕上了,那么它们有什么存在的价值呢?如果仅仅为了页面

显示,我可以负责任的说,完全没有必要写上这些 value 来浪费时间。可是我们更关心,到底用户选择了三种车的哪一个。当用户在页面上做了选择后,选项的 value 会被赋值到 cars 这个变量里。换句话说,程序会默默地将用户的选择记录下来,以后要把用户信息存放如数据库的时候,自然可以做到所谓的"心里有数"。值得高兴的是, value 的内容,没有太多的限制,你可以使用数字,可以使用字母或汉字,这都取决于你。但在下还是希望尽量不要使用汉字,前面说过了。汉字牵扯到了复杂的编码问题。另外,既然是用来记录用户的选择,所以每个 option 的 value 当然应该是不同的,如果有特殊的需求,将其中几个项目的 value 弄成相同,以实现某种分类的目的,当然也可以,HTML 都是允许的。

有些读者,可能已经急于想知道。如何编程玩弄这些用户输入的数据了,敬请大家放心,开胃菜您不妨先尝,主菜再后头呢☺

剩下最后一个要介绍的单选按钮控件了。老规矩,把代码写出来然后分析

```
<input type="radio" name="InfoShared">是  
<input type="radio" name="InfoShared" checked="">否
```

看到这里,某些读者也许已经可以会心一笑了,因为这些元素写出来都是大同小异,可以说没有任何难度了。这里只需要特别提出来的,就是这两个单选框的名字,他们的名字是一样,这是有特殊原因的。一般来说,名字可以用来区分控件,就像前面说的要管教不同的孩子一样。可是单选按钮的存在意义在于,当用户选择了其中一个,就代表他放弃了剩下的选项。所以单选按钮后面写的字,通常都是"是","否"这样彼此互斥的单词。用户不能既选择是,又选择否。所以我们把它的名字取成一样,就是对外宣称,它们两个,是在一起算作一组的。如果你想放上另外一组单选钮,那么当然要换上一个全新的名字。

细心的读者可以进一步注意到,单选钮中并没有 value 这个挥之不去的可恶属性,这里也解释一下。用户的选择,是通过另外一种方式存放的,下一章主菜上席的时候,马上就会提到,读者可以先记住一下,后面自然会豁然开朗。

最后, checked 属性,代表默认选择。



HOMEWORK: 读者请把学到的部分好好总结归纳一下，设计出一个个性化一些的注册页面来，这里说的个性化，指的是需要用户填写的内容而非页面美工样式。只有自己写代码，才能更快的掌握。

2.2 让页面漂亮起来

不可否认，即使拥有了上一节提到的所有知识，我们的页面看起来仍然是丑陋不堪。清一色的白底黑字，字体也是清一色的大小，无论如何设计 Table，始终看起来土的掉渣。这一节，就带领大家慢慢的将页面打扮起来。不过事先声明，在下并非一个成功的化妆师，我只能把胭脂水粉奉上面前，至于如何涂抹上色，则要看大家自己的功力。

2.2.1 字体和图片

— 个性化美文的问世

懂得把玩文字的高手，可以通过一些特殊的字体润色，达到表达感情的目的，我们来看看这个曾经相当出名的个人网站的介绍（动漫网站。名字为“云和山的彼端”）

生人止步！当你看到此等文字，你已身在云和山的彼端（什么？盗版？！去你哥的，本人这样用是因为本人觉得爽！本人还想去告什么XX剑3侵权咧！）这里是动漫的异端，接近死气森森的噩梦城的边缘——总之不是你该来的地方！

说得清楚点，本站是为满足你最低级的本能的需求而存在的，提供你只有在梦魇里才能见到的有关动漫所有最基本的臃肿的垃圾，包括每月1G的经典动画MP3、大到XM一张的天X喜孝都筑X彦的非压缩CG、尽情增加尽情嘲弄你那小不拉几的XG硬盘的经典卡通连载剧场，还有占着XG的硬盘不拉屎、绝对原创、绝对霸道、绝对不更新的原创空间，以及专门放含对本站歌功颂德文章的动漫绝对论坛，由一个正常得变态和一个变变得正常的站长联手主持，有神的就来丢文章灌水打屁！

不错，你来错地方了！本站是有钱人和公费上网者的天堂！网络速度在30K以下或穷得叮当响的可怜虫请在每日18时后自动滚开，以免拖慢网络速度。以上。

暂时不用管内容到底写了些什么，不过该站长明显认识到，不同的字体，同一段文字，读起来就会有不同的感觉。这个中缘由想来不必在下过分强调，我们跨世纪的一代，应该懂得不用婆婆妈妈，直入主题。

改变字体可以通过 3 种手段，

- 1 改变字体类型。比如由默认的宋体（中文系统）改为楷体。
- 2 改变字体颜色和大小。比如上面那个例子，它的站长就使用了改变大小的技巧。
- 3 一些特殊效果。常见的有加粗，斜体和下划线

不再多说废话。直接奉上代码。

```
<table>
  <tr>
    <td>
      <font color="blue" size="3">Oh my God!</font>
    </td>
    <td font-size="9pt" font-color="blue" font-family="宋体"
      Font-weight="bold">
        Oh my God!
      </td>
    </tr>
  </table>
```

注意到红色的字体和紫色的字体。他们实现的效果是相同的。应该说，红色是一种简便的用法，且最原始标准的 HTML 语法，并不提供对这种字体设置方式的支持。好消息是现在读者可以放心大胆的使用，如今它也作为一种规范生存了下来。它的优点是使用简单，随时随地。在下一节只使用它的 color 和 size 功能，我们来仔细分析一下该元素。

前面的 font 和 input 元素很相似，不必多说，它就是该元素的名称，显而易见，后面的 color 和 size 则是它的属性。再次证实了 html 其实就是一大堆元素通过它们自己不同的属性的结合，由于属性千变万化，出来的页面自然也是风格迥异。Color 可以有两种书写方式，我们在前面的 bgcolor 中已经有所提及，此处不再赘述。size 属性则是指字体的大小，可以设置 1-7 一共 7 个整数，1 为最小的字体，7 则是最大。一般我们都设置成 3，中等偏小的就可以。

此外，font 元素可以使用 face 属性来设置字体。在下把这个留给大家让大家自行探索 😊

我们这次的重点，其实是放在下面的紫色代码部分。

一般来说，页面上的字体风格不要超过 3 种，让人家看起来既不失变化，又整齐大方。在下刚会用字体的时候为班上做主页，曾经神经线搭错，一个页面用了 10 多种的风格，结果看起来就像一件打满补丁的丐帮工作服，这里一堆，那里一坨，可谓惨不忍睹。于是，人们把页面中用到的字体，提到页面的最前面来，把它们放到样式表（CSS）里。（关于 CSS 的部分，本书不会讲述，有兴趣的读者请自行查询相关资料）

紫色字体的部分，就是经常被使用作放到样式表里的一种字体定义方式。

该字体定义方式可以放到大部分元素中，用来设置和定义该元素内部的字体。基本上，能够在页面上显示出字的元素，都得吃它这套。

最为常用的几种属性：

Font-size: 字体大小，可以用 pt 为单位，精确的定义字体的大小。我们一般看到的 Windows 中的字体大小都是 9pt 或者是 10pt。

Font-color: 字体颜色，和前面说过的 color 是一样的。

Font-weight:字体的磅数。就是我们经常说的粗体。一般我们将这个属性设置为 bold。这个属性允许我们对粗体进行非常详细的设置，换句话说——我们可以定义这个粗体，到底有多“粗”。其实现在我们使用不到那么多，一个 bold 就够大家玩弄了。一般好的网站，注重的是字体的颜色和大小，粗体多粗，都不在大家的考虑范围。如果一定要坚持研究这个粗体，可以去 DHTML 这本字典里（前面提到过），查看对 font-weight 的详细介绍。

Font-style:设置是否斜体。默认是正常字体。如果你需要让字歪斜一些，则可以用到这个属性。基本用法是 Font-style="Italic"

既然没有讲 CSS，建议大家在写网页的时候，仍然使用<font...>这种简单的方式，如果某些地方，对字体要求高了，再把紫色代码那种写法加进去，（具体的用法，就像我上面所写，放到 td 或者其他元素中即可）

需要了解的是，我们现在学习的是 ASP/HTML 的掌握，要将来大家长年累月的去积累，不可一蹴而就。

二 图文并茂才是王道

做网页最容易发生的错误，就出现在学习了字体那节以后。很多人从网上下载了美轮美奂的字体装饰自己的网站，各种风格的字体交相辉映，煞是好看。可是把网站发布后，其他人看起来，却全部老老实实变成土土的宋体。其中的原因很简单，网站用的字体，访问者机器上没有，自然只能用系统默认的宋体来代替。

这怎么办.....总不能在打开网站的时候说，请大家下载某某字体才能达到最佳观看效果吧。解决的方法，就是图片。

图片的这种妙不可言的用法，让网站上各种搞怪字体也争相涌现出来。不过归根到底，那只是图片用法的一种。在下特地放到这里，也主要是想起一个承上启下的作用◎且把闲话休提，来看看图片的用法。

放上一句图片代码

```

```

可以失望的看到，HTML 实在脱不了这个俗套，还是和 Font 那个死样类似。不必说，img 就是元素名称，后面的则是它的属性。Width 和 height 可以从字面直接得出意思，就是图片放上去以后的宽度和高度。如果设置的宽度与高度与原来的图片不合，浏览器会拉伸或压缩要显示的图片。如果你不知道图片到底多大，又希望原尺寸显示到页面上。那么这两个属性都可以忽略不写，关键是这个 src 属性。

src 的值，就是图片文件的文件名。在支持 HTML 发帖的论坛上，经常会出现所谓的硬盘图现象。什么叫硬盘图呢？刚涉猎 HTML 的朋友，想要使用 img 元素显示图片。于是写成下面这个样子：

```

```

如此写法的目的很明显，想在页面上显示 C 盘上那个名字为 avator.gif 的图片。在本机上看得时候，显示完全正常。因为你的机器上，确实拥有这么一个图片在 C 盘上。和这一节前面的那个字体设置，其实是犯了一样的错误，访问页面的人如果没有字体，还可以用系统默认的宋体来表示，但是访问者没有图片，就只能看到一个血红的大叉（找不到图片时的显示），这就是所谓的硬盘图。

为了让大家看起来都爽，我们就得在这个路径上下下功夫了。回头到第一天的讲座，里面提到了一个虚拟目录的概念，此处就派上了用场。

例子中的 src="images\avator.gif"，用的就是虚拟路径。我们在存放网站的目录下，建立一个 images 子目录，并把 avator.gif 拷贝到该目录下。（如果看不懂拷贝这些概念。建议先学一些计算机文件操作的基础知识）。此时如果网站的访问者调用图片，就不是到他们自己的机器上七翻八找了。而是通过虚拟路径。直接从服务器上找到制作者想显示给他们的图片，并正确的显示出来。读者务必要仔细理解上面这段话，不懂得部分，可以回头参看第一天所说到的虚拟目录结合起来理解。

我再把话说通俗一点，图片为什么网站访问者看不到，因为他没有这张图。怎么让他有这张图呢？我们就把图片放到服务器上，并让他直接访问服务器查看该图片。Src 其实就是文件的路径，你写成 C:\avator.gif,访问者的浏览器就在它自己的 C 盘上找，你写成 D:\avator.gif,它就到 D 盘上找，而你写成虚拟目录，它就能登上你放置网站的服务器来找。此时你只要在服务器上把文件放好，路径书写正确即可。一个完整的图片路径可能是这种形式：

http://www.google.com/images/logo_sm.gif

但你在代码里不需要这么写。直接用"images\logo_sm.gif"就可以。这就是所谓的相对路径。它是相对于 <http://www.google.com> 这个网站的路径，它代表到网站的根目录下寻找 images 子目录，然后进入该子目录寻找 logo_sm.gif 文件。

img 元素还有其他可以使用的属性。再次在下也依旧惯例的将它们省略掉。仅仅已经介绍的几个属性，就够大家喝一壶了。关键仍在那张图片好看不好看上。就算把 img 的所有属性摸透，图片不好看，也是白搭。

至此，第二天的内容全部结束，明显要比喝咖啡点鼠标的第一天，辛苦多了。如果有些读者依旧像第一天那般，觉得其实什么都没有学到，在下深表遗憾的同时，也告诉大家，HTML 只是 ASP 的基础，真正的战斗，下一章才开始.....



HOMEWORK: 用这天所学的所有知识，设计出一个漂亮的注册页面来，为了最后的效果，允许大家不择手段◎算是对该天学习的一个总结

第三日：仙风云体术

掌握 HTML 并非一件困难的事情，不过写着写着，有些问题渐渐凸显了出来。以注册页面为例子，刚才的那些，实在太过简单和单调，本着交互的想法，许多读者可能开始希望加一些控制效果上去。比如大部分站长都不希望用户名超过 20 个字母，有些人给自己起名字像写诗一样，没完没了；另一些人则对于注册信息毫不在意，乱填一通。值得欣慰的是，所有的这些，我们都是可以在代码中写好，直接于客户端控制并处理的。当用户注册完毕真正将信息传递到服务器的时候，服务器不需要再费神去做检查，某种程度上减轻了服务器的负担。这部分代码，理论上说，可以使用 JavaScript 和 Vbscript 中的任意一种，不过据在下观察，大部分客户端代码，都是由令人爱恨交加的 JavaScript 写成，为了站好迎接 asp 最后一道岗，我们还是迈向这一步---JavaScript 的学习。为了便于大家查阅，先放出 JavaScript 官方帮助文件的下载：

[Javascript 参考手册](#)

3.1 JavaScript 是一门简单的脚本语言

掌握脚本语言绝对比掌握一门真正的语言要来得容易——尽管 JavaScript 玩深了也是一门重要的学问。好在目前为止，在下腹中无才，而读者也暂时不需要了解那么多，我们决定来个浅入浅出的语言教学。

第一天我们说过，任何一种语言，万变不离其中，无法脱离以下的讲解步骤

1. 输入和输出
2. 变量和常量
3. 判断语句
4. 循环结构
5. 高级技巧

除了最后一条高级技巧，我现在搜肠刮肚也总结不出什么东西来，姑且放着，让我们

1-4 条的从头说起。

3.1.1 输入与输出，变量与常量

输入和输出，我们最好能跟页面紧密结合起来。离开了页面的 JavaScript，就像离开了土壤的花朵，立刻丧失了其强悍的生命力。

先说说输入。很显然，我们在第二天，已经详细的介绍了如何在页面上写入东西，注册页面的那些输入框明显就是和用户交流的一个最好手段之一。输入的过程，我们其实仍然没有全部完成——因为，我们昨天那么书写，一点 JavaScript 的痕迹也没有见到，谁来接收用户的输入，又有谁来处理，用户满头大汗的在注册页面上敲敲打打，到头来竹篮打水一场空。那么，我们需要进行第一个讲解步骤，就是 JavaScript 和页面元素值是如何结合在一起的。

这里不得不提变量的概念。第一天已经说过变量的含义--用于临时记录信息，并对它进行操作。

我们先来看看下面这段语句：

```
var b;
```

```
b=10;
```

相当简单，只有两句话。第一句称作声明，第二句则是赋值。在 JavaScript 或是将来遇到的所有语言中，任何一个变量，在使用前，都最好将它声明一次。这是一个良好的习惯。那么，究竟什么是声明呢。非常简单，就像几个流氓接头的时候，喜欢问得一句话，"小子，你混哪里的？"既然出来混，当然需要名正言顺。变量声明正是一个让毫无意义的字母数字组合，变得名正言顺的过程。一个正式的语言中声明变量，还需要严格的说明该变量是什么类型，是整数，还是字符串，又或是数组等等。幸亏今天我们玩得是 JavaScript，这些我们都不必管，我们只需要简单的：

```
var 变量名;
```

注意每一句 JavaScript 语句都是用分号结束，它代表一个语句的结束。初学者很容易

忘记这个。所以希望刚学的时候，大脑里给自己提个醒，不停的默念着“分号分号”，自然不会漏掉。

`var` 就是变量声明使用的特殊关键字。所谓关键字，指的是这种字母组合在 JavaScript 是有一定含义的，是一开始就被定义了功能。比如 `var`，它就是完全为了变量声明而存在于世界上。

JavaScript 对大小写有着严格的要求，如果你写 `Var` 或者 `vaR` 都是毫无意义的字符，它们也不能起到变量声明的作用。所以除非特殊需要，最好通篇的代码都使用小写。目前在下自己在使用的时候，就没有发现有几个关键字是大写。

在 `var` 后面，跟的就是你希望使用的变量名。考虑到将来你会给这个变量赋值，你会对它进行操作，请不要将名字取到 200 多个字母这么长。变量名的选择，有三点的要求

1 不要和关键字一样。

如果我们写出 `var var;`（本意是声明一个变量，变量的名称是 `var`）然而这样的语句，计算机是不认识的，因为它不知道你吃的这么饱，要把声明的语句写两遍。它并不知道后面的 `var`，其实是变量的名称。所以代表了特别功能的关键字，都不能用作变量名。

2 不要使用特殊的字符，不要将数字作为变量名的开头。

给变量取名字不是给 QQ 设置昵称。最重要是你将来使用方便。一般我们取变量名甚至连数字都不用，常用到的就是 26 个英文字母和下划线“_”，它们的排列组合已经足够我们使用了。发现很多书讲变量声明的时候，告诉读者不要这么做不要那么做，还列出那些字符不能作为变量名，在下认为大可不必，真正写程序的人，并不知道哪些字符不能用作变量，我们只需要知道，那些可以做变量，够我们使用就好

3 变量不要重复声明。

这个道理太简单了。意义也显而易见，不必多说。

OK，我们回头来看前面写的

```
var b;
```

```
b=10;
```

在声明了变量了以后，我们就开始了迫不及待的使用了。学计算机语言，一定要知道，等号这玩意，在语言中，很多时候并不代表了逻辑上的相等关系。这里的 `b=10`，我们叫它赋值过程。它的作用是把 10 这个数字，附到 b 这个变量里去。所以将来要是看到

```
b=b+1
```

这样不合道理的式子，就不用太过惊慌，它并没有代表 b 和 b+1 相等的意思。它仅仅是一个赋值过程，代表把 b+1 以后的值，再附到 b 里面去。

连起来写：

```
b=10;
```

```
b=b+1;
```

经过了这两个语句，这时候的 b 里面的值，就是 11

有的人想问。如果能让 b 被赋值成 var 这个关键字怎么办？前面不是说过不能用关键字吗？

我们注意到，赋值语句还有一种写法

```
b="var"
```

当一个字符串被双引号给“关”起来的时候，它就失去了原来的任何含义，无论是变量还是关键字，这时候，它就乖乖的做一个字符串。因此，以下语句

```
a="var"
```

```
b="a"
```

```
c=a+b+";";
```

执行后，c 被赋的值就是 `var a;`

(注意到最后的相加，其实是字符串组合的过程。第一个分号是字符串，而第二个分号则是代表语句结束)

那么所谓的输入，我们可以暂时的把它理解为，将用户在页面上所填入的值，用户所做

的操作（比如用户在下拉式菜单里进行选择），记录到变量里。回到页面来，我们一个个的把用户的输入，进行处理。

上一节我们清楚地知道，注册页面上的元素可以的分这么几类。字符串输入（包括普通的输入和密码输入），下拉式选框，单选按钮。其实还有复选框，用户的点击事件等等，输入种类之多，让人无所适从，我们暂且看到一个说一个。

先说到字符串输入，这应该是最简单的部分。我们把用户名和密码两部分拿出来做处理：

```
<input name="txtUserName" type="text" value="">  
<input name="txtPassword" type="password" value="">
```

这两个输入框，分别被命名为 txtUserName（名字）和 txtPassword（密码）。

下面这段代码：

```
var username;  
var password;  
username=document.all.txtUserName.value;  
password=document.all.txtPassword.value;
```

随时要注意，JavaScript 是大小写敏感的。不要写错哪怕是一个字母的大小写。上面那段文字，言简意赅，应该不难看懂。先是两个声明语句不必多说，它们分别声明了变量 username 和 password。后面的两句，不用说也知道是在做一个赋值动作。可是，等号右边乱七八糟的一大串，除了 txtUserName 和 txtPassword 我们知道是那两个控件的名称外，其他皆一头雾水。读者放心，且听在下分说。

首先我们要先确认，需要被赋值到变量里的，是什么东西。究竟是要把整个控件都赋值到变量里（这其实也是可行的），还是仅仅需要用户输入的内容？赋值前，一定把“要赋什么”这个问题给搞明白。在这节里面，我们基本可以确认的是，只对用户输入的内容感兴趣。而前面说过，用户输入的内容，被放到文本框的 value 属性里。所以，等号右边的一部分，

应该就不难理解了：txtUserName.value 就是代表着 txtUserName 这个文本控件上用户输入的内容。乱七八糟的片断里，现在只剩下前面的 document.all 了。

我们可以把 document 看作整个 HTML 文档，而 document.all 则是指文档中的所有元素。这样的话，document.all.txtUserName 我们可以看作是一个梯形的层级关系，翻译成白话可以这么说“HTML 文档里的所有元素中的 txtUserName”。蓝色字体分别以此代表着 document.all 和 txtUserName。归根到底来说，写到这么长，是为了定位到这个网页元素。仅仅写一个 txtUserName.value，计算机会报错的。为什么需要这么长的定位，在下觉着大约要花上十几万字来说明，就留给读者在今后的学习中慢慢体会。附带说明一下，document 后面的 all 是必不可省，有的人想当然的认为，这里的 all 既然带表所有元素，那干脆直接把 txtUserName 安插上去就好，其实不然。先把它理解成王八的臀部——规定，将来自然明了。

严格的说，输入已经教完了。为了不让读者产生思维定势，在废话几句。刚才读者看到我们这么写语句：

```
username=document.all.txtUserName.value;
```

其实，没有人规定。Username 只能赋值成文本框的 value。我们可以把这句话写成这样：

```
username=document.all.txtUserName;
```

此时的 username 就代表着 txtUserName 这个控件。而 username.value 才是用户输入的值。灵活的运用赋值算式和变量，会达到意想不到的结果。

现在我们来说一说输出。

最为常见的输出，莫过于弹出提示框。这种突兀的输出方式，居然广泛的流传在大江南北。不可否认它是一种引起注意的方法。如果音箱开的够大声。还能伴随着一个巨响“咚”，让人心跳加速，一身冷汗，有怯寒的功效。以前有一个整人的网页，就是由 JavaScript 编

写的，叫做“白痴确定练习站”，一共设计有 100 多个弹出框，进入的朋友需要一个一个点掉，非常烦人，现在在下就教大家怎样开始输出旅程的第一步。

先给大家看一个弹出框的样子☺



安装了 WindowsXp SP2 的朋友可能快和这个弹出框绝缘了，因为现在增加了安全性。页面上会出现这样一行字

为帮助保护您的安全，Internet Explorer 已经限制此文件显示可能访问您的计算机的活动内容。单击此处查看选项...

不过出于负责人的态度，在下还是介绍一下，给大家看一段完整的 JavaScript 代码

```
1 <script language="JavaScript">
2     var mymessage;
3     mymessage="我是一个菠菜";
4     alert(mymessage);
5 </script>
```

方便起见给每行弄个行号。第一行和最后一行这种方式，大家也许似曾相识。没错，它就和我们学过的 `td` 或者 `font` 一样，`script` 我们也可以看作是一种页面元素——叫做脚本语言的页面元素。而 2-4 行，则是脚本语言的内容。

前两行刚刚学习过，大家都明白。变量 `mymessage` 被赋值为“我是一个菠菜”。而

弹出框居然就是第四行描写的那么简单。alert 这个单词就是警告的意思，难怪每个弹出框都令人心浮气躁。语法相当简单：**alert()**；括号里放上要显示在弹出框中的内容。我们可以在括号里放上变量（像刚才那段代码里所写），也可以简单地写成

```
alert("我是一个菠菜");
```

注意加上双引号，要不然计算机认为**我是一个菠菜**是变量。

把这段代码，放到一个空白的 asp 或者 html 页面中，那么这个页面被打开时，它就会被执行。页面上弹出"我是一个菠菜"的框来。

学到弹出框，其实可以不再管其他的输出方法，因为更高级的输出技巧，需要一些额外的知识，我会在后几章的例子用到，那时再像读者分说不迟。

学习任何一项知识都有一个痛苦的过程，就是，学得越多，问题越多，后来几乎觉得问题问都问不完。而如果完全没接触，反而会问不出问题来。我现在就想给读者提个问题，当用户输入用户名以后，希望在页面上弹出一个框。上面写着，"您好**，欢迎注册"（其中**代表用户输入的用户名）该怎么实现。

我开始学习的时候，几乎不假思索，就把代码写出

```
1 <script language="JavaScript">
2     var username;
3     username = document.all.UserName.value;
4     alert("您好"+username+"，欢迎注册");
5 </script>
```

代码完全没错，但写完后我傻了眼。我怎么知道用户什么时候"输入完"用户名，总不能用户在用户名的文本框里打个"a"我就弹出这个欢迎注册的窗口吧。我必须确认用户确实把

用户名输入完毕。

可是，用户什么时候输入完成，是他的自由，他可以在早上 10 点打开注册页面，在用户名那里输入"a"，然后到夜里 12 点，再把后面的"b"给加上。我们需要用户给我们一个反馈，代表他的输入已经完成。这就是注册页面中，最为经常出现的"注册"按钮。

```
<input name="btnRegister" type="button" onclick="welcomemsg()" value="
输入完毕">
```



特别强调一下，读者不要用拷贝粘贴的方法拷贝文档中的代码，因为 word 中的双引号可能是全角的，而代码中的双引号必须是半角的。所以请读者自行书写。

上面是一句 HTML 语句，

它的作用就是：

显示一个按钮（`type="button"`）在页面上

按钮上的名称是 btnRegister（`name="btnRegister"`）

按钮上面写着"输入完毕"（`value="输入完毕"`）

新鲜的语句就是 onclick 这句话。它的含义是，当点击时（`onclick`），执行 JavaScript 的函数 `welcomemsg`（`"welcomemsg()"`）

所谓函数，是具有一定功能并且拥有返回值得代码段。我们不能毫无条例的将代码从头写到尾。我们可以将一个注册页面的代码分为几块，比如，这块处理用户名，那块处理密码，再一块处理用户的所在地等等。所以我们暂且把处理用户名这段代码叫做 `welcomemsg`，它就是函数的名称，而这个函数不需要任何返回值，它只需要完成我们所说的，弹出个窗口就可以。所以我们按照函数的规则，

函数的格式：

```
function 函数名()
```

```
{
```

```
    函数中的代码
```

```
}
```

注意大括号是 JavaScript 中的一个重要的概念，它通常用来包含一个拥有意义的结构（函数，循环和判断语句的内容都会用到大括号）

改动一下刚才被我所谓一起呵成写下的代码：

```
<script language="JavaScript">
```

```
    function welcomemsg()
```

```
    {
```

```
        var username;
```

```
        username = document.all.UserName.value;
```

```
        alert("您好"+username+"，欢迎注册");
```

```
    }
```

```
</script>
```

把那个按钮的语句加入到注册页面中，并写好以上那段函数一并加入到页面中。（JavaScript 的代码可以放到<head>里面）。当屏幕出现欢迎注册的提示框时，代表着一个简单的输入和输出教程就此可以告上一个段落咯。



HOMEWORK: 改动注册页面。实现效果, 当用户点击“输入完毕”的代码后, 将用户的资料组成字符串弹出来, 格式自定。包括用户名密码和等所有文本框的信息。

开始学习的朋友, 对于如何使用 JavaScript 获得下拉式菜单或者 radio 单选框的动作, 都存在一些疑问。一般来说, 触发的条件, 仍然会使用 `onclick`。而难点是这两个控件, 根本没有什么 `value` 之类的属性供大家 happy。暂时不给大家介绍的原因在于, 它们其实算是一种“组”控件, 用程序控制“组”控件, 需要遍历这个“组”内的所有元素 (请看不懂这句话的读者暂且忍耐下对在下的愤慨——因为戏肉马上就来了)。除此之外, 还要引入立刻开始介绍的概念——判断语句与循环结构。

3.1.2 判断语句与循环结构

我们想象一下教育孩子的情形 (管你是否拥有小孩, 对比你小的孩子呼来喝去, 是人类的本能)。我们很喜欢说这样一句话, “如果水开了, 你 Y 最好不要把手放进去”。虽然不无恐吓的意味, 不过真的把手放进去, 恐怕也真的不会太好受。值得注意的是这里还用了另外一个重要的概念——“如果, 则”这样典型的判断语句。当纷繁复杂的情况被摆在电脑面前的时候, 我们有必要让它知道, 遇到什么样不同的情况, 该做什么样不同的处理, 而这个, 靠的就是我们要讲述的“判断语句”。

像人类说话一样, 一个判断语句的结构, 在各种语言中, 其实都差不多——如果怎样怎样, 那就如何如何。为了给大家加深印象, 在下将用比较的形式来说

人话: 如果你恨他, 送他到纽约, 因为那里是地狱。(出自影片《北京人在纽约》)

英文: If you hate him, send him to New York, for it is hell.

JavaScript:

```
if (you hate him)
{
    send him to New York;
```

```
}
```

请注意，大括号又出现了。

令人高兴的是，命令计算机做事，从来不需要跟它解释理由，所以我们省略了"因为那里是地狱"这样仅仅用于增加感染力的语句。而 JavaScript 的判断结构也浮出水面来

判断语句的形式

if (判断条件为真)

```
{  
    需要执行的语句;  
}
```

为了马上脱离抽象，走进具体的殿堂来。在下准备了个简单的例子，读者主要注意下判断条件的写法。

程序的功能，是判断 txtUserName 这个文本框栏位，如果用户输入的字母超过 10 个，或者是用户什么都不填，就弹出"非法的用户名"作为提示，反之则弹出"合法的用户名"。

HTML 部分

```
<input name="txtUserName" type="text" value="">
```

```
<input name="btnRegister" type="button" onclick="welcomemsg()" value="输入完毕">
```

JavaScript 部分

```
<script language="JavaScript">
```

```
function welcomemsg()  
{  
    var username;  
    username = document.all.UserName.value;  
    if ((username.length() < 10) && (username != ""))  
    {  
        alert("合格的用户名");  
    }  
}
```

```
    }  
    else  
    {  
        alert("非法的用户名");  
    }  
}  
</script>
```

相信大家可以轻松看出，else 表达的意思就是“除了 if 的判断条件以外的情况”。

判断语句是可以嵌套的。意即可以写成这种形态：

```
if()  
{  
    if()  
    {  
        if()  
        {.....  
    }  
}
```

理论上是可以无限写下去，但是请读者不要吃到这么饱，层次越多，自己看的越乱。条理清晰是写程序所需要遵守的第一原则。

我们现在来看看那个还不算短的判断条件

```
((username.length()<10)&&(username!=""))
```

username 这个前面说过了。是存放着页面控件 UserName 值的变量。而 username.length，则代表着该字符串的长度了。这种点某某的方式（如.length），在 JavaScript 中广泛存在。我们暂且把.length 叫做 username 这个字符串的函数（只是这么叫而已，不要跟前面的函数混淆起来）。这种类型的函数还有很多，他们也都代表着不同的功能。在下把它们总结起来（只列出最常用的），请看此处

[charAt 函数](#) [charCodeAt 函数](#) [concat 函数](#) [fromCharCode 函数](#)
[indexOf 函数](#) [replace 函数](#) [split 函数](#) [substring 函数](#)
[toLowerCase 函数](#) [toUpperCase 函数](#)

除此之外，语句中还掺杂着不少令人作呕的符号。除了我们都认识的小于号以外，还有许多莫名其妙的括弧，和一些不知所云的等号，感叹号这样企图搞乱我们思想的东西。在下面细细把它们说一说。

判断语句，其实离不开逻辑两个字。当我们陈述条件的时候，往往会希望这个条件丰满一些，或者说，表达的内容更多一些。所以我们修正一些前面对于判断的那种说法。原来是“如果 A,那么 B”，这种语句对于计算机，是不严谨的，因为 A 的概念非常含糊，并且严格的计算机，不能允许有病句的存在——我们不能说“如果馒头，那么肉包”这样没头没脑的判断句。为了严谨起来。我们稍作改动，句子变成这样：“如果 A 是真的，那么就执行 B”

看起来似乎是差的不太多，实际区别很大。改动过的句子，要求“如果”后面的东西，必须是一个带有逻辑性的语句，它必须是一个合乎语法标准的条件。（说的这么抽象，其实有个简单的鉴别方法，就是你把它翻译成人类语言，不是病句，就一切 OK）

有鉴于此，需要引入了一些逻辑上的修饰词汇来对条件进行更深一步的描述——而如今纷繁复杂的逻辑世界，其实逃不脱“与，或”二字，它们如同太极八卦的阴阳一样，掌控了逻辑世界的全部奥妙。所有放在判断语句“如果”后面的假设条件，都可以用一些简单的小事件通过此二字连接而成：

如果你是人，并且是男人，那么你一定喜欢女人。

虽然这句话在当今社会已经不完全对了，我们却仍然可以注意到“是人”与“是男人”被奇妙的牵引起来，而如果说的更详细一点，就是：

如果你是人这件事是真的，并且你是男人这件事也是真的，那么你一定喜欢女人。

我们当然不必事必躬亲的加上"是真的"这个仅仅为了说明逻辑关系的无聊词汇（在 JavaScript 中，也是可以省略掉此类的说法）。

"或"的例子我也不再举，想必大家都是聪明人。我们把判断语句中的逻辑条件，人话与 JavaScript 的区别列举如下：

人话

JavaScript

是，等于

`==` 注意是两个等号

不是，不等于

`!=`

与

`&&`

或

`||` 这也是两个同样的字符。该字符每个键盘的位置不同，通常是在反斜杆那个按钮上。

我们可以把逻辑符号，理解成四则运算符号那样。只是它代表的是一种逻辑关系，就像小时候，乘号总是稳吃加号一样，逻辑符号之间，也存在着优先级关系。表示逻辑相等和逻辑不等的 `==` 和 `!=`，优先级就会低于与、或（`&&`、`||`），同样的，逻辑符号想提升自己的地位，也只能通过括号来完成。这就解释了刚才的那句话，为什么那么多奇怪的括号。除了 `if` 本身需要带的一对括号以外，其他都是用于提升优先级。

对比如下：

人话

如果你是人，并且是男人，那么你一定喜欢女人。

JavaScript

```
if ((你==人)&&(你==男人))
```

```
{
```

```
    一定喜欢女人；
```

```
}
```

逻辑关系一目了然,此时再看最前面那个控制拥护名长度的判断语句,也就相当明白了。



HOMEWORK: 撰写一个页面。上面有 3 个文本输入框与一个按钮。你作为用户,分别在框中输入 3 个数字,当用户点击按钮时,计算出前两个数字相加是否和第三个数字相等。把结果以弹出窗口的形式显示。

大家都从刚才的例子里知道,教育小孩,要使用判断语句-_-。只是如今的小孩子管教越来越麻烦,拿个鸡毛掸子正襟危坐就唬住他们的时代,已经一去不复返了。我们除了频繁的要灌输判断语句外,还需要另外一项重要的技能——重复。

所谓重复,就是我们马上要讲到的循环结构。循环并非简单的重复,它是拥有目的性和条件性的。历史上最出名的重复例子,就由大数学家高斯那个 BT 的班主任所提出。从一加到一百,求其总和。虽然聪明的高同学巧妙的发现规律,但相信其他同学应该依然笔耕不辍的坚持着有"目的性,条件性"的"循环"结构——并没有简单重复,但重复的过程有它的规律。我们就用这个高斯的成名作,来看看一个循环结构。是如何使用 JavaScript 实现。

代码如下:

```
<script language="JavaScript">

    //当两个斜杆出现的时候,则后面的东西是注释,这样的内容不会被解析
    //但是放着也不会报错,通常是给开发者自己看的

    var n,sum; //使用逗号是同时申明两个变量的方法
    sum=0;
    for (n=1;n<=100;n++) //for 语句的重要部分
    {
        //又见大括号

        sum=sum+n; //循环语句的主体,将会被循环执行。
    }

    alert(sum);

</script>
```


我们现在模拟一下计算机的流程，看看代码会被怎样的取执行。

- 1 声明变量 `n` 和变量 `sum`
- 2 将 `sum` 赋值为 0
- 3 开始一个循环，循环的条件是这样的。将变量 `n` 慢慢增加（每次增加 1），只要 `n` 小于或者等于 100，循环都要继续下去。
- 4 循环的主体。作用是把 `sum` 的值加上 `n` 的值（注意，在循环内部里，`n` 是慢慢增加的），并重新赋给 `sum`
- 5 `n` 递增到 101，循环结束
- 6 将结果弹出

除了 3，4 两步，其他应该都浅显易懂，循环的公式化结构如下：

```
for (循环条件) //for 语句的重要部分
{
    循环主体;
}
```

循环主体，就是普普通通的代码（比如 `sum=sum+n;`）相信不用赘述。而循环条件的写法。

似乎就有些光怪陆离。一般的说，循环条件，是由 3 个语句组成。

表达式 1；表达式 2；表达 3

表达式 1 通常用来给循环变量赋值。所谓循环变量，就是将来用于决定循环次数的那个变量，比如我们的例子中，循环变量就是 `n`。也允许在 `for` 语句外给循环变量赋值，最夸张的是，如果你已经在循环外部赋值了，则表达式 1 可以省略。

比如：

```
for (n=1;n<=100;n++)
```

可以写作

```
n=1;
```

```
for (;n<=100;n++)
```

表达式 2 通常是循环条件，一般为逻辑表达式，当这个逻辑表达式不被满足，或者说不成立，循环就会结束。此部分最为紧要，关系到循环是否能够正确地结束。亲爱的读者，写程序的时候，千万要注意到，不要制造死循环。以上面的高斯算术题代码来说，如果你把 `n<=100` 改成 `n>0`，则这个条件永远被满足（因为 `n` 一直大于 0），这样就可以创造一个地道的死循环出来；相反地，使用 `n==0`（请不要忘记刚刚学习到的逻辑判断符，这里是两个等号），条件不成立（因为 `n` 一开始就大于 0），则循环一次都不会被执行。

表达式 3 当第二个表达式执行完毕，就会执行此表达式。其实循环条件中除了第一个表达式是从头到尾只执行一次以外，二三两个语句和循环主体，都是随着循环的次数交替执行，所以刚才的 `n++` 并不神秘，它是 `n=n+1` 的另一种写法而已，所以，我们把刚才的程序执行步骤中的 3 和 4 再度细化，

- 1 将 `n` 赋值为 1
- 2 检查 `n` 是否小等于 100.....表达式 2
- 3 确认小等于 100，执行循环主体
- 4 将 `n` 加上 1，并赋给它自己.....表达式 3
- 5 检查 `n` 是否小等于 100.....表达式 2
- 6 确认小等于 100，执行循环主体
- 7 检查 `n` 是否小等于 100.....表达式 2

按照这个规律一直继续进行下去，直到 `n=101` 的那天来临。表达式 2 的条件不再被满足，成功跳出循环。

为了在给大家店感觉。随手写上几个循环的例子（省略变量声明部分）

计算 10 的阶乘

```
for (n=1;n<=10;n++)  
{  
    sum=sum*n;  
}
```

写 50 遍我爱你，并一次性弹出来。

```
outputstring="";  
for (n=1;n<=50;n++)  
{  
    outputstring=outputstring+"我爱你";  
}  
alert(outputstring);
```

从 1 开始，加到 100，如果和超过 4000 就停下来

```
for (n=1;sum<=4000;n++)  
{  
    sum=sum+n;  
}
```

许多人发现,for 的表现力虽然很强，但是形式稍嫌麻烦。有的时候，不希望有个什么循环变量在那里好死不死的计数。睿智的先辈们显然也看到了这一点，JavaScript 中还提供另外一种循环方式:while,结构简单至极

while (循环条件)

```
{  
  
    循环主体;  
  
}
```

只需要给出一个循环条件，在满足条件的前提下，语句就会被一直执行，什么循环变量，都可以统统抛到脑后，在下仍然以高斯那个家伙的东西来做实验

```
n=1;  
  
while (n<=100)  
{  
  
    sum=sum+n;  
  
    n++;  
  
}
```

结构可以说返璞归真，比起 for 来要亲切易懂很多。在下偏好使用这种形式，而读者当然也是仁者见仁，智者见智，哪个爽选哪个吧：)

把 while 交代以后，就告别小节结束，给众位读者一个小小的演练机会



HOMEWORK: 编写一个网页，提供用户名的输入。如果用户输入的字符中含有 a, 则告知用户名不合法。(可能需要用到 `substring`, 请读者仔细研读关于字符串函数的内容)

3.2 JavaScript 的高级部分

用那么短的一节来介绍别人要写一本书的内容。相信读者都会觉得怪怪的，仿佛学到了点什么，其实又空洞无物。在下已经不厌其烦的声明，我并非在编写一本四库全书，或者是辞海之类的全面书籍，我仅仅是在写一个目录。如果你发现解决一个简单的问题时用了大量的代码，那么你就要去考察一下是否存在一些 JavaScript 自带的函数没有被用上。在下本准备结束此章，良心却始终过意不去，终于准备将压箱的宝贝拿出来献丑。

3.2.1 基于对象的 JavaScript 语言

看到**对象**二字，相信许多读者已经开始暗暗摇头，仿佛在下已经开始一步步地走向俗套。并开始用传统的枯燥文字开始讲学。在下也是不太愿意如此，只是如果不掌握这个，就没法迈进 JavaScript 的高级殿堂。只能做到尽可能的通俗易懂，希望读者谅解。

那么，什么是对象呢？

我们可以简单的认为，对象就是一种类型。每一个对象。都是由属性(properties)和方法(methods)两个基本的元素构成的。其中的属性二字，相信大家都不会陌生，它广泛的存在于前面提到的 HTML 的元素中。我们把输入框这一类东西，叫做一个对象，我们把按钮这类东西，叫做对象，我们可以把任何类型的东西，都叫做对象。这么叫，并非只是仅仅一个称呼，我们要在写代码的时候用上他们。比如，字符串对象拥有 length 属性，意味着所有的字符串，都有 length 属性，我们可以在任何一个字符串后面加上".length"来查看它的长度和大小。而字符串的方法，就是我在前面反复提到的所谓"字符串函数"。你会发现，所有的字符串都可以使用**.substring** 来取得子字符串，这正是因为，substring 就是字符串对象的方法。

上面的是纯理论内容，颇为绕口，且在理解上还有所困难，而事实上，对象的属性和方法才是我们需要关心的内容，比如我们会关心到底字符串会有哪些方法可供我们操纵，纯理论的内容，只是让我们在编写程序的时候保持意识上的清醒。对于代码的掌握也会高一个层次。

让我们看看下面的内容

算术函数的 Math 对象

功能：提供除加、减、乘、除以外的运算。如对数，平方根等。

1 主要属性

Math 中提供了 6 个属性，它们是数学中经常用到的常数 **e**、以 10 为底的自然对数 **ln10**、以 2 为底的自然对数 **ln2**、3.14159 的 **PI**、1/2 的平方根 **SQRT1-2**、2 的平方根为 **SQRT2**。

也就是说，我们可以直接使用 **Math.e**，**Math.PI**（注意大小写）直接在计算中当作一个常量使用。比如下面这段代码：

```
function testcode()
{
    var radius;
    var acreage;
    radius=5;
    acreage=Math.PI*radius*radius; //计算半径为 5 的圆面积
}
```

2 主要方法

绝对值：Math.abs()

正弦余弦值：Math.sin(),Math.cos()

反正弦反余弦：Math.asin(),Math.acos()

正切反正切：Math.tan(),Math.atan()

四舍五入：Math.round()

平方根：Math.sqrt()

基于乘方次的值：Math.Pow(基数,此方数)

下面的这段话很重要，当我们在一个函数或者方法的括号里写上变量的时候，我们就称之为参数。它的作用是，将外部的数据传递到函数内部供函数计算。我们看上面那个计算圆面积的函数 `function testcode`，很多时候我们并不想仅仅是把半径为 5 的面积算出来就完了，我们希望，调用 `testcode` 的时候，将半径传入到函数中，然后在返回给我们面积。要求应该不算过分。我们看看经过改造后的代码

```
function testcode(radius)
{
    var radius; //由于 radius 已经被当作参数传进来，则下面的代码就不需要再声明
    var acreage;
    radius=5; //我们不再需要为 radius 进行赋值
    acreage=Math.PI*radius*radius; //计算圆面积
    return(acreage); //返回圆的面积
}
```

将来如果想计算半径为 5 圆的面积。可以简单的象这样：

```
var a;
a=testcode(5); //将 testcode 函数返回值赋给变量 a
```

为什么突然提到这个？`Math` 这个对象的方法，让在下无法再回避该问题。我们可以看到，`Math` 的方法，基本都是需要参数的。它的整个调用方式，和函数基本一模一样，因为说到底，方法就是一种函数。我们可以轻松的这么做来获得 10 的平方根：

```
a=Math.sqrt(10); //注意参数要符合要求，比如这个方法，如果传入-1,就会出错
```

由于 `Math` 是系统的内部对象，因此，实际书写的时候，我们可以省略掉 `Math` 属性或方法前面的 "`Math.`"，如 `Math.sqrt(10)` 可以简单的写成 `sqrt(10)`

日期及时间对象

功能：提供一个有关日期和时间的对象。

Date 对象没有提供直接访问的属性。只具有获取和设置日期和时间的方法。

1. 获取日期的时间方法

`getFullYear()`: 返回年数

`getMonth()`: 返回当月号数

`getDate()`: 返回当日号数

`getDay()`: 返回星期几

`getHours()`: 返回小时数

`getMinutes()`: 返回分钟数

`getSeconds()`: 返回秒数

`getMilliseconds()`: 返回毫秒数

和前面的 Math 方法一样, 这些方法同样是需要传入参数的。但是我们知道数字可以作为参数传入, 如果想传入日期, 是不是对格式要有所要求? 答案是肯定的。我们需要事先声明一个日期变量:

```
var MyDate;
```

```
MyDate=new Date()
```

可以看出, 这样声明的日期变量, 应该是一个默认值 (1770/1/1 0 点 0 分 0 秒)。此时取它的年月日或是小时分秒, 毫无意义。所以, 一般来说, 获取日期时间的方法, 通常要和设置日期和时间的方法进行配合。

2 设置日期和时间:

`setYear()`:设置年

`setDate()`:设置当月号数

`setMonth()`:设置当月份数

`setHours()`:设置小时数

`setMintes()`:设置分钟数

`setSeconds()`:设置秒数

`setMilliseconds()`:设置毫秒数

只有设置了时间和日期，将来经过变化后，获取的方法才有意义。设置的方法很简单。

下面这段代码：

```
var MyDate;  
MyDate=new Date();  
MyDate.setYear(1999);  
MyDate.setMonth(10);  
MyDate.setDate(10);
```

执行以后，MyDate 就代表着 1999 年 10 月 10 日。

如果嫌麻烦。这样的语句也可以达到一样的效果

```
var MyDate;  
MyDate =new Date(1999,10,10)
```

这时候，如果使用

```
a=getMonth(MyDate);
```

变量 a 就会被赋值为 10

上面介绍的一些对象，对将来的程序会有比较大的作用。而 JavaScript 还有很多其他

的对象等待读者的发掘，在下懒惰，只能介绍到此，高级的对象和它们的方法，需要读者自己查询帮助或者别人的代码进行学习。师傅引进门，修行靠自身嘛。

3.2.2 JavaScript 中的数组

在下在初学程序的时候，非常的不幸，从来没有人跟我说过数组。在下凭着一股傻气，居然也杀出了一小片天地（原因是在下也一直做着“一点技术含量都没有的事情”）。后来对程序越来越喜，竟然发狠开始往深处研究程序（这在以往是绝无仅有的事情，在下的懒惰足以毁灭在下的任何热情和对未来的憧憬）从而才慢慢接触数组这种东西。可以负责任的说，程序中的每种东西，都有它存在的价值，我们先来看看，如果没有数组会有什么结果。

假设现在页面上有 10 个输入框，用户可以在输入框中输入数字（假设页面上 10 个输入框的 name 是这样的:txtNo1,txtNo2,txtNo3,txtNo4,txtNo5,.....一直到 txtNo10。

我们先把这些值都取下来（用以往学过的知识）

```
var a,b,c,d,e,f,g,h,i,k  
a=document.all.txtNo1.value;  
b=document.all.txtNo2.value;  
c=document.all.txtNo3.value;  
d=document.all.txtNo4.value;  
e=document.all.txtNo5.value;  
f=document.all.txtNo6.value;  
g=document.all.txtNo7.value;  
h=document.all.txtNo8.value;  
i=document.all.txtNo9.value;  
k=document.all.txtNo10.value;
```

本来挺爽快，此时在下说，请把这 10 个数，排个序.....不知道读者准备怎么处理。最

智慧的读者。也得周旋于这 10 个变量间,进行焦头烂额的处理。比如你仅仅取出上面的 a,b 由小到大的排序:

```
var temp
if (a>b)
{
    temp=a;
    a=b;
    b=temp;
}
```

这样,a 就会存放较小的数,而 b 则存放较大的那个。我们可以看到,由于变量的名称关系,我们不能像阿拉伯数字那样去编写循环,然后把 10 个数都两两比较一番,而每次比较,都要用掉几行代码。(上面的例子明确的显示出这一点),从数学上计算出,10 个数字两两比较的话,要比对 $10 \times 9 / 2 = 45$ 次(不要问我为什么。如果不知道的话可致电你初中的数学老师,联络感情的同时也必定能得到答案),如果写程序时如此枯燥乏味的事情,保证天下程序员有 90%都要提着裤子逃跑的。

为什么刚才说"由于变量名称的关系"?我们可以注意观察,10 个数字两两比较的话,如果我们给 10 个数字编号,比如 1 到 10。比较的过程我们可以如此描述:

- 1 将第一个数和序号为 2 一直到 10 的数进行比较
- 2 将第二个数和序号为 3 一直到 10 的数进行比较(2 号数已经与 1 号数在第一个过程里比较过了)
- 3 将第三个数和序号为 4 一直到 10 的数进行比较
-
- 4 将第 n 个数和序号为 n+1 一直到 10 的数进行比较(当然,n 最大为 9)

整个比较过程,由于编上了序号,让我们看起来,似乎像一个前面学过的条件循环。而我们如果可以通过循环语句 把本来需要书写的上百句程序语句 缩短到 10 行以内解决掉,

人生显然会变得美好许多。为变量编号的过程，其实就用上了数组的定义。所谓数组，我们大致就可以理解成一组编上序号的变量。

数组的定义语句如下：

```
var myarray;  
  
myarray = new Array(); //注意大小写
```

此后，myarray 就代表着一个数组。我们可以这样给 myarray 变量赋值（以上面那 10 个数的赋值过程为例子）

```
myarray [0]=document.all.txtNo1.value; //请注意，数组是从 0 开始编号的  
myarray [1]=document.all.txtNo2.value;  
myarray [2]=document.all.txtNo3.value;  
myarray [3]=document.all.txtNo4.value;  
myarray [4]=document.all.txtNo5.value;  
myarray [5]=document.all.txtNo6.value;  
myarray [6]=document.all.txtNo7.value;  
myarray [7]=document.all.txtNo8.value;  
myarray [8]=document.all.txtNo9.value;  
myarray [9]=document.all.txtNo10.value;
```

一旦赋值完毕，我们马不停蹄的开始比较工作，将数字由小到大排列，并把最小的数放在 myarray[0]中，次小的放在 myarray[1]中，以此类推：

```
var ij;  
  
for (i=0;i<8;i++)  
{
```

```
for (j=i+1;j<9;j++) //请注意 j 的起始值是 i+1
{
    var temp; //存放两个数交换时的临时数据

    if (myarray[i]>myarray[j])
        //如果 myarray[i]>myarray[j] , 意味着顺序需要调换
        {
            temp=myarray[i];
            myarray[i]= myarray[j];
            myarray[j]=temp;

            //3 个语句借助临时变量 , 完成了 myarray[i]和 myarray[j]的顺序调换
        }
    }
}
```

比起前面说的数百行代码，无论从结构上还是理解程度上来说，都胜了不是一星半点。数组的好处在此暴露无遗。我们甚至可以利用它再度简短代码——HTML 有一个特性，如果一种类型的页面元素用同样的 name,则它们会变成一个序号由 0 开始的页面元素数组。我们把 10 个输入框都用 txtNo 这个名字。则赋值语句就可以简化为：

```
for (i=0;i<10;i++)
{
    myarray [i]=document.all.txtNo[i].value;
}
```

在下私下里不太推荐这么做，因为让 HTML 来为你编号数组，由于不显示在代码里，很多时候会不太清楚哪个编号对应哪个页面元素，此种受制于人的感觉窝囊至极，而且在正

式介绍服务器端代码前,也不太可能在页面上放下几十个同样类型的输入框让用户输入(毕竟这也是要用 HTML 代码写的),所以暂时还是希望读者们在赋值语句上不要太过的偷懒。



HOMEWORK: 这个作业有些难度,读者如果实在无法完成也不要自怨自艾:
模拟真实的乘法计算过程计算 100 的阶乘(由于供计算的数字型变量有大小限制,所以结果是用乘积来存放)

3.2.3 附加内容:算法



警告,由于本章艰涩难懂,18 岁以下青少年请在家长陪同下观看

想要把算法在一天之内说清楚,是基本没有可能的事情。大学里一本"数据结构",已经可以让充满智慧的理科生们哀嚎遍野,在下本人自诩聪明,也没敢怠慢这计算机系的压轴科目。

说起来,数据结构的东西,并不复杂,生吞活剥的话,一天半时间,应该也可以收拾得服服帖帖。只是要完全掌握并灵活应用,没有几千行的代码基础,却绝对拿不下来。那么说到这么多,究竟,算法是什么呢?

所谓算法,归根到底来说,其实是运用有限的计算机语言,来解决实际问题的一种途径和方法。简单的来讲,即使是从 $1+2+3+\dots+100$ 这种实际问题。计算机也可能被编写成两种形式。即真的一个一个从头加起,和 $(1+100)*100/2$ 这两种。亲爱的读者们,一定清晰的认识到,计算机的极大优点,就是对于枯燥乏味的事情,勇敢地重复,并且从来不出错。我一度也被这种任劳任怨的性格所打动,所以尽量的把脏活累活都交给它来做,而很少考虑

到,某些代码,效率的低下已经到达令人发指的程度。为此,我准备把最经典和著名的一些算法,提到日程上来,与大家共同分享。

著名的排序算法

相信刚才的那番排序,读者还有深刻的印象.....真是相当的简单啊。不知道这涉及了什么算法,居然要被提到日程上来。占用读者宝贵的时间。我们首先来看看,人来处理这个问题,都会怎么做(不排除有个别读者,独辟蹊径,希望用更繁琐的办法来解决)

每次都取出所有数中最小的那个,把它抽出来,放到一边,如此往复,第一次拿到最小的,然后是次小得,最后是最大的那个。亲爱的读者,这种方法对我们来说,做起来,绝对比上一种方法简单,因为我们的大脑运算速度太快,能够在很短的时间里,找出一组数中最小的。读者在想,这个事情,实在太过 easy,即使交给计算机,它也不会有太多怨言。毕竟身为人类的我们,都不认为它有多繁琐。电脑更应该甘之如饴了

那么在下请问大家,找到最小的数,该怎么找?

反应迅速的读者,立刻回应了过来,看一眼就知道阿。是的,因为数量仅仅是 10 个。如果今天摆出的是 10000 个数,看一眼,显然就没那么简单了吧。我们通常的"看"法会这样,从第一个数开始,记住第一个数,然后继续往下看。如果遇到比第一个数小的,则抛弃第一个,将这个稍微小一点的数记下来,如此一直"看"到最后,一共必须进行 10000 次的比较,就可把最小的数字找出。所以,此项工作的工作量,将是

$10000+9999+9998.....+1=50005000$ 次比较

虽然数字看起来比较恐怖,但毕竟我们的对象是 10000 个随机数据。而刚才的整个过程,就是著名的**冒泡排序法**。取出最小的那个数的过程,可以称之为"冒泡"。它的工作量是 $n(n+1)/2$ 。其中 n 就是需要排序的数字个数。这似乎就是刚才那个数组章节,所在是用的方法。冒泡排序在计算机界广泛的被使用起来,因为它逻辑简单,并且速度也并不算缓慢。

聪明而无聊的人们,意识到,我们并没有充分的利用排序过程中的所有信息量。或者说,

我们利用信息的态度不够吝啬。经常上网的朋友，一定曾经遇到一个很经典的数学问题，即有 12 个球，其中一个在质量上有问题（反正和其他球不一样重，但究竟是偏重还是偏轻，则无从得知）。题目要求利用天平，3 次将该球找出。我们这里不能浪费大家宝贵的时间去讨论这个与程序算法基本没有太多关系的题目，但是此题的解法，却给我们一个警醒，它非常充分的利用了上一次的結果，让一个近乎不可能解决的问题得到了解决。

我们深入的研究可以发现，找最小数的过程其实满浪费的。我们做了那么多次的比较，仅仅是为了找出那个最小的，实在是太过奢侈。

假设有这么 7 个数字，序号已经被我们编成 1-7。我们来看看上一章冒泡排序的弊端。

7 个数字为：

49, 38, 65, 97, 76, 13, 27

第一个循环的结果：//将第一个数字和其他 6 个数字进行比较，并进行交换

13, 49, 65, 97, 76, 38, 27 //第一组比较将最小的数放在了最前头

而期间我们也得到了一些信息：

38 > 13, 27 > 13, 其余的数字大于 38, 38 和 27 则还没有机会比较 //这是 6 次比较过程中得到的信息

第二组比较后的结果：//将第一个数字和其他 6 个数字进行比较，并进行交换

13, 27, 65, 97, 76, 49, 38

仔细考虑第二组的比较过程，我们会发现，49 和 38 又做了一次比较。而事实上，我们早已经知道，49 在第一组的第一次比较调换中已经败下阵来了

不用再继续下去，结论已经很明了：**冒泡排序虽然算法简单，信息的利用却不够完全。**

1962 年有一个生活比较无聊的大哥，提出了使用分治法进行排序。我们来看看，什么是分治法：

1 随便找一个基准数字（一般是使用需要排序的数列中的第一个），把数列中，

小于该数字的，都放到左边，而大于该数字的放到右边。

2 分别把左边和右边的两个数列，都按照第一步的方法重新划分。将得到的子数列不停的按照此种规则划分下去。

3 一直到分无可分为止，排序即宣告结束。

从第二步已经隐约可以看出，划分到左边的数字，基本已经失去了和被划分到右边数字见面的缘分。一个 100 个数字的无序数列的排序任务，也许就变成 29 个数字的排序任务加上 70 个数字的排序任务（根据基准数字的选择，划分结果会有所不同，且基准数字不再参与排序）。空说无凭，我们还拿刚才那 7 个数字进行举例：

7 个数字为：

49, 38, 65, 97, 76, 13, 27

第一组：

第一次划分的结果：//以 49 作为基准数字

38, 13, 27, 49, 65, 97, 76

第二组：

分别对两个子数列进行划分

子数列 1：38, 13, 27 的划分结果：13, 27, 38 //38 为基准数字

子数列 2：65, 97, 76 的划分结果：65, 97, 76 //65 为基准数字

第三组

再进行 2 次排序，排序过程即宣告完成，这就是著名的快速排序。

比较次数也立刻可以统计出来：

第一组 6 次比较，第二组 4 次比较，第三组 2 次比较，一共是 $6+4+2=12$

而冒泡排序： $7*6/2=21$

优劣程度可见一斑

一个良好的经过考虑的算法，可以大幅度提升运行效率。这就是学习算法的目的。也许某些读者已经看到头晕，但是噩梦仍未结束——

递归的探讨

把这个放到 ASP 的介绍里，连我自己都有些犹豫。因为说到底，目前为止，ASP 中的语言，只是非常简单的脚本语言。除了一些基本的元素外，实在是能省则省。在下却认为，如果许多读者也许将 ASP 作为踏入程序殿堂的一块垫脚石，则不能把程序的魅力完全展现给读者，是在下的失职。

在下擅自决定，使用一个自行编写，而又不大成熟的例子『五子棋』作为例子，因为在下曾经和几个程序算法上，造诣都不是很深的同仁们，出于好玩的目的，各自编写五子棋的 AI，然后进行比赛，在下总算对这个有那么一点点地发言权。比赛最后是在下以微弱的优势胜出，仅仅因为在下在算法上使了一个小小的心眼。

我们那个时候的五子棋其实没有什么 AI 可言。说到底，就是一个“活三工具”^①。所谓活三工具，就是拼命的制造活三，务求让走棋的人看不过来，终于出现漏掉的情况，从而失误。而程序本身，也在制造活三的同时，不遗余力地封锁着敌人的活三。可以想见，只要编写者本身的局势判定没有问题，则两者写出的“活三工具”就是棋力相当的两个对手，除非运气好，否则整盘落满，也不会杀死对方。后来对弈起来，证明确是如此。

电脑思考的特点，就是在于它会按照我们设定的规则去进行。所以，人的对弈千变万化，而电脑之间的对弈却只有几种套路。想到了此点，在下特地在棋局中加入了如下判定：每次进行对弈的时候，记录整盘棋局以及最后的胜败结果。如果下次对弈，前 5 步都和前面曾经失败的某盘一模一样，则在选择应该于何处活三的时候（很多情况，想要活三会有不同的走法），故意避开曾经导致失败的那个位置。仅仅因为这个小设定，在下的程序让其他同仁一筹莫展并最终放弃了比赛。而在下在心中窃喜的同时，妄图仔细思考这个活三工具，想从算法的角度上，对其进行仔细的分析 and 改造。

首先，电脑应该对局势有一个正确的判定。这是最基本的要求。电脑必须知道，现在的局势，己方对于敌人的威胁程度，和敌人对己方的威胁程度。而衡量这个，从算法角度上来

说,并不是一个困难的过程。具体实现,可能要牵扯到语言的范畴,而看到此章节的某些朋友,也许仍然没有此基础,在下在此卖个关子。有兴趣的朋友可以自行到在下的网站上下载源代码。

有了这个基础,其实算法仍然停留在相当低级的程度。"活三工具"的最大缺点,就是对于人有目的走出的招数,缺乏基本的判断。其中最为基本的就是"双活三", "双活三"还未形成前,我的程序不能有预见性的看出它的潜在危险性,而仍然自己顾自己的走,等"双活三"一旦形成,再开始堵就太迟了。换句话说,只会分析当前局面的程序,视野只有一步。

有鉴于此,我们即将对程序进行大手术。而读者也可以从这里,了解到算法的另外一个重要部分:**递归**

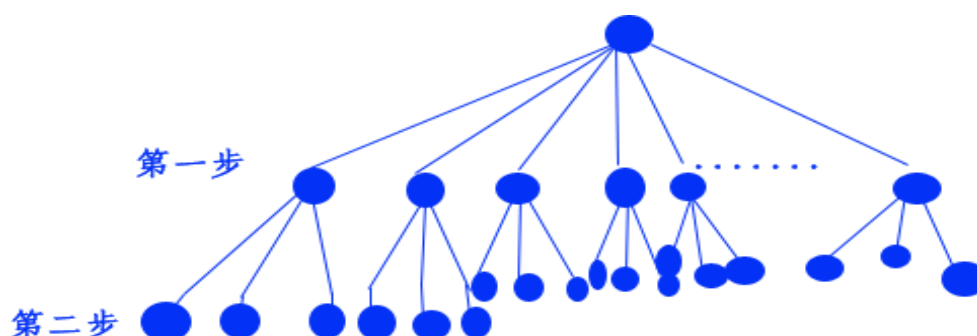
要讲述递归,我近乎遇到了无法逾越的困难,因为如果说递归,则必须把程序放进来,而这又违反了我写此教程耀通俗易懂的决定。思前想后,还是决定,用最口语话的方法和最简单的例子,进行概括,只求让读者对递归,有一个粗浅的认识。

回到五子棋上来,为了提高程序的视野,我们必须这么做,让程序模拟敌人走一步棋,形成一个新的棋局,然后程序分析该棋局对自己的威胁程度。同时,程序模拟自己,在棋局中走下一步,同样也进行敌我情况的分析。这里,遇到的比较恶心的事情就是,如何模拟,我怎么知道对方要走什么,对方的意图并没有那样的明显。

考虑到计算机的速度,实在是快的莫名其妙。我大胆的让计算机将 15*15 的棋局,凡是空下来的位置,都进行了缜密的分析和思考:P。虽然最后电脑显然发现了我的险恶用心,它用铁一般的事实告诉我,如果不注意算法,即使把深蓝搬过来,也是于事无补的。

我们且看看,如果用刚才在下提出的笨方法,电脑需要做多少的工作。

假设视野为 3(考虑 3 步是一个棋手的基本素质),而现在棋盘上已经有了 10 个棋子,轮到电脑走。由于拥有 $255-10=245$ 个空位,这趟漫长的棋局分析之旅,就是以 245 作为起始基数。



如图所示，步数越多，所需要考虑的情况也就越多。这个结构的顶端，只有 245 种情况，而下一步，就会有 245×244 种情况需要计算机来考虑。每多走一步，都是数量级的增加。

那么，所谓的递归，这里就可以有一个说明。我会特别的放一段代码，进行棋局的分析工作。这段代码的作用，就是根据规则，模拟下一个子（无论是模拟电脑还是模拟敌人），然后把落子后得整个棋盘，传递给它自己-----其结果当然是，如果没有规定视野是多少的话，子越落越多，直到最后无子可落为止。注意，模拟下的这个子，只是所有需要进行分析棋子的一部分。换句话说，代码自己调用自己，而这个自己调用自己的过程，又被放在一个循环里。递归就是这样的一种思路，它让程序把一条路走到底，然后慢慢的回头，以达到把所有的路线探索完毕的目的。

如果读者完全看不懂上面那段话，我只能再用另外一种具体的方式进行阐述，请读者继续看下去。

假设视野为 3，且落好 10 个子的判断，将一共有 14526540 这么多种情况需要分析。而递归就提供了这么一种手段，让我们通过语言描述这 14526540 种情况的时候，轻松一点。代码的执行过程将会是这样

- 1 模拟敌人落子。 //此时是第一层
- 2 模拟己方落子应对。//此时是第二层，它的棋局由第一层演变过来
- 3 模拟敌人落子。 //此时是第三层。它的棋局由第二层演变过来
- 4 电脑分析局势

- 5 一次分析过程结束
- 6 模拟敌人落子。 //此时是第三层，但是它将是不同于刚才步骤 3 的落子，因为需要考虑所有情况，所以更换其它位置落子，进行新的局势分析
- 7 电脑分析局势
- 8 一次分析过程结束

.....

一直到第三层所有该考虑的棋子，都考虑过一遍。

回到第二层，找一个和步骤 2 不一样的棋子，继续进行庞大的分析工作

当所有的情况，都被电脑找过一遍后，这次的分析过程就告结束。而递归的代码也全部运行完毕。

总结一下。递归需要

- 1 一段自我调用的程序
- 2 一个结束条件

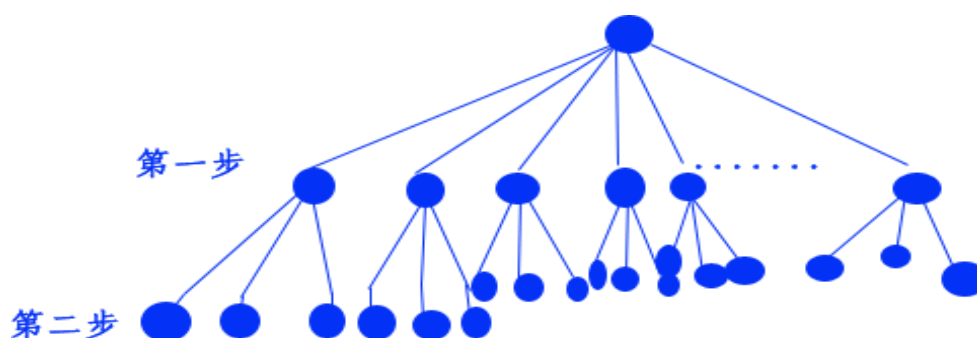
刚才的结束条件，就是我们反复提到的"视野为 3"，就是当递归进行到第三层的时候，无论局势如何，都不再自我调用下去，而是将属于第三层的所有分析做完，然后乖乖的回到第二层，具体的实现，不在此写出，以免干扰还在慢慢理解中的读者。

2.1 树和剪枝

我们当然不希望，2.2 节说的代码把棋局模拟到落满为止，起码，如果发现自己或对手获胜，这个循环，已经完全可以终止。而视野，也是减少循环个数的一个重要手段之一。

但是，我们做的，真的已经足够了吗。仅仅是计算 3 步，就需要考虑 14526540 种棋局。对资源，显然是一种极大的浪费。我仔细思索了以后，对算法又进行了改进。

首先把"树"的概念介绍一下。再把刚才那副图找出来。



最上面的那个点，我们叫它做“根”，

而每条拓展出去的分枝，则可以叫“树枝”，同时，树枝的顶端，我们称呼它为节点。

我们可以把它看作，一个雌雄同体的家伙，生出一大堆娃娃了。娃娃再生小娃娃，子子孙孙无穷匮也。而每个娃娃，都是一个节点，且除了那个最早的老大“根节点”外，每个节点都有它的父亲（父节点），他也有一些亲兄弟（兄弟节点），至于他的伯父叔叔这些，我们暂时不予考虑。如果读者疑惑为什么用父节点，和兄弟节点这种似乎歧视女性的叫法，这个我也不是很清楚，在下尊重妇女同志已经不是一年两年了，此次实在是规矩使然，在下也无可奈何。

这颗树的一共有 14526540 条最小的分枝，也就是最底层的节点。数量庞大到骇人听闻的地步。电脑虽然任劳任怨，我们做主人的，也不能不负责任地把这么多种情况，统统扔给电脑分析。据当时我当时在 AMD 雷鸟 750 的测试结果，当数量超过 50000 以后，速度已经令人无法忍受。一千多万的情况，都不敢让电脑跑完。

稍微留意一下，可以注意到，就这么把 200 多格都摊进去，实在是太浪费了。没有人下棋的时候东下一个西下一个的胡搅蛮缠。所以，将**这颗树的枝条剪去**，让最后的枝条数目减少，就是首要的工作。

稍加思考后，我把程序搜索的范围，减少到了棋盘上已经拥有的子，方圆 2 步的空间里。举例来说，假设棋盘上原来只有 1 个子，那么所需要搜索的个数就是 24。瞬间把数量级减少了一位，速度的增加自然是不言而喻。

除此之外，视野为 3 的判定，也可以进一步压缩。也许，电脑在模拟走下一步后，敌人已经是死局，此时，如果再模拟敌人落子，就毫无意义。如果按照原来 14526540 那种算

法，这一下，就可以省却 $244*243$ 种局势的判定，而这里，还有一个很有意思的部分。

如果电脑在计算到第三步（即模拟自己落子，模拟对方落子，再模拟自己落子）的时候，发现可以杀死对方，那么聪明的读者，是否知道，这样的信息，对于剪枝有什么样的帮助呢？

我一开始也没有想通这一层。杀死对方就杀死对方嘛，接着寻找其它枝条就可以，何必要大的剪枝动作呢？说什么树啊，剪枝阿显然很麻烦。我们把自己放在电脑的位置上来进行思考，问题应该就明朗了。

我走一步，敌人走一步.....

此时我发现，有一个位置，只要落子，敌人就挂了。作为我们。当然把这个位置，作为这一层考虑的终结，既然杀掉他的棋，何必考虑其它位置呢。所以，此次剪枝，就是将这个胜利节点的所有兄弟都咔嚓掉，这些兄弟对于我们已经没有任何利用价值：)

我们能不能再贪心一点呢.....古往今来善良很勤劳的中国劳动人民，经常教育我们，人的欲望是无止境的，而欲望也是人类发展的最重要动力源泉。我们知道，五子棋走到一定的个数的时候，棋盘上，实际上已经存在一种**必胜**的下法了。必胜下法的出现原因，往往是对方的布局失误，或者是对方漏堵了一些重要的位置。

既然有可能存在，我们就有必要去寻找它，这对该五子棋的水平提高大有裨益。必胜的概念，就是无论敌人走什么棋进行应对，都无法改变失败的结局。因此，摸索该路线时所需要的剪枝，直可以用四个字来形容“大刀阔斧”。

我们把搜索必胜下法的视野，设定成 5。即包括敌人的考虑在内的将来 5 步棋子。为了便于说明，仍然使用最为笨拙的全盘搜索法（即开始说的，把没放棋子的位置，全部考虑一遍），这就大约有 $244*243*242*241*240$ 种棋局，不用把它真的乘出来都可以知道，这样大范围的搜索，毫无意义，耗时耗力。没办法，只好再度祭出大剪子来。

仍然使用换位思考的方法，代替电脑分析棋局

我走一步，敌人走一步。我再走一步，敌人亦然

我下第三步。

直至此步，我都无法杀死敌人。分析已经结束。这个结果直接告诉大剪子说，**我走的第一步，并不是必胜路线的一部分**，整个枝条可以剪去。

这相当于，一刀下去，省掉了 $243 \times 242 \times 241 \times 240$ 种棋局。

换句话说，寻找必胜路线的走法，最坏的情况，也只需要考虑

$243 + 242 \times 241 + 241 + 241 \times 240$ 种情况。大家不必考虑前面那个公式怎么得出。但可以明显地发现到，从一个天文数字，变成一个还算可以接受（约 14 万），靠的就是剪枝的技巧。

至此，五子棋算是介绍结束。最后的必胜路线计算方法，本人还没有实现，概因没有竞争，就缺少了写下去的动力。单单把思想放在此处，算是对读者有一个交待。有兴趣的读者，可以继续研究下去，如果有更好的方法，在下也甚感高兴，毕竟在下的拙作，起到了那么一点点抛砖引玉的作用。

第三天的要学习的内容，似乎已经结束了，第三天的任务可能很重，特别是后面的算法部分，更绝对让人痛不欲生。在下在网站上已经贴出了关于数据结构和算法的完整文章连接，那部分转贴的东西是纯理论内容，没有任何修辞润色，在下最早看的时候，自己都看到抓狂，请读者有些心理准备。

我不是个随便的人，我随便起来不是人，谢谢大家！

第四日：醉仙望月步

没错，似乎越来越难了。许多读者开始对后面几天的教程，充满了恐惧和无奈。学习的过程就是如此，由浅入深，阻力可能会越来越大，在下也无法改变这个事实。学习计算机语言的不二法门，前面也已经一再强调，就是练习练习再练习。唯有等你看到代码时双眼放光，露出饥渴难耐的表情时，方算功德圆满。我们今天的教学，才算正式的踏入 ASP 的殿堂，我们要来揭开服务器端代码神秘的面纱。

4.1 粉墨登场的 VBScript

首先要声明的是（似乎前面已经声明过了），没有任何人规定，客户端代码必须用 JavaScript，服务器端代码必须用 VBScript，这两种语言的作者，都和在下没有亲戚关系，在下自然也不会徇私。只是在下注意到两件事情

- 1 在同一个页面里用一种语言同时写服务器端和客户端时，有时候会让人看到糊里糊涂，特别是语言的编写工具没有正确的高亮显示时。所以很多人的作法是故意用两种语言以示区别。
- 2 一般网站都是用 JavaScript 来写客户端代码。将来我们拿来主义的时候（客户端代码可以从任何的网站上取到，而服务器端代码则不行），就可以很方便的阅读和修改。而第一个原因既然说了两者需要不同，服务器端代码只能留给 VBScript。

我们可以确信一件事，服务器端代码的存在原因有二，一是为了与用户完善的交互，并存储用户的数据，二则是为了安全性，服务器端代码和运行逻辑，是任何网页制作这都不愿意泄露出来的。所以，介绍 VBScript 之前，有必要先讲讲，网页数据的传递。

为了便于大家查阅，放出 VBScript 官方帮助文件的下载：

[VBScript 参考手册](#)

4.1.1 网页数据的传递

所谓网页数据的传递,就是用户这边的静态页面数据,如何传递到服务器端并通知服务器端处理。前面已经说过,从运行流程来看,服务器端的代码要先于客户端代码运行,那么,当用户看到页面时,也意味着,服务器端代码早已经运行结束,此时还如何能传递到服务器端呢?聪明的人们早考虑好这点,他们在网页上特地设计了一种机制,叫做"POST",当客户端 POST 动作被执行的时候,一些指定的数据,就会被提交给服务器端,而客户端这边看到的现象是,页面被刷新,或者转移到另外一个页面去(到底是刷新还是转移,取决于 POST 的目标)

我们来看看,POST 语句生活的地方:

```
<form name="frmRegister" method="post" action="test.asp">
    <input name="txtUserName" type="text" value="">
    <input name="btnSubmit" type="submit" value="提交填写内容">
</form>
```

出现的新鲜物事不多,除了一行 form 和 type 是 submit 的 input 元素以外,其余部分我们都耳熟能详。在下现在来介绍一下给大家认识。

一个页面上可以有許多 form,每个 form 都有它内部的数据。当它们被提交的时候(一次只会有一個 form 被提交),form 内部的内容将会以一种特殊的格式传递到服务器去。该格式就是象这样的:

元素名称=元素的值

每一个我们见过由<input 开头的页面元素,再加上<select>,这些元素,统一称为"表单元素"。只要他们被写到<form>的里面,他们就会随着 form 的提交,被抛送到目的页面去。

form 的各个属性的作用如下:

name : 每个元素都有,就不介绍了

method: 设置如何抛送数据到服务器,有两种方式, **get** 和 **post**。当使用 **get** 的时候,数据的大小被限制在网页地址的最大长度(即 2048)之内,而 **post** 则要大得多,长度近乎没有限制(因为那个限制基本很难达到)

action: 指定抛送数据的目的页面路径。可以写成相对路径。本例用的是 **test.asp**

翻看 DHTML 可以看出 form 还有其他的属性,只是讲述起来太过麻烦,而对读者暂时用处不大,如此按下不表,留给读者将来自行学习。

页面提交的方式,有两种:一是在页面上放置上面写到的这个语句:

```
<input name="btnSubmit" type="submit" value="提交填写内容">
```

该 HTML 将会在页面上显示一个写着“提交填写内容”的按钮,点击该按钮后代码中该句所在的 form 里包含的内容即被提交

另一种则是在 JavaScript 中,调用需要提交的 form 对象的 submit 方法(不要告诉我你已经忘记了什么叫对象和方法),以上面那个 form 为例:

```
<script>
    document.all.frmRegister.submit();
</script>
```

两者的功能是完全相同,区别只是一个通过按钮立时触发,另一个由于是客户端代码,则自由的多(你甚至可以在输入框的 onclick 事件里放上这段代码,那每一次输入框被点击都会提交给服务器)

回头再来看看被提交的格式

前面说过,表单元素里大部分都是用<input>打头的,它们都有一个 value 属性。除了 type="radio"这个异数外,其他元素的 value 属性所代表的东西都是相当明确的。我们后

面将用代码说明这一切。暂时先绕过来，看看，既然抛送了，该如何去接收。

4.1.2 VBScript 的输入和输出

数据的接受。就是服务器端的事情了。曾记否 JavaScript 的代码，需要用<script>元素框起来，用以表示这段代码并非 HTML。其实<script>元素是有它的属性,它的完整形式如下：

```
<script language="JavaScript" >
```

如果不特别书写 language 和 Runat 属性，默认值就是 JavaScript 和在客户端运行。

language="JavaScript"表示语言为 JavaScript，

language="VBScript"则表示代码语言为 VBScript，

如果加上 Runat="Server"表示为服务器端代码。注意没有 Runat="Client"这种写法，因为默认就是在客户端运行。

由上可知，服务器端代码在 ASP 文件里，需要特别的写成这种形式

```
<script language="VBScript" Runat="Server">.....</script>
```

人们都知道这么写的麻烦，因此，马上有人发明了，用简单的<%作为服务器端代码的开始，而%>作为服务器端代码结束这种皆大欢喜的写法。在下心怀感恩的将这种简便的方法再教给亲爱的读者们。当然，出于责任心，在下还是在前面将麻烦的那种给讲了，选择的权力交给了读者自己。（不过我们将在下面的代码中，省却这两个符号，除非为了特别和其他客户端代码分开）

且把闲话休提，我们这节，是想看看服务器端代码，是如何输入和输出的。输入的过程，刚才其实已经讲到一半。一组类似于"用户名=*"的值，被一个提交（submit）的动作，提交到了服务器端页面来。我们现在关心的就是。如何把这些数据，变成程序认识的，可存放于变量中的东西。先看下面这段代码

```
<%
```

注意 Vbscript 的注释方式与 JavaScript 不同，它是以单引号作为注释的开头

```
dim txtusername,txtpassword 'VBScript 的声明语句

txtusername=Request("txtUserName") '取得客户端数据 txtUserName 的值

txtpassword=Request("txtPassWord") '取得客户端数据 txtPassWord 的值

%>
```

先不管代码写了些什么，我们暂时假设，该 ASP 页面，接收了另外一个页面传过来的一组数据，该数据中**只包含**有 txtUserName 这个输入框，和输入框的值。

此时再来看代码，代码中的注释写的很清楚每一句的功用。读者也了解到了，Request 是用来获取由客户端 submit 到该页面来的数据的，在 Request 后加上一个名称，比如本例的 txtUserName，就可以取到客户端抛送数据中该名称所对应的值。所以，当你写着客户端抛送部分代码的时候，就得花点心思考虑这里的接收问题了，否则 Request 后的字符串写什么都心里没数，取值自然也会取错。

本例前面提过，数据只包含 txtUserName，那么多此一举的取名称为 txtPassword 的客户端传递数据，将会如何？读者放心，并不会出现任何错误，仅仅是这样取值的结果就是竹篮打水一场空而已。例中的 txtpassword 将会被赋值为空字符串。

VBScript 的输入，就这么几行字，差不多已经交待完毕。某些读者也许还沉迷在上一章的 var 变量申明里，一时转不过弯。VBScript 和 JavaScript 在结构上有着较大的不同，将会要特别拿出一节进行探讨。我们再回过头来，讲讲**输出**。

弹出窗口，对于服务器端代码来说，当然是没有什么必要的。否则如果 1000 个用户同时来访问，都想弹出点什么，服务器那里就得飞出 1000 个窗口来，这显然不科学。而最重要的是，服务器端代码的输出，给谁看？如果是给服务器自己看，那没有什么必要。普天之下，服务器前面坐个大活人的情况，十分之一耳，为了这么十分之一的人，硬生生的输出点什么来，太不值得。我们还是希望，服务器端代码的输出，仍然是可以给用户看到的。给用户看到的，那才叫输出，而解决的问题就实质上发生了变化，由"服务器端代码的输出问题"转变为"如何在服务器端执行客户端代码"，因为谁都知道，只要可以执行客户端代码，输出给客户实在是一件简单不过的事情。我们来看下面这个例子。

有位站长，想把所有的用户名长度控制，都交给服务器端来解决，一来是他认为他的服

务器举世无双的强横，二是网站访问的人数不多，不想一会儿客户端代码一会儿服务器端代码的麻烦。那么他的代码就会写成这样：

```
dim txtusername 'VBScript 的声明语句

txtusername=Request("txtUserName") '取得客户端数据 txtUserName 的值

if len(txtusername)>20 or len(txtusername)<5 then '判断语句

    Response.write "<script>alert('用户名长度不合规范');</script>"

end if
```

有注释的情况下，似乎并不难看懂。多出来的那句 `Response.write "....."`，就是问题之所在了。所有 `Response.write` 后面的内容，都会被解析成客户端代码。你可以写上 HTML，JavaScript 都没有问题。要注意的是，这段代码执行的效果，仅仅相当于，在一个写好的 HTML 里面插入一句话，执行的动作仍然是由浏览器来触发。我们可以看到下面这段混合代码：

```
<script language="JavaScript">

    function testCODE()

    {

        <% Response.write "alert( 'hello' );" %>

    }

</script>
```

只要 `testCODE` 这个函数不被调用到，里面被添上的 `alert('hello')` 这句话自然就无法被执行。因此，很多读者在调试过程中发现 `Response.write` 并不像直接写客户端代码那么方便，问题大约都出在此。

4.1.3 一个 VBScript 的例子

为了便于讲述，在下特别在此准备了一份 asp 的源代码，供大家消遣娱乐学习使用。在下在所有必要的地方都加入了注释，请大家务必看完它（申明一下，都素在下原创）：

该程序一共分为 2 个文件，一个是直接访问的 index.asp 文件，一个是用于后台处理的 chesscode.asp 文件。请读者自行拷贝编辑到虚拟目录下进行调节测试。

index.asp 的代码 :(VBScript 大小写不敏感 ,所以读者会发现 if 的大小写经常变来变去 ,在下写多了就控制不过来)

```
<html>

<head>

<title>抽棋子游戏</title>

</head>

<body>

<p>

<font color='blue'>

规则很简单~~<br>

大家轮流取子，每次只能从一排中取，当然，哪一排由你决定。<br>

取的个数也没有限制，一下子将整排拿光都没有关系。不过<font color='red'>

谁如果取了所有棋子中的最后一个，就算他输</font><br>

人和电脑轮流下。<br>

没有太多功夫设计界面，读者见谅

</font>

</p>

<table border=1>

<%

'HTML 和 VBScript 混合写的情况将来会经常发生 ,请大家注意颜色 ,HTML

部分我都已经使用蓝灰色，服务器端代码都有阴影

'由于每次取子结束后都会回到该页面，所以页面需要传入参数来告诉它，现

'在每排各还剩多少子

RowI=Request("RowI")
```



```
RowII=Request("RowII")

RowIII=Request("RowIII")

If RowI="" then

    RowI=3 '如果没有取到值，表示第一次玩，就赋予初值

End If

If RowII="" then

    RowII=5 '如果没有取到值，表示第一次玩，就赋予初值

End If

If RowIII="" then

    RowIII=7 '如果没有取到值，表示第一次玩，就赋予初值

End If

'此处的 vbscript 代码用于决定页面上是否显示第一排棋子

If RowI>0 then

%>

<tr>

    <td>    第一排的棋子：</td>

<%

    for intI=1 to RowI %>

        <td bgcolor="black">

            <font color="white">

                <b>棋子<%=Cstr(intI)%></b>

            </font>

        </td>

    <% 'Cstr 是强制字符串转换

        '百分号和等号连写是一种重要的用法，后面要着重拿出来介绍(1)

    next 'Vbscript 的 for 循环，请大家注意用法

%>
```

```
</tr>

<%
    end if

    '此处的 vbscript 代码用于决定页面上是否显示第二排棋子

    if RowII>0 then

%>

<tr>

    <td>    第二排的棋子： </td>

<%

    for intI=1 to RowII %>

        <td bgcolor="black">

            <font color="white">

                <b>棋子<%=Cstr(intI)%> </b>

            </font>

        </td>

    <% next%>

</tr>

<%

    end if

    '此处的 vbscript 代码用于决定页面上是否显示第三排棋子

    if RowIII>0 then

%>

<tr>

    <td>    第三排的棋子： </td>

<%

    for intI=1 to RowIII %>

        <td bgcolor="black">
```

```
<font color="white">

<b>棋子<%=Cstr(intI)%> </b>

</font>

</td>

<% next

'由于循环的次数不同，所以可能每个 tr 中的 td 数量不同

'这在 HTML 中仍然是允许的

%>

</tr>

<%

end if

%>

</table>

<table>

<form name="chesscode" method="post" action="chesscode.asp">

<tr>

<td colspan="2" height="30px">

<!--这是 html 的注释方式-->

<!--colspan 代表这里的一个 td,整合了 2 个 td 的内容，当你要达成这

种效果时：第一行整行显示一些东西，而后几行则分格显示，-->

<!--你就需要把第一行的 td 加上 colspan ,它代表"合并包括该格以后的

n 个单元格"-->

</td>

</tr>

<tr>

<td>

<%if rowI+rowII+rowIII=1 then%>
```

兵败如山倒阿，凡事都要动脑~~

<input type="button"

value="重来" onclick="recreate()">

<%else%>

轮到你走了，你想从第

<input type="text" name="whichrow">排，

拿走<input type="text" name="howmuch">个棋子

</td>

<td>

<input type="button"

value="确认" onclick="checkthenumber()">

</td>

<%end if%>

</tr>

<input type="hidden" name="RowI" value=<%=RowI%>>

<input type="hidden" name="RowII" value=<%=RowII%>>

<input type="hidden" name="RowIII" value=<%=RowIII%>>

<!--以上是用隐藏的客户端元素存放每行的个数数据，由于是 input 元素的一种，它们也将在 submit 时被提交-->

</form>

</table>

<script language="JavaScript" >

function checkthenumber()

{

//将该函数放于此处而不是代码的最开头，是因为下面用到的服务器端变量

RowI,RowII,RowIII 在开头的时候仍未被赋值

//此处略去不写对数字还是字母的判断，有兴趣的读者可以自行研究

parseInt 方法

```
var rowsel,nosel;

rowsel=document.all.whichrow.value;

nosel=document.all.howmuch.value;

if (rowsel>3)

{

    alert("总共只有 3 排");

    return;

}

if (((rowsel==1 )&&(nosel> <%=RowI%>))||

    ((rowsel==2 )&&(nosel> <%=RowII%>))||

    ((rowsel==3 )&&(nosel> <%=RowIII%>)))

{

    alert("请不要超过该行的数目");

    return;

}

document.all.chesscode.submit();

}

function recreate()

{

    window.location="index.asp?RowI=3&RowII=5&RowIII=7";

}

</script>

</body>

</html>
```

chesscode.asp 的代码

<%

Dim RowNo(3)

'注意，VBScript 的变量是可以不申明就使用，但在下不推荐这么做

RowNo(1)=Request("RowI")

RowNo(2)=Request("RowII")

RowNo(3)=Request("RowIII")

RowSel=Request("WhichRow")

NoSel=Request("Howmuch")

RowNo(RowSel)=RowNo(RowSel)-NoSel

'下面是 AI 算法部分。不感兴趣的读者可跳过

Dim temp(3)

Dim varifyresult

varifyresult=false

temp(1)=RowNo(1)

temp(2)=RowNo(2)

temp(3)=RowNo(3)

IF temp(1)+temp(2)+temp(3)=1 then

'VBScript 的判断语句，注意到后面有一个 then

outputstr="长江后浪推前浪，在下服了"

else

intI=1

varifyresult="false"

while varifyresult="false"

'VBScript 的循环，和 JavaScript 中有些类似，

'以 End While 或者 wend 结束

If RowNo(intI)>0 then

RowNo(intI)=RowNo(intI)-1

```
varifyresult=valifyArray(RowNo(1),RowNo(2),RowNo(3))
```

```
End IF
```

```
If RowNo(intI)=0 and varifyresult="false" then
```

```
    If intI<3 then
```

```
        RowNo(intI)=temp(intI)
```

```
        intI=intI+1
```

```
    Else
```

```
        RowNo(intI)=temp(intI)
```

```
    For intI=1 to 3
```

```
        IF temp(intI)>0 then
```

```
            RowNo(intI)=RowNo(intI)-1
```

```
        Exit For
```

```
    End IF
```

```
Next
```

```
varifyresult="true"
```

```
End IF
```

```
End IF
```

```
Wend
```

```
outputstr="你拿棋子后，电脑从第"+Cstr(intI)+ _
```

```
"排拿走"+Cstr(temp(intI)-rowno(intI))+ "个棋子"
```

```
End IF
```

```
Function valifyArray(i,j,k)
```

```
'算法所用函数，VBScript 的函数以 End Function 结尾
```

```
'这个函数完全为算法所存在，对算法不感兴趣的朋友可以只注意一下语法
```

```
Dim RowData(3,3) '数组的声明
```

```
Dim RowNo(3) '数组的声明
```

```
Dim intI

RowNo(1)=i

RowNo(2)=j

RowNo(3)=k

If i+j+k=1 then

    valifyArray="true"

    exit function

End IF

If cint(i)=cint(j) and cint(j)=cint(k) and cint(i)=1 then

    valifyArray="true"

    exit function

End IF

For intI=1 to 3

    Dim temp

    temp=RowNo(intI)

    For intJ=2 to 0 step -1 'vbscript 中的渐小循环

        Dim tempi

        tempi=2^intJ

        If cint(temp)>=cint(tempi) then

            temp=temp-tempi

            RowData(intI,intJ+1)=1

        Else

            RowData(intI,intJ+1)=0

        End IF

    Next

Next

Next
```



```
Dim Sum(3)

For intI=1 to 3

    For intJ=1 to 3

        If RowData(intJ,intI)<>0 then

            Sum(intI)=Sum(intI)+1

        End If

    Next

    If Sum(intI) mod 2<>0 then

        valifyArray="false"

        '这是函数返回值得一种方式，注意不是用 return

        '返回值得格式：函数名=值。然后继续执行，直到 Exit Function，

        '或者 End Function 出现

        Exit Function

    End IF

Next

IF Sum(1)=2 and Sum(2)=sum(3) and sum(2)=0 then

    valifyArray="false"

Else

    valifyArray="true"

End IF

End Function

%>

<script>

    //这段已经是客户端代码了

    alert("<%=outputstr%>");

    window.location="index.asp?RowI= <%=RowNo(1)%> &RowII= <%=RowNo(2)

%> &RowIII= <%=RowNo(3)%>";
```

</script>

能注释的地方，在下已经尽量加上了注释。这里主要讲一下里面不是 VBScript 语言本身的东西。读者一定注意到上面有这么一种用法：<%=变量名称%>。这种用法，是为了让客户端能够使用服务器端的变量。但是，该用法的位置很重要，因为大家都知道，变量的值可能一直在改变，那么最后交给客户端的，到底是变量什么时候的值呢？我们可以这样理解，如果把所有的服务器端代码（包括<%= %>的部分）都提取出来，看起来就像一个完整的程序，此时<%=变量名称%>所处在这个提取出来程序中的那个部分变量的值，就是给客户端的值。

举例来说

```
<%  
    Dim a  
    a="1"  
%>  
  
<script>alert("<%=a%>");</script>  
  
<%    a="2" %>  
  
<script>alert("<%=a%>");</script>
```

这样就会先弹出 1，后弹出 2 来。

读者在还没有熟悉 VBScript 的情况下，突然看这种混排的代码，一定会有很多的不适应。那在下不敢怠慢，马上引入下一节，VBScript 和 JavaScript 的对比。让读者对 VBScript 的语法有更进一步的了解

4.2 JavaScript 和 VBScript

本人一直是 VB 类语言的忠实支持者。在过去的一段血雨腥风的日子，总是充斥着

和舍友关于语言的不少争论。我们现在通过比较的方式,来学习一下 VBScript 的各种语法,其实语言之间并没有谁优谁劣,还是那句话,管它白猫黑猫,能逮耗子,就是好猫。

输入与输出不用细说,因为前面已经用专门的篇幅介绍过。

4.2.1 判断语句与循环结构

判断语句与循环结构:

JavaScript:

```
if (条件 1&&条件 2||条件 3)
```

```
{
```

```
    判断语句内部;
```

```
}
```

```
for (int i;i<=10;i++)
```

```
{
```

```
    循环内部语句;
```

```
}
```

```
while (条件)
```

```
{
```

```
    循环内部语句;
```

```
}
```

VBScript:

```
If 条件 1 and 条件 2 or 条件 3 then
```

```
    判断语句内部
```

```
End if
```

```
For i=1 to 10
```

```
    循环内部语句
```

```
Next
```

While 条件

循环内部语句

Wend '或者用 End While

VBScript 大小写不敏感,关键字我都已经用蓝色的字体标注出来了。我们可以注意到,两种语言在表现方式上还是有所区别。JavaScript 会将一个内容(比如循环或者函数的内部)用大括号括起来,而 VBScript 则习惯使用 End If,End Function 等方式。所以,VBScript 的代码书写时缩进尤其重要,否则有可能出现忘记关门的情形。在下的习惯是,写好 If 语句 就补上 End If,写好 While 就添上 End While 或者 Wend,而里面的内容最后才补上,主要就是为了避免遗忘。循环和判断的层次多了,出问题改起来也会很麻烦。

4.2.2 数组与高级函数

与 JavaScript 中的方法类似,VBScript 也内置了不少常用的函数。老规矩,先放出字典的下载地址:

[点击下载 VBScript 官方帮助文件](#)

和 JavaScript 的帮助文件使用方式一样。读者可以自行参阅。

我们先来说说数组的区别

JavaScript 的数组声明与赋值

```
var myarray=new Array();
```

```
for (i=1;i<10;i++)
```

```
{
```

```
    myarray[i]=i;
```

```
}
```

VBScript 的数组声明与赋值

```
Dim myarray(10)

For i=1 to 10

    myarray(i)=i

Next
```

2 者放在一起比较，用法就跃然纸上了。两段代码都是给数组变量进行赋值，注意 VBScript 的数组元素调用用的是小括号，而 JavaScript 则是使用中括号。另外，可以看到 VBScript 的数组声明是需要**事先说明数组大小**的。而 JavaScript 则是见招拆招，一开始都不用说多大，后面使劲用就可以。

再说明一下所谓的方法和函数。我们在 JavaScript 中，经常用到方法这个词，因为 JavaScript 是一个基于对象的脚本语言。而 VBScript 则不同，它是一个基于过程的脚本语言。到处都充斥着“函数”。其实以我们现在的水平，不用管什么方法还是函数，关键注意的是用法上的区别。举字符串为例子进行说明：

JavaScript 的字符串方法：

```
var a;

a="我是字符串";

则

a.length();    //字符串的长度

a.replace("字","非字")  //将字符串中的"字"替换为"非字"

a.substring(0,3)      //返回从原字符串开始，长度为 3 的子字符串
```

VBScript 的字符串函数

```
Dim a

a="我是字符串"
```

则

`Len(a)` '字符串的长度

`Replace(a,"字","非字")` '将字符串中的"字"替换为"非字"

`Mid(a,1,3)` '返回从原字符串开始, 长度为 3 的子字符串

可以看到,VBScript 的函数和 JavaScript 方法,除了在用法上,名称也有略微的差别。更多的函数,在下在此不便一一介绍,读者还是养成查询官方手册的习惯。而 VBScript 这章,在此就算匆匆带过。给大家布置一份家庭作业



HOMEWORK: 改造本章中的程序,让程序无论从界面还是人工 AI 上都有所提高(原来的 AI 并不会走出迷惑敌人的步数,如果人按照必胜法则走,电脑就和瘫痪了一样)

第五日：天剑

上一章的 VBScript ,让人有些摸不着头脑 ,盖因服务器端代码和客户端代码交替出现 ,群魔乱舞的场面让大家有点无所适从。好在在下近日来修生养性 ,准备放下屠刀 ,立地成佛。今天要给大家讲的 ,就是纯粹给服务器端使用的东西——数据库。

5.1 数据库简介

如果之前完全对数据库毫无认识 ,那么此章的压力将会非常非常的大。在在下念大学的时候 ,数据库部分的内容 ,可以把 A4 大小的书 ,装到 400 多页 ,而现在却要求读者一天内学会 ,有点考试前夕抱佛脚的味道。

所谓数据库 ,其实只是一种有序的数据存储方式。举个最简单的例子——留言板 ,在没有数据库的时候 ,如何实现留言板内容的存储呢。我们得把留言人姓名 ,留言人的内容 ,统统存放在一个文件里。格式可能是这样子的 :

留言人姓名 : ***

留言人内容 : *****

下一次读取的时候 ,就用字符串查找的办法 ,找到每一个留言人姓名和每一个留言人内容 ,并将跟在他们后面的数据读取出来。例如读取留言人姓名的方法 ,就是取每一个"留言人姓名"和"留言人内容"间的字符串。

可是数据的存储和读取 ,并非那么简单。因为有些用户开始恶作剧 ,他把自己的名字取作"留言人内容"。于是 ,计算机在进行读取的时候 ,就会发现一个"留言人姓名"对应着两个"留言人内容" ,计算机无所适从 ,因为它不清楚 ,哪个"留言人内容"才是真真用来标识的。有鉴于此 ,我们的格式就需要进行更换 :

留言人姓名 : *** @@

留言人内容 : ***** ##

其中的@@就作为一个姓名结束的标识符,而##则作为内容的结束标识符,虽然仍然是有用户,可能在名字里加上@@,或者留言里加上##符号影响计算机,但机会毕竟要小一些。

有些读者可能已经开发嫌烦了,因为这里还涉及到文件的操作——读取和写入。写入还好说,读取的时候要需要根据特殊符号进行分析,考虑到各种可能出现的突发情况。最可恨的是,留言信息并非仅仅要个名字和内容就算完了的,我们有的时候希望知道,这个留言留下的时间,它有多少人回复,等等等等。如此焦头烂额的文件操作,能不能有个人帮我搞定它?答案当然是有——它就是数据库。

5.1.1 数据库中的表

我们把留言中所需要的信息,全部给归拢起来,做为一组有实际意义的信息,这就用到了所谓“**数据库表**”的概念。一个数据库中的表,可以由很多**栏位**:如留言人姓名,留言主题,留言内容,留言日期等等。这些是需要我们在把表建立起来之前,就必须在大脑里设计好的东西。我们顺着思路走一次:

问题:要存放什么东西?

答案:关于留言的一些数据。那么,我们就把数据表的名称叫做:留言板。为了不出差错,只好整点洋文,名称定为“**BBS**”,我们准备用这个东西,存放留言过程中的所有数据。

问题:那么,留言中到底有什么数据需要存放呢?

答案:雁过拔毛,留言人姓名是必不可少。留言的主题,留言的内容,留言的时间,还有它的回复,回复人的姓名,回复的主题,回复的内容。

仔细分析过要存放的数据,就会发现,其实所谓的回复,只是另外一种形式的留言,它们的特点仅仅是,拥有了一个“父亲”。所以我们把所有的留言进行编号,并把留言分为两种类型,一种的留言,父亲为空,它们是最根部的留言,而另一种留言,它们的父亲则是一个留言编号,它代表这则留言是它父亲的“回复”。(我们先假设留言的回复只有一层,这样问题会简单些)

问题：这些内容了解了以后，该干什么？

答案：建立数据表，并确定它的栏位。

现在的课题变的相当简单，如何在一个数据库中建立表。

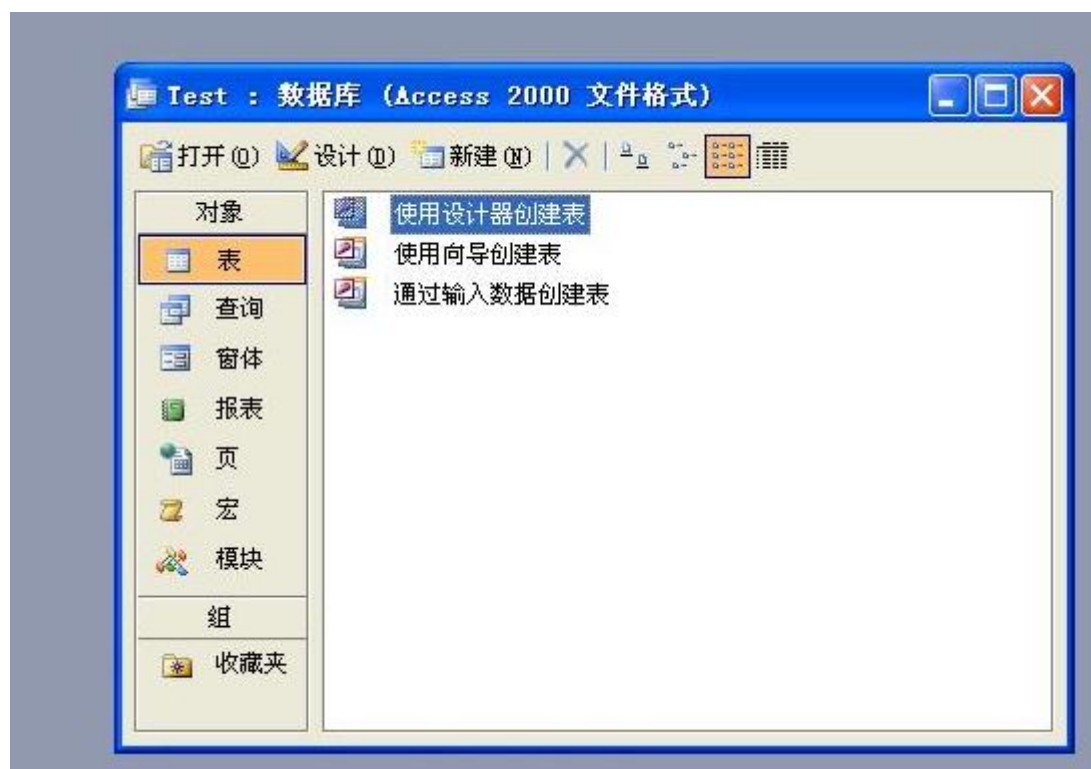
等等，事情发展的有点太快了，读者现在似乎仅仅是知道了为什么要使用数据库，对于该使用什么数据库，和如何操作，都一无所知。这就开始建立表，一定让人丈二和尚摸不到头。现在，隆重为大家介绍一下，ASP 网页的最佳搭档——**ACCESS 数据库**。

如果有幸你的机器上，已经安装了 Office，那么一定能在开始菜单里，找到启动 "Microsoft Office Access" 这个应用程序的地方。如果你没有安装 Office，那么就去装一个。在下没法控制您获得 Office 的手段，却仍然必须在这里强调，希望大家使用正版软件。

我们先打开 Microsoft Office Access:

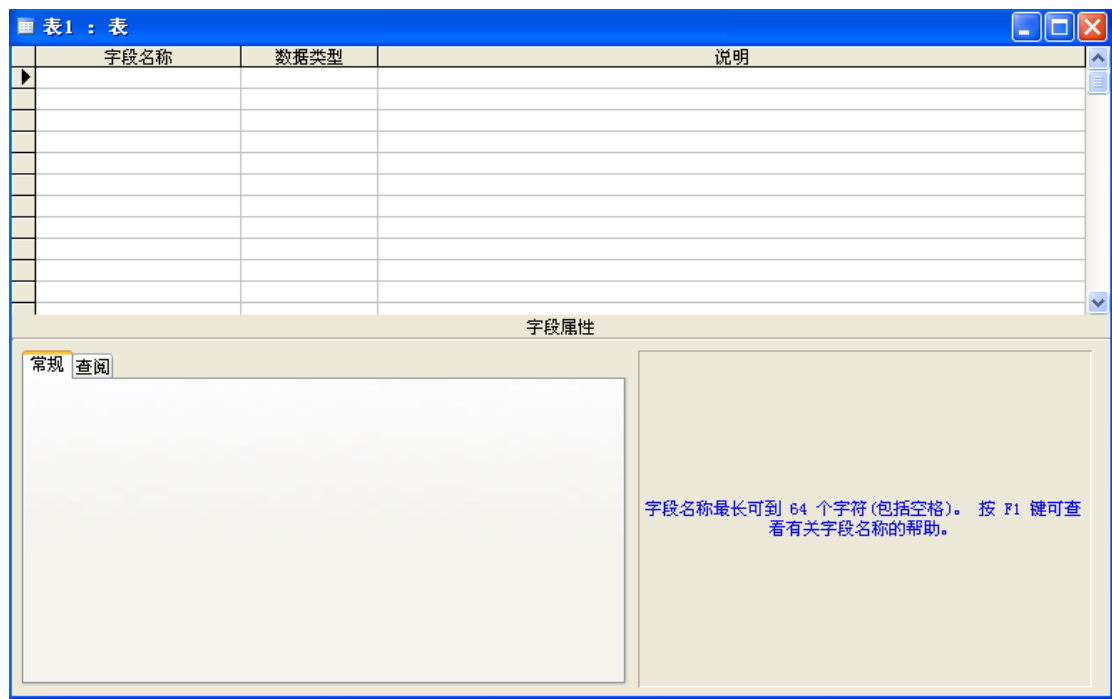
文件→新建空数据库

在系统的提示下写好数据库文件的名称（通常是 **.mdb 文件**）。然后出现了这么一个界面（我把数据库文件的名字取名叫 test.mdb）：



读者可以看到,即使微软公司也无法免俗,开门见山的,就嚷嚷着要创建表,还提供了 3 种方式。读者需要记住一点,ACCESS 是简化过了的数据库,商业级的数据库操作起来可没有这么简单,在下推荐大家有机会,还是系统的将数据库好好学习一下,特别是 Oracle 数据库的认证,能考就尽量把它考下来◎当今社会,有一技之长防身已经是相当重要了。

我们这就开始设计表了,在刚才的页面上,选择"使用设计器创建表",出现了以下界面:



可以填写的部分有:字段名称,数据类型和说明。其中的说明是可以省略不填的,它仅仅是用来你下次打开数据表的时候,对自己当时设计的栏位有所认识,有的时候表设计多了,看到自己写的栏位名称,都不知道当初为什么这么设计,说明就起到这么一个备注的作用。重要的就是"字段名称"和"数据类型"了。

所谓的字段名称,就是我刚才提到过的"栏位"。我们已经仔细想过,我们的表,需要有这么几个栏位:留言序号,留言人,留言标题,留言内容,留言时间和父亲序号一共 6 个栏位。我们把栏位名称分别都用英文来表示——做为字段名称:

`m_id,userid,subject,content,m_date,f_id`

把这些字段名称——填入到设计器中,在下现在讲述一下数据类型。ACCESS 中提供选择的数据类型一共有这么几种:

文本：较小的文本，最多只能容纳 255 个英文字符（或者 128 个汉字）

备注：较大的文本，通常超过 255 字符的文本都使用它

数字：数字

日期/时间：具有日期格式的字符串

货币：钱~~

自动编号：属于数字的一种。一旦定义为自动编号，则我们在新增数据时可以不用理会该栏位，它会自动增加。

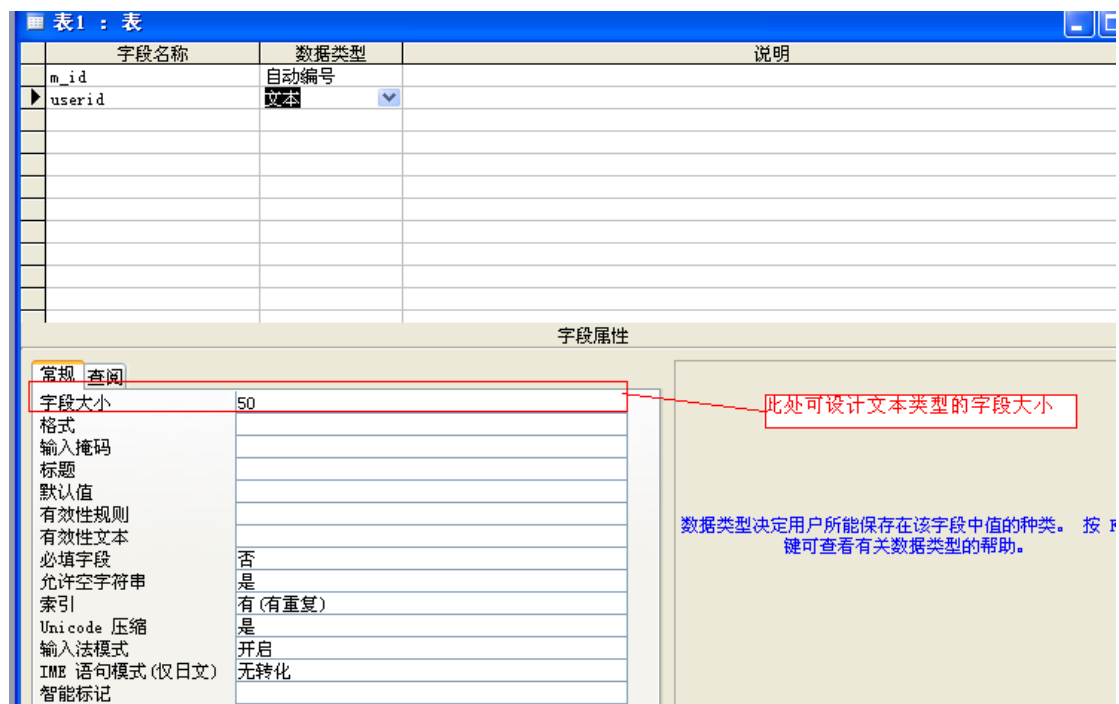
是/否：被指定为该类型的栏位，只能有"是"和"否"两种值

OLE 对象：通常是指在数据库中存放图片，音乐等等数据。

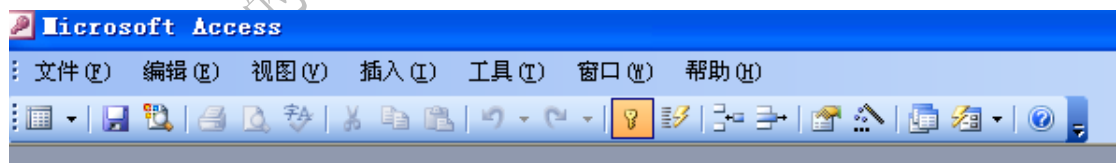
超链接：这个无需说明。就是网页上的那种链接。

我们可以很容易的对号入座。`m_id` 设计为**自动编号**，`userid,subject` 为**文本类型**，`content` 为**备注类型**，`m_date` 使用**日期/时间**。`f_id` 为**数字**（这个父亲 ID 需要我们来指定，因此不能自动编号）

注意，当选择每个数据库类型时，下面都会出现一个该类型的一些数据。我们可以制定很多东西。比如文本类型，我们可以限定它的大小（目前有用到的，就是这个了）。其它的属性，大家可以自行研究（再学完本章后研究，因为现在有可能看不懂）。如图所示：



一个完整的表，还需要为它设计一个**主键**。主键的定义，就是该表中随着记录增加，绝对不会重复的栏位。在数据库的学习里，设置主键被作为一个非常重要的部分，它决定着数据库的表设计是否合乎规范，我马上会在以后的内容里介绍到主键的重要性，此刻暂时按下不表，以设计为重。很显然，我们这张表里，m_id 这个栏位就完全符合条件，选择到 m_id 这个字段，点击如下图所示突出的那个钥匙按钮，就把 m_id 成功的设置为主键



在文件中选择保存，ACCESS 将会让你给表取个名字——我们早已准备好 :BBS。至此，一张用于存放留言数据的留言表，就生成了，这些数据都被存放在开始进入 ACCESS 时新建空白数据库所问到的那个 mdb 文件里，它就是数据库文件。

5.1.2 插入，选择，删除和更新

INSERT 语句

在 ACCESS 中，要实现插入数据是一件再容易没有的事情。读者只需要点开刚才建立的那张表，一目了然的数据输入方式，会让你对微软的爱慕之情，又添上几分。而我们这里要讲的插入数据，则是需要通过代码来实现。

从表的设计可以看出，插入数据，就是个简单的填空过程。我们给每个栏位填好值，就算是进行了一次插入操作。用代码实现这部分，我们首先得明白一件事：必须掌握一门新的程序语言——SQL。请读者放心，SQL 的简单运用，远比脚本语言来的轻松，我们不用从循环和判断语句学起，更可以暂时把什么数组抛于脑后。SQL 语句只有 4 种基本形式——插入数据语句，选择数据语句，删除数据语句和更新数据语句。这四种形式根据情况的不同千变万化，而我们则盯住它们的基本语法即可。第一步要学的，就是如何使用插入语句填空。

插入语句的基本形式如下：

```
Insert into 表名 ( 栏位 1, 栏位 2, 栏位 3..... ) Values  
( 栏位 1 的值, 栏位 2 的值, 栏位 3 的值..... )  
Where 条件  
--SQL 语句和 VBScript 一样，大小写并不敏感。
```

注意，除了蓝色字体关键字间的空格以外，语句间是连贯的，在下主要是为了让读者看清楚才用换行将它们分开。插入语句中的最后部分 **where 条件** 是难点所在，好在我们在编写简单的 ASP 页面时，根本用不上它。所以，where 和之后的条件语句在正常使用都可以省略。

语句的意思简单明了，每次执行 Insert 语句，都往数据库中插入一条数据。而插入语句的格式，摆明就是一个填空的过程。既然是填空，意味着你的栏位“一个也不能少”，所有的栏位都必须填上值，也就是说它们必须都出现在 Insert 语句中，如果你认为该栏位根本不需要有值，那么当时在设计表的时候，就需要设置该栏位的属性为“允许空”。唯一免俗的，就是如 m_id 这样的自增栏位，它的填空过程，计算机会代你完成，而除此之外的，都是需

要手动来做。

假设有个叫 “Peter” 的用户，于 2005 年 8 月 24 日这天，在留言板上留下了以 “讨伐飞行岛在下” 为题目的战斗檄文，而内容则是简单而不负责任的这么几个字 “在下欠钱不还”，我们就以这个为例，写一句完整的插入语句☺

```
Insert into BBS (userid,subject,content,m_date,f_id) Values
```

```
("Peter","讨伐飞行岛在下","在下欠钱不还",#2005-8-24#,0)
```

聪明的读者一定注意到，有的值两边使用引号，有的使用#号，有的则干脆置之不理。是的，这些用法都取决于栏位的类型：字符类（包括文本和备注，货币等）都使用引号，日期类使用#号，而数字类型则什么符号也不必使用。有时你会发现用错也不会出现问题（比如日期类型也可以用引号放在值得两边），但还是严格按照格式来走比较好，当一个页面的问题多到你无法定位时——初学语言经常遇到这种情况——就需要你对每个可能出错的地方严格把关。上面的语句经过执行（由什么执行和如何执行将在后面的章节讲到），数据表 BBS 里就会增加一笔记录。

我们也暂且不要管 SQL 语句如何与可爱的 VBScript 进行结合，这一节把精力放到 SQL 语言上来。

SELECT 语句

增加了数据，问题的重点就转移到读取数据上来了。许多数据库书都喜欢把读取数据的部分放在一开始来说。诚然，这是 SQL 语言的基础，可是它并不符合我们人类思维的习惯，试问谁又希望在不知道如何建立表和插入数据的情况下，就开始涉足所谓的读取呢？在下故意倒过来说，希望读者能够接受这种方式。

读取数据用的是 SELECT 语句，它应该是 SQL 语言中出镜率最高的咚咚了。我们仍然采取举例配合理论的方式来学习。我们先看看 SELECT 语句的语法和结构：

```
Select 栏位 1，栏位 2，栏位 3 From 表名
```

```
Where 条件
```

其中，栏位就不需要像前面的插入语句那样，写的满满，这次的工作不是填空。列出来的栏位，都是我们希望从中获得值的——但如果你确实需要获得所有栏位的值，则可以用*号代替：`Select * From 表名`。表名自然是必不可少，而 where 后面的条件，则是此次说明的重点了。

例：我们一直往 BBS 里面塞数据，该表已经成功存放了成百上千条留言或回复。此时要求写一个网页，把所有客户留言（**但不包括客户的回复**）给选出来，只需要列出它们的标题即可。

上面的这句话，可以分解成这么几个部分。

- 1 **数据来源表**：BBS
- 2 **要求被读取的栏位**：BBS 中的 subject（主题栏位）
- 3 **条件**：所有不包括客户回复的数据——这我们在插入数据时，就有所规定，最顶层的留言，他们的父亲 ID 都是 0。

如此分解后，便可尝试将语句写出来了：

```
Select subject From BBS
```

```
Where f_id=0
```

读者应该会发现，where 后面的语句，和前面学习的 VBScript 的判断语句有所类似。

我们可以再举多个更复杂的例子进行练习：

要求：从表 BBS 中选出这样的数据，发表日期在 2005 年 8 月 26 日之前，所有“愤怒的葡萄”的发言（不包括他对其他人的回复），需要列出这些留言的标题，内容和时间。并把它们按照发表日期排序。

语句如下：

```
Select subject,content,m_date From BBS
```



```
Where userid="愤怒的葡萄" and m_date>#2005-8-26# and f_id=0
```

```
Order By m_date desc
```

条件的写法几乎和 VBScript 一样。大部分条件都是针对栏位的。注意日期两边的#。另外,SQL 中字符串两边的引号,都是单引号

Order By 就是排序的语法, **desc** 表示倒序排列, 而 **asc** 则表示正序排列。

本例将 m_date 置于 Order By 的后面, 表示根据 m_date 栏位倒序排列。Order By 后面的栏位, 必须是出现在 Select 之后那些栏位的其中之一, 否则会出错(比如我们就不能写 Order By f_id)

要求: 从表 BBS 中选出这样的数据, 无论是留言还是回复, 把所有标题中包含“飞行岛在下”字样的数据选出来, 列出这些数据的标题, 发言人和发表日期, 并按照发言人的名称升序排列。

```
Select subject,userid,m_date From BBS
```

```
Where subject like '%飞行岛在下%'
```

```
Order By userid asc
```

既然要求是标题中包含某某字段, 自然不能用等号或者是不等号 (< > 为不等号) 来诠释条件。我们需要使用到的, 就是 **like** 和通配符 **%**。

SQL 中有两种通配符: **%** 和 **?**。**%** 可以代表任意数目的字符, 而 **?** 只能代表一个字符。当你对条件知道的比较精确的时候, 就使用 **?** 来加快读取数据的速度。而本例中的包含, 就只能使用 **%**。

由于 **%** 可以代替任何字符, 甚至包括空字符。因此 “**%飞行岛在下%**” 这个词组就涵盖了所有包含 “飞行岛在下” 的词组或句子。而如果我们条件是: 找出所有 userid 中姓张并且只有名字长度为 2 的所有留言, 通配符条件就可以写成 **like '张?'**

基本上到此为止, SELECT 语句的学习已经够我们在 ASP 中使用了, 但是个性坚忍的同学可以继续看下去。后面是一些 SELECT 语句的高级技巧了



警告, 以下内容艰涩难懂, 18 岁以下青少年请在家长陪同下观看

要求：从表 BBS 中选出这样的数据，把所有发帖的数量超过 5 个（不包括回复）的用户都列出来，显示方式为 userid 和发帖数量。

```
Select userid,Count(userid) From BBS
```

```
Where f_id=0
```

```
Group By userid
```

```
Having Count(userid)>5
```

在 SELECT 后面已经开始出现，不是栏位的东西了。我们先从 Group By 分析起。

Group By 是一种分组的手段，本例就是按 userid 分组，因为计算发帖数量的时候，我们实际上就是进行了一个分组的动作——把所有不包括回复的帖子（f_id=0）按照用户名分组，然后数一下每个用户旗下有多少的帖子。有的时候，可以通过两个栏位来分组，其实分组的规则很简单，就是把 Group By 后面的栏位值相同的，归为一组，达到物以类聚，人以群分的目的。

Count，顾名思义，就是数数的过程。而 **Having**，则是专门用在 Group By 后面，作用和 Where 相同，它只能作用于分组条件（比如我们就不能把 f_id=0 这个条件放到 Having 的后面）。此处的条件，就是 Count(userid)>5，而 Count(userid)实际上就是在计算被选出的 userid 数据组中，每个组有多少条记录。

要时刻保持警觉的是，一旦使用了 Group By，就**不能**在 Select 后随便写栏位。Group By 已经限制住了最后显示出来数据的数目——以本例来说，如果没有发帖数量超过 5 个这个条件，选出来的数据条数应该就是所有用户的数目（如果只有一个用户才拼命发帖，数据表里只有他一个 userid，那么 Group By 选出来的数据也将只有一个）。所以，规则是，Select 后面的栏位，必须在 Group By 后面出现过，或者是一种计数方式（比如 Count 或后面讲到的 Max）。**换言之**，Group By 后选出的数据，实际上是组级别的，所有后面的 Having 条件或是前面的 SELECT，都是基于组的操作。

如果读者对上面这段解释云里雾里，在下深表抱歉，请查阅相关书籍，因为这个部分说起

来牵连甚广，在下不能寥寥数语代过，而它们并非 ASP 学习的重点所在，只能由读者凭兴趣学习。

要求：从表 BBS 中选出这样的数据，每个用户最后发的那个帖子。（无论是回复还是留言）
显示方式是用户名，帖子标题，帖子内容

```
Select userid,subject,content From BBS  
Where m_id in  
(  
    Select Max(m_id) From BBS  
    Group By userid  
)
```

将帖子按照用户分组以后，最后发的帖子必然是该组中 m_id 最大的那个（m_id 一直是递增的），所以后半句 SELECT 语句，其实是选出每个用户最后发的帖子的 m_id。Max(m_id) 就是选择该组中最大的 m_id，它和 Count 类似，都是基于组的操作。

将该结果嵌套到前半句 SELECT 语句中：m_id in 代表条件是 m_id 是后面选择出来的内容中的一部分。SELECT 语句的嵌套一般都是通过 in 来进行，类似的语句还有：

```
Select subject From BBS  
Where subject in  
(  
    Select subject From BBS  
    Where m_id=2  
)
```

上面这个语句的作用是，选出所有标题与 m_id 为 2 一样的留言或回复。

用以上的几个例子，把基本的 SELECT 用法都算介绍了一遍，这一节需要读者仔细琢磨，最好能配合一些专门的 SQL 书籍进行观看。给读者留个小作业：



HOMEWORK: 编写 SELECT 语句, 要求从 BBS 表中选出所有回复“愤怒的葡萄”的用户名称 (提示: 使用 in 语句)

多张表之间的 SELECT 语句

不难认识到, 一张数据表是远远不够的。在众多拥有留言功能的网站中, 无一不是厚颜无耻的先要求用户在上面注册后, 才拥有留言的权力。而正如前几章的网页所说, 制作一个注册页面不能简单地让人家用户敲个名字就算注册成功。伴随该用户的详细信息, 都希望被存放起来——我们需要一张人员表。

此处不再赘述人员表的设计过程, 请读者自行完成。而做为例子的人员表, 在下将安排它拥有这样的信息:

表名: **usertable**

栏位:

Userid 分配的用户 ID.....自增型, 主键

Username 用户名称.....文本类型

Password 用户密码.....文本类型 (注意, 虽然页面上的密码用星号代替, 但存放
到表里的, 就无需遮遮掩掩, 都是真刀真枪)

ComeFrom 用户所在地.....文本类型

等等....., 读者可以根据需要添加其他栏位

很显然, 刚刚建立的 usertable, 需要和之前的 BBS 表有所联系。原来的 BBS 中存放的用户信息, 太过粗糙和不负责任了——仅仅是存放了名字而已。现在这份专业存储用户信息的事情, 已经交由 usertable 来完成。BBS 表中只需要存放一个路标, 通过这个路标, 可以轻松的到 usertable 中将用户的详细信息提取出来。这个路标, 就是 usertable 的主键——前面说过, 唯一在表中不重复, 作为标识的咚咚——userid。

为了玉成 BBS 和 usertable 之间的美事。我们需要对 BBS 的 userid 进行一些更改——

—由于是和另一张表的自增类型主键关联，该 userid 也必须是数字形式。

打开 ACCESS，右键点击 BBS 表，选择“设计视图”，即可重新到达表结构设计页面，将 userid 由文本形式修改为数字形式，即大功告成。



这种关联性，实际上只有天知地知，你知我知。ACCESS 提供了一种方法，在内部将这两张表联系起来——这就是外键的概念。当一个表的主键 (usertable 中的 userid) 作为一个栏位在另一张表中存在时 (BBS)，我们就认为 userid 是 BBS 的外键。通过 ACCESS 可以将这种联系表现出来。请读者参考 ACCESS 的帮助中“关系”的相关章节。在下不在此介绍，是因为在下并不推荐将表的关系交给 ACCESS 管理，原因如下：

- 1 一旦建立了外键关系，意味着两张表的这两个栏位唇亡齿寒。如果你在 usertable 中不小心删除了一个用户的纪录，那么该用户的所有文章，都会因为建立好的“关系”而被全部删除。删除是一个不可逆的过程，在很多时候会带给我们很多不方便。
- 2 每一次我们更改被“关系”管理的栏位属性——比如从文本改为数字类型。ACCESS 都会阻止你的操作，并让你先将关系删除。这又凭空添加了不少麻烦。特别是在我们设计的表并没有那么复杂，经常改动的时候，这尤为烦人。

当然，读者也一定看到，严谨是这么做的优点。当你的网站已经步入成熟，表结构和代码已经基本稳定下来的时候，你就可以考虑建立这样的关系。关系的建立步骤并不复杂，请读者自行学习☺

如何自由的从两张有关系的表中，自由的查询数据，成了问题的焦点。我们仍然以例子的形式来说明：

要求：选出这样的数据，希望得知在 2005-8-26 日后回复的用户（只有回复，不包括单独发帖）都来自于哪里。希望能列举出用户姓名，帖子主题和用户来源地的信息。

```
Select username,BBS.subject,usertable.ComeFrom
From BBS,usertable
Where BBS.f_id<>0 and BBS.m_date>#2005-8-26#
and BBS.userid=usertable.userid
```

语句的变化是显而易见的，除了 From 后面出现了以逗号隔开的两张表以外，Select 关键字后面跟随的栏位写法也作了一些调整。调整的主要目的，就是区分两个表的栏位，很多时候，两张表的栏位名称可能相同的有很多（本例只有一个 userid），因此，“表.栏位”的写法必不可少，后面的条件也是如此。而本例的重中之重，就是最后一个条件语句，它说明了 BBS 表栏位和 usertable 表栏位的关系。它是一个重要的限制条件，因为题目中的，其实有个默认的要求：BBS 表和 usertable 表之间，通过 userid 一一对应。如果缺少此句，则查出来的结果将包含整张 usertable 表的数据（很显然没有任何条件约束该表），请读者仔细咀嚼。

如果读者觉得，这么写实在是非常麻烦，SQL 中还提供另外一种写法：给表命名

```
Select b.username,a.subject,b.ComeFrom
From BBS a,usertable b
Where a.f_id<>0 and a.m_date>#2005-8-26#
and a.userid=b.userid
```

为 From 后的表添加另外一个代名称（本例的 a 和 b），并在整个 SQL 语句中都用代名称来书写，就是一种减轻负担的方法，在下对字母 a,b 的简单尤为喜爱，基本遇到多表混查的时候，都少不了他们的帮忙

其实多张表查询就是单张表的升级版，区别在于，多表查询时，Where 条件的中心，往往会落在这几张表的栏位关系上，希望读者给予重视。

UPDATE 语句

终于有人会发现，插入的数据，会出现需要更新的情况。比如某读者洋洋洒洒的写上数千字，希望对在下从人格和世界观上都进行无情的鞭笞，偏偏一切就绪并发表后，该读者注意到他竟然鬼使神差的写了不少的错别字——读者怕被内心阴暗的在下抓住把柄笑话，他急切地希望能够编辑文章，重新发布。将已有的数据进行更新——这就是 Update 语句的历史使命。

我们先看看它的语法：

再次强调，本章中的所有 SQL 语句，换行都是为了读者阅读方便，其实整句话都是连贯的

Update 表名

Set 栏位=栏位需要更新的值

Where 条件

经历了 Insert 和 Select 语句的读者们，在看到这幅行头，相信已经颇为镇定了。国际惯例，仍然用例子向读者说明：

要求：将所有帖子标题中含有“飞行岛在下”字样（包括回复）的文章内容，都更新为“Don't Cry for me, Argentina.”

Update BBS

Set content='Don't Cry for me, Argentina.'

Where subject like '%飞行岛在下%'

应该是很简单就可以看明白，更新语句中的条件至关重要，一个不小心，可能会把你不想更新的东西也一并改了，到时候后悔莫及，因此一定要慎重

要求：将所有帖子标题中含有“飞行岛在下”字样（包括回复）的发帖时间改为 2005-8-27，而发帖人的姓名都改为“在下的 fans”

Update BBS,usertable

Set BBS.m_date=#2005-8-27#,usertable.username='在下的 fans'

Where subject like '%飞行岛在下%'

and BBS.userid=usertable.userid

语句中的某些写法，类似于多表查询 SELECT 语句。注意 Update 语句中的表名是不能像多表查询那样用另外的名字来替代，必须老老实实的将“表名.栏位”写完全。当然，最后一个两表关联的限制条件，一样是必不可少。

Update 语句并不复杂，上面两个例子已经可以全面阐述它的用法，还是提醒读者对于条件的把握。如果实在不放心，一般的做法是先将 where 后的条件，套到 select 语句中进行查询，查询出来的记录通常都是更新语句所要更新的对象，这样即可保万无一失。

DELETE 语句

所有的数据库操作，不逃脱，增查删改四字。删除数据的作用，相信不用在下废话，他是一个不可逆行的操作，没有什么商量的余地，出于负责的态度，这尚方宝剑此刻才交给读者，希望读者能好好掌握和使用（其实无论是 SQL Server 还是 Oracle 都提供了类似于回滚的机制用来避免误删除，但这对我们来说，实在没有了解的必要，我们就暂且把删除认为是不可逆）

删除语句的语法

Delete From 表名

Where 条件

单从这简单的语法，即可看出，破坏实在是比建设要困难得多（Insert 语句一般来说是所有 SQL 语句中最麻烦的，读者以后写多了就会有所认识）。随便举上一个例子，读者即可完全明白——和更新语句一样，一定要把条件写好。

要求：删除所有标题或内容中出现“猪狗不如”的帖子（包括回复）

Delete From BBS

Where subject like '%猪狗不如%' or content like '%猪狗不如%'

语法相当简单，只要能保证不误删和漏删，请读者都尽管放心的撰写和执行：)

至此为止，SQL 语法算是介绍结束。当然，SQL 的博大精深，非吾辈能在短短几页内讲述明白。囫囵吞枣的读者们，也只好从后面大量的例子中，再度巩固和学习了。我们马上就要将网页和 SQL 进行挂钩。

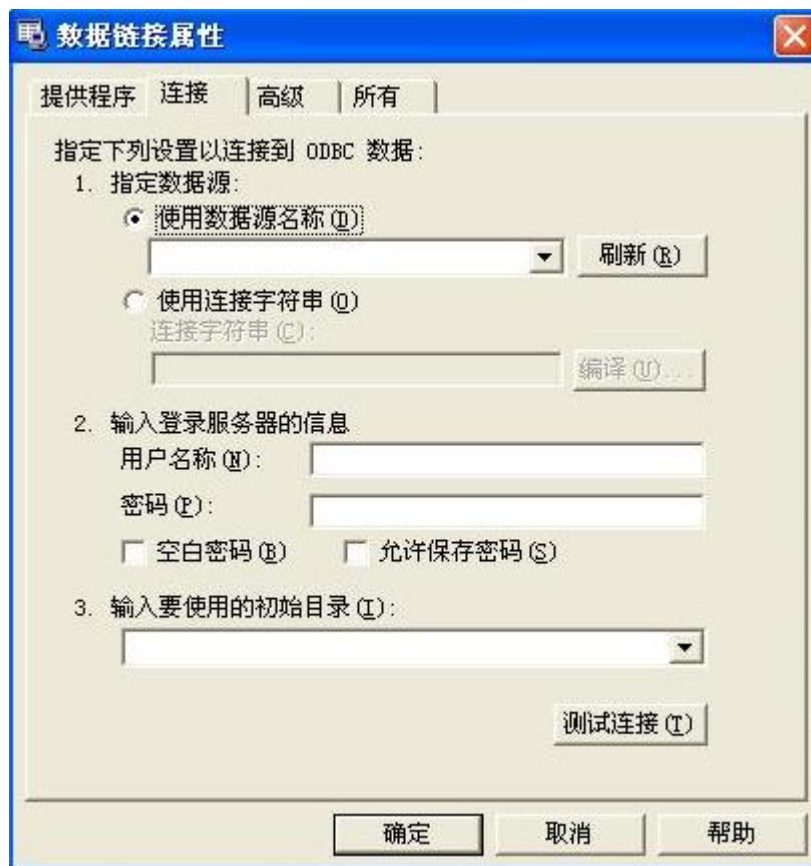
5.2 网页中的数据库连接

5.2.1 数据库连接

说到底，我们还是关心 asp 多一点。要把数据库和网页上的代码连接起来，我们还需要一根桥梁。

幸亏这根桥梁的搭建方式，非常简单，尽管也许有的读者觉得不太地道.....

随便在哪个文件夹下（桌面上也可以），建立一个文件，文件的名称不重要，但它一定要以 udl 作为扩展名（udl 的扩展名代表该文件是数据库连接文件）。建立成功后，双击打开（下面用 test.udl 作为例子），如图所示



- 1 将指定数据源选择为“使用连接字符串”，并点击[编译](#)。
- 2 在[文件数据源](#)选项卡中，选择[新建](#)。
- 3 在[创建数据源](#)窗口里，选择“[Driver to Microsoft Access\(*.mdb\)](#)”
- 4 在弹出的[创建新数据源](#)的窗口里，随意取一个数据源的名称（本例使用 **test**），点击[下](#)

一步

- 5 点击[完成](#)

出现以下窗口：



点击**选择**，在弹出的文件选择窗口中，选择我们一直用来做人体实验的数据库文件©，然后不断点击确定到底。（不论出现什么画面，只管确定便是）

最后我们回到开始的**数据库连接属性**页面

我们可以发现，在连接字符串下方的文本框里，出现了类似于以下的字符串（读者机器上的连接字符串可能与在下的有所不同，主要的区别就在 mdb 文件的位置）

这么长的字符串，并非所有的东西都有用，我们把最为重要的信息提取出来：

```
Driver={Driver do Microsoft Access (*.mdb)}; uid=admin;pwd=;
```

```
dbq=C:\Documents and Settings\陈思\桌面\Test.mdb;
```

为了排版方便，读者看起来也许是换行的效果，其实他们是一整个的数据库连接字符串，中间没有任何换行字符。而中间蓝色字体部分，可以看出就是 mdb 文件的路径。上面的这个步骤，我们已经完成了数据库与代码连接桥梁的必要步骤，而那个被利用的 udl 文件，已经可以过河抽板的删除。当然读者如果是个念旧之人，在下也不敢阻止您将其留到天长地久。

打开 ASP 文件，我们创建一个进行数据库连接的函数（该函数是服务器端代码）

```
Function GetConnection
```

```
Dim tempconnstring
```

```
Dim ConnLaputa

Set ConnLaputa= CreateObject("Adodb.Connection")

tempconnstring= "driver={microsoft access driver (*.mdb)};"

tempconnstring= tempconnstring + "uid=admin;pwd=dbq="

tempconnstring= tempconnstring + Server.MapPath(".")

tempconnstring= tempconnstring + "\Database\test.mdb"

ConnLaputa.ConnectionString= tempconnstring

ConnLaputa.Open

Set GetConnection=ConnLaputa

End Function
```

代码基本分为几个步骤：

- 1 创建对象。**CreateObject** 是一种创建对象的方法，数据库连接是一个对象，名称是 **ADODB.Connection**
- 2 拼装连接字符串。我们知道，网页中不允许存在绝对路径（原因前面已经说过多次），所以需要数据库文件的相对路径——这意味着，你需要把数据库文件拷贝到虚拟目录下。面来。**Server.MapPath(".")**就代表着包含该段代码的文件所在位置的虚拟路径。可以看出，例子中的数据库文件放在连接代码文件所在目录的 Database 子目录下
- 3 **ConnLaputa.ConnectionString** 为连接字符串属性，将拼装好的变量赋值给它。
- 4 **ConnLaputa.Open** 打开数据连接

将来如果需要创建数据库连接，可以通过简单的代码直接调用该函数：

```
Set ConnLaputa = Server.CreateObject("ADODB.Connection")

Set ConnLaputa=GetConnection
```

完成了以上操作，就成功地建立了数据库连接，总算是迈出关键性的一步。以后手写顺了，就不用做太多飞鸟尽，良弓藏的事情，那个 udl 文件完全可以不用创建，连接字符串的

形式并不复杂，自己书写即可。

5.2.2 SQL 代码的执行

从之前的 SQL 的代码中，我们可以把语句分为两大类：读取数据，写入数据（包括插入，更新和删除）这样的分类，并非为了照顾 VBScript 的不安情绪，实在是形势所迫。读取的 SQL 语句执行后，势必要返回一堆数据来，而服务器端代码当然责无旁贷，必须照单全收。相比之下，写入语句则是属于执行之后即可拍屁股走人的类型。有鉴于此，我们特地把本来简单的 SQL 语句执行过程，弄得复杂一些，引入一个叫做**数据集**的对象。

```
Set RsLaputa= Server.CreateObject ("ADODB.RecordSet")
```

又见 CreateObject，只是此次创建的对象变成 ADODB.RecordSet。数据集对象用于存放读取语句执行后所获得的数据。我们现在以表 BBS 为例子，编写一个简单的留言显示网页：

```
<table bgcolor="#CCCCCC" class="normal"
  cellpadding=0 cellspacing=1>
  <tr  bgcolor="#E8EEF7">
    <td align="center"><B>以往的留言内容</B></td>
  </tr>
  <%
    Set ConnLaputa= Server.CreateObject ("ADODB.Connection")
    Set ConnLaputa= GetConnection
    Set RsLaputa= Server.CreateObject ("ADODB.Recordset")
    Set RSLaputa= ConnLaputa.execute(".....")
    While Not RSLaputa.Eof
  %>
  <tr  bgcolor="#E8EEF7">
```

```
<td align="left"><%=RSLaputa("username")%>说道 : </td>

</tr>

<tr bgcolor="white" >

<td align="left" style="width:400px">

<%=RSLaputa("content")%> <BR><BR>

</td>

</tr>

<%

RSLaputa.movenext

Wend

%>

</table>
```

注解如下：

- 1 ConnLaputa 就是数据库连接变量，它通过函数 GetConnection 被赋值为数据库连接。
- 2 While 是循环大家都知道，While Not 代表当后面的条件不成立的时候，执行循环
- 3 数据集是很多条数据的集合，读取的时候，只能一条条的顺序读取，每次循环读取一条，而当所有的数据读取结束的时候，数据集的 EOF 属性就为真（条件为真的就代表条件成立），此时如果继续读取，页面就会报错，所以需要作为循环条件而跳出循环。
- 4 红色省略号部分就是需要执行的 SQL 语句，请读者自行写出读取留言者，留言内容的 SELECT 语句（需要用 BBS 和 usertable 两张表），执行 SQL 语句的语法就是：

数据库连接变量.Execute(SQL 语句)

比如：ConnLaputa.Execute("Update BBS Set f_id=0")

删除，更新和添加语句与此相同，不再赘述，读者只需要把精力放在拼装 SQL 字符串上即可。

- 5 RSLaputa("username")和 RSLaputa("content")代表数据集里的栏位的值：曾记否 SELECT 语句后面跟随着，就是栏位的名称，因此刚才说的 SQL 语句要仔细书写，因为只有正在执行的 SELECT 语句中出现过的栏位，才能出现在数据集里。

6 **MoveNext** 代表让数据集继续开始读取下一条数据，如此往复，即可完成对数据集里所有数据的读取。

貌似复杂的东西，原来是这么的简单。给读者一个至今为止最大的作业，希望不要有人吐血晕倒。我仔细研究过下面这个作业题目，所有的功能都可以用上面介绍的知识来实现，读者遇到不懂的地方，可以多动脑筋，也是一个复习前几章的好机会。



HOMEWORK: 编写没有翻页功能的留言板，用户输入名称才能留言。当用户不存在时，往 `usertable` 里添加记录。留言板要求可以通过点击按钮编辑和删除留言。有兴趣的朋友，可以将注册页面也和数据库联系起来。

到这里，在下准备不负责任的结束此章。亲爱的读者们，入门教学就是如此，它让你可以进行简单的开发，而真正的提高，需要将来你自己的磨练。有人曾问过在下，学习 ASP 要多久，在下也说不上答案来，视每个人的基础和学习能力而定，但受苦受累，却绝对是无法避免的，读者看到这里，也应该有这样的觉悟。

我不是个随便的人，我随便起来不是人，谢谢大家！

第六日：剑圣

严格说，此节的内容，才算是 ASP 的正题，大部分的书，都把在下的这节所要说的内容，放到最前面来讲。在下一直对此表示疑惑，因为在下的开发经验来看，这些放在最前面的东西，结果其实应用的最少，甚至一些简单的功能，连想都懒得想，直接去网络上拷贝现成代码。大部分的时间，都会浪费在一些简单功能上面——比如上章的数据库联接，以及一些 JavaScript 的按钮事件编写上。在下更帮人家做过商业网站的后台页面（大部分是服务器端内容，页面美工设计要求很低），基本上就是写数据库操作和页面的提交。不要小看了数据库代码编写，因为每个你涉及到的数据库操作，比如留言板的管理，人员的管理等等，都有添加，选择，删除，更新四种操作，写起来不厌其烦，那个时候的 ASP，就仿佛一只体态臃肿的熊。但是话说回来，在下并不敢省略此节，他们毕竟是 ASP 一个重要的组成部分，还是要对亲爱的读者们负上责任。

6.1 网页中的计数器

6.1.1 Application 对象

记住每一个过往人员，一直是我们这些网页编写者的小小心愿。我一个朋友曾经认为，每天最大的乐趣，就是盯着他网站的计数器发呆（不过在下倒是认为是这家伙以查看计数器变化为由，自己刷新页面来制造虚假网页访问量）。运用之前学过的知识，我们可以通过数据库完成计数器的功能，代码很简单，设计一张计数器的数据库表，在首页里加入代码，每次首页被读取的时候，就更新数据库，将数据表中特定栏位值加一即可。

我们现在就来做这件事：

建立一张表：

表名：**counter**

栏位：

CounterID 每个表的必备 ID.....自增型，主键

CounterNo 计数值得存放栏位,数字类型

网页代码如下:(服务器端代码)

```
Set ConnLaputa= Server.CreateObject ("ADODB.Connection")  
  
Set ConnLaputa= GetConnection  
  
Set RsLaputa= Server.CreateObject ("ADODB.Recordset")  
  
Set RSLaputa=  
ConnLaputa.execute("Update counter Set CounterNo=CounterNo+1")
```

'注意上面语句并无换行。仅为显示需要

这样,每次该页面被访问到,数据库就会被更新,栏位 CounterNO 值就可以代表访问量。(当然我们如果怕别人嫌弃网站没人气,可以把基数设置的很大。比如一开始就把 CounterNo 设置为 2000,这样别人看起来,你的网站似乎就会比较火热,当然这种卑鄙手段,在下从原则上是反对的◎)

不过,并非每个网页,都要搞个数据库出来和它配合的。如果为了计数器,而特别开放一个数据库连接,并且在每个用户访问的时候都频繁的读写数据库,对于服务器的开销将会相当相当的大——读者可以立刻试验出来,将页面迅速刷新几十次,很可能就会出现数据库操作超时的情况。而你要想象到,当用户量大的时候,服务器的负担可能比你自已按刷新的时候还要重上许多。如此不智慧的做法,应当予以坚决打击。

这时候我们想到,如果有一个服务器变量,它不因哪个用户关闭页面而消失,只要有用户触发它,它就活到服务器关机的那一天该有多好。这个变量,我们就可以名正言顺的拿来作计数器——毕竟变量的读写操作可比数据库要轻松的多。只要我们在服务器关机前,将变量的值存放到什么地方(比如文件)就好。一个优秀的程序员,就应该懂得做白日梦!偏偏可爱的工程师们早已体谅到我们这些关注网站访问量的群体,他们设计出一种完全符合上面条件的东西来,供我们玩耍——Application 对象。

本着普度众生的想法,Application 对象的存在,让技术器问题变得简单起来。它被创

建于第一个用户访问之时，并坚持到生命周期结束（**当有一段时间没有人再连线该服务器，对象就宣告终止，一般为 20 分钟**）。如此重情重义的对象，不用来做计数器，倒实在是可惜了。

对象的格式是这样：（服务器端代码）

`Application("变量名称")=变量的值`

其实简单来说，Application 对象就是在编程中使用的一个变量集合。而这些变量拥有前面说过的那些特性。我们用来让计数增加的，就可以这么写：

`Application("counter")= Application("counter")+1`

仅仅让计数器添加是不够的。万一今天网站拥挤，大家一起浏览的时候，这个计数器就会不那么准确。比如我和你一起访问，都读取到 `Application("counter")=3` 这个信息。接着我们俩的页面都把 `Application("counter")` 变成 4，就算运行结束了。而实际上，两个人访问，计数器才增加了 1。

为了避免这种情况，特地为 Application 对象设计了一种**锁操作**。

Application.Lock

`Application("counter")= Application("counter")+1`

Application.Unlock

`Application.Lock` 就是锁。而 `Application.Unlock` 就是解锁。当 Application 被锁住的时候，将无法被访问到。这样，用户需要等到计数器因为其他用户增加完毕了以后，才轮到对他的纪录。准确性就大大提高了。

这样，变量中将会一直存放着计数器的数值，直到 Application 对象消亡为止。

莫怪在下突兀，突然必须介绍另外一个知识点。

其实在 ASP 的网站里，可以编写一个命名为 `global.asa` 的文件。无论用户访问什么页面，只要是这个网站旗下的，都得先访问到 `global.asa`。一般来说，大家会比较喜欢把一些常用的代码——如数据库连接字符串等等，在这里先处理清楚。所以，当一个用户在你的

网站里打开 index.asp 的时候 其实意味着 服务器端解析并执行了 global.asa 和 index.asp 两个文件。

为什么要说这个呢？~~不知道读者有没有发现，刚才的 Application("counter") 只有增加语句，却没有赋值语句——我们当然不能把赋值语句写到读者访问的页面里。除非你想每次读者访问该页面，辛苦加了半天的计数器都会被复位。而真正可以被赋值的，当然就是 Application("counter") 被创建的时候。

我们在 global.asa 中（只能在这个文件里），对 Application("counter") 做手脚：

```
Sub Application_OnStart 'Sub 就是没有返回值得函数
```

```
'当第一个用户访问网站，Application_OnStart 就会被触发
```

```
'之后除非对象终止，再也不会运行
```

```
Application("counter")=0'计数器赋值为 0
```

```
End Sub
```

现在剩下的问题就是。如何在 Application 对象消失之前，把变量的值永久的保存下来——既然对象是在服务器上自己消失的。我们自然也不能按照前面说的那种写数据的方法来写（因为 Application 对象消失意味着无人访问数据库，此时的数据库连接对象早已消失——它可是页面一关掉就没了的对象）。与 Application_Onstart 遥相呼应的，还有一个 Application_Onend，而它，就是救计数器出火海的兄弟了。

代码如下：

```
Sub Application_OnEnd()
```

```
countFile= Server.MapPath(".")+"/counter.txt"
```

```
Set fso=CreateObject("Scripting.FileSystemObject")
```

```
Set file=fso.CreateTextFile(countFile,true)
```

```
file.WriteLine(Application("counter"))
```

```
file.close
```

End sub

代码中似乎多了不少新东西。我们期待能将计数器的东西，写入到文件中去。于是乎这个 Sub 里面，充斥着对于文件的写操作。文件的写操作分为几个步骤

1 先确定文件的文件名和它的虚拟路径。其中 `Server.MapPath(".")` 就是用来获取当前虚拟目录的路径。如果将 `Mappath` 后面的参数 `"."` 改为 `"counter"` 则表示当前虚拟目录下 `counter` 子目录的路径。

2 创建文件对象。

3 调用文件对象的 `CreateTextFile` 方法，第一个参数是该文件的虚拟路径和文件名，第二个参数如果为 `true`，表示创建一个新文件（如果原来有同名文件将被清空），而如果为 `false`，则表示创建一个新文件，但如果原来有同名文件，则打开该同名文件，将来的写文件操作将会把数据写到这个文件的后面，并不改动原来的内容。`CreateTextFile` 将会返回一个文件对象。

4 调用上一步返回的文件对象中的 `WriteLine` 方法，进行写文件操作

5 关闭文件对象

`Application_OnEnd` 将在 `Application` 对象会面死神之际执行，可保证将计数器中的值写入到文件中去。当然，这样的话，我们的 `Application_OnStart` 当然也必须进行更改，否则每次都用 0 来做基础值，就前功尽弃咯

Sub Application_OnStart()

```
countFile= Server.MapPath(".")+"/counter.txt"
```

```
Set fso=CreateObject("Scripting.FileSystemObject")
```

```
Set file=fso.OpenTextFile(countFile)
```

```
Application("counter")=file.ReadLine()
```

```
file.close
```

End sub

和写文件的区别在于，读文件需要使用 `OpenTextFile` 方法，该方法同样返回一个文件对象。`ReadLine` 表示读取文件中的一行——因为写文件我们也是一行行的写，读自然可以一行行的读。读出来的值就会被放到 `Application("counter")` 中去。

至此，一个还算不太耗费资源的计数器就完成了。

至于读者准备如何将 Application("counter")显示到页面上，则当然是仁者见仁，智者见智的事情，在下也不便过问。

6.1.2 Session 对象

本来事情已经完结，因为功能似乎已经基本实现。只是大家都是追求完美的人，对于这样似是而非的解决方法，并不能算太满意。问题出在，这样的访问量，虚假的成分很大，如果有一个无聊（应该说令人感动的）的用户，他发现每次访问该页面，访问量都芝麻开花似的节节高。他时不时地就进来刷一刷，并乐此不疲（会如此做的用户似乎就只有那个站长吧，其他人怕是没这闲工夫）。晚上你照例打开自己的网站一看，乖乖，今天迎来了 2000 个访问量，兴奋莫名下，却不知道都出自一人手笔……

开始时当然觉得这是件好事，可其他用户如果发现你的网站，访问量胡乱增长，其实却没什么人留下信息，就会有点对站长的人品产生怀疑。为了保证做人厚道，我们只能开始重新设计这个计数器的计数方案。而最关键的部分在于，我们如何可以知道，现在来访问的人，到底是不是已经来过呢？其实如果同一个人隔了很长时间再来，我们是可以算作访问量增加的，否则若只有新来的用户才能增加访问量，且不论技术上可否实现，万一你的网站今天来的都是老客户，则计数器将会让你的面子很难看——明明那么多人来过，访问量却不增加。为了满足这样的要求，我们就基本有了一种构思，需要在人来访问的时候，服务器记录来访者的信息，然后一段时间后，服务器自动删除该消息。符合这种构思的东西，就是我们马上就要讲的——Session 对象。

如果说，Application 是一个大浴池，那么 Session 就是每间浴室的钥匙。Session 的语法和 Application 很象：

`Session("变量名称")=变量的值`

他们两者的不同是，Application 对象在任何访问者进来后，都可以共用（所以我们前面加入了锁操作）。而 Session 是针对个人的，一个用户访问引起的 Session 变量变化，和

其他用户，没有任何联系。就仿佛这个变量，是为用户而单独设计的（可以想象成，为你分配了一个账号，你拥有了自己的私人空间那种感觉）

Session 对象开始于用户访问该网站，如果在一段特定时间内用户不提交任何页面，则该对象就宣告结束——貌似完全符合我们刚才的要求——我们在 Application 计数器增加的时候，检查该用户是否有纪录（用户一进入网站我们就让他录口供），就可以知道是不是同一个人。

```
If IsEmpty(Session("counter")) then
```

```
    Application.Lock
```

```
    Application("counter")= Application("counter")+1
```

```
    Application.Unlock
```

```
    Session("counter")=true
```

```
End If
```

IsEmpty 是一个检查变量是否为空的函数。检查 Session("counter")是否为空。如果为空则计数器加 1，并且为 Session("counter")赋值。如果下次页面再被访问而 Session 的生命周期并为结束，计数器就分毫不动☺

计数器只是 Session 的一种用法。学习 Session，最重要的是一定要弄清楚，它是属于用户每个人的。所以每个人的 Session("counter")可能都不一样。但是，重要的信息请勿放置到其中，因为我们对它的过期时间并不能把握得很准——它过期的时候也不会跳出来提示你什么，这是读者需要注意的。

6.2 高级的 ASP 语法

6.2.1 Cookies 对象

经常为用户登录界面上，会看到一些奇怪的字眼，如“保存一周”，“保存一年”之类。当我们登录该网站一次以后，将来的一段时间里（也没有人去数过到底是一周还是一年），

都不必再讨厌的输入用户名和密码——直接会以我们的身份进入。不知道是否有读者想过，到底是谁，记录了我们的这些信息？如果是服务器在记录，岂不是增加了很多服务器的负担么？

上一节有认真学习的，自然而然的会想到可能可以用 Session 来完成上述的高难度事件。只是，读者也许有所不知，Session 有着一个致命的弱点：一旦浏览器关闭，则一切均告完蛋——什么 Session 都不管事儿。

其实，我们可以如此处理。用户信息都放到客户端的机器上来，每次用户访问网站，服务器就检查用户机器上这些存放的信息，达到之前说的那个“保存**天”的目的。而这就是网上流传很久的 Cookies 了。

Cookies 的用法，要用到两个 ASP 中可能是经常用到的东西，我们上面也是见过的哦。这就是 Request 和 Response。

给 Cookies 赋值（即写到客户端的机子上）：

Response.Cookies("user")="飞行岛在下"

从 Cookies 中取值：

Request.Cookies("user")就是 user 这个 Cookie 中包含的值。

如果 Cookies 没有存放任何值，将返回空字符串

因此，判断 Cookies 是否为空，并非使用 IsEmpty，而是像这样直接这么写：

IF Request.Cookies("user")="" then

用法和前面说过的 Application 和 Session 略有所不同。

由于是存放在客户机器上，因此，它不可能被作为计数器使用。什么是作用于服务器端，什么是作用于客户端，这非常重要，它是写 ASP 程序时所需要时刻都在心中铭刻的。

那么，刚才也看到了，那些登陆页面上还有着“保存一年”“保存一月”的选项，这意味着 Cookies 是可以指定过期时间（所谓过期，就是该 Cookie 会在客户端失效），虽然在下认为这个实在没有任何必要——要不就一直留着，要不就永远别留，岂不干脆。不过还是得介绍下它的语法：

Response.Cookies("字段名").expires=时间函数+N，

例如：

`Response.Cookies("name2").expires=date+1`，表示 Cookies 保存 1 天

`Response.Cookies("name2").expires=Hour+8`，表示 Cookies 保存 8 小时

`Response.Cookies("name2").expires=Year+2`，表示 Cookies 保存 2 年

有了 Application 和 Session 的基础，Cookies 总算是容易让人接受。只要记住它的使用范围和方法即可。

有了上面的知识，我们只需在登陆页面里对用户 Cookies 进行检查，如果存在则直接登陆，如果不存在则必须重新输入用户名和密码。有些读者的浏览器，可能设置了比较高的安全性，将会禁止 Cookies 的使用，需要在 Internet 选项中打开它。

6.2.2 Server 端对象及 Server 端相关信息

这个题目看来可谓无聊至极。在下自己写的都双手发抖。不过想到它们能带来的快乐，在下又觉得实在有必要给大家说出来。

我们能在网上看到很多 BBS 的论坛斑竹，为了显示自己无上权力时，经常放出话说，“***如果再发帖，则封 IP”之类的话语。我们既然来写网页，当然需要享受这种带来绝对腐化的绝对权力。而这些特殊的信息，如获取客户端的 IP，需要的就是 Sever 端对象和 Server 端相关信息。

Request.ServerVariables 语法：

顾名思义，ServerVariables 表示的一些特殊的网络信息。比如，有时希望获得服务器端或者客户端 IP 地址，有时希望获得被打开的 ASP 页面所在的虚拟路径等等。

`Request.ServerVariables("LOCAL_ADDR")` 返回服务器端 IP 地址

`Request.ServerVariables("REMOTE_ADDR")` 返回客户端 IP 地址

`Request.ServerVariables("APPL_PHYSICAL_PATH")` WWW 根目录的实际路径，指的是服务器端的路径

`Request.ServerVariables("PATH_INFO")` ASP 文件所在目录的虚拟路径

在下经常使用它，一般是在本页面提交到本页面，或者在页面里加参数跳转回自己的时候。

比如下面这种情况（当前文件是 test2.asp）：

```
Response.Redirect "test2.asp?user=sandycs"
```

'Redirect 代表页面跳转

'这个语句就是自己跳转到自己，并加入 user=sandycs 的参数

这本来没有什么问题。可是如果我有一天将该文件名称改变为 test3.asp，则这句语句也势必发生改变，万一页面中类似的语句很多，就容易出现漏掉的情形。于是我们可以改造该语句：

```
MyPath=Request.ServerVariables("PATH_INFO")
```

'此时的 MyPath 就是完整的虚拟路径，路径里包含着该 asp 的文件名

```
Response.Redirect MyPath+"?user=sandycs"
```

这样就可以省略一切烦恼。

`Request.ServerVariables("PATH_TRANSLATED")` ASP 文件所在目录的实际路径

关于虚拟路径和实际路径的区别，已经在前面说明。说实话，似乎基本用不到实际路径。只是特别写出来和虚拟路径区别开。

以上所写的语法，都是返回一个字符串。一般都会被直接用作显示。其中的 IP 地址，也可以用作地址段屏蔽等等的高级技巧（比如将 214.48.23.0-214.48.23.255 的地址全部屏蔽）

Server 对象：

这里只介绍一下 Server 对象中最为常用的 3 种。

`Server.MapPath` 这个相信大家都不陌生。因为刚才已经使用过了

假设我们把网站的根目录设置在 D:\root 下面（根目录的设置一般是在 IIS 里面完成的，且虚拟目录的实际路径和根目录的设置是完全没有关系的）

而目前网络所在目录为 D:\test

`Server.MapPath("/")` 将会返回 `D:\root`

`Server.MapPath("/abc.txt")` 将会返回 `D:\root\abc.txt` (注意斜杆和反斜杆)

`Server.MapPath(".")` 将会返回 `D:\test`

Server.HTMLEncode:HTML 编码

不知道，有没有读者，在数据库那章的留言板介绍部分想过一个问题。如果一个用户，撰写留言板的内容时，直接使用 HTML 代码留言，而你在显示该用户留言内容的时候，按照原样显示出来的时候，浏览器会把他的留言当作 HTML 来解析

比如 `ha` 这样的留言，就会被显示为 **ha**，而肯定也有些别具用心的读者，使用 `table` 或者 `td, tr` 之类会打乱页面结构的代码，就会让页面编写者非常的不爽了。于是我们希望，数据库存储留言的时候，和原来真实的字符略有不同，但是显示出来的时候既和原来输入的看起来一样，又不会被浏览器解析掉而破坏页面。而这种将原先字符转变为另一种形式的方法，我们就叫它做“编码”。HTML 编码就是这么一种特殊的，针对 HTML 的编码形式。

用法如下：

`Server.HTMLEncode("需要被编码的字符串")`

经过该步骤后，返回值就是编码过后的字符串了，如果仔细观察，会发现其实区别不大。

26 个英文字母还是被保留下来，而空格，尖括号则会发生改变。比如 `<font` (注意空格) 会被编码成 `<font `，编码过后的数据在浏览器下就会真实显示出尖括号或者空格来，而不会被解析。

Server.CreateObject : 创建服务器端对象

这个相信不用赘述。因为前面的数据库联接部分已经进行过类似的尝试。

可以被 `CreateObject` 创建的对象，我们称他为 ActiveX 对象，想要正确地使用它，同样必须调整浏览器的安全设置。(有的读者会将浏览器调整为禁止 ActiveX 对象)

用法如下：

`Server.CreateObject("ActiveX 对象名称")`

大部分时候，`Server` 这个前缀是可以省略掉的。目前比较经常用到的 ActiveX 对象也只有表示数据库的 `ADODB` 和表示文件处理的 `Scripting.FileSystemObject`

第六章看来是结束最快的一章。我数了一下，前后 6 页不到，比第一章的内容还少。在下不禁反省了一下，是在下开始偷工减料呢，还是确实已经不需要讲上那么多？纵观这章的内容，也许一时间，读者会觉得，一下子也用不到例子上来。除了一个简单的计数器和一点点的登陆技巧外，基本上算是一无是处。在下只能请读者在将来的编程中，多自觉地采用一些不太熟悉的技术，主要是为了拓宽自己的思路，当复杂的问题来临时，可以游刃有余，解决问题也多了很多途径，这样在下就老怀大慰拉。

我不是个随便的人，我随便起来不是人，谢谢大家！

第七日：剑神

其实，到这里为止，读者编写一个简单的 ASP 页面骗骗 MM，已经绝对不是一件困难的事情了。严格的说，如果去找工作，已经可以将自己的技能里，骄傲的将 ASP 作为你一项“熟练并精通”的编程语言载入你的简历中去——并非弄虚作假，如果把上面教过的知识都灵活应用的话，已经可以胜任大部分 ASP 的工作。剩下的一些，就是本日的任务了。我会交给大家一些常用到的难点解决方法和技巧。相信将来即使公司出现 ASP 面试，也不会超过本日的难度。因此，可能会遇到一些艰涩难懂的部分，还望读者能多看几遍，将它们吃透，将来赶英超美，指日可待。

7.1 页面刷新的问题

读者一定奇怪了，页面刷新有什么学问可言。我们先来看看下面这个简单的问题：在页面上画上两个下拉式选单。当选择第一个的时候，第二个下拉式选单随着第一个下拉式选单的选择而发生变化。举个例子来说，第一个选单代表国家，而第二个代表省份。当选中中国的时候，则第二个选单显示为中国的省份。看似简单的要求，实际做的时候却存在着一个奇怪的问题。

假设数据库中有两张表，

表名:CountryTable

栏位:ID 自增型栏位

栏位:CountryName 字符串型栏位。存放国家名称

表名:ProvinceTable

栏位:ID 自增型栏位

栏位:CountryID 数字型栏位。存放国家 ID（与上张表相关）

栏位:ProvinceName 省份栏位。存放省份名称

我们先按正常的思路来做：(假设数据库连接变量 connLaputa 都已经弄好)

1 放上两个下拉式选单

```
<select name="country" onchange="countrysel()">

<%

    strSql="Select * From CountryTable"

    Set RSCountry=CreateObject("ADODB.RecordSet")

    Set RsCountry=connLaputa.Execute(strsql)

    While Not RSCountry.EOF

%>

        <option value="<%=RSCountry("ID")%>">

            <%=RSCountry("CountryName")%> </option>

    <%      RSCountry.MoveNext

        Wend

%>

</select>

<select name="province"> </select>
```

第二个下拉式选单，由于不能在编写网页时确定。暂时不写出 option

2 当国家选单的选择事件发生时，通过 JavaScript 函数 `countrysel` 提交页面的选择给本页面。(此部分略) 另需注意的是：当页面刚进入的时候，并不会触发选择事件。所以第一次将省份的下拉式选单带出的动作，需要读者另行处理。(将来不再赘述此点)

3 将第二个下拉式的菜单代码里面放入服务器端代码。我们假设提交的国家选单的值被存放在 `CountryID` 变量里。

```
<select name="province">

<%

    strSql="Select * From ProvinceTable where CountryID="
```

```
strSql= strSql+Cstr(CountryID)

Set RSProvince=CreateObject("ADODB.RecordSet")

Set RSProvince =connLaputa.Execute(strsql)

While Not RSProvince.EOF

%>

<option value="<%= RSProvince("ID")%>">

    <%= RSProvince ("ProvinceName")%> </option>

<%    RSProvince.MoveNext

Wend

%>

</select>
```

代码并不复杂,做起来貌似也并不麻烦。不过,大家是否注意到,提交的过程,会带来一次的页面刷新。当页面上的东西很多时——比如注册页面,就会带来很大的麻烦。整个页面闪烁一次的感觉,绝对不好受。换句话说,我们不能提交数据。

是的,我们不能通过提交数据,让那个该死的刷新动作出现。但是大家都知道,HTML 页面在读取完成后,是否还能再对上面的 HTML 代码作修改呢——似乎只能再让浏览器读取一次阿。好在当时的设计者,并没有把事情做绝,他们提供了一种方式,通过 Javascript 重写页面控件内部的 HTML 代码,以达到让页面上的东西更新的目的。这个重要的控件属性就是:InnerHTML。

几乎每个 HTML 的控件都有 InnerHTML 的属性。但是有的是可以重写的,有的不行。在下把不能重写的控件列举于此:

COL, COLGROUP, FRAMESET, HTML, STYLE, TABLE, TBODY, TFOOT, THEAD, TITLE, TR

我们的 Select 元素,显然是可以下手的对象。为了不让页面刷新,我们可以这么做,在页面读取的时候,事先将所有的省份数据存放到 JavaScript 变量中。将来当国家菜单选

择事件触发的时候，我们就通过这些变量来重写省份下拉式菜单的代码。

把上面的代码改正如下：

```
<select name="country" onchange="countrysel()">

<script>

    var provin=new Array();

    var provinID=new Array();'一个用于存放名称，一个用于存放 ID

</script>

<%

    strSql="Select * From CountryTable"

    Set RSCountry=CreateObject("ADODB.RecordSet")

    Set RsCountry=connLaputa.Execute(strsql)

    While Not RSCountry.EOF

%>

    <option value=" <%=RSCountry("ID")%>">

        <%=RSCountry("CountryName")%> </option>

    <script>

        provin[<%=RSCountry("ID")%>]=new Array();

        //这是多维数组的声明方式，

        //从此以后，provin[<%=RSCountry("ID")%>]本身又是一个数组

        provinID[<%=RSCountry("ID")%>]=new Array();

    <%

        strSql="Select * From ProvinceTable where CountryID="

        strSql=strSql+Cstr(RSCountry("ID"))

        Set RSProvince=CreateObject("ADODB.RecordSet")

        Set RSProvince =connLaputa.Execute(strsql)

        Dim intI '用于数组序数的增长
```

```
intI=0

While Not RSProvince.EOF

'这里是从数据库读取省份的过程

%>

provin[<%=RSCountry("ID")%>][<%=intI%>]

    = "<%=Cstr(RSCountry("ProvinceName"))%>";

provinID[<%=RSCountry("ID")%>][<%=intI%>]

    = "<%=Cstr(RSCountry("ID"))%>";

<% intI=intI+1

    RSProvince.MoveNext

Wend

%>

</script>

<%    RSCountry.MoveNext

Wend

%>

</select>
```

看起来似乎有些五彩斑斓，红色字体就是改动和增加的部分，它就是将数据表的数据，全部存入变量的过程。变量采取多维数组的模式存取，第一维是 CountryID,第二维则是序号。

而我们可以写一个 countrysel 函数，进行 onclick 事件后的处理：(该函数必须写在刚才的代码之后，因为刚才的代码才开始定义 provin 和 provinID 两个数组。)

```
function countrysel()

{

    var countryID;

    countryID=document.all.country.value;
```



```
var inti;

var outputstring;//准备用来存放拼装后的 HTML 代码。

for (inti=0;inti<provin.length;inti++)
{
    outputstring=outputstring+"<option value='"
    outputstring=outputstring+provinID[countryID][inti]
    outputstring=outputstring+"'>"
    outputstring=outputstring+provin[countryID][inti]
    outputstring=outputstring+"</option>"
}

document.all.province.innerHTML=outputstring;

//上面的这句语句就是 innerHTML 的用法
}
```

函数执行后，第二个下拉式菜单就会被更新，而页面则没有什么变化，属于理想的刷新效果。只是如果 province 表比较大的时候，则第一次读入页面的速度会很慢——在下以前做过一个数据量较大的页面，第一次读入时间竟然到达 5 秒之久，这是任何一个用户都难以忍受的。所以，读者选用第二种方法的时候，应该先进行认真的考察。

7.2 XML 的启示

刚才也说到，其实我们解决的并不完美。主要的情况是，有时候会要求 3-4 个的下拉式菜单之间发生关系——我们的变量将会变成 4 维数组，且上节也提到过，数据量大时候的页面首次读入速度，也是个令人抓狂的问题。

仔细分析一下，何必如此傻傻的将所有数据全部捞取出来呢？事实上，选择“中国”的用户，压根就不关心美国到底有什么省。我们最终还是殷切的盼望着，当用户选择国家的时候，系统才去捞省份的数据——尽管貌似必须通过提交页面来实现。

在下可以负责的告诉大家，我们还是可以绕过提交页面——工序有些复杂，但实现之后却绝对是爽利的很。但此之前，在下需要稍微介绍一下 XML 的技术。

XML 可以通过一本 300 页的书来和大家讲述。看本章之前，在下推荐一本写得不错的 XML 书籍给大家——应该说，无论你将来准备掌握程序世界的哪个部分，XML 都是你的必修课，不合格不给毕业证那种课程。

这本 XML 的书籍由台湾的两只老虎工作室写成，用语生动幽默，知识面涵盖甚广。值得读者仔细阅读，下载地址是：

[无废话 XML 下载地址](#)

XML 我们可以把它看作是一种简单的数据库 和 ACCESS 较为类似。而它的构成形式，与 HTML 非常相似，只是规矩更加严格。让我们看看 XML 的示例代码：

```
<COUNTRY>
  <NAME>China</NAME>
  <PROVINCE>
    <NAME>Beijing</NAME>
  </PROVINCE>
  <PROVINCE>
    <NAME>ShangHai</NAME>
  </PROVINCE>
</COUNTRY>
```

看起来和 HTML 非常相似，也是有节点，也是有属性等等。只是 XML 要求很多：

- 1 节点一旦开门，必须关门。否则报错。
- 2 大小写敏感。<COUNTRY>开的门不能使用</cOuntry>来关
- 3 节点层次分明，每种程序语言都可以方便的访问每一个标准的 XML 节点中的元素和值。
- 4 XML 文件经常是自行拼装。数据需要如何组合，全看编写者自己来决定。在下这里只是提供了一种方案。其实设计 XML 的架构也是一门功夫，不过我们用到 XML 的部分，

都不甚高级，只要构架和读取方便，就可以满足要求。

整段 XML 存放了关于国家和省份的数据，由于数据量较小，看起来结构清晰易懂。不过这并非我们这次所需要的。当年黄裳的九阴真经里写过，绕过页面刷新而又能提交数据的方法，只能通过 XML 进行。用户可以在前台（客户端），抛送组合好的 XML 代码（本例就是国家的信息）给后台（服务器端），服务器端经过处理，重新组合 XML 代码（本例为该国家所对应的省份数据），再次抛送回客户端。而这整个过程，都不会有页面的转向，也不会对页面进行刷新。

此等奇技淫巧，竟也能登大雅之谈，在下当时也甚为诧异。只是按着一试，竟然也诚如真经所说，页面纹丝不动，完全没有刷新的意愿，而数据也达到了及时更新的目的。在下特地把代码帖于此，供世人参考：

首先当然是页面部分。主要是用 Javascript 重写上面的 countrysel 函数（原来的数组变量存储和读取过程都可以统统放弃），我们把上面那章的代码略作修改如下：

```
<select name="country" onchange="countrysel()">
<%
    strSql="Select * From CountryTable"
    Set RSCountry=CreateObject("ADODB.RecordSet")
    Set RsCountry=connLaputa.Execute(strsql)
    While Not RSCountry.EOF
%>
        <option value="<%=Cstr(RSCountry("ID"))%>">
            <%=Cstr(RSCountry("CountryName"))%> </option>
    <%
        RSCountry.MoveNext
    Wend
%>
</select>
```

省去变量存储的代码可谓是一身轻松。我们先准备一个函数,专门用于抛送 XML 数据,之后的程序只要调用它即可。

```
function postxml(method,strxml)
```

```
{
```

```
    var strurl;//该变量将用于存储抛送路径
```

```
    var objHttp=new ActiveXObject("Microsoft.XMLHTTP");
```

//我们把抛送路径定为 `mainbackground.asp`,而不是本身,是为了区分开来客户端和服务端页面。让整个网站系统看起来条理清晰并且分工明确。

//所谓的 `method`,其实是为了将来使用,当传入不同的 `method`,我们也会在 `mainbackground.asp` 页面里做区分处理,这样这个 `xml` 抛送函数就可以多处使用,而不是仅仅抛送国家数据获取省份那么简单了。

```
    strurl="mainbackground.asp?action="+method;
```

//下面的语句,类似于服务器端的 `CreateObject`,不同的是它写在客户端代码里,意味着建立的是客户端对象。且是 Javascript 的语法。

```
    //XMLDOM 就是网页中最为常用的 XML 对象
```

```
    var objDom = new ActiveXObject("MICROSOFT.XMLDOM");
```

```
    //loadXML 用于将拼装好后的 xml 字符串读入到创建好的 xmldom 对象中
```

```
    //本例的 xmldom 对象就是 objDom,而 strxml 则是由外部函数当作参数传入的。
```

也就是说,拼装 `xml` 的过程并不在本函数内。

```
    objDom.loadXML(strxml);
```

//`open` 方法是发送前的必经步骤,可以理解是为发送打开了一扇门。并把发送目的地也指明了。

//第一个参数 `POST` 代表是发送,如果写为 `GET`,表示获取页面的 HTML 代码(效果和你在网页上点右键,然后选“查看源文件”是一样的)。

```
    //第二个参数是需要抛送的后台页面。不必多言
```

```
    //第三个参数表示是否同步。如果参数值为 false 则数据抛送后继续执行下面的代码。
```

如果参数值为 `true`,则会一直等待到抛送成功为止(有的时候因为网络状况,抛送过程会

比较缓慢)

```
objHttp.open("POST",strurl,false);
```

//send 方法代表正式开始抛送 ,此时需要把 objDom 作为参数 ,表示发送的是该对象。

```
objHttp.send(objDom);
```

//通过 load 方法将抛送后由后台返回的数据进行接收 (注意 , xml 要求返回的数据也

是 xml 的标准格式 , 否则将会接收到空字符串) , 将接收到的数据存入 objDom 中

```
objDom.load(objHttp.ResponseStream);
```

//返回 objDom 给该函数的调用者

```
return(objDom);
```

```
}
```

```
function countrysel()
```

```
{
```

```
var countryID;
```

```
countryID=document.all.country.value;//仍然是先获得 countryID
```

//准备拼装 xml 字符串 , 并调用上述函数进行抛送

//同样要建立 XMLDOM 对象

```
var objDom = new ActiveXObject("MICROSOFT.XMLDOM");
```

```
strxml="<COUNTRY>"+countryID+"</COUNTRY>";
```

//调用 postxml 进行 xml 的抛送

//GETPROVINCE 是我们自己定的名称。后台页面将会接到该参数。并根据参数选择

所要进行的操作 (主要是为将来扩展使用)

```
objDom=postxml("GETPROVINCE",strxml);
```

```
var returnvalue;
```

//这部分需要在后台页面写出时再进行讲述

//显然后台页面我们希望返回一个拼装好的下拉式菜单 HTML 代码

```
outputstring=objDom.selectSingleNode("//returnvalue").text
```

```
document.all.province.innerHTML=outputstring;
```

```
}
```

一个拼装的相当简单的字符串被抛到后台：<COUNTRY>countryID</COUNTRY>。

中间的 countryID 自然是用户选择的国家 ID。现在开始写服务器端的后台页面代码：

以下的代码将会被放置到 mainbackground.asp 文件中

```
Action=Request("action")
```

'Action 就是在抛送 xml 数据时附带的一个页面参数，请仔细查看客户端代码

```
IF Action="GETPROVINCE" THEN
```

'当得知传入的 action 参数为 Getprovince,表示需要获得省份

'这主要是为了将来扩展使用，比如 4 个下拉式菜单的时候，就需要通过 action 区分

'下面的函数是初始化数据库连接，而 GetConnection 函数在下也不再写出

```
Set MyConn=CreateObject("Adodb.Connection")
```

```
Set MyConn=GetConnection()
```

'创建服务器端的 XML 对象

```
Set objGetDom =CreateObject("MSXML2.DOMDocument.3.0")
```

'这里设定 objGetDom 为同步。关于同步异步，前面已经有简要的说明

```
objGetDom.async = false
```

'load Request 代表将所有的传入的 XML Dom 对象都读取出来。

```
objGetDom.load Request
```

```
Dim CountryID
```

'我们可以看到，传入的 xml 字符串非常简单，所以要获得 country 节点的值，可以直接使用下面的语句

'如果 xml 字符串的结构较为复杂，我们就通过路径的形式来获取节点的值，举个例子来说 以上面我们介绍过的那个 xml 例子来说(就是前面把省份名字都列出来的 xml 代码例子)，如果想获得 Beijing 这个值，就需要使用 selectSingleNode("//Country/Province/Name")

```
CountryID=objGetDom.selectSingleNode("//Country ").text
```

'得到 CountryID，就可以从数据库中获得所有的省份

```
Dim strSql
```

```
strSql="Select * From ProvinceTable where CountryID="

strSql=strSql+Cstr(CountryID)

Set RsProvince=CreateObject("ADODB.RecordSet")

Set RsProvince=MyConn.Execute(strsql)

Dim strOutputstring '拼装返回字符串

strOutputstring="<returnvalue>"

While Not RsProvince.EOF

    strOutputstring=strOutputstring+"<option value="

    strOutputstring=strOutputstring+Cstr(RsProvince("ID"))

    strOutputstring=strOutputstring+">"

    strOutputstring=strOutputstring+Cstr(RsProvince("ProvinceName"))

    strOutputstring=strOutputstring+"</option>"

    RsProvince.MoveNext

Wend

strOutputstring=strOutputstring+"</returnvalue>"

'将拼好的字符串抛送回去

Dim objBackDom

Set objBackDom = CreateObject("MSXML2.DomDocument")

objBackDom.loadxml(strOutputstring)

'下面的两句话,就是建立了和客户端页面的通信,同时终止该后台页面的运行

'而已经载入 xml 的 objBackDom,也成功地完成了历史使命。

objBackDom.save Response

Response.end

End IF
```

至此,整个 xml 抛送的后台服务器处理过程就告完成。配合好客户端代码,这样的页面显示效果,就是大部分用户可以接受的了。当然它也有它的弊端,首先服务器端页面调试

很不方便，因为抛送过程是后台进行，我们只能通过客户端页面的显示情况来看是否后台运作正常。其次是与前面的数组解决方法不同，这种方法每次改选下拉式菜单都会引起后台的服务器操作，在单笔数据量较大的时候，用户也许会产生反感，他们可能会偏向喜欢长痛不如短痛的方式（即数组解决方式），宁可第一次读取时间长一些，换来后面的速度感。尽管如此，我们本着学习的目的，总算是对 ASP 中的一个著名问题，有了个轮廓与粗浅的认识，将来可以在技术运用方面也多了些可走的路。



HOMEWORK: 编写一个四级的下拉式菜单。第一级带出第二级的数据，第二级带出第三级，以此类推。题材可自拟。注意，**每个下拉式菜单都应该有默认的选择**，不要出现某级菜单因为没有父亲菜单的事件触发而无数据的情况。

7.3 出错处理及其应用

有些读者可能已经颇不急待的开始编写网页，甚至网站都开始小有规模。不知道是否有注意到，尽管你已经非常的小心，竭尽所能的考虑周到并注意代码编写的规范，可是你的网页，还是有出错的可能性？因为用户的古怪刁钻，绝对是你难以想象的。据报道曾有国外的大妈那鼠标当作脚踏板，还致电客服抱怨按钮太小.....

当程序运行到出错状态时——要出错非常的容易，比如对一个空的变量取长度（注意空变量不是空字符串），比如做除法的时候除数为 0 等，错误会在用户浏览时跳出来，骚扰用户平静的心湖。很多时候，错误并非由我们这些代码编写者造成。举例来说，我们在网页上做一个类似于计算器的四则运算功能，运算数和运算符号都由用户来输入，此时，就可能出现意外的情况。用户在输入运算数的时候，根本不考虑什么溢出情况，直接输入超过程序接收范围的数字让计算机来算，而我们又未作控制，页面就自然报错。我们无法每个程序都去仔细考虑它出错的可能性，相反地，只要在弹出错误框前，拦截下来，告诉电脑“我知道

错了，你别乱弹”，这个过程就是在下所说的[出错处理](#)。

我们先看看 Javascript 的出错处理代码范例，及其运行效果：

准备一个有可能出错的页面先：(html 代码，只把最关键的部分写出来)

```
<table>

  <tr>

    <td><input type="text" name="num1"> </td>

    <td>÷ </td>

    <td><input type="text" name="num2"> </td>

    <td>= </td>

    <td><input type="text" name="result"> </td>

  </tr>

  <tr>

    <td>

      <input type="button" value="计算" onclick="calc()">

    </td>

  </tr>

</table>
```

只是一个用于计算除法的简单页面。当用户点击按钮时，就开始计算。而计算的函数就是 calc。

我们把 calc 写出来：

```
<script>

function calc()

{

  var num1;

  var num2;

  num1=document.all.num1.value;

  num2=document.all.num2.value;
```

```
document.all.result.value=num1/num2;

}

</script>
```

为了制造程序错误，我特地把 result 后面的.value 给去掉了。我们立刻来试验一下该网页出错的样子。

打开页面，点击“计算”按钮。出现以下的提示框：

（有些读者的电脑可能并不会跳出该框来，这和浏览器的设置有关系。但是，无论如何左下角都会有一个惊叹号进行提示，点击惊叹号即可看到错误报告）



相信很多人都对这种错误报告相当熟悉——太多制作不良的网站弹出过这样的东西。仍然保留错误代码，但是在下对上面的函数进行这样的处理：

```
<script>

function calc()

{

    try

    {
```

```
var num1;

var num2;

num1=document.all.num1.value;

num2=document.all.num2.value;

document.all.result.value=num1/num2;

}

catch(e)

{

    //如果 try 语句括起来的部分出错，则直接跳往此处

    alert("代码好像有错哦~~~");

}

finally

{

    //此处的代码无论出错都会被执行

    alert("无论错误正确，都会执行的语句");

}

}

</script>
```

重新观察该页面（刷新），可以欣慰得看到，出错的提示框没了，取而代之的是“代码好像有错哦”的提示框，且 finally 所包括起来的代码也会被执行——换句话说，错误被包装起来了。这当然是件令人欣慰的事情。只需要记住，catch 语句只会在出错时被执行，而 finally 则是不计对错，一概运行的。

catch 后面的(e)，代表的是捕捉的错误类型变量。这个变量，记录着有关错误的信息。它有两个属性：description 和 number。一个代表错误的详细描述，另外一个则代表错误号。这个变量虽然可用可不用，但最好把它跟随在 catch 的后头作为一种标准的写法——并非在下想如此强制要求，实在是因为不这么做，你的错误处理语句就不会生效。

如此一来，大部分没有把握的函数，我们都可以给它来这么一手，避免一些让用户觉得不友好的尴尬局面。

这么好用的功能，当然不会让客户端独美，服务器端依然也有着自己的错误处理部分（说白了，其实是 VBScript 的错误处理部分）

格式如下：

`On Error Goto errhandle` errhandle 是自己命名的错误处理部分名称，类似函数

可能出错的代码

.....

`Exit` '如果不退出，则 errhandle 的语句依然会执行，因为是顺序执行下去的

`errhandle`

错误处理代码，和 Javascript 中的 catch 部分相同

上面的代码已经写的很明白，我们可以在出错的时候直接跳入另一段代码继续执行下去，简便易行，算是居家旅行之必备良药。但它也并非 VBScript 最凶狠的部分，VBScript 中的出错处理中有一绝招——无视代码错误，继续运行。

`On Error Resume Next`

可以理解为一把尚方宝剑，它会强行的把它以下的代码执行结束，令人心动不已。而如果你想查查到底出错没有（否则这么被它强制的往下执行，总觉得自己会被完美的表象所欺骗），也有简便易行的手段，我们在代码的最后添上这样的判断：

`If err.Number <> 0 then`

'如果刚才那一大坨代码出错了就执行这里的语句

'因为出错了，err.Number 就会不等于 0，因为它是错误代号。

`End IF`

总的来说，错误处理机制提供了一种缓和的，可控制的异常发生处理方案。主要是增加

用户的界面友好度——当然我们会麻烦一点。虽然好用，也不推荐读者每个函数都加上错误处理那样极端，因为有些代码，比如打印一个字符串，是永远不会出错的（除非电脑有问题），错误处理本身也会带来一定的系统开销，并非万金油哦☺

我不是个随便的人，我随便起来不是人，谢谢大家！

后记

虽然想说的东西还有很多，可作为一本书，还是要保证它的系统完整性。七日的学习，实际上并不轻松。在下虽然有些掏心掏肺的感觉，却仍然觉得一门知识，是很难一下子完全的掌握下来的。仍需要读者们大量的练习与实践，同时也欢迎读者到在下的网站上提出问题，我一般是知无不言，言无不尽。[点此进入](#)，欢迎您的到来。

本书耗时 3 个月，虽然在下花了很多心血，却仍然难以避免纰漏。特别到后面几章，代码多起来的时候，在下自己都有些检查不过来，毕竟是肉眼凡胎。方才从前言看到后记，竟有恍如隔世的感觉，本以为会无法完成此书，看来还是**有志者事竟成**。愿用这句话作为本书的终结，与读者共勉。

我不是个随便的人，我随便起来不是人，