以前一直用vbscript来操作文件夹，才发现原来使用JavaScript也是可以的，肯定不如vbs用的简单，不过学习一下还是不错的

在Windows平台上, js可以调用很多Windows提供的ActivexObject,本文就使用js来实现文档处理, 和使用js编写ActiveX做一个简单介绍.

复制代码 代码如下:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">  
<html>  
 <head>  
 <title> New Document </title>

</head>  
<script type="text/javascript">  
 function readFolder(){  
 var filePath = "d:\\test\\";  
 var fso = new ActiveXObject("Scripting.FileSystemObject"); //加载控件  
 var f = fso.GetFolder(filePath);  
 var underFiles = new Enumerator(f.files); //文件夹下文件  
for (;!underFiles.atEnd();underFiles.moveNext()){   
 var fn = "" + underFiles.item();   
 //alert(fn);  
 var content = readFile(fn,fso);  
 alert(content);  
 }

}  
function readFile(path,fso){  
var f1 = fso.GetFile(path);  
var fh = fso.OpenTextFile(f1, 1/\*reading\*/);  
 var content = '';  
 while ( !fh.AtEndOfStream ) {  
 content += fh.ReadLine();  
 }  
 fh.close()  
 return content;  
}

function writeExcel(){  
 var ExcelApp = new ActiveXObject("Excel.Application");  
 var ExcelSheet = new ActiveXObject("Excel.Sheet");  
 ExcelSheet.Application.Visible = true;  
 ExcelSheet.ActiveSheet.Cells(1,1).Value = "This is column A, row 1";  
 ExcelSheet.SaveAs("d:\\TEST.XLS");  
 ExcelSheet.Application.Quit();  
}  
</script>  
 <body>  
 <input type="button" value="遍历文件夹" onclick="readFolder()">  
<input type="button" value="写excel" onclick="writeExcel()">  
 </body>  
</html>

JavaScript中ActiveXObject对象是启用并返回 Automation 对象的引用。使用方法：

newObj = new ActiveXObject( servername.typename[, location])

ActiveXObject 对象语法有这些部分：其中newObj是必选项。要赋值为 ActiveXObject 的变量名。  
servername是必选项。提供该对象的应用程序的名称。  
typename是必选项。要创建的对象的类型或类。  
location是可选项。创建该对象的网络服务器的名称。

切记：ActiveX是微软的东西，故而这玩意儿只有IE才支持！

javaScript中利用ActiveXObject来创建FileSystemObject操作文件

一、功能实现核心：FileSystemObject 对象

要在javascript中实现文件操作功能，主要就是依靠FileSystemobject对象。

二、FileSystemObject编程

使用FileSystemObject 对象进行编程很简单，一般要经过如下的步骤： 创建FileSystemObject对象、应用相关方法、访问对象相关属性 。

（一）创建FileSystemObject对象

创建FileSystemObject对象的代码只要1行：  
var fso = new ActiveXObject("Scripting.FileSystemObject");  
上述代码执行后，fso就成为一个FileSystemObject对象实例。

（二）应用相关方法

创建对象实例后，就可以使用对象的相关方法了。比如，使用CreateTextFile方法创建一个文本文件：  
var fso = new ActiveXObject("Scripting.FileSystemObject");  
var f1 = fso.createtextfile("c:\\myjstest.txt",true");  
（三）访问对象相关属性  
要访问对象的相关属性，首先要建立指向对象的句柄，这就要通过get系列方法实现：GetDrive负责获取驱动器信息，GetFolder负责获取文件夹信息，GetFile负责获取文件信息。比如，指向下面的代码后，f1就成为指向文件c:\test.txt的句柄：  
var fso = new ActiveXObject("Scripting.FileSystemObject");  
var f1 = fso.GetFile("c:\\myjstest.txt");  
然后，使用f1访问对象的相关属性。比如：

复制代码 代码如下:

var fso = new ActiveXObject("Scripting.FileSystemObject");  
var f1 = fso.GetFile("c:\\myjstest.txt");  
alert("File last modified: " + f1.DateLastModified);

执行上面最后一句后，将显示c:\myjstest.txt的最后修改日期属性值。  
但有一点请注意：对于使用create方法建立的对象，就不必再使用get方法获取对象句柄了，这时直接使用create方法建立的句柄名称就可以：

复制代码 代码如下:

var fso = new ActiveXObject("Scripting.FileSystemObject");  
var f1 = fso.createtextfile("c:\\myjstest.txt",true");  
alert("File last modified: " + f1.DateLastModified);

三、操作驱动器（Drives）

使用FileSystemObject对象来编程操作驱动器（Drives）和文件夹（Folders）很容易，这就象在Windows文件浏览器中对文件进行交互操作一样，比如：拷贝、移动文件夹，获取文件夹的属性。  
（一）Drives对象属性  
Drive对象负责收集系统中的物理或逻辑驱动器资源内容，它具有如下属性：  
l TotalSize：以字节（byte）为单位计算的驱动器大小。  
l AvailableSpace或FreeSpace：以字节（byte）为单位计算的驱动器可用空间。  
l DriveLetter：驱动器字母。  
l DriveType：驱动器类型，取值为：removable（移动介质）、fixed（固定介质）、network（网络资源）、CD-ROM或者RAM盘。  
l SerialNumber：驱动器的系列码。  
l FileSystem：所在驱动器的文件系统类型，取值为FAT、FAT32和NTFS。  
l IsReady：驱动器是否可用。  
l ShareName：共享名称。  
l VolumeName：卷标名称。  
l Path和RootFolder：驱动器的路径或者根目录名称。

（二）Drive对象操作例程

下面的例程显示驱动器C的卷标、总容量和可用空间等信息：

复制代码 代码如下:

var fso, drv, s ="";  
fso = new ActiveXObject("Scripting.FileSystemObject");  
drv = fso.GetDrive(fso.GetDriveName("c:\\"));  
s += "Drive C:" + " – ";  
s += drv.VolumeName + "\n";  
s += "Total Space: " + drv.TotalSize / 1024;  
s += " Kb" + "\n";  
s += "Free Space: " + drv.FreeSpace / 1024;  
s += " Kb" + "\n";  
alert(s);

四、操作文件夹（Folders）

涉及到文件夹的操作包括创建、移动、删除以及获取相关属性。  
Folder对象操作例程 :  
下面的例程将练习获取父文件夹名称、创建文件夹、删除文件夹、判断是否为根目录等操作：

复制代码 代码如下:

var fso, fldr, s = "";  
// 创建FileSystemObject对象实例  
fso = new ActiveXObject("Scripting.FileSystemObject");  
// 获取Drive 对象  
fldr = fso.GetFolder("c:\\");  
// 显示父目录名称  
alert("Parent folder name is: " + fldr + "\n");  
// 显示所在drive名称  
alert("Contained on drive " + fldr.Drive + "\n");  
// 判断是否为根目录  
if (fldr.IsRootFolder)  
alert("This is the root folder.");  
else  
alert("This folder isn't a root folder.");  
alert("\n\n");  
// 创建新文件夹  
fso.CreateFolder ("C:\\Bogus");  
alert("Created folder C:\\Bogus" + "\n");  
// 显示文件夹基础名称，不包含路径名  
alert("Basename = " + fso.GetBaseName("c:\\bogus") + "\n");  
// 删除创建的文件夹  
fso.DeleteFolder ("C:\\Bogus");  
alert("Deleted folder C:\\Bogus" + "\n");

五、操作文件（Files）

对文件进行的操作要比以上介绍的驱动器（Drive）和文件夹（Folder）操作复杂些，基本上分为以下两个类别：对文件的创建、拷贝、移动、删除操作和对文件内容的创建、添加、删除和读取操作。下面分别详细介绍。  
（一）创建文件  
一共有3种方法可用于创建一个空文本文件，这种文件有时候也叫做文本流（text stream）。  
第一种是使用CreateTextFile方法。代码如下：

复制代码 代码如下:

var fso, f1;  
fso = new ActiveXObject("Scripting.FileSystemObject");  
f1 = fso.CreateTextFile("c:\\testfile.txt", true);

第二种是使用OpenTextFile方法，并添加上ForWriting属性，ForWriting的值为2。代码如下：

复制代码 代码如下:

var fso, ts;  
var ForWriting= 2;  
fso = new ActiveXObject("Scripting.FileSystemObject");  
ts = fso.OpenTextFile("c:\\test.txt", ForWriting, true);

第三种是使用OpenAsTextStream方法，同样要设置好ForWriting属性。

复制代码 代码如下:

var fso, f1, ts;  
var ForWriting = 2;  
fso = new ActiveXObject("Scripting.FileSystemObject");  
fso.CreateTextFile ("c:\\test1.txt");  
f1 = fso.GetFile("c:\\test1.txt");  
ts = f1.OpenAsTextStream(ForWriting, true);

（二）添加数据到文件

当文件被创建后，一般要按照"打开文件－>填写数据－>关闭文件"的步骤实现添加数据到文件的目的。  
打开文件可使用FileSystemObject对象的OpenTextFile方法，或者使用File对象的OpenAsTextStream方法。  
填写数据要使用到TextStream对象的Write、WriteLine或者WriteBlankLines方法。在同是实现写入数据的功能下，这3者的区别在于：Write方法不在写入数据末尾添加新换行符，WriteLine方法要在最后添加一个新换行符，而WriteBlankLines则增加一个或者多个空行。  
关闭文件可使用TextStream对象的Close方法。

（三）创建文件及添加数据例程

下面的代码将创建文件、添加数据、关闭文件几个步骤结合起来进行应用：

复制代码 代码如下:

var fso, tf;  
fso = new ActiveXObject("Scripting.FileSystemObject");  
// 创建新文件  
tf = fso.CreateTextFile("c:\\testfile.txt", true);  
// 填写数据，并增加换行符  
tf.WriteLine("Testing 1, 2, 3.") ;  
// 增加3个空行  
tf.WriteBlankLines(3) ;  
// 填写一行，不带换行符  
tf.Write ("This is a test.");  
// 关闭文件  
tf.Close();

（四）读取文件内容

从文本文件中读取数据要使用TextStream对象的Read、ReadLine或ReadAll 方法。Read方法用于读取文件中指定数量的字符；ReadLine方法读取一整行，但不包括换行符；ReadAll方法则读取文本文件的整个内容。读取的内容存放于字符串变量中，用于显示、分析。在使用Read或ReadLine方法读取文件内容时，如果要跳过一些部分，就要用到Skip或SkipLine方法。  
下面的代码演示打开文件、填写数据，然后读取数据：

复制代码 代码如下:

var fso, f1, ts, s;  
var ForReading = 1;  
fso = new ActiveXObject("Scripting.FileSystemObject");  
// 创建文件  
f1 = fso.CreateTextFile("c:\\testfile.txt", true);  
// 填写一行数据  
f1.WriteLine("Hello World");  
f1.WriteBlankLines(1);  
// 关闭文件  
f1.Close();  
// 打开文件  
ts = fso.OpenTextFile("c:\\testfile.txt", ForReading);  
// 读取文件一行内容到字符串  
s = ts.ReadLine();  
// 显示字符串信息  
alert("File contents = ‘" + s + "‘");  
// 关闭文件  
ts.Close();

（五）移动、拷贝和删除文件

对于以上三种文件操作，javascript各有两种对应的方法：File.Move 或 FileSystemObject.MoveFile用于移动文件；File.Copy 或 FileSystemObject.CopyFile用于拷贝文件；File.Delete 或 FileSystemObject.DeleteFile用于删除文件。  
下面的代码演示在驱动器C的根目录下创建一个文本文件，填写一些内容，然后将文件移动到\tmp目录下，再在目录\temp下面建立一个文件拷贝，最后删除这两个目录的文件：

复制代码 代码如下:

var fso, f1, f2, s;  
fso = new ActiveXObject("Scripting.FileSystemObject");  
f1 = fso.CreateTextFile("c:\\testfile.txt", true);  
// 写一行  
f1.Write("This is a test.");  
// 关闭文件  
f1.Close();  
// 获取C:\根目录下的文件句柄  
f2 = fso.GetFile("c:\\testfile.txt");  
// 移动文件到\tmp目录下  
f2.Move ("c:\\tmp\\testfile.txt");  
// 拷贝文件到\temp目录下  
f2.Copy ("c:\\temp\\testfile.txt");  
// 获取文件句柄  
f2 = fso.GetFile("c:\\tmp\\testfile.txt");  
f3 = fso.GetFile("c:\\temp\\testfile.txt");  
// 删除文件  
f2.Delete();  
f3.Delete();

六、结 语

通过以上对FileSystemObject的各种对象、属性和方法的介绍和示例，相信你已经对如何使用javascript语言在页面中操作驱动器、文件和文件夹有了清晰的认识。但是上述提及的例程都非常简单，要全面、灵活地掌握javascript文件操作技术，还需要大量的实践练习。而且还有一点提醒大家，由于涉及到在浏览器中进行文件读写这样的高级操作，对于默认的浏览器安全级别而言，在代码运行前都会有一个信息提示，这点请在实际环境中提示访问者注意。