




文件处理与页面打印

管理科学与工程学科
耿方方



主要内容

- 文件对象处理
- 页面打印

- 在网站的开发过程中，经常需要对文件及文件夹进行操作，这些操作可以借用JavaScript中的文件处理对象实现。另外，用户还可以使用JavaScript实现常用的打印功能。

- 在JavaScript中实现文件操作功能主要依靠FileSystemObject对象。该对象用来创建、删除和获得有关信息，以及通常用来操作驱动器、文件夹和文件的方法和属性。该对象包含的对象和集合说明如表：

对象/集合	说明
FileSystemObject	主对象。包含用来创建、删除和获得有关信息，以及通常用来操作驱动器、文件夹和文件的方法和属性
Drive	对象。包含用来收集信息的方法和属性，这些信息是关于连接在系统上的驱动器的，如驱动器的共享名和它有多少可用空间。这里需要注意的是，Drive并非必须是硬盘，也可以是RAM磁盘等等。并非必须把驱动器实物地连接到系统上；它也可以通过网络在逻辑上被连接起来。
Drives	集合。提供驱动器的列表，这些驱动器实物地或在逻辑上与系统相连接。Drives集合包括所有驱动器，与类型无关。要可移动的媒体驱动器在该集合中显现，不必把媒体插入到驱动器中
File	对象。包含用来创建、删除或移动文件的方法和属性。也用来向系统询问文件名、路径和多种其他属性
Files	集合。提供包含在文件夹内的所有文件的列表。
Folder	对象。包含用来创建、删除或移动文件夹的方法和属性。也用来向系统询问文件夹名、路径和多种其他属性
Folders	集合。提供在Folder内的所有文件夹的列表
TextStream	对象。用来读写文本文件

- 1、动态创建FileSystemObject对象
- 要对文件进行相应的操作，必须对FileSystemObject对象进行实例化，也就是动态创建FileSystemObject对象
- 语法：
- `fso=new ActiveXObject (“Scripting.FileSystemObject”) ;`

- 2、FileSystemObject对象的方法
- (1) GetAbsolutePathName () 方法
- GetAbsolutePathName () 方法根据提供的路径返回明确完整的路径，也就是说如果路径提供了从指定驱动器的根开始地完整引用，那么它就是明确和完整的。如果路径指定的是映射驱动器的根文件夹，那么完整的路径将只能由一个路径分隔符“\”结束。
- 语法：
- object.GetAbsolutePathName (pathspec)
- object:必选项。FileSystemObject对象的名称；
- pathspec必选项。要变为明确完整路径的路径说明。该参数相应设置如表：

pathspec	说明
“c:”	返回当前的完整路径
“c:..”	返回当前路径的上一级路径
“c:\\”	返回当前路径根目录
“c:*..*\\myfile”	在当前路径后加上“*..*\myfile”
“myfile”	在当前路径后加上“myfile”
“c:\\..\\..\\myfile”	返回当前路径以myfile文件名结尾

注意：表中c指并不是c盘，而是服务器端当前路径的盘符。

- 2、FileSystemObject对象的方法
- (1) GetAbsolutePathName () 方法
- 假设当前的路径为d:\word\javascript, 下面对GetAbsolutePathName () 方法的应用进行说明:
- 例如, 获取当前路径的上一级目录, 代码如下:
 - `var fso=new ActiveXObject("Scripting.FileSystemObject");`
 - `var driv=fso.GetAbsolutePathName("d:..");`
 - 运行结果: d:\word。
 - `var fso=new ActiveXObject("Scripting.FileSystemObject");`
 - `Var driv=fso.GetAbsolutePathName("nn");`
 - 运行结果: d:\word\javascript\nn;

- 2、FileSystemObject对象的方法
- (2) GetBaseName () 方法
- 将以字符串的形式返回指定路径中最后成分中的基本名称，不包含文件扩展名。语法：
- object.GetBaseName (path)
- 例如：获取d:\word\javascript\mycolor.htm路径中的最后成分地文件名称mycolor，代码：
- ```
function ShowBaseName(filespec)
```
- ```
{var fso, s="" ;
```
- ```
 fso=new ActiveXObject(Scripting.FileSystemObject);
```
- ```
    s+=fso.GetBaseName(filespec);
```
- ```
 alert(s);}
```
- ```
ShowBaseName( "d:\word\javascript\mycolor.htm" );
```

- 2、FileSystemObject对象的方法
- (3) GetDriveName () 方法
- 该方法根据指定路径返回包含驱动器名称的字符串。语法：
- `object.GetDriveName(path);`
- path:路径说明, 将根据其中成分返回驱动器名称。
- (4) GetDrive () 方法
- 该方法用于返回指定路径中驱动器的Drive对象。语法：
- `object.GetDrive(drivespec);`

- 2、FileSystemObject对象的方法
- (4) GetDrive () 方法
- 例如：获取d:\word\javascript路径中的盘符d:，并以驱动器的D:形式进行显示。代码如下：
- ```
function GetDriveLetter(path)
```
- ```
{
```
- ```
 var fso, s="" ;
```
- ```
    fso=new ActiveXObject( "Scripting.FileSystemObject" );
```
- ```
 s+=fso.GetDrive(fso.GetDriveName(path));
```
- ```
    alert(s);
```
- ```
}
```
- ```
GetDriveLetter( "d:\word\javascript" );
```

- 2、FileSystemObject对象的方法
- (5) GetExtensionName () 方法
- 用于返回指定路径中的最后成分扩展名的字符串。
- path:必选项。返回其扩展名的指定路径。
- 例如：获取d:\word\javascript\nn.text目录中nn文件的扩展名text，代码如下：
- `<script>`
- `function ShowExtensionName(filespec)`
- `{`
- `Var fso, s= “” ;`
- `Fso=new ActiveXObject(“Scripting.FileSystemObject”);`
- `s+=fso.GetExtensionName(filespec);`
- `alert(s);}`
- `ShowExtensionName(“d:\\word\\javascript\\nn.text”);`
- `</script>`

- 2、FileSystemObject对象的方法
- (6) GetFileName () 方法
- 返回指定路径的最后成分，但指定的路径不能只是驱动器说明，也可以是共享路径。语法：
- `object.GetFileName(pathspec)`
- `pathspec`:必选项。指定文件的路径（绝对或相对路径）

- 2、FileSystemObject对象的方法
- (7) GetParentFolderName () 方法
- 根据指定路径中的最后成分返回其父文件夹名称的字符串。语法：
 - object.GetParentFolderName (path)
 - path: 必选项。文件名所在的完整路径。
- (8) GetSpecialFolder () 方法
- 该方法返回指定的特殊文件夹对象。语法：
 - object.GetSpecialFolder (folderspec)
 - folderspec: 必选项。返回特殊文件夹的名称，参数如表：

常数	值	说明
WindowFolder	0	Window文件夹，包含了由Windows操作系统安装的文件
SystemFolder	1	包含库、字体以及设备驱动程序的System文件夹
TemporaryFolder	2	用于存储临时文件的Temp文件夹。

- 2、FileSystemObject对象的方法
- (9) GetTempName () 方法
- 该方法返回一个随机产生的临时文件或文件夹名，有助于执行那些需要临时文件或文件夹的操作。语法：
- `object.GetTempName () ;`
- 说明：
- GetTempName () 方法并不创建文件。它只提供一个临时文件名，可以通过 CreateTextFile来创建文件。

- 案例1-1:
- `<script language="JavaScript">`
- `var strFile = "d:\\test.txt";`
- `var objFSO = new ActiveXObject("Scripting.FileSystemObject");`
- `// 检查文件是否存在`
- `if (!objFSO.FileExists(strFile)) {`
- `// 创建文本文件`
- `var objStream = objFSO.CreateTextFile(strFile, true);`
- `document.write("创建文本文件: " + strFile + "
");`
- `objStream.Close(); // 关闭文件`
- `}`
- `else`
- `document.write("文本文件: " + strFile + "已经存在
");`
- `</script>`

- Drive对象负责收集系统中的物理或逻辑驱动器资源内容，如驱动器的共享名和有多少可用空间。在使用该对象时，不一定非要把驱动器实物地连接到系统上，也可以通过网络在逻辑上连接起来。需要说明的是，在这里所说的驱动器不一定非是硬盘，也可以是RAM磁盘等。
- 1、动态创建Drive对象
- 为使用Drive对象来获取驱动器的相关信息，必须首先创建Drive对象。该对象是通过FileSystemObject对象的GetDrive（）方法来创建的。
- 例如：对C盘驱动器创建一个Drive对象，代码如下：
- ```
var fso=new ActiveXObject("Scripting.FileSystemObject");
```
- ```
var s=fso.GetDrive("C:\");
```

- 1、Drive对象的属性
- (1) FreeSpace属性
- 向用户返回指定驱动器或网络共享上的可用空间的大小，只读。
- `object.FreeSpace`
- (2) IsReady属性
- 如果指定驱动器已就绪，则返回True，否则返回False
- 注意：针对可移动媒体的驱动器和CD-ROM驱动器来说。
- (3) TotalSize属性
- 以字节为单位返回驱动器或网络共享的所有空间大小。

- 案例1:
- `<script language="javascript">`
- `<!--`
- `function DriveSize(DriveName) {`
- `var fso=new ActiveXObject("Scripting.FileSystemObject");`
- `var s=fso.GetDrive(DriveName.value);`
- `if (s.IsReady) {`
- `var str, str1, AllSize=0.0;`
- `str="当前驱动器的名称为:"+s.DriveLetter+"\n";`
- `AllSize=s.TotalSize/1024/1024/1024;`
- `str=str+"当前驱动器的大小为:"+parseInt(AllSize*10)/10+"\n";`
- `AllSize=s.FreeSpace/1024/1024/1024;`
- `str=str+"当前驱动器的可用空间为:"+parseInt(AllSize*10)/10;`
- `alert(str);`
- `}else`
- `alert("该驱动器无效。")`
- `}`
- `//-->`
- `</script>`

- 1、Drive对象的属性
- (4) DriveType属性
- 返回一个值，表示所指定驱动器的类型。
- `object.DriveType`
- (5) SerialNumber属性
- 返回连续的十进制数字，用于唯一标识磁盘卷。
- 注意：针对可移动媒体的驱动器和CD-ROM驱动器来说。

- 案例2:
- `<script language="javascript">`
- `function dtype(Drivename) {`
- `var fso=new ActiveXObject("Scripting.FileSystemObject");`
- `var s=fso.GetDrive(Drivename.value);`
- `var t="",n="";`
- `switch(s.DriveType) {`
- `case 0: t="找不到该驱动器";break;`
- `case 1: t="移动硬盘";break;`
- `case 2: t="固定硬盘";break;`
- `case 3: t="网络资源";break;`
- `case 4: t="CD-ROM";break;`
- `case 5: t="RAM";break;`
- `}`
- `if (s.IsReady)`
- `n="系列号为:"+s.SerialNumber;`
- `alert(t+"\n"+n);`
- `}`
- `</script>`

- 1、Drive对象的属性
- (6) AvailableSpace属性
- 返回在所指定的驱动器或网络共享上可用的空间大小。
- `object.AvailableSpace`
- (7) FileSystem属性
- 返回指定驱动器所使用的文件系统的类型，可能的返回类型包括FAT、NTFS和CDFS。
- (8) Path属性
- 返回指定文件、文件夹或驱动器的路径。

- 1、Drive对象的属性
- (9) RootFolder属性
- 返回一个Folder对象，表示指定驱动器的根文件夹，只读。
- object.AvailableSpace
- (10) ShareName属性
- 返回指定驱动器的网络共享名。如果object不是网络驱动器，那么ShareName属性返回长度为0的字符串（“”）。
- (11) VolumeName属性
- 设置或返回指定驱动器的卷名，读/写。
- object.VolumeName [=newname]
- newname:可选项。如果提供了这个部分，那么newname就将成为指定的object的新名称。

- File对象可以获取服务器端指定文件的相关属性，如文件的创建、修改、访问时间。也可以对文件或文件夹进行复制、移除或删除的操作。
- 1、动态创建File对象
- 为使用File对象对指定文件的所有属性进行访问，必须首先创建File对象，该对象是通过FileSystemObject对象的GetFile（）方法创建的。
- GetFile（）方法根据指定路径中的文件返回相应的File对象。语法：
- `object.GetFile(filespec)`
- `filespec`:必选项。指定文件的路径（绝对或相对路径）
- 例如，将qq.txt文件以File对象进行实例化，代码如下：
- `var fso=new ActiveXObject(“Scripting.FileSystemObject”);`
- `var s=fso.GetFile(“E:\\word\\JavaScript\\qq.txt”);`

- 2、File对象的方法
- (1) Copy () 方法
- Copy () 方法对由单个 File 或 Folder 所产生的结果和使用 FileSystemObject.CopyFile或FileSystemObject.CopyFolder所执行的操作结果一样。其中，后者把由object所引用的文件或文件夹作为参数传递。但是，后两种替换方法能够复制多个文件或文件夹。
- 将指定文件或文件夹从一个位置复制到另一个位置。
- object.Copy(destination[, overwrite]);
- destination: 必选项。复制文件或文件夹的目的位置。不允许通配字符。
- overwrite:可选项。 Boolean值，如果要覆盖已有文件或文件夹，则为True（默认），否则为false。

- 2、File对象的方法
- (2) Delete () 方法
- 删除指定的文件或文件夹。
- `object.Delete(force);`
- `force`:可选项。Boolean值, 如果要删除设置了只读属性的文件或文件夹, 则为True, 否则为false。
- (3) Move () 方法
- 将指定的文件或文件夹从一个位置移动到另一个位置。
- 语法: `object.Move(destination);`
- `destination`: 必选项。移动文件或文件夹的目的位置, 不允许通配字符。

- 案例3: `<script language="javascript">`
- `function filemove(fname,mname){`
- `<!--`
- `function filecopy(sname,dname) {`
- `var fso, f;`
- `fso = new ActiveXObject("Scripting.FileSystemObject");`
- `f = fso.GetFile(sname.value);`
- `f.Copy(dname.value);`
- `alert("文件复制成功");`
- `}`
- `function filedelete(fname) {`
- `var fso, f;`
- `fso = new ActiveXObject("Scripting.FileSystemObject");`
- `f = fso.GetFile(fname);`
- `f.Delete();`
- `alert("文件删除成功");`
- `}`
- `var fso, f;`
- `fso = new ActiveXObject("Scripting.FileSystemObject");`
- `f = fso.GetFile(fname);`
- `f.Move(mname);`
- `alert("文件移动成功");`
- `}`
- `//-->`
- `</script>`

- 2、File对象的方法
- (4) `OpenAsTextStream ()` 方法
- 打开指定的文件并返回一个TextStream对象，可以通过这个对象对文件进行读、写或追加。语法：
- `object.OpenAsTextStream ([iomode, [format]])`
- `iomode`: 可选项。指明输入/输出的模式，可以是3个常数之一：`ForReading`、`ForWriting`或`ForAppending`。

常数	值	描述
<code>ForReading</code>	1	以只读方式打开文件。不能写这个文件
<code>ForWriting</code>	2	以写方式打开文件。如果存在同名文件，那么它以前的内容将被覆盖。
<code>ForAppending</code>	8	打开文件并从文件末尾开始写

- 2、File对象的方法
- (4) `OpenAsTextStream ()` 方法
- 打开指定的文件并返回一个TextStream对象，可以通过这个对象对文件进行读、写或追加。语法：
- `object.OpenAsTextStream ([iomode, [format]])`
- `format`:可选项。使用三态值中的一个来指明打开文件的格式。如果忽略，文件将以ASCII格式打开。

常数	值	描述
<code>TristateUseDefault</code>	-2	使用系统默认值打开文件
<code>TristateTrue</code>	-1	以Unicode方式打开文件
<code>TristateFalse</code>	0	以ASCII方式打开文件

文件处理对象

File对象

- 案例4:

- ```
function TextStreamTest (fname, Addname, n)
{
 var fso, f, ts, s;
 var ForRWA=0, ForReading=1, ForWriting=2, ForAppending=8;
 var TristateUseDefault=-2, TristateTrue=-1, TristateFalse=0;
 fso = new ActiveXObject("Scripting.FileSystemObject");
 var s1=Addname.innerHTML;
 if (fname.value!="")
 {f = fso.GetFile(fname.value);
 switch(n) {
 case 1: ForRWA=ForWriting;break;
 case 2: ForRWA=ForAppending;break;}
 if (n>0)
 {
 ts = f.OpenAsTextStream(ForRWA, TristateUseDefault);
 var s1=Addname.innerHTML;
 ts.Write(s1);
 ts.Close();
 }
 }
```

```
ts = f.OpenAsTextStream(ForReading, TristateUseDefault);
s = ts.ReadLine();
ts.Close();
}
return(s);
}
function run(n)
{
 document.form3.textarea1.innerHTML=
 TextStreamTest(document.form5.text1,
 document.form4.textarea2,n)
}
```

- 3、File对象的属性
- (1) Attributes属性
- 设置或返回文件或文件夹的属性。根据不同属性为读/写或只读。语法：  
object.Attributes [= newattributes]
- newattributes: 可选项。如果提供了这个部分，那么newattributes将成为指定的object的新属性值。newattributes可以是下表中任意的逻辑组合。

- 3、File对象的属性
- (1) Attributes属性

| 常数         | 值   | 描述                  |
|------------|-----|---------------------|
| Normal     | 0   | 普通文件。不设置属性          |
| ReadOnly   | 1   | 只读文件。属性为读/写         |
| Hidden     | 2   | 隐藏文件。属性为读/写         |
| System     | 4   | 系统文件。属性为读/写         |
| Volume     | 8   | 磁盘驱动器卷标。属性为只读       |
| Directory  | 16  | 文件夹或目录。属性为只读        |
| Archive    | 32  | 文件在上次备份后已经修改。属性为读/写 |
| Alias      | 64  | 链接或者快捷方式。属性为只读      |
| Compressed | 128 | 压缩文件。属性为只读          |



### ■ 案例5:

```
function ShowFileInfo(filespec) {
 var fso, f, s;

 fso = new ActiveXObject("Scripting.FileSystemObject");
 f = fso.GetFile(filespec);
 switch(f.Attributes) {
 case 0: s="普通文件";break;
 case 1: s="只读文件";break;
 case 2: s="隐藏文件";break;
 case 4: s="系统文件";break;
 case 32: s="文件在上次备份后已经修改";break;
 case 33: s="只读文件(已修改)";break;
 case 34: s="隐藏文件(已修改)";break;
 case 128: s="压缩文件";break;}

 if (f.Attributes==1 || f.Attributes==33) {
 if (confirm("当前文件为"+s+"\n是否将其改为可写文件")) {
 f.Attributes=f.Attributes-1;}
 }else
 alert("当前文件为:"+s);
}
```

### ■ 3、File对象的属性

#### ■ (2) DateCreated属性

■ 返回指定文件或文件夹的创建日期和时间，只读。语法：

■ `object.DateCreated`

#### ■ (3) DateLastAccessed属性

■ 返回最后修改指定文件或文件夹的日期和时间，只读。语法：

■ `object.DateLastAccessed`

#### ■ (4) DateLastModified属性

■ 返回最后修改指定文件或文件夹的日期和时间，只读。语法：

■ `object.DateLastModified`

- 案例6:
- ```
function ShowFileData(filespec) {
```
- ```
 var fso, f, s;
```
- ```
    fso = new ActiveXObject("Scripting.FileSystemObject");
```
- ```
 f = fso.GetFile(filespec);
```
- ```
    var d=f.DateCreated;
```
- ```
 Cdate = new Date(d) ;
```
- ```
    d=f.DateLastModified;
```
- ```
 Mdate = new Date(d) ;
```
- ```
    d=f.DateLastAccessed;
```
- ```
 Adate = new Date(d) ;
```
- ```
    s="当前文件的创建时间为:"+Cdate.toLocaleString()+"\n当前文件的修改时间为:"+Mdate.toLocaleString()+"\n当前文件的访问时间为:"+Adate.toLocaleString();
```
- ```
 alert(s);
```
- ```
}
```

- 3、File对象的属性
- (5) Name属性
- 设置或返回指定文件或文件夹的名称，读/写。语法：
 - `object.Name [= newname]`
 - `newname`:可选项。如果提供了这个部分，`newname`将成为指定的`object`的新名称。
- (6) Size属性
- 对于文件，以字节为单位返回指定文件的大小。对于文件夹、以字节为单位返回文件夹中包含的所有文件和子文件夹的大小。语法：
 - `object.Size`
- (7) Type属性
- 返回关于文件或文件夹类型的信息。例如，对于以`.txt`结尾的文件将返回“文本文档”。语法：
 - `object.Type`

- 案例7:
- `<script language="javascript">`
- `<!--`
- `function ShowFileData(filespec) {`
- `var fso, f, s;`
- `fso = new ActiveXObject("Scripting.FileSystemObject");`
- `f = fso.GetFile(filespec);`
- `s=f.type+"类型的"+f.name+"文件的大小为: "+(f.size)+"b";`
- `alert(s);`
- `}`
- `//-->`
- `</script>`

- 3、File对象的属性
- (8) ShortName属性
- 返回短名称。语法：
object.ShortName
- (9) Drive属性
- 返回指定文件或文件夹所在驱动器的驱动器号，只读。语法：
object.Drive
- (10) ParentFolder属性
- 返回指定文件或文件夹的父文件夹对象，只读。语法：
object.ParentFolder
- (11) Path属性
- 返回指定文件、文件夹或驱动器的路径。语法：
object.Path

- Folder对象可以获取服务器端指定文件夹的相关属性，它与File对象的实现过程基本相同。只是Folder对象针对的是文件夹，File对象针对的是文件。
- 1、动态创建Folder对象
- 使用Folder对象对指定文件夹的所有属性进行访问，必须创建Folder对象，该对象是通过FileSystemObject对象的GetFolder（）方法创建的。
- GetFolder（）方法根据指定路径中的文件返回相应的Folder对象。

语法：

```
object.GetFolder(filespec)
```

filespec:必选项。指定文件夹的路径（绝对或相对路径）。

- 2、Folder对象的方法与属性
- Folder对象的属性和方法与File对象中的属性和方法基本相同，只是其功能针对的不是文件而是文件夹。在Folder对象中有两个属性是File对象所没有的：
 - (1) Files属性
 - 返回一个Files集合，由指定文件夹中包含的所有File对象组成，包括设置了隐藏和系统文件属性的文件。语法：
 - `object.Files`
 - (2) IsRootFolder属性
 - 如果指定的文件夹是根文件夹，则返回true，否则返回false。语法：
 - `object.IsRootFolder`

页面打印

使用execWB方法进行打印

- 对页面进行打印，主要是通过WebBrowser组件的execWB()方法来实现的，可以通过该方法来实现IE浏览器中菜单的相应功能。WebBrowser组件是IE内置的浏览器控件，无须用户下载。它的优点是客户端独立完成打印目标文档，减轻服务器负荷；缺点是源文档的分析操作复杂，并且要对源文档中要打印的内容进行约束。
- 在使用WebBrowser组件时，首先要在<body>标记的下面用<object>...</object>标记声明WebBrowser组件。

```
<object id="WebBrowser" classid="C1SID:8856F961-340A-11D0-A96B-00C04Fd705A2" width="0" height="0">
```
- </object>
- 下面给出execWB () 方法的语法：

```
WebBrowser.ExecWB (nCmdID, nCmdExecOpt [, pvaIn] [, pvaOut])
```
- WebBrowser:必选项。WebBrowser控件的名称。
- nCmdID: 必选项。执行操作功能的命令。
- nCmdExecOpt:必选项。执行相应的选项，通常值为1。

页面打印

使用execWB方法进行打印

- 下面给出在IE浏览器中WebBrowser组件的execWB () 方法的一些功能：

WebBrowser.ExecWB(1,1)	打开
WebBrowser.ExecWB(2,1)	关闭现在所有的IE窗口，并打开一个新窗口
WebBrowser.ExecWB(4,1)	保存网页
WebBrowser.ExecWB(6,1)	打印
WebBrowser.ExecWB(7,1)	打印预览
WebBrowser.ExecWB(8,1)	打印页面设置
WebBrowser.ExecWB(10,1)	查看页面属性
WebBrowser.ExecWB(15,1)	撤销
WebBrowser.ExecWB(17,1)	全选
WebBrowser.ExecWB(22,1)	刷新
WebBrowser.ExecWB(45,1)	关闭窗口体无提示

■ 案例8

- `<script language="javascript">`
- `<!--`
- `function webprint(n)`
- `{`
- `switch(n)`
- `{`
- `case 0:document.all.WebBrowser.Execwb(7,1);break;`
- `case 1:document.all.WebBrowser.Execwb(6,1);break;`
- `case 2:document.all.WebBrowser.Execwb(6,6);break;`
- `}`
- `}`
- `//-->`
- `</script>`

- 在打印页面时，有时只需要打印网页中的部分内容，可以将要打印的内容以框架的形式进行显示，然后用window对象的print()方法打印框架。
- 在打印页面中的框架时，首先需要将要打印的框架获得焦点，可以用内置变量parent来实现。
- 内置变量parent指的是包含当前分割窗口的父窗口。也就是在一窗口内如果有分割窗口，而在其中一个分割窗口中又包含着分割窗口，则第2层的分割窗口可以用parent变量引用包含它的父分割窗口。
- 语法：
 - `parent.mainFrame.fcous()`;
 - 参数mainFrame表示框架的名称。

- 案例9:
- `<tr>`
- `<td height="264" rowspan="2"> </td>`
- `<td width="666" height="25" class="word_orange">当前位置: 系统查询 > 借阅信息打印
>>> </td>`
- `<td width="58" align="center" class="word_Green"><a href="#"
onClick="parent.contentFrame.focus();window.print();">打印</td>`
- `<td rowspan="2"> </td>`
- `</tr>`
- `<tr>`
- `<td height="240" colspan="2" align="center" valign="top" bgcolor="#FFFFFF"><iframe
name="contentFrame" src="19-content.htm" frameborder="0" width="100%"
height="100%"></iframe></td>`
- `</tr>`
- `<tr>`
- `<td> </td>`
- `<td colspan="2"> </td>`
- `<td> </td>`
- `</tr>`

- 设置页眉页脚主要通过WshShell对象的相关方法实现的。WshShell对象是WSH（WSH是Windows Scripting Host的缩写，内嵌于Windows操作系统中的脚本语言工作环境）的内建对象，主要负责程序的本地运行、处理注册表、创建快捷方式、获取系统文件夹信息及处理环境变量等工作。WshShell对象的相关方法如下表所示。

方 法	说 明
CreateShortcut()	创建并返回WshShortcut对象
ExpandEnvironmentStrings()	扩展PROCESS环境变量并返回结果字符串
Popup()	显示包含指定消息的消息窗口
RegDelete()	从注册表中删除指定的键或值
RegRead()	从注册表中返回指定的键或值
RegWrite()	在注册表中设置指定的键或值
Run()	创建新的进程，该进程用指定的窗口样式执行指定的命令

- 设置页眉页脚主要应用WshShell对象的RegWrite () 方法。RegWrite () 方法用于注册表中设置指定键或值。
- 语法：

```
WshShell.RegWrite(strName, anyValue[, strType])
```
- strName: 用于指定注册表的键或值，若strName以一个反斜杠结束，则该方法设置键，否则设置值。strName参数必须以根键名HKEY_CURRENT_USER、HKEY_LOCAL_MACHINE 、 HKEY_CLASSES_ROOT 、 HKEY_USERS 或 以HKEY_CURRENT_CONFIG开头。
- anyValue: 用于指定注册表的键或值的值。当strType为REG_SZ或REG_EXPAND_SZ时，RegWrite () 方法自动将anyValue转换为字符串。若strType为REG_DWORD, 则anyValue被转换为整数。若strType为REG_BINARY, 则anyValue必须是一个整数。
- strType: 用于指定注册表的键或值地数据类型。RegWrite () 方法支持的数据类型为REG_SZ、REG_EXPAND_SZ、REG_DWORD和REG_BINARY。若其他的数据类型作为strType传递，RegWrite返回E_INVALIDARG。

- 案例10: var HKEY_RootPath="HKEY_CURRENT_USER\\Software\\Microsoft\\Internet Explorer\\PageSetup\\";
- function PageSetup_del() {
- try{
- var WSc=new ActiveXObject("WScript.Shell");
- HKEY_Key="header";
- WSc.RegWrite(HKEY_RootPath+HKEY_Key, "");
- HKEY_Key="footer";
- WSc.RegWrite(HKEY_RootPath+HKEY_Key, "");
- } catch(e) {} }
- function PageSetup_set() {
- try{
- var WSc=new ActiveXObject("WScript.Shell");
- HKEY_Key="header";
- WSc.RegWrite(HKEY_RootPath+HKEY_Key, "&w&b页码, &p/&P");
- HKEY_Key="footer";
- WSc.RegWrite(HKEY_RootPath+HKEY_Key, "&u&b&d");
- } catch(e) {} }
- //-->

页面打印

分页打印

- 在页面打印时，可以利用CSS样式中的page-break-before（在对象前分页）或page-break-after（在对象后分页）属性进行分页打印，并利用<thead>和<tfoot>标记在打印的每一个页面中都显示表头和表尾。
- (1) thead标记
- thead用于设置表格的表头。
- (2) tfoot标记
- tfoot用于设置表格的表尾。
- (3) page-break-after属性
- 该属性在打印文档时发生作用，用于进行分页打印，但是对于
和<hr>标记不起作用。
- 语法：
- page-break-after: auto | always | avoid | left | right | null

■ page-break-after属性的参数说明

参 数	描 述
after	设置对象后出现页分割符。设置为always时，始终在对象之后插入页分割符
auto	在对象之后自动插入页分割符（当对象前没有多余空间时插入分割符）
always	始终在对象之后插入页分割符
avoid	未支持。避免在对象后面插入分割符
left	未支持。在对象后面插入页分割符，直到它到达一个空白的左页边
right	未支持。在对象后面插入页分割符，直到它到达一个空白的右页边
null	空白字符串。取消了分割符设置

想一想

- 1、表格导出到word并进行打印