Объект `FileSystemObject`

http://farinadesign.narod.ru/BaseOfWeb/Lectures/DOM3.html

Contents

Производить операции как над объектами файловой системы в целом (файлы, папки, диски), так и над содержимым отдельно взятого файла можно с помощью встроенного в MSIE компонента ActiveX, под названием **`FileSystemObject`**.

Однако следует иметь в виду, что для работоспособности данного компонента необходимо соответствующим образом настроить режим безопасности MSIE. А именно, отредактировать секцию Элементы ActiveX и модули подключения, для доступа к которой нужно выбрать команду **`Сервис|Свойства`** обозревателя и далее в появившемся диалоговом окне выбрать вкладку **`Безопасность`**. Следует быть очень осторожным при редактировании настроек этой секции, т. к. разрешение доступа к файловой системе клиентской машины влечет за собой значительное увеличение вероятности успеха ее взлома.

Итак, перед тем, как приступить к операциям над файловой системой клиента, нам нужно создать объект типа **`FileSystemObject`**. Делается это с помощью следующей конструкции:

```
var fso = new ActiveXObject("Scripting.FileSystemObject")
```

после чего можно приступать к использованию методов и свойств этого объекта. Заметим, что инициализация объекта **`FileSystemObject`** происходит только один раз, вне зависимости от количества попыток разработчика создать объект этого типа.

Объектную модель **`FileSystemObject`** образуют следующие объекты и коллекции:

- Непосредственно сам **`FileSystemObject`** как основной объект.
- Объект **`Drive`**, служащий для получения различной информации о дисках.
- Коллекция объектов **`Drives`**, являющаяся представлением всех определенных в системе дисков.
- Объект **`File`**, реализующий операции над файлами такие, как создание, удаление и перемещение файла, а также служащий для обработки различных характеристик файла (путь, имя, размер и т. п.).
- Коллекция **`Files`**, служащая для представления списка файлов, содержащихся в файле.
- Объект **`Folder`**, содержащий методы и свойства для операций над папками, а также для получения их характеристик.
- Коллекция **`Folders`**, служащая для представления списка папок, содержащихся в другой папке.
- Объект **`Textstream`**, реализующий операции чтения и записи над текстовыми файлами.

Операции над дисками

Для получения информации о диске необходимо воспользоваться методом **`getDrive()`** объекта **`FileSystemObject`**, который, принимая в качестве аргумента путь к диску, возвращает объект типа **`Drive`**. В табл. 5.9 содержится описание свойств этого объекта, каждое из которых соответствует какой-либо характеристике диска.

Таблица 5.9. Свойства объекта FileSystemObject	
Свойство	**Характеристика диска**
TotalSize	Общий размер в байтах
AvailableSpace	Размер свободного места в байтах
DriveLetter	Буква диска
DriveType	Тип диска: 1 — переносной (**removable**), 2 — непереносной (**fixed**), 3 — сетевой (**network**), 4 — **CDROM**, 5 — **RAM**-диск
SerialNumber	Серийный номер
FileSystem	Тип файловой системы (FAT, FAT32, NTFS)
isReady	Доступен ли для использования
VolumeName	Метка тома
ShareName	Сетевое имя
Path	Путь
RootFolder	Корневая папка

Для проверки, существует ли диск с заданным именем, предназначен метод **DriveExists()** объекта **FileSystemObject**, принимающий в качестве аргумента имя диска и возвращающий **true**, если такой диск существует и **false**, если — нет.

Операции над папками

Для получения объекта, представляющего какую-либо папку, нужно воспользоваться методом **getFolder()** объекта **FileSystemObject**, по аналогии с **getDrive()**, описанным выше. Для манипулирования папками и их характеристиками можно воспользоваться либо тем значением (объект типа **Folder**), которое будет возвращено в результате вызова **getFolder()**, либо специальными методами объекта **FileSystemObject**. В табл. 5.10 содержится описание тех возможностей, которые предоставляют **Folder** и **FileSystemObject** для управления папками и их характеристиками.

Таблица 5.10. Методы, реализующие операции над папками	
Операция над папкой	**Метод**
Создание	**FileSystemObject.CreateFolder**
Удаление	**Folder.Delete** или **FileSystemObject.DeleteFolder**
Перемещение	**Folder.Move** или **FileSystemObject.MoveFolder**
Копирование	**FoIder.Copy** или **FileSystemObject.CopyFolder**
Получение имени	**FoIder.Name**
Выяснение существования	**FileSystemObject.FolderExists**
Получение имени родительской папки	**FileSystemObject.GetParentFolderName**
Получение Folder-представления системной папки	**FileSystemObject.GetSpecialFolder**

Здесь аргументом каждого метода из перечисленных в правой колонке таблицы будет значение пути, соответствующее той операции, которую выполняет данный метод. Под системной понимается одна из трех папок:

1. Папка Windows. Соответствующим ей возвращаемым значением является о.
2. Системная папка (то есть та, в которой хранятся библиотеки, шрифты и драйверы устройств). Соответствующим ей возвращаемым значением является 1.
3. Временная папка (то есть предназначенная для хранения временных файлов). Соответствующим ей возвращаемым значением является 2.

Далее приводится табл. 5.11, содержащая описание свойств объекта **Folder** за исключением тех из них, которые определены также и для объекта **File**.

Таблица 5.11. Некоторые свойства объекта Folder	
Свойство	**Значение**
Files	Коллекция файлов (объект **Files**), содержащихся в данной папке
isRootFolder	Является ли данная папка корневой (**true**, если да, и **false**, если нет)
SubFolders	Коллекция папок (объект **Folders**), вложенных в данную

Операции над файлами

Как уже говорилось ранее, для представления файлов служит объект **File**, который можно получить с помощью вызова метода **getFile()**объекта **FileSystemObject**. Далее мы опишем по отдельности свойства и методы этого объекта.

Существуют три способа создания текстового файла. Каждый из этих способов подразумевает использование отдельного метода какого-либо из объектов **FileSystemObject** или **File**. Рассмотрим их по порядку.

- **FileSystemObject.CreateTextFile(name,overwrite,Unicode)** — создает текстовый файл с именем **name**. Аргумент **overwrite** при этом указывает, должен ли быть перезаписан файл в том случае, если файл с таким именем уже существует (**true** — файл может быть перезаписан, **false** — нет). Необязательный аргумент **Unicode** указывает, символы какого набора используются для содержимого файла (**true** — Unicode, **false** — ASCII).
- **FileSystemObject.OpenTextFile(name,iomode,create,format)** — открывает или создает файл с именем name. Необязательный аргумент **iomode** может принимать одно из трех значений:
    - **1** — файл предназначен только для чтения,
    - **2** — файл предназначен для записи,
    - **8** — файл предназначен для добавления данных в его конец.

  Необязательный аргумент **create** указывает, должен ли быть создан новый файл, если файл с именем **name** отсутствует (**true** — файл будет создан, **false** — нет). И последний, также необязательный, аргумент **format** служит для задания формата содержимого файла. Он может принимать одно из трех специальных значений:

    - **TristateTrue** — содержимым файла является текст Unicode,
    - **TristateFalse** — ASCII,
    - **TristateUseDefault** — используется набор символов, по умолчанию заданный в системе.

- **File.OpenAsTextStream(iomode,format)** — открывает файл, представленный объектом **File**. Аргументы этого метода полностью эквивалентны аргументам метода **FileSystemObject.OpenTextFile()**.

Последний метод является одним из четырех методов, определенных для объекта **File**. Остальные три метода предназначены для проведения операций перемещения, копирования и удаления файлов. В табл. 5.12 представлены эти методы, а также их аналоги, определенные для объекта **FileSystemObject**.

Таблица 5.12. Базовые методы объекта File	
Операция	**Метод**
Перемещение	**File.Move** или **FileSystemObject.MoveFile**
Копирование	**File.Copy** или **FileSystemObject.CopyFile**
Удаление	**File.Delete** или **FileSystemObject.DeleteFile**

Методы **Move** и **Copy** обладают двумя аргументами: **destination** (обязательный) — путь, куда будет перемещен или скопирован файл, и **overwrite** (необязательный) — следует ли переписать файл, если он уже присутствует в приемнике копирования, заданном первым аргументом. Методы **MoveFile** и **CopyFile**, помимо **destination** и **overwrite**, обладают еще одним обязательным аргументом **source**, значением которого является путь и имя перемещаемого или копируемого файла.

Метод **Delete** обладает одним необязательным аргументом **force**, значением которого может быть **true** (удалить файл, даже если он обладает атрибутом "только для чтения") и **false** (не удалять файл, если он имеет атрибут "только для чтения"). Метод **DeleteFile**, помимо **force**, обладает еще одним, обязательным аргументом **source**, значением которого является путь и имя удаляемого файла.

Для проверки, существует ли файл с заданным именем, может быть применен метод **FileExists()** объекта **FileSystemObject**, принимающий в качестве аргумента путь и имя файла и возвращающий true, если такой файл существует, и **false**, если — нет.

Для чтения данных из текстового файла у объекта **File** определены следующие три метода, представленные в табл. 5.13.

Таблица 5.13. Методы объекта File, реализующие операцию чтения

Метод	Описание
Read(chars)	Читает из файла **chars** байт
ReadLine()	Читает из файла одну строку
ReadAll()	Читает все содержимое файла

Если при чтении из файла необходимо пропустить некоторое количество данных, то для этих целей можно воспользоваться одним из двух методов: **Skip(chars)** — пропускает **chars** символов, или **skipLine()**, который пропускает одну строку.

Следующие три метода объекта **File**, приведенные в табл. 5.14, предназначены для записи данных в файл.

Таблица 5.14. Методы объекта File, реализующие операцию записи

Метод	Описание
Write(text)	Записывает в файл **text**
WriteLine(line)	Записывает в файл строку **line**
WriteBlankLines(lines)	Записывает в файл **lines** пустых строк

После того, как работа над содержимым файлом закончена, файл нужно закрыть. Для этих целей служит метод **close()** объекта **File**, не имеющий аргументов.

При чтении или записи данных из файла с помощью **FileSystemObject** корректно обрабатываются только те данные, которые представлены в кодировке ср1251. Поэтому попытка загрузить в какую-либо область Web-страницы неанглоязычные данные в кодировке, отличной от указанной, приведет к тому, что эти данные будут отображаться так, как если бы для этой Web-страницы была задана кодировка ср1251, вне зависимости от ее действительной кодировки, и, следовательно, вместо данных мы увидим бессмысленный набор символов. Тем не менее, эти данные можно отправить на сервер и, указав соответствующую им кодировку в качестве аргумента конструктора объекта, читающего информацию из входного потока (например, **inputstreamReader**), корректно эти данные обработать.

Теперь рассмотрим табл. 5.15, содержащую свойства объекта **File**, а также те характеристики файла, которым они соответствуют.

Таблица 5.15. Свойства объекта File	
Свойство	**Значение**
Attributes	Атрибуты файла
DateCreated	Дата создания
DateLastAccessed	Дата последнего обращения
DateLastModified	Дата последней модификации
Drive	Буква диска
Name	Полное имя
ParentFolder	Folder-представление родительской папки
Path	Полный путь
ShortName	Имя в формате 8.3
ShortPath	Путь в формате 8.3
Size	Размер в байтах
Type	Тип файла из списка зарегистрированных в Windows типов

Всеми вышеперечисленными свойствами обладает и объект **Folder**.

Свойство **Attributes** может принимать следующие значения:

- 0 — атрибуты файла или папки не установлены;
- 1 — файл или папка предназначены только для чтения;
- 2 — скрытый;
- 4 — системный;
- 8 — объект является меткой тома;
- 16 — объект является папкой;
- 32 — архивный;
- 64 — объект является мягкой или жесткой ссылкой;
- 128 — сжатый файл.

Значения 8, 16, 64 и 128 являются значениями только для чтения, и соответствующие им атрибуты, будучи заданными, снять невозможно.

Манипулирование путями и именами файлов

Следующие методы объекта **FileSystemObject** образуют инструментарий, соответствующий заголовку подраздела.

- **BuildPath(path,name)** — формирует полный путь с помощью конкатенации своих аргументов. Первый аргумент (**path**) задает путь к файлу, второй (**name**) — его имя. Оба аргумента обязательны. При необходимости добавляется разделитель \.
- **GetAbsolutePathName(path)** — возвращает полный путь к объекту, заданному обязательным аргументом **path**. При этом **path** рассматривается как путь, указанный относительно текущей папки. Формирование пути эквивалентно формированию абсолютного URL на основе базового и относительного Internet-адресов.
- **GetBaseName(path)** — возвращает имя, представленное последним компонентом пути, заданного обязательным аргументом **path**. Если объект обладает расширением, то в возвращаемом значении оно отсутствует.
- **GetExtensionName(path)** — возвращает расширение (расширенное имя) последнего компонента пути, заданного обязательным аргументом **path**.

Коллекции дисков, папок и файлов

Объектная модель **FileSystemObject** определяет три типа коллекций:

- **Drives** — свойство **FileSystemObject**, значением которого является список всех дисков, имеющихся на той машине, где был инициализирован этот объект. Каждый диск коллекции представлен объектом **Drive**.
- **Folders** — свойство объекта **Folder**, значением которого является список всех папок, содержащихся в представленной данным **Folder**. Каждая папка коллекции представлена объектом **Folder**.
- **Files** — свойство объекта **Folder**, значением которого является список всех файлов, содержащихся в папке, представленной данным **Folder**. Каждый файл коллекции представлен объектом **File**.

Объекты, представляющие каждую из вышеперечисленных коллекций, обладают только двумя свойствами: **Count**, значением которого является количество элементов в данной коллекции, и **item(key)**, значением которого является элемент, заданный с помощью своего обязательного аргумента **key**. Как правило, последнее свойство не используется, но вместо него для последовательной обработки коллекций применяется объект **Enumerator**, определенный в языке JScript.

Далее следует список методов объекта **Enumerator**:

- **atEnd()** — возвращает **true**, если был достигнут конец коллекции и **false** в противном случае;
- **item()** — возвращает текущий элемент коллекции;
- **moveFirst()** — делает текущим первый элемент коллекции;
- **moveNext()** — делает текущим следующий элемент коллекции.