Часть III

Языки написания сценариев

В этой части...

Четыре главы, составляющие эту часть книги, предлагают вниманию пользователя обширный материал, посвященный “хиту наших дней”, — языкам написания сценариев. С помощью этих инструментальных средств обеспечивается разработка Web-приложений, выполняющих сложнейшие задачи, — начиная от поиска информации в Internet и завершая обработкой интерактивных запросов пользователей. Не секрет, что эти возможности могут применяться в “разрушительных” целях, хотя и “столовый нож оружием может стать”.

Освоив данный материал, вы научитесь пользоваться основными возможностями языков написания сценариев. Вполне естественно, что “нельзя объять необъятное”, поэтому многое придется осваивать в ходе практической работы. А теперь перейдем к описанию возможностей и приемов работы с языком VBScript.

Глава 10

Основы VBScript

В этой главе...

Базовые понятия

Объекты в VBScript

Объектная модель Microsoft Internet Explorer

Язык VBScript "вырос" из недр языка Visual Basic for Applications, сохранив все лучшие качества своего предка, — простоту в освоении и применении. Используя средства этого языка, можно создавать сценарии, возможности которых ничем не отличаются от возможностей сценариев JavaScript (а может даже превосходят последние). Сначала рассмотрим краткую историю возникновения и развития языка.

Краткая историческая справка

Вряд ли будет преувеличением назвать язык программирования BASIC наиболее простым инструментальным средством, позволяющим конструировать вполне работоспособные программы, которые могут принести их создателю коммерческий успех. Недаром этот язык изучается во всех начальных курсах информатики. Конечно, в настоящее время он уже потерял былую привлекательность, но все же по-прежнему остается примером языка, в котором можно программировать без особых трудностей.

Свое триумфальное шествие этот язык программирования начал теперь уже в далеком 1963 году. Именно тогда в стенах Дартмутского колледжа появилась первая версия этого языка. Причем он не пребывал в гордом одиночестве. В то время существовало целое семейство языков высокого уровня: FORTRAN, Cobol, Algol, Ada, PL/1, Focal, Pascal и т.д. И все они "сошли со сцены", за исключением BASIC и Pascal. Естественно, что при этом они подверглись серьезным изменениям (появился Delphi и Visual Basic for Applications), но основа осталась прежней.

Язык Visual Basic for Applications появился в 1991 году. В распоряжении программиста оказалась визуальная среда разработки программ, а также стандартные библиотеки, допускающие некоторую степень "автоматизации" в процессе разработки программ. Именно в это время получил широкое распространение Internet, а также язык Web-страниц — HTML. Естественно, что Web-страницы служили не только для отображения статической информации, но и обеспечивали интерактивное взаимодействие с пользователем. На то время единственным языком, обеспечивающим подобные возможности, был JavaScript.

Но к моменту появления браузера Microsoft Internet Explorer 3.0 фирма Microsoft вспомнила о своем любимом детище, включив поддержку языка Visual Basic Scripting Edition, или просто VBScript. В то время появился язык JavaScript, разработанный Netscape (причем изначально назывался LiveScript). Поскольку в то время большим успехом пользовался язык программирования Java, компания Netscape в рекламных целях воспользовалась названием JavaScript, как только получила лицензию от Sun. В ответ на это Microsoft разработала язык JScript для своего браузера Internet Explorer.

В дальнейшем я буду рассматривать примеры программ, ориентированных исключительно на Internet Explorer. И не потому, что испытываю особую любовь к программным продуктам фирмы Microsoft, а в силу того, что этот браузер входит в комплект поставки любой операционной системы из семейства Microsoft, поэтому многие именно на нем и останавливаются.

Ну а теперь перейду к краткому описанию синтаксиса языка, которого, впрочем, будет вполне достаточно, чтобы новички смогли приступать к написанию сценариев на VBScript.

> Обратите внимание на то, что при описании синтаксиса квадратные скобки (`[]`) обозначают необязательные для указания параметры.

Типы данных

Приверженцы и почитатели классического варианта BASIC наверное помнят о том, что для ввода символьной переменной использовался знак доллара:

```
a$="сегодня вы получите много буказоидов"
```

В VBScript все немного по-другому: единственный тип данных, называемый `Variant`, пригоден для хранения данных любых типов, традиционно применяемых в языках программирования (начиная от целочисленного типа и завершая объектами). Если же требуется точно определить применяемый в данном случае тип данных, можно воспользоваться специальными функциями преобразования (указаны в табл. 10.1). В связи с особенностями реализации типов данных в VBScript могут возникать ошибки из-за некорректной интерпретации. Например, поля экранной формы имеют текстовый формат. Поэтому если попытаться сложить значения двух полей (например, 5 и 4), в результате произойдет конкатенация двух текстовых значений (результат — "54"), а не 9, как вы вправе ожидать в данном случае.

Как правило, тип `Variant` применяется для определения числовых или символьных данных (что наиболее часто происходит при написании сценариев). Интересно отметить, что этот тип данных на самом деле является составным и может включать *подтипы*, охватывающие все значения, которые могут применяться в языках программирования. Все применяемые подтипы перечислены в табл. 10.1.

Таблица 10.1. Подтипы данных типа Variant

Название подтипа	*Применяемая функция преобразования*	*Краткое описание*
`Empty`	-	Применяется для новых переменных, которые еще не инициализированы
`Null`	-	Применяется в том случае, если переменная не содержит запрещенные данные

Окончание табл. 10.1

Название подтипа	***Применяемая функция преобразования***	***Краткое описание***
Error	-	Этот подтип предназначен для хранения номеров ошибок
Boolean	CBool(x)	Применяется для обозначения логических переменных, принимающих два допустимых значения: True или False
Byte	CByte(x)	Наиболее "компактный" тип данных, допустимые значения: 0...255
Integer	CInt(x)	Целый тип данных, допустимые значения: −32768...32768
Long	CLng(x)	Длинный целочисленный тип данных, допустимые значения: −2147483648...2147483647
Currency	CCur(x)	Специальный числовой формат для денежных величин, допустимые значения: −922337203685477,5808... 922337203685477,5807
Single	CSng(x)	Вещественный тип данных с плавающей точкой одинарной точности (для отрицательных чисел допустимые значения: −3,402823E38...−1,401298E-45; для положительных числе допустимые значения: 1,401298E-45...3,402823E39)
Double	CDbl	Вещественный тип данных с плавающей точкой двойной точности (для отрицательных чисел допустимые значения: −1,797693124E38...−1,401298E-45; для положительных чисел допустимые значения: 1,401298E-45...3,402823E39)
Date, Time	Cdate	Определяет число в формате отображения времени и даты от 1 января 100 года до 31 декабря 9999 года
String	Cstr	Символьный тип данных
Object	-	Определяет ссылку на объекты, обычно ole, html и ActiveX

Обратите внимание на то, что первые три подтипа, перечисленные в таблице, исполняют чисто "информационную" роль. Значение Empty изначально присваивается переменной, которая объявлялась в операторе Dim, но пока что не получила какое-либо значение. Для числовых типов переменных это значение равно 0, для строковых — " ". Переменной может присваиваться значение Null, которое означает отсутствие данных. Это значение может присваиваться переменной автоматически во время выполнения каких-либо операций над ней. Непустое значение Error свидетельствует о том, что во время выполнения процедуры возникла ошибка.

При определении подтипов данных используются *литералы*, представляющие собой символьные константы. Литералы бывают четырех видов: числовые (целые, с плавающей точкой, с фиксированной точкой), строковые (последовательность символов, заключенная в двойные кавычки), булевы (целочисленные константы, принимающие значения `0` и `1`), а также даты и времени. Обратите внимание на то, что при записи литералов даты и времени применяются знаки `#`:

```
#31-1-2004 19:00
```

Если требуется выполнить преобразование одного подтипа данных в другой, используются специальные функции, приведенные во втором столбце табл. 10.1.

Константы, переменные и массивы

В VBScript приняты две формы объявления переменных:

- явное объявление — с помощью оператора `Dim`;
- неявное объявление — путем указания имени переменной.

Синтаксис оператора Dim имеет следующий вид:

```
Dim название_переменной
```

В качестве имени переменной допускается любая комбинация символов, общая длина которой не превышает 255, за исключением символов пробела, точки, восклицательного знака, а также символов `#`, `$`, `&` и `@`.

В данном случае отсутствует чувствительность к изменению регистра символов, поэтому этот факт можно не отслеживать.

С помощью одного оператора Dim разрешается объявлять несколько переменных, разделяемых запятой.

> Объявлять переменные можно путем использования альтернативных операторов `Public` и `Private`. Эти конструкции были унаследованы из языка Visual Basic for Applications и объявляют глобальные и локальные переменные, соответственно.

Как и в любом другом языке программирования высокого уровня, в VBScript предусмотрено использование массивов. Причем допускается применение одномерных и многомерных массивов, которые могут быть статическими и динамическими.

Ниже приводится синтаксис объявления массива:

```
Dim название_массива(n1, n2,...)
```

Как видите, здесь используется тот же оператор, что и в случае объявления простой переменной. Отличие заключается в использовании индексов (`n1`, `n2`,...), указанных в круглых скобках. Причем нумерация индексов массива начинается с 0. В данном случае определяются *статические* массивы, на что указывает числовой индекс. Если же индекс не указывается, определяется так называемый *динамический* массив. При этом заранее неизвестно количество измерений, а также элементов, образующих массив.

Ниже приводится пример объявления динамического массива:

```
Dim Cars()
```

Если требуется переопределить массив, указав его размерность, применяется оператор `Redim`. При этом используется тот же синтаксис, что и в случае оператора `Dim`. В случае переопределения массива возникает угроза "затирания" прежних значений. Во избежание этого в комбинации с оператором `Redim` следует использовать ключевое слово `Preserve`:

```
Dim Cars()
...
Redim Cars(5)
...
Redim Preserve Cars(Ubound(Cars)+1)
```

Обратите внимание на тот примечательный факт, что необходимость в применении ключевого слова `Preserve` возникает только при втором переопределении массива. Причем в этом случае применяется функция `Ubound`, с помощью которой определяется максимальный индекс массива `Cars`.

> Если нужно быстро очистить содержимое массива, используется оператор `Erase` (в виде `Erase(a)`, где `a` — название массива).

Константы

Достаточно часто возникает ситуация, когда в программе задаются значения переменных, которые не изменяются в дальнейшем. При этом проще всего воспользоваться *константами*. Константы в VBScript определяются с помощью ключевого слова `Const` (как в "добром старом" BASIC). В этом случае применяется следующий синтаксис:

```
Const название_константы = присваиваемое_значение
```

Ниже приведены примеры некоторых констант "в действии":

```
Const CarName = "Волга" 'Строковая константа
Const MaxBall = "12" 'Числовая константа
Const NewYear = #1-1-2004#
```

Операторы

Выполнение различных действий с переменными, константами и литералами осуществляется с помощью *операторов*. Каждый оператор находится в своей собственной строке, причем для его завершения не требуется использование какого-либо разделителя. При указании нескольких операторов в одной строке применяется двоеточие.

Если встречается длинный оператор, который не помещается в одной строке, следует воспользоваться символом продолжения (подчеркивание).

Каждая хорошая программа должна включать комментарии, которые позволят вам разобраться в происходящем через довольно продолжительное время, когда назначение отдельных конструкций "сглаживается" из памяти. В VBScript комментарии вводятся с помощью символов одинарных кавычек или оператора `REM` (как в классическом BASIC).

Арифметические операторы (`+`, `-`, `*`, `/`) выполняют арифметические действия над указанными операндами. Скобки изменяют порядок выполнения операций (как и в любом другом языке программирования).

Операции сравнения выполняются с помощью операторов сравнения: равенство (`=`), неравенство (`<>`), больше (`>`), меньше (`<`), больше или равно (`>=`), меньше или равно (`<=`). Для сравнения объектов между собой используется оператор `Is`.

В VBScript применяется ряд операторов, которые обрабатывают логические (булевы) данные: отрицание (`Not`), конъюнкция (`And`), дизъюнкция (`Or`), исключающее ИЛИ (`Xor`), импликация (`Imp`) и эквивалентность (`Eqv`).

Присваивание переменной объекта осуществляется с помощью оператора Set или просто с помощью знака '=':

```
Set a = значение
a = значение
```

Условные операторы

Условные операторы предназначены для изменения обычного хода выполнения сценария VBScript. В этом случае используются три формы:

- If...Then
- If...Then...Else
- Select Case

Первая синтаксическая конструкция позволяет выполнять оператор (или набор операторов) в случае, если истинно выражение, используемое в качестве проверочного условия.

Первая форма оператора имеет следующий вид:

```
If условие then оператор
```

Если требуется выполнить группу операторов, применяется вторая форма условного оператора:

```
If условие Then
     Операторы
End If
```

Ниже приводится наиболее обобщенная форма условного оператора:

```
If условие1 Then
     [первая группа операторов]
[ElseIf условие2 Then
     [вторая группа операторов]]
...
[Else
     [n-я группа операторов]]
End If
```

Здесь происходит последовательная проверка условий. Если ни одно из условий не является истинным, выполняется блок операторов, который соответствует оператору Else. Затем сценарий передает управление операторам, которые находятся после конструкции End If.

При наличии большого количества проверяемых условий целесообразнее воспользоваться конструкцией Case Else. Ниже приводится описание применяемого в этом случае синтаксиса:

```
Select Case проверяемое_выражение
     [Case первый перечень значений
          [первая группа операторов]]
     [Case второй перечень значений
          [вторая группа операторов]]
          ...
     [Case Else
          [n-я группа операторов]]
End Select
```

В данном случае производится однократное вычисление значения проверяемого_выражения. Затем производится его сравнение со значениями,

находящимися в блоках Case. Если соответствие установлено, выполняется нужная группа операторов из данного блока, после чего передается управление оператору, который находится после End Select.

Если соответствие не установлено, выполняются операторы из блока Case Else.

А теперь рассмотрим операторы, предназначенные для организации циклов в VBScript.

Операторы циклов

Ниже перечислены операторы, применяемые для организации циклов в VBScript:

- Do...Loop
- For...Next
- For Each...Next

Форма цикла Do..Loop позволяет выполнять соответствующую группу операторов до тех пор, пока значение выражения остается истинным или ложным. Здесь выделяются две формы цикла: Do While и Do Until.

Ниже представлен синтаксис этих двух разновидностей циклов:

```
Do While условие завершения цикла
     группа операторов
Loop
```

Перед началом выполнения внутренних операторов цикла производится проверка условия завершения цикла. Если это условие истинно, выполняются операторы цикла. При этом изменяется само условие. Выполнение цикла продолжается до тех пор, пока условие не станет ложным. Эта конструкция называется циклом с *предусловием*. Здесь перед первым выполнением цикла производится проверка условия, и если оно ложно, цикл не выполняется ни разу.

Во время цикла с *постусловием* проверка условия осуществляется после завершения первой итерации цикла:

```
Do
     группа операторов
Loop Until условие завершения цикла
```

Форма цикла Do...Until эквивалентна циклу Do...While. Этот цикл относится также к категории циклов с предусловием, а его выполнение продолжается до тех пор, пока условие завершения цикла остается ложным.

Формы циклов, описанные выше, характеризуются тем, что заранее неизвестно количество итераций. Если же требуется выполнить четко определенное количество итераций, следует воспользоваться формой цикла For...Next. Ниже приводится описание соответствующего синтаксиса:

```
For счетчик = начальное значение To конечное значение [Step приращение]
     Набор операторов
Next
```

При первом выполнении цикла переменной счетчик присваивается значение, которое задается параметром начальное значение. Затем при каждом выполнении цикла происходит изменение значения счетчика на величину приращения. Причем значение приращения может быть положительным или отрицательным. Если параметр Step не указывать, значение счетчика будет увеличиваться на единицу после каждой итерации цикла.

Если же требуется организовать цикл по элементам некоего массива или объектам семейства объектов, причем заранее неизвестно количество этих дискретных элементов, следует воспользоваться циклом `For Each...Next`. Ниже приводится описание соответствующего синтаксиса:

```
For Each элемент In группа
     Набор операторов
Next
```

Обратите внимание на то, что с помощью параметра `группа` задается название массива или набора объектов. На каждой итерации цикла переменная-элемент включает ссылку на элемент массива (объект набора). Как только завершится перебор всех элементов массива, работа цикла заканчивается.

Иногда возникает потребность в немедленном выходе из цикла. Для этого применяется оператор `Exit`, который выступает в двух формах:

- `Exit...Do`: немедленный выход из цикла `Do...Loop`;
- `Exit For`: немедленный выход из цикла `For...Next` и `For Each...Next`.

А теперь перейдем к описанию одной из наиболее полезных синтаксических конструкций VBScript — *процедур*.

Процедуры

Как и в любом другом языке программирования, в VBScript часто возникает потребность в многократном выполнении набора операторов (например, отображать экран справки в нескольких местах). При этом соответствующие операторы оформляются в виде процедуры.

Процедуры в VBScript бывают двух типов:

- процедуры-подпрограммы: `Sub`;
- процедуры-функции: `Function`.

Процедура-подпрограмма выполняет ряд действий, но при этом не возвращает программе какие-либо значения. Ниже приводится соответствующий синтаксис:

```
Sub название_процедуры ([перечень параметров])
     выполняемые операторы
End Sub
```

С помощью перечня параметров можно передавать процедуре данные или получать некоторые вычисленные значения.

Для вызова процедур-подпрограмм применяется оператор `Call название_процедуры(параметры)`. Если же вызывать этот вид процедуры простым указанием имени, тогда параметры указываются без применения скобок.

Ниже продемонстрированы два варианта вызова процедуры-подпрограммы:

```
Call CalcMoney(FirstSum, Result)
CalcMoney FirstSum, Result
```

Процедура может также задаваться в виде *процедуры-функции*, которая также получает набор каких-то внешних данных, но при этом возвращает результат, ассоциированный с ее именем. Впоследствии такую процедуру можно использовать в сценариях VBScript подобно тому, как это происходит со *встроенными функциями*. Обратите внимание на синтаксис процедуры-функции:

```
Function название процедуры ([перечень параметров])
     Набор операторов
     Название процедуры = значение
End Function
```

Передача параметров процедурам осуществляется *по ссылке* и *по значению*.

При передаче параметра по значению вместо адреса переменной процедура получает копию переменной. Поэтому, независимо от результатов обработки процедурой, значение исходной переменной изменяться не будет. Для передачи по значению применяется ключевое слово `ByVal`:

```
Sub CalcMoney(ByVal FirstSum, Result)
     Result = FirstSum * 100
End Sub
FirstSum = 5
Result = 10
CalcMoney FirstSum
CalcMoney FirstSum, Result
document.write Result //здесь выводится 10, а не 500
```

При передаче по ссылке, определенной по умолчанию, передается фактический адрес переменной-параметра.

В VBScript существует целый ряд встроенных функций, большинство из которых приведены в табл. 10.2.

Таблица 10.2. Встроенные функции VBScript

Синтаксис	*Описание*	*Пример использования*
`Abs(число)`	Модуль числа	`Abs(-1)`
`Atn(число)`	Арктангенс числа	`Ath(0,9)`
`Cos(число)`	Косинус угла, определенного в радианах	`Cos(1)`
`Exp(число)`	Экспонента — e в степени x	`Exp(2)`
`Int(число)`	Целая часть числа	`Int(1,79)`
`Log`	Натуральный логарифм	`Log(5)`
`Sin(число)`	Синус угла, определенного в радианах	`Sin(1)`
`Sgr(число)`	Квадратный корень	`Sgr(2)`
`Tan(число)`	Тангенс угла, определенного в радианах	`Tan(1)`
`Asc(символ)`	Вычисление ASCII-кода символа	`a=asc("d")`
`Chr(число)`	Вычисление символа, соответствующее ASCII-коду	`b=Chr(110)`
`Fix(число)`	Округление до ближайшего целого числа	`a=Fix(2.5)` `(результат - 2)`
`Hex(число)`	Шестнадцатеричное представление числа	`b=Hex(13)`

Продолжение табл. 10.2

Синтаксис	*Описание*	*Пример использования*
`Oct(число)`	Восьмеричное представление числа	`c=Oct(20)`
`rnd(число)`	Случайное число в диапазоне от 0 до 1	`a=rnd(1)`
`Randomize`	Начальная установка генератора случайных чисел	`Randomize`
`sgn(число)`	Определение знака числа	`sgn(-5)`
`Date`	Вычисление текущей даты	`date`
`Day(выражение)`	Определение дня недели	`Day(date)`
`Time`	Показания системных часов	`time`
`Timer`	Количество секунд, прошедших от полуночи	`timer`
`Now`	Вычисление текущих даты и времени	`now`
`Hour(выражение)`	Вычисление часа	`b=hour(now)`
`Minute(выражение)`	Вычисление минуты	`c=Minute(now)`
`Month(выражение)`	Вычисление месяца	`d=Month(now)`
`Second(выражение)`	Вычисление секунды	`e=Second(now)`
`Year(выражение)`	Вычисление года	`f=Year(now)`
`Weekday(выражение)`	Вычисление дня недели	`g=Weekday(now)`
`Instr (начало, строка, искомая подстрока)`	Возвращает номер символа в строке, с которого начинается исходная подстрока	`If Instr(1,mail,@) then ...`
`Lcase (строка)`	Преобразование символов строки в строчные буквы	`a=Lcase("ПРивЕТ") ... a="привет"`
`Ucase (строка)`	Преобразование символов строки в заглавные буквы	`a=Ucase("ПРивЕТ") ... a="ПРИВЕТ"`
`Left (строка,N символов)`	Левая часть строки длиной N символов	`a=Left("Привет",3) ... a="При"`
`Right (строка,N символов)`	Правая часть строки длиной N символов	`a=Right("Привет",3) ... a="вет"`
`Len (строка)`	Длина строки	`a=Len("Привет") ... a=6`
`Ltrim (строка)`	Удаляет начальные пробелы	`a=Ltrim(" Привет ",) a="Привет "`
`Rtrim (строка)`	Удаляет конечные пробелы	`a=Rtrim(" Привет ",) a=" Привет"`
`Trim (строка)`	Удаляет начальные и конечные пробелы	`a=Ttrim(" Привет ",) a="Привет"`
`Mid (строка, начало, N символов)`	Часть строки с позиции "начало" и длиной N символов	`a=Mid("Привет",3,2) a="ве"`

Окончание табл. 10.2

Синтаксис	*Описание*	*Пример использования*
`Left(строка,N символов)`	Левая часть строки длиной N символов	`a=Left("Привет",3)` `a="При"`
`Space (N)`	Строка из N пробелов	`a=Space(5)`
`String (N символов, символ)`	Строка из N символов	`a=String(5,"A")` `a="AAAAA"`
`Ltrim (строка)`	Удаляет начальные пробелы	`a=Ltrim(" Привет ",)` `a="Привет "`

Примеры вызова процедур и функций в VBScript

Ниже приводятся примеры процедуры-подпрограммы, применяющей две функции языка VBScript: `InputBox` и `MsgBox`. Эти две функции отображают окно с полем ввода и окно вывода (с результатом). Программа запрашивает у пользователя ввод температуры (в градусах Фаренгейта), переводит ее в градусы Цельсия и отображает результат. Вычисление температуры происходит в функции `Celsius`:

```
<SCRIPT LANGUAGE="VBScript">
<!-
Sub ConvertTemp()
temp = InputBox("Please enter the temperature in degrees
                F.", 1)
MsgBox "The temperature is " & Celsius(temp) & " degrees C."
End Sub
-></SCRIPT>
```

Ниже приводится пример использования процедуры-функции. Здесь функция `Celsius` переводит значение температуры (по шкале Фаренгейта) в шкалу по Цельсию. Когда вызывается функция из подпрограммы `ConvertTemp`, переменная, содержащая параметр-значение, передается функции. Результат вычислений возвращается вызывающей процедуре, а затем отображается с помощью `MsgBox`:

```
<SCRIPT LANGUAGE="VBScript">
<!-
      Sub ConvertTemp()
            temp = InputBox("Please enter the temperature in
                  degrees F.", 1)
            MsgBox "The temperature is " & Celsius(temp) & "
      degrees C."
End Sub

Function Celsius(fDegrees)
      Celsius = (fDegrees - 32) * 5 / 9
End Function
->
</SCRIPT>
```

> Для включения сценария на языке VBScript в HTML-код используется пара элементов `<SCRIPT LANGUAGE="VBScript">` и `</SCRIPT>`. Элементы комментария (`<! -` и `- >`) гарантируют, что сценарий не будет отображаться в браузерах, которые не поддерживают VBScript.

Прежде чем использовать процедуры в любой программе, следует их объявить. Для этого их включают в самом начале раздела `<HEAD>` HTML-страницы. Весь последующий текст также должен находиться в разделе `<HEAD>`.

Как упоминалось ранее, любые данные передаются процедурам в виде параметров. При создании подпрограммы (или функции) после ее имени должны указываться круглые скобки. Любые параметры внутри них отделяются запятыми. В следующем примере параметр `fDegrees` передает значения функции `Celsius` с целью их дальнейшего преобразования:

```
<SCRIPT LANGUAGE="VBScript">
<!-
Function Celsius(fDegrees)
      Celsius = (fDegrees - 32) * 5 / 9
End Function
->
</SCRIPT>
```

Для вызова функции в программе необходимо поставить ее название справа от переменной или выражения, которому будет возвращаться результат выполнения функции, например:

```
<SCRIPT LANGUAGE="VBScript">
      <!-
            Temp = Celsius(fDegrees)
      ->
</SCRIPT>
```

или

```
<SCRIPT LANGUAGE="VBScript">
      <!-
            MsgBox "The Celsius temperature is " & Celsius(fDegrees) &
                  " degrees."
      ->
</SCRIPT>
```

Чтобы вызвать одну процедуру из другой, необходимо указать имя первой процедуры со значениями требуемых параметров, каждый из которых должен отделяться от предыдущего запятой. Причем в данном случае оператор `Call` не требуется. Но если вы все-таки используете его, следует заключить любые параметры в круглые скобки. Следующий пример иллюстрирует две возможности вызова процедуры `MyProc`. В первом случае используется оператор `Call`, во втором — нет, причем действие выполняется одно и то же.

```
<SCRIPT LANGUAGE="VBScript">
      <!-
            Call MyProc(firstarg, secondarg)
            MyProc firstarg, secondarg
      ->
</SCRIPT>
```

Обратите внимание на то, что круглые скобки опускаются при вызове подпрограммы, если оператор `Call` не применяется.

Методы вызова сценариев VBScript

Любая Web-страница, включающая активное содержимое (сценарий), состоит из двух относительно независимых частей. Первая — это HTML-код самой страницы, содержащий необходимые элементы управления (ссылки, кнопки, формы для ввода информации и т.д.), вторая — сценарии, которые начинают работать при различных событиях, происходящих с элементами управления.

Рассмотрим основные приемы вызова сценариев.

Сценарий, вызываемый неявным образом

Листинг 10.1. Вызов простейшего сценария неявным образом

```
<HTML>
<HEAD>
      <SCRIPT language="VBScript">
            Sub but_onclick
            Alert "Всем привет"
            End sub
      </SCRIPT>
</HEAD>

<BODY>
      <FORM>
            <input type="button" name="but">
      </FORM>
</BODY>
</HTML>
```

Результат выполнения этой программы показан на рис. 10.1.

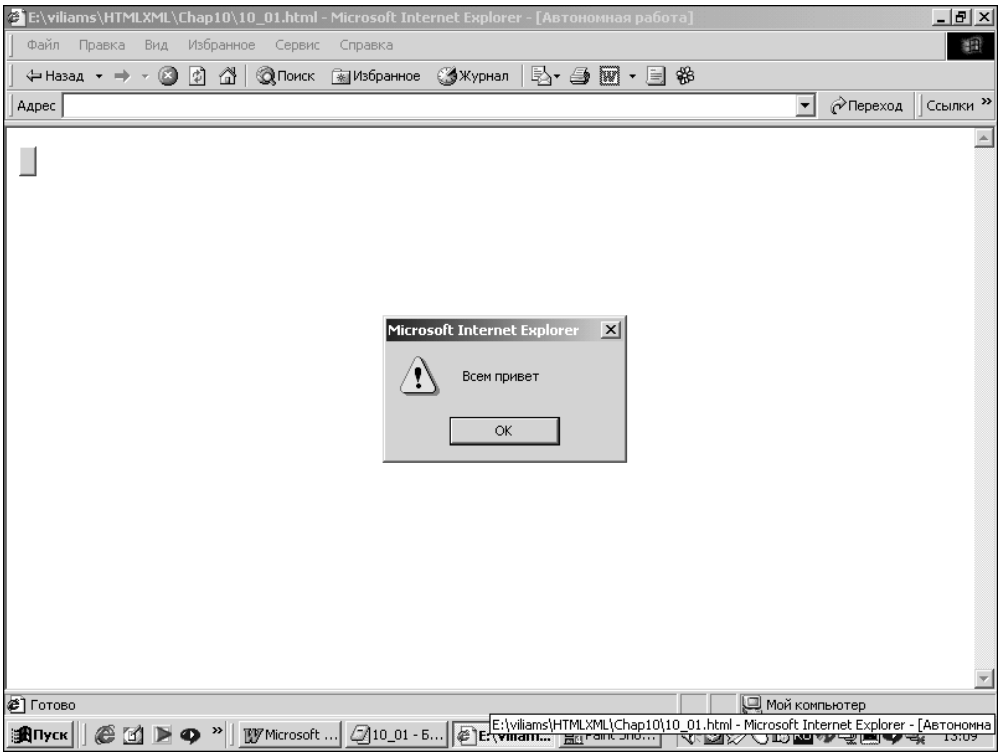

Рис. 10.1. *Результат выполнения первого завершенного сценария VBScript*

Вызов сценария с помощью элемента управления

Листинг 10.2. Вызов сценария с помощью элемента управления

```
<HTML>
      <BODY>
            <script language="VBScript">
```

```
                Alert "Всем привет"
            </SCRIPT>
        </BODY>
</HTML>
```

Код программы в этом случае выполнится сразу после загрузки документа (рис. 10.2).

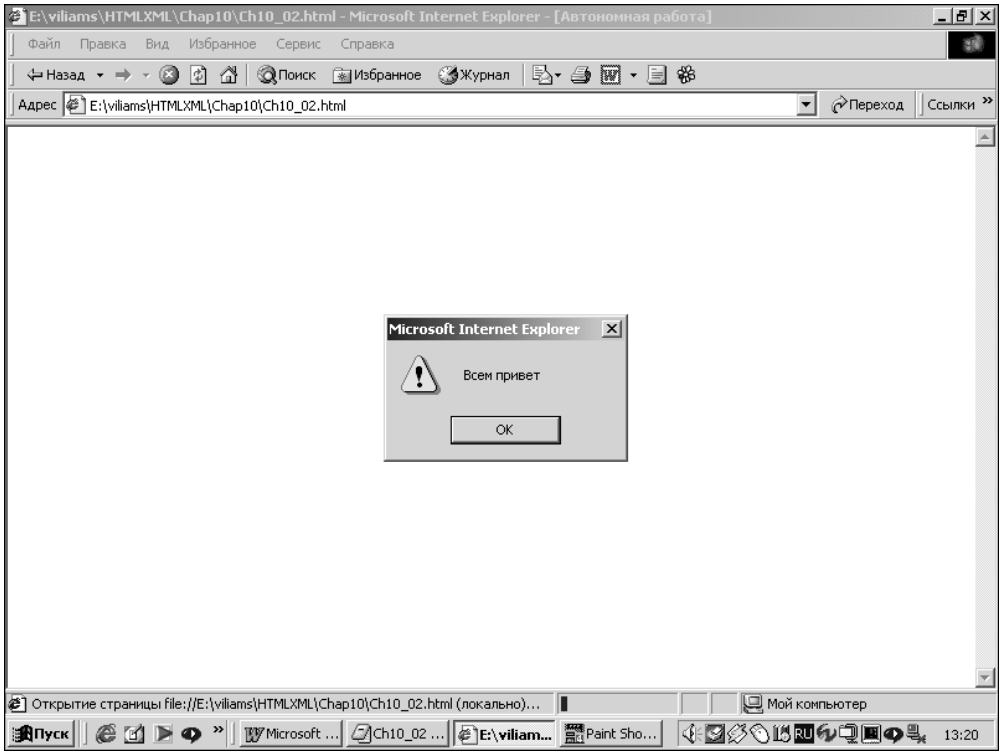

***Рис. 10.2.** Загрузка сценария с помощью элемента управления*

Объекты в VBScript

Сначала дадим некоторые основные определения. *Объект* в VBScript — это простой объект, методами, свойствами и событиями которого может управлять пользователь. *Метод* — это процедура или просто набор команд, сообщающих объекту о том, что следует выполнить некоторую задачу. *Свойство* — это некоторый вид параметра объекта. *Событие* — это сигнал, который генерируется в том случае, если с объектом что-то происходит. Методы, свойства и события выбираются разработчиком исходя из потребностей задачи. Так, если в вашей программе нужно отследить время окончания работы, то имеет смысл обращать внимание на событие, сигнализирующее об этом. В противном случае эта информация бесполезна.

А теперь перейдем к описанию двух основных функций, позволяющих генерировать стандартные диалоговые окна.

Функции `InputBox` и `MsgBox` (см. предыдущий раздел) применяются практически всеми сценариями на языке VBScript.

Функция InputBox приводит к отображению диалогового окна, в котором содержится поле ввода, а также две кнопки OK и Cancel. Применяемый в этом случае синтаксис имеет следующий вид:

```
InputBox(подсказка, заголовок, заданное по умолчанию значение, [x, y])
```

С помощью параметров подсказка и заголовок определяются подсказка для пользователя и заголовок диалогового окна, соответственно. Параметр заданное по умолчанию значение определяет значение, отображаемое в поле при открытии диалогового окна. Необязательные параметры x и y определяют координаты левого верхнего угла диалогового окна.

Функция MsgBox выводит сообщения для пользователя. Применяемый в этом случае синтаксис имеет следующий вид:

```
MsgBox(строка сообщения, тип окна, заголовок окна)
```

С помощью параметра строка сообщения определяется отображаемое в диалоговом окне сообщение. Параметр тип окна задает вид отображаемого окна. Этот параметр представляет собой целое число, которое фактически является суммой трех целых чисел, определяющих все параметры отображаемого окна (количество и типы кнопок, вид пиктограммы, стандартная кнопка). Эти значения сведены в табл. 10.3–10.5, соответственно. С помощью параметра заголовок окна определяется название окна.

Таблица 10.3. Типы отображаемых кнопок

Значение	*Результат*
0	Кнопка OK
1	Кнопки OK и Cancel
2	Кнопки Abort, Retry и Ignore
3	Кнопки Yes, No и Cancel
4	Кнопки Yes и No
5	Кнопки Retry и Cancel

Таблица 10.4. Выводимые пиктограммы

Значение	*Результат*
0	Пиктограммы не отображаются
16	Отображается пиктограмма останова (X)
32	Отображается пиктограмма вопроса (?)
48	Отображается пиктограмма восклицания (!)
64	Отображается информационная пиктограмма (I)

Таблица 10.5. Кнопка, заданная по умолчанию

Значение	*Результат*
0	Первая кнопка
256	Вторая кнопка
512	Третья кнопка

При выполнении функции MsgBox возвращаются целые значения, описание которых приведено в табл. 10.6.

Таблица 10.6. Значения, возвращаемые функцией MsgBox

Нажатая кнопка	*Возвращаемое значение*
OK	1
Cancel	2
Abort	3
Retry	4
Ignore	5
Yes	6
No	7

Работа с объектами

Для включения объектов в HTML-документ следует использовать пару дескрипторов <OBJECT></OBJECT> (см. главу 1), а для установки значений его свойств — дескриптор <PARAM>. Если у вас есть опыт работы с Visual Basic, то использование дескрипторов <PARAM> будет напоминать вам размещение какого-либо компонента или элемента управления на форме. Например, следующие значения дескрипторов <OBJECT> и <PARAM> добавляют на HTML-страницу элемент ActiveX Label:

```
<OBJECT
      classid="clsid:99B42120-6EC7-11CF-A6C7-00AA00A47DD2"
      id=lblActiveLbl
      width=250
      height=250
      align=left
      hspace=20
      vspace=0>
      <PARAM NAME="Angle" VALUE="90">
      <PARAM NAME="Alignment" VALUE="2">
      <PARAM NAME="BackStyle" VALUE="0">
      <PARAM NAME="Caption" VALUE="Простая метка">
      <PARAM NAME="FontName" VALUE="Arial">
      <PARAM NAME="FontSize" VALUE="20">
      <PARAM NAME="FontBold" VALUE="1">
      <PARAM NAME="FrColor" VALUE="0">
</OBJECT>
```

При работе с объектами можно присваивать или устанавливать свойства, а также вызывать методы точно так же, как в случае с любым средством управления формой. Например, в следующем коде включаются дескрипторы <FORM>, которые могут использоваться для управления свойствами двух элементов Label:

```
<FORM NAME="LabelControls">
      <INPUT TYPE="TEXT" NAME="txtNewText" SIZE=25>
      <INPUT TYPE="BUTTON" NAME="cmdChangeIt" VALUE="Change Text">
      <INPUT TYPE="BUTTON" NAME="cmdRotate" VALUE="Rotate Label">
</FORM>
```

Обработчик события нажатия кнопки `cmdChangeIt` изменяет код, соответствующий объекту `Label`:

```
<SCRIPT LANGUAGE="VBScript">
<!-
Sub cmdChangeIt_onClick
      Dim TheForm
            Set TheForm = Document.LabelControls
            lblActiveLbl.Caption = TheForm.txtNewText.Value
End Sub
->
</SCRIPT>
```

Объектная модель Microsoft Internet Explorer

Объектная модель браузера Microsoft Internet Explorer достаточно сложная, хотя при написании сценариев основную роль играют объекты, связанные с дескрипторами HTML-страниц.

Корневым объектом в иерархии является `Window`, который "порождает" все остальные объекты (рис. 10.3). Объекты `Form` ассоциированы с HTML-формами и включают разнообразные элементы управления. Все эти элементы будут подробно описаны в главе 12 (при рассмотрении возможностей языка JavaScript).

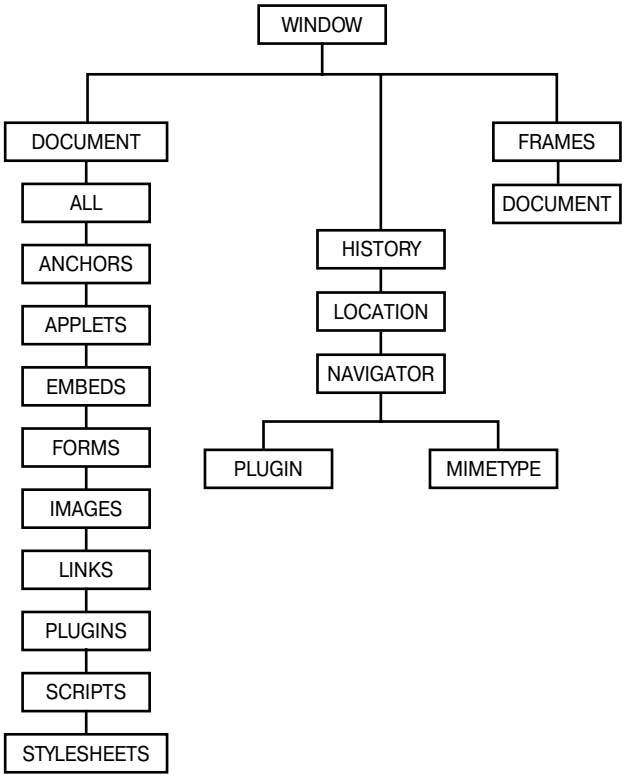

Рис. 10.3. *Объекты Microsoft Internet Explorer*

Следующая глава посвящена разработке HTML-форм и языку написания сценариев Perl. Там же будут подробно описаны события, связанные с вводом данных в формы. Мне же остается пожелать вам успехов в дальнейшем изучении языка VBScript. Несмотря на свою простоту, этот язык стал основой для написания вирусов и "троянских коней". Поэтому отнеситесь к изучению возможностей языка VBScript со всей серьезностью.