Вадим Дунаев

Основы VBScript

Здесь рассматриваются основные понятия, синтаксис и объекты VBScript без претензий на исчерпывающую полноту. Тем не менее, приведенный материал можно использовать в качестве краткого справочника и элементарного учебника для начинающих. Нумерация разделов согласована с моей книгой "**HTML, скрипты и стили**", ***2-е издание, 2008г.***.

Принципы программирования сценариев на VBScript такие же, что и на JavaScript. Однако имеются весьма заметные отличия в синтаксисе и наборах встроенных средств. В этой главе мы не будем рассматривать VBScript столь же подробно, как JavaScript, а остановимся лишь наиболее важных, на мой взгляд, особенностях. Для тех, кто уже знаком с JavaScript, ниже приводится список основных особенностей VBScrip, отличающих его от JavaScript:

- выражения на VBScript не зависят от регистра;
- комментарии выделяются одинарным апострофом (верхней одинарной кавычкой `'`);
- объявление и присвоение значений переменным возможно как с помощью оператора присваивания `=`, так и с помощью специальных операторов (`Dim`, `Let`, `Set`, `Private`, `Public`);
- задание массивов;
- определения функций и процедур;
- вызовы функций, процедур и методов объектов допускают как бесскобочный, так и скобочный синтаксис при передаче параметров;
- обработчики событий вызываются в выражениях не с точечным, а с дефисным синтаксисом.

13.1. Ввод и вывод данных

Для вывода сообщения существует функция `MsgBox`, а для ввода — `InputBox`. С некоторым приближением их можно рассматривать как аналоги методов `alert()` и `prompt()` в JavaScript.

13.1.1. Функция *MsgBox*

Выводит диалоговое окно с сообщением и набором кнопок; возвращает числовое значение, показывающее, какая кнопка нажата. Возможны две формы вызова:

`MsgBox(`*`сообщение`*`, `*`число`*`, `*`заголовок`*`)`
`MsgBox `*`сообщение`*`, `*`число`*`, `*`заголовок`*

Синтаксис с круглыми скобками для записи вызова функции `MsgBox` используется, чтобы присвоить возвращаемое значение произвольной переменной:

```
x=MsgBox("Добро пожаловать!", 20, "Приветствие")
```

Первый параметр обязателен. Второй числовой параметр указывает, какая картинка и кнопки появятся в диалоговом окне. Значения числового параметра, определяющего набор кнопок на панели, приведены в табл. 13.1.

Таблица 13.1. *Значения числовой константы, определяющей кнопки*

Константа	Значение	Кнопки
`vbOkOnly`	0	**OK**
`vbOkCancel`	1	**OK**, **Отмена** (Cancel)
`vbAbortRetryIgnore`	2	**Стоп** (Abort), **Повтор** (Retry), **Пропустить** (Ignore)
`vbYesNoCancel`	3	**Да** (Yes), **Нет** (No), **Отмена** (Cancel)
`vbYesNo`	4	**Да** (Yes), **Нет** (No)
`vbRetryCancel`	5	**Повтор** (Retry), **Отмена** (Cancel)
`vbDefaultButtonI`	0	По умолчанию активна первая слева кнопка
`vbDefaultButton2`	256	По умолчанию активна вторая слева кнопка
`vbDefaultButton3`	512	По умолчанию активна третья слева кнопка
`vbDefaultButton4`	768	По умолчанию активна четвертая слева кнопка

Значения числового параметра, определяющего картинку, приведены в табл. 13.2.

Таблица 13.2. *Значения числовой константы, определяющей картинку*

Константа	Значение	Описание	Картинка
`vbCritical`	16	Важное сообщение	Перекрестие
`vbQuestlon`	32	Вопрос	Знак вопроса
`vbExclamation`	48	Предупреждение	Восклицательный знак
`vblnformatlon`	64	Информационное сообщение	Буква "i"

Две константы, представленные в табл. 13.3, определяют режим работы диалогового окна.

Таблица 13.3. *Режимы диалогового окна функции* `MsgBox`

Константа	Значение	Описание
`vbApplicationModal`	0	Пока пользователь не нажмет кнопку, работа приложения останавливается
`vbSystemModal`	4096	Пока пользователь не нажмет кнопку, работа всей системы останавливается

Чтобы использовать одновременно несколько установок, необходимо просто сложить соответствующие значения числового параметра. Например, если требуется вывести картинку с вопросом и две кнопки **Да** (Yes) и **Нет** (No), то следует задать числовой параметр, равным 4 + 32 = 36.

Пример (рис. 13.1):

```
MsgBox "Добро пожаловать!", 20, "Приветствие"
```

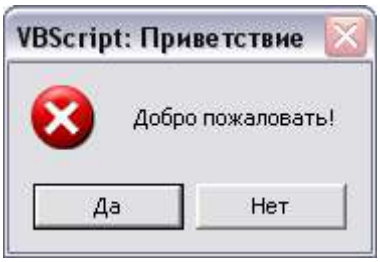

Рис. 13.1. Один из вариантов диалогового окна, вызываемого функцией `MsgBox`

В табл. 13.4 перечислены возвращаемые значения, которые присваиваются переменной `intButtonClicked`.

Таблица 13.4. *Возвращаемые функцией* `MsgBox` *значения*

Константа	Значение	Нажатая кнопка
`VbOK`	1	**OK**
`vbCancel`	2	**Отмена** (Cancel)
`vbAbort`	3	**Стоп** (Abort)
`vbRetry`	4	**Повтор** (Retry)
`vbIgnore`	5	**Пропустить** (Ignore)
`vbYes`	б	**Да** (Yes)
`vbNo`	7	**Нет** (No)

Если требуется, чтобы сообщение выводилось в нескольких строках, то следует использовать функцию `Chr(13)`, возвращающую служебный символ перевода каретки в качестве разделителя строк. Например,

```
MsgBox "Привет!" + Chr(13) + "Для продолжения щелкните на кнопке OK"
```

13.1.2. Функция *InputBox*

Выводит диалоговое окно с полем ввода данных и двумя кнопками **OK** и **Отмена** (Cancel). Возвращает текст в поле ввода данных, если был щелчок на кнопке **OK**, или значение `Empty`, если был щелчок на кнопке **Отмена** (Cancel). Возможны две формы вызова:

```
InputBox(подсказка, заголовок, исходное_значение[, x, y])
InputBox подсказка, заголовок, исходное_значение[, x, y]
```

Синтаксис с круглыми скобками для записи вызова функции `InputBox` используется, чтобы присвоить возвращаемое значение произвольной переменной:

```
x=InputBox("Введите пароль", "Вход в систему", "", 1000, 2000)
```

Параметр *подсказка* представляет собой текст, выводимый внутри диалогового окна, *исходный_текст* — содержимое поля ввода при открытии окна, *x* и *y* — необязательные соответственно горизонтальная и вертикальная координаты относительно левого верхнего угла экрана монитора, измеряемые в твипах (1 твип = 1/1440 дюйма). На рис. 13.2 показан пример диалогового окна.

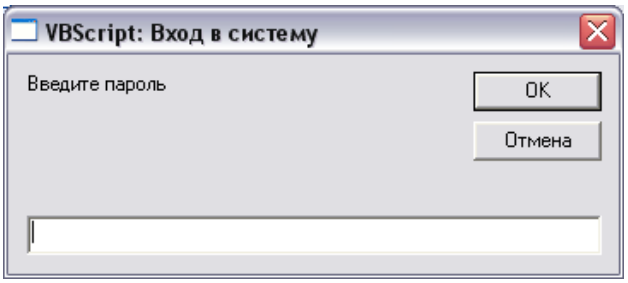

Рис. 13.2. Пример диалогового окна, вызываемого функцией `InputBox`

В сценариях, выполняющихся браузером, кроме функций `MsgBox` и `InputBox` можно использовать методы `alert()`, `confirm()` и `prompt()` объекта `window`.

13.2. Типы данных

Язык VBScript, как и JavaScript, является языком с так называемым *свободным типом данных*. Это означает, что переменная будет иметь тип, который определяется типом присваиваемого значения. При этом одна и та же переменная может иметь то один тип, то другой. Каким либо специальным образом тип переменной в тексте программы не объявляется. Обычно переменным присваиваются значения в виде строки символов, заключенных в двойные кавычки, или числа (без кавычек).

Внутреннее представление этих значений соответствует одному из следующих типов, перечисленных в табл. 13.5.

Таблица 13.5. *Типы данных в VBScript*

Тип данных	Примеры	Описание значений

`String` (Строковый или символьный)	`"Привет всем!"`
	`"д.т. 123-4567"`
	`"Сегодня 30.11.2004г."`
		Последовательность символов, заключенная в двойные кавычки

`Integer` (Целый)		Целые числа в диапазоне от –32 768 до 32 767
`Long` (Длинный)		Целые числа в диапазоне от –2 147 483 648 до 2 147 483 647
`Single` (Одинарный)		Вещественные числа с плавающей точкой одинарной точности в диапазоне примерно от –3.4e38 до –1.4e–45 для отрицательных чисел и от 1.4e–45 до 3.4e38 — для положительных
`Double` (Двойной)		Вещественные числа с плавающей точкой двойной точности в диапазоне примерно от –1.8e308 до –4.9e–324 для отрицательных чисел и от 4.9e–324 до 1.8e308 — для положительных
`Byte` (Байт)		Целые числа от 0 до 255
`Currency` (Денежный)		Числа в диапазоне от –922 337 203 685 477.5808 до 922 337 203 685 477.5808
`Boolean` (Логический, булевский)	`True` `False`	`True` (истина, да) или `False` (ложь, нет); возможны только два значения
`Date` (Дата)		Содержит число, представляющее дату от 1.01.100 до 31.12.9999
`Null`	`Null`	Этот тип данных имеет одно значение — `Null`, обозначающее отсутствие какого бы то ни было допустимого значения
`Empty` (Пустой)	`Empty`	Этот тип данных имеет одно значение — `Empty`, обозначающее, что переменная не инициализирована
`Error`		Содержит номер ошибки
`Object` (Объект)		Программный объект, определяемый своими свойствами

Для преобразования данных из одного типа в другой служат специальные функции. Например, числа можно хранить как данные одного из 6 типов. Однако не все эти типы равноценны с точки зрения точности представления чисел и занимаемого места в памяти. Если вы хотите преобразовать число 3.14 в целое, то тип `Byte` окажется более экономным, чем `Integer`. Для приведения числового значения переменной `x` к типу `Integer` используется функция `Cint(x)`, а для приведения к типу `Byte` — функция `CByte(x)`. Перечень функций преобразования типов приведен в *разд. 13.7.1*.

Примеры:

```
15                ' Целое число
-25.67            ' Вещественное число с фиксированной точкой
-28.5e-3          ' Вещественное число с плавающей точкой
"Привет всем!"    ' Строка символов
```

Дату и время можно представить непосредственно с помощью цифр, используя ведущий и заключительный символ `#`. При этом разделителем составляющих даты является либо дефис (`-`), либо прямой слэш (`/`), а разделителем составляющих времени — двоеточие (`:`); дата от времени разделяются пробелом. Однако формат отображения (а не хранения) даты определяется настройками компьютера.

Примеры:

```
#8-12-2004#
#8-12-2004 17:12:36#
#8/12/2004#
#8/12/2004 17:12#
```

Значение `Empty` имеет объявленная переменная, которой еще не присвоено значение. Это значение ведет себя как 0 в операциях над числами и как пустая строка — в операциях над строками. Значение `Empty` нельзя присвоить с помощью оператора присваивания, оно назначается автоматически при создании переменной без присвоения ей начального значения.

Значение `Null` означает отсутствие допустимого значения. Переменная может получить его в результате некоторых операций над ней, в том числе и путем непосредственного присвоения ей этого значения. В отличие от `Empty`, значение `Null` можно присвоить переменной с помощью оператора присваивания.

> ***Внимание!***
> В VBScript, в отличие от JavaScript, строковые значения заключаются только в двойные кавычки. Одинарные кавычки для этой цели не допускаются.

13.3. Переменные и операторы присваивания

Переменная является контейнером для хранения данных. Данные, сохраняемые в переменной, называют *значениями* этой переменной. Переменная имеет имя — последовательность букв, цифр и символа подчеркивания без пробелов и знаков

препинания, начинающаяся обязательно с буквы или символа подчеркивания. Таким образом, имя переменной не должно начинаться с цифры или знака препинания.

Переменную можно объявить с помощью оператора `Dim`:

```
Dim имя_переменной
```

Одним оператором `Dim` можно объявить несколько переменных, например:

```
Dim x, y, myvar
```

Вместе с тем, объявленная оператором `Dim` переменная, которой не присвоено конкретное значение, уже имеет значение `Empty`. Чтобы убедиться в этом, выполните HTML-код из листинга 13.1.

Листинг 13.1. Проверка значения объявленной оператором `Dim` переменной

```
<HTML>
  <SCRIPT LANGUAGE=VBScript>
    Dim x
    MsgBox x=Empty, 0, "Значение x равно Empty?"
  </SCRIPT>
</HTML>
```

В результате выполнения этого HTML-кода браузером Internet Explorer появится диалоговое окно, показанное на рис. 13.3.

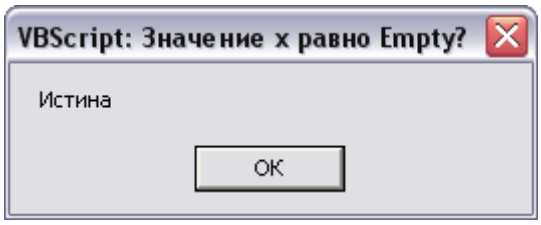

Рис. 13.3. Диалоговое окно с сообщением о проверке значения переменной, которой не присвоено значение

Чтобы присвоить переменной значение, используют следующие выражения:

```
имя_переменной=значение
Let имя_переменной=значение
Set имя_переменной=значение
```

Оператор `Let` не является обязательным для присвоения значения переменной. Оператор `Set` используется для присвоения переменной ссылки на объект, в том числе и при создании нового объекта с помощью специальной функции `CreateObject()`.

Примеры:

```
x="Привет всем!"
Let NumberOfDays=365

Set MyTextBox=txtcontrol
MyTextBox.Value="Привет всем!"
```

Вместо оператора `Dim` можно использовать операторы `Private` и `Public` для объявления соответственно локальных (частных) и глобальных переменных.

13.4. Массивы

Оператор `Dim` может использоваться не только для создания обычных переменных, но и для задания массивов. Массив может быть *статическим* (с постоянным количеством элементов) или *динамическим* (с заранее неопределенным количеством элементов). Массивы могут быть *одномерными* и *многомерными* (до 60 измерений).

Одномерный статический массив определяется следующим образом:

```
Dim имя_массива(n)
```

Здесь `n` — количество элементов массива минус 1. Индексация элементов массива начинается с 0. Таким образом, число `n` в определении массива — индекс последнего элемента.

Многомерный статический массив задается аналогичным образом, но в круглых скобках через запятую указываются количества элементов минус 1 по каждому измерению. Например, следующий оператор задает двумерный массив из 8 строк и 3 столбцов:

```
Dim myarray(7, 2)
```

Для получения значения элемента массива используется выражение вида:

```
имя_массива(индексы)
```

Например, для получения значения элемента двумерного массива `myarray`, находящегося в 3-й строке и 2-м столбце, можно воспользоваться таким выражением:

```
myarray(2, 1)
```

В уже заданном статическом массиве нельзя изменить количество элементов и размерностей. В динамическом массиве, напротив, это можно делать сколько угодно раз. Динамический массив задается аналогично статическому, но в круглых скобках ничего не указывается:

```
Dim имя_массива()
```

Динамический массив отличается от статического тем, что позволяет устанавливать и изменять количество элементов в программе по мере необходимости. В случае многомерных массивов количество элементов можно изменить только для последнего измерения.

После определения динамического массива можно использовать оператор назначения его длины (количества элементов):

```
Redim имя_массива(n)
```

Здесь `n` — количество элементов массива минус 1.

Пример:

```
Dim myarray()       ' Определение динамического массива
Redim myarray(1)    ' Установка длины массива, равной 2
' Определение элементов массива
myarray(0)="Вася"
myarray(1)="Маша"
' Вывод строки со значениями элементов массива: "Вася Маша"
MsgBox myarray(0)  + " " + myarray(1)
```

Если в программе изменяется длина массива, то значения его уже определенных элементов могут быть потеряны.

Пример:

```
Dim myarray()         ' Определение динамического массива
Redim myarray(1)      ' Установка длины массива, равной 2
' Определение элементов массива
myarray(0)="Вася"
myarray(1)="Маша"
Redim   myarray(2)    ' Установка новой длины массива, равной 3
myarray(2)=3.14       ' Установка значения 3-го элемента массива
' Вывод строки со значениями элементов массива: строка пуста
MsgBox myarray(0)  + " " + myarray(1)
```

Для сохранения значений динамического массива при изменении его длины используется ключевое слов `Preserve` (предохранить):

```
Redim Preserve имя_массива(n)
```

Пример:

```
Dim myarray()         ' Определение динамического массива
Redim myarray(1)      ' Установка длины массива, равной 2
' Определение элементов массива
myarray(0)="Вася"
myarray(1)="Маша"
Redim Preserve myarray(2)     ' Установка новой длины массива, равной 3
myarray(2)=3.14               ' Установка значения 3-го элемента массива
' Вывод строки со значениями элементов массива: "Вася Маша 3.14"
 MsgBox myarray(0) + " " + myarray(1) + " " + Cstr(myarray(2))
```

При работе с массивами, особенно динамическими, часто требуется знать их длину. Для этой цели служат две функции:

- `Lbound` — возвращает наименьший индекс для данного массива;

- `Ubound` — возвращает наибольший индекс массива. Количество элементов всегда на 1 больше этого значения.

В следующем примере длина существующего массива `myarray` увеличивается на 1 с сохранением всех его уже имеющихся значений:

```
Redim Preserve myarray(Ubound(myarray) + 1)
```

13.5. Константы

Переменные, значения которых нельзя изменять, называются *константами*. Лучше сказать, что константы — это именованные и неизменяемые значения.

Константа задается следующим оператором:

```
Const имя_константы=значение
```

Имя константы задается так же, как и имя переменной.

В VBScript имеется большое количество предопределенных констант, некоторые из них приведены в табл. 13.6—13.13.

Таблица 13.6. *Константы системных цветов*

Константа	Значение	Цвет
vbBlack	&h00	Черный
vbRed	&hFF	Красный
vbGreen	&hFF00	Зеленый
vbYellow	&hFFFF	Желтый
vbBTue	&hFF0000	Голубой
vbMagenta	&hFF00FF	Малиновый
vbCyan	&hFFFF00	Циан
vbWhite	&hFFFFFF	Белый

Таблица 13.7. *Константы сравнения*

Константа	Значение	Описание
vbBinaryCompare	0	Сравнение двоичных чисел
vbTextCompare	1	Сравнение текста
vbOatabaseCompare	2	Сравнение той части информации в базе данных, где эта константа была применена

Таблица 13.8. *Константы даты и времени*

Константа	Значение	Описание
vbSunday	1	Воскресенье
vbMonday	2	Понедельник
vbTuesday	3	Вторник
vbWednesday	4	Среда
vbThursday	5	Четверг
vbFrlday	б	Пятница
ybSaturday	7	Суббота
vbFirstJan1	1	Неделя, когда наступило 1 января текущего года
vbFirstFourDays	2	Первая неделя, в которой было, как минимум, 4 дня нового года
vbFlrstFullWeek	3	Первая полная неделя года
vbUseSystem	0	Использование формата установок даты и времени, принятой для этого компьютера
vbUseSystemDayOfWeek	0	Использовать установленный на данном компьютере первый день недели

Таблица 13.9. *Константы форматов даты*

Константа	Значение	Описание
vbGeneralDate	0	Показывает дату и/или время в формате, принятом на данном компьютере. Для вещественных чисел указывает дату и время. Для целых — только дату. Для чисел, меньших 1, отображается только время
vbLongDate	1	Показывает дату в полном виде в формате, принятом на данном компьютере
vbShortDate	2	Показывает дату в кратком виде в формате, принятом на данном компьютере
vbLongTime	3	Показывает время в полном виде в формате, принятом на данном компьютере
vbShortTime	4	Показывает время в кратком виде в формате, принятом на данном компьютере

Таблица 13.10. *Константы для работы с содержимым файла*

Константа	Значение	Описание
`ForReading`	1	Открывает файл только для чтения
`ForWritIng`	2	Открывает файл для записи. Если существует файл с таким же названием, сначала очищает его содержимое
`ForAppending`	8	Открывает файл для записи в конец

Таблица 13.11. *Строковые константы*

Константа	Значение	Описание
`vbCr`	`Chr(13)`	Возврат каретки
`vbCrLf`	`Chr(13) & Chr(10)`	Возврат каретки и перевод строки
`vbLf`	`Chr(10)`	Перевод строки
`vbNewLine`	Знак перевода строки, соответствующий данной платформе	
`vbNullChar`	`Chr(0)`	Символ, имеющий значение 0
`vbNullString`	Строка, имеющая нулевое значение (это может быть не только пустая строка)	
`vbTab`	`Chr(9)`	Горизонтальная табуляция

Таблица 13.12. *Логические константы*

Константа	Значение	Описание
`TristateTrue`	1	Истина
`TristateFalse`	0	Ложь
`TristateUseDefault`	2	Значение, используемое по умолчанию

Таблица 13.13. *Константы типа переменной*

Константа	Значение	Описание
`vbEmpty`	0	Не определена (по умолчанию)
`vbNull`	1	Не содержит корректных данных
`vblnteger`	2	Целое число
`vbLong`	3	Длинное целое

vbSingle	4	Вещественное обычной точности
vbDouble	5	Вещественное двойной точности
vbCurrency	6	Переменная в принятом формате для валюты
vbDate	7	Переменная в принятом формате для даты
vbStnng	8	Строковая переменная
vbObject	9	Объект
vbError	10	Переменная в принятом формате для ошибки
vbBoolean	11	Логическая переменная
vbVariant	12	Признак (используется только для массива признаков)
vbDataObject	13	Объект для доступа к данным
vbDecimal	14	Десятичное число
vbByte	17	Байт
vbArray	8912	Массив

13.6. Операторы

В программах на VBScript каждое выражение обычно размещается в отдельной строке и не завершается никаким специальным символом окончания. При необходимости расположить в одной строке несколько выражений последние разделяются двоеточием. Запись одного выражения (обычно очень длинного) можно перенести на другую строку. При этом используется знак продолжения — пробел и следующий за ним символ подчеркивания.

> ***Примечание***
> В JavaScript одиночное выражение в одной строке можно завершать, а можно и не завершать точкой с запятой. Несколько выражений в одной строке разделяются точкой с запятой. При переносе записи выражения на другую строку знаки продолжения не предусмотрены.

13.6.1. Комментарии

Оператор комментария позволяет вставить в программу неинтерпретируемый текст, служащий лишь программисту. Он предваряется одинарным апострофом (одинарной верхней кавычкой). Все, что расположено правее его, является комментарием, а не кодом программы. Комментарий может располагаться с начала строки или в строке с программным кодом, но правее его. Комментарии могут быть многострочными. В этих случаях каждая строка комментария должна начинаться с апострофа.

Пример:

```
' Пример программы,
' которая выводит диалоговое окно
x="Привет"      ' Это текст сообщения
MsgBox x        ' Открытие диалогового окна с сообщением
```

Кроме того, для выделения комментария допустимо использование устаревшего оператора `Rem`.

> ***Примечание***
> В JavaScript для однострочных комментариев используются символы `//`, а для многострочных — `/*...*/`.

13.6.2. Арифметические операторы

Применение арифметических операторов (табл. 13.14) к числовым данным подчиняется правилам математики. Однако они могут быть применены и к данным других типов. В частности, оператор сложения + для строковых данных выполняет их склейку, т. е. приписывает второй операнд к концу первого.

Таблица 13.14. *Арифметические операторы*

Оператор	Название	Пример
+	Сложение	X+Y
-	Вычитание	X-Y
*	Умножение	X*Y
/	Деление	X/Y
^	Возведение в степень	X^Y

13.6.3. Операторы сравнения

Результатом вычисления элементарного выражения, содержащего оператор сравнения (табл. 13.15) и операнды (сравниваемые данные), является логическое значение, т. е. `True` или `False`. Так, если условие выполняется (верно, справедливо), то возвращается `True`. В противном случае возвращается `False`.

Таблица 13.15. *Операторы сравнения*

Оператор	Название	Пример
=	Равно	X=Y
<>	Не равно	X<>Y
>	Больше, чем	X>Y
>=	Больше или равно (не меньше)	X>=Y

<	Меньше, чем	X<Y
<=	Меньше или равно (не больше)	X<=Y

Обратите внимание, что операторы равенства и присваивания обозначаются одинаково. Интерпретатор VBScript различает их по контексту применения в выражении.

Примечание

В JavaScript для оператора равенства используются два подряд следующих символа =, а для оператора присваивания — одинарный символ =.

13.6.4. Логические операторы

Логические данные, обычно получаемые с помощью элементарных выражений, содержащих операторы сравнения, можно объединять в более сложные выражения. Для этого используются логические (булевские) операторы (табл. 13.16).

Таблица 13.16. *Логические операторы*

Оператор	Название	Пример
Not	Отрицание (НЕ)	!X
And	И	X And Y
Or	ИЛИ	X Or Y
Xor	Исключающее ИЛИ	X Xor Y
Eqv	Эквивалентность	X Eqv Y
Imp	Импликация	X Imp Y

Логические выражения принимают значения `True` (истина) или `False` (ложь). Смысл первых трех операторов такой же, как и операторов `!`, `&&` и `||` в JavaScript *(см. разд. 11.7.5)*. Действие остальных операторов можно выразить, используя булеву алгебру, через НЕ, И и ИЛИ так, как показано в табл. 13.17.

Таблица 13.17. *Выражения, эквивалентные логическим операторам*

Оператор	Эквивалентное выражение
X Imp Y	Not X Or Y
X Eqv Y	(X Imp Y) And (Y Imp X)
X Xor Y	(X And Not Y) Or (Not X And Y)

Примечание

В JavaScript нет операторов исключающего ИЛИ, эквивалентности и импликации. Однако нетрудно написать собственные функции, которые будут вычислять соответствующие выражения.

13.6.5. Строковые операторы

К строковым данным применим оператор *склейки (конкатенации)*. В результате действия этого оператора к концу первой строки приписывается вторая строка. В качестве символа оператора склейки строк можно использовать + и &.

Пример:

```
x="Саша"
MsgBox x + " и " + "Маша"       ' Окно с сообщением "Саша и Маша"
MsgBox x & " любит " & "Машу"   ' Окно с сообщением "Саша любит Машу"
```

Примечание

В JavaScript оператор сложения, примененный к строке и, например, числу, возвращает строковое значение. Поэтому преобразование чисел в строки можно выполнять как прибавление числа к пустой строке (`"" + число`). В VBScript подобное выражение приведет к ошибке "Несоответствие типа". В то же время выражение вида `"число1" + число2` дает в результате сумму двух чисел. Далее, выражение вида `"" + "число1" + число2` не приводит к ошибке и возвращает сумму чисел. Это происходит от того, что VBScript старается преобразовать операнды смешанных типов в числовой тип, но при этом пустые строки и строки с одними пробелами он не превращает в 0. Выражение `"" + "число1" + число2` выполняется слева направо и, следовательно, сначала получается строка, содержащая *число1*, а затем число, равное сумме двух чисел.

13.6.6. Операторы условного перехода

Оператор *If*

Оператор условного перехода `If` позволяет реализовать структуру условного выражения

```
если ..., то ..., иначе ...
```

Оператор `If` имеет несколько вариантов синтаксиса:

```
If условие Then выражение

If условие Then
  выражения
End If

If условие_1 Then
  [выражения_1]
[ElseIf условие_2 Then
  [выражения_2]] ...
```

```
[Else
  [выражения_n]]
End If
```

Здесь квадратные скобки указывают лишь на то, что заключенные в них элементы синтаксической конструкции не являются обязательными. Обратите внимание, что блоков вида `ElseIf` может быть сколько угодно, в то время как блок вида `Else` может быть использован не более одного раза.

Оператор *Select Case*

Когда требуется вычислить значение выражения и сравнить его со значениями из заданных списков, удобно использовать оператор `Select Case`:

```
Select Case тестируемое_выражение
  [Case список_значений_1
    [выражения_1]]
  [Case список_значений_2
    [выражения_2]]
  ...
  [Case список_значений_n
    [выражения_n]]
  [Case Else
    [выражения_n+1]]
End Select
```

Здесь квадратные скобки указывают лишь на то, что заключенные в них элементы синтаксической конструкции не являются обязательными. Список значений может состоять из одного или нескольких элементов. В последнем случае они разделяются запятыми.

С помощью оператора `Select Case` вычисляется значение тестируемого выражения, которое последовательно сравнивается со значениями из списков блоков `Case`. Если значение тестируемого выражения совпадает с каким-нибудь значением из списка какого-нибудь блока `Case`, то выполняются выражения, соответствующие этому блоку. Если значение тестируемого выражения входит в несколько списков, то выполняются выражения самого первого блока `Case`. После выполнения выражений блока `Case` управление вычислительным процессом передается оператору, непосредственно следующему за `End Select`. Если ни один из списков значений, заданных в блоках `Case`, не содержит значения тестируемого выражения, то выполняются выражения блока `Case Else` (при его наличии). Если блок `Case Else` отсутствует, то управление передается оператору, непосредственно следующему за `End Select`.

Пример:

```
x=InputBox("Сколько Вам лет?")
Select Case x
  Case Empty
```

```
    MsgBox "Вы ничего не ввели"
  Case 20,21,22,23,24,25
    MsgBox "У Вас все впереди"
  Case 40,41,42,43,44,45
    MsgBox "Вы достигли главного"
  Case Else
    MsgBox "Вы ищете свой путь в жизни"
End Select
```

Примечание

В JavaScript близким по смыслу к `Select Case` является оператор `switch`.

13.6.7. Операторы цикла

Оператор цикла обеспечивает многократное выполнение блока программного кода до тех пор, пока не выполнится некоторое условие. В VBScript имеются несколько операторов цикла.

Оператор *For ... Next*

Оператор цикла со счетчиком циклов имеет следующий синтаксис:

```
For счетчик=начальное_значение To конечное_значение [Step приращение]
  выражения
Next
```

Здесь *счетчик* — просто имя переменной. При выполнении оператора цикла сначала переменной *счетчик* присваивается начальное значение. Если это значение не равно заданному конечному значению, то выполняются выражения, указанные в теле оператора цикла. В противном случае выражения не вычисляются, а управление передается оператору, непосредственно следующему за `Next`. Затем значение счетчика увеличивается на величину приращения, указанную после ключевого слова `Step` (если его нет, то на 1). Новое значение счетчика сравнивается с конечным значением и далее все повторяется описанным выше способом.

Пример:

```
' Заполнение массива квадратами первых десяти натуральных чисел
Dim myarray(9)
For i=1 To 10
  myarray(i-1)=x^2
Next
```

Если необходимо прекратить выполнение оператора цикла до достижения счетчиком конечного значения, то можно использовать оператор выхода из цикла `Exit For`.

Примечание

В JavaScript оператором выхода из цикла является `break`.

Оператор *For Each ... Next*

Оператор `For Each...Next` позволяет организовать цикл по элементам массива или по объектам из некоторого множества, когда их количество заранее не известно. Этот оператор выполняет заданные выражения для каждого элемента из указанного множества. Он имеет следующий синтаксис:

```
For Each элемент In множество
  выражения
Next
```

Параметр *множество* задает имя массива или коллекцию объектов. Параметр *элемент* — переменная, которая в процессе выполнения оператора цикла автоматически принимает в качестве значения ссылку на элемент массива или множества объектов.

Пример:

```
' Определение длины массива
Dim myarray(25)
Count=0
For Each i In myarray
  Count=Count + 1
Next
```

В листинге 13.2 приведен HTML-документ, в котором определена форма, состоящая из поля ввода данных и кнопки. Сценарий выводит на страницу перечень имен всех форм (в примере только одна форма) и список имен элементов форм и значений атрибутов `VALUE`. Внешний вид этого документа показан на рис. 13.4.

Листинг 13.2. Список элементов форм

```
<HTML>
  <HEAD><TITLE>Список элементов форм</TITLE></HEAD>
  <FORM NAME=myform>
    <INPUT TYPE=TEXT NAME=field VALUE ="Привет">
    <P>
    <INPUT TYPE="BUTTON" NAME=mybutton VALUE="Щелкни здесь"
  </FORM>
  <P>
  <SCRIPT LANGUAGE=VBS>
    For Each i In document.forms
      document.write("Форма " + i.name + "<br>")
      For Each j In i.elements
        document.write("Элемент " + j.name + " - " + j.value + "<br>")
      Next
    Next
  </SCRIPT></HTML>
```

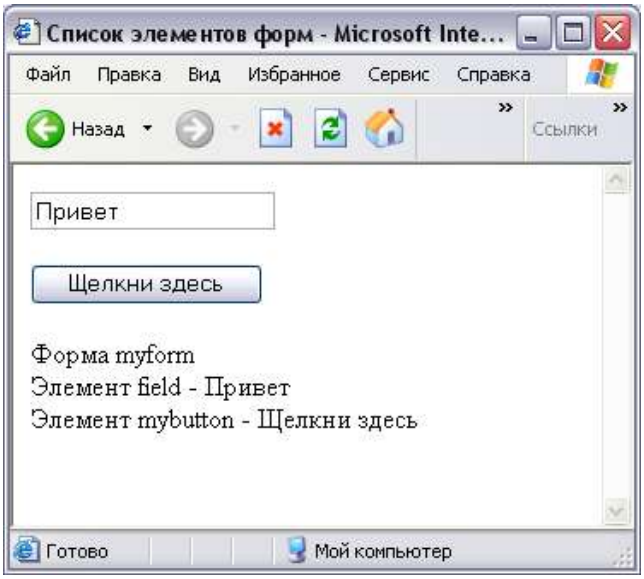

Рис. 13.4. Элементы формы и отображение сведений о них

В листинге 13.3 приведен код, отличающийся от предыдущего только тем, что сценарий выводит на страницу перечень имен всех тегов (элементов коллекции `all` объекта `document`). Внешний вид этого HTML-документа показан на рис. 13.5.

Листинг 13.3. Список элементов документа

```
<HTML>
  <HEAD><TITLE>Список элементов документа</TITLE></HEAD>
  <FORM NAME=myform>
    <INPUT TYPE=TEXT NAME=field VALUE ="Привет">
    <P>
    <INPUT TYPE="BUTTON" NAME=mybutton VALUE="Щелкни здесь"
  </FORM>
  <P>
  <SCRIPT LANGUAGE=VBS>
    For Each i In document.all
      document.write(i.tagname+"<br>")
    Next
  </SCRIPT>
</HTML>
```

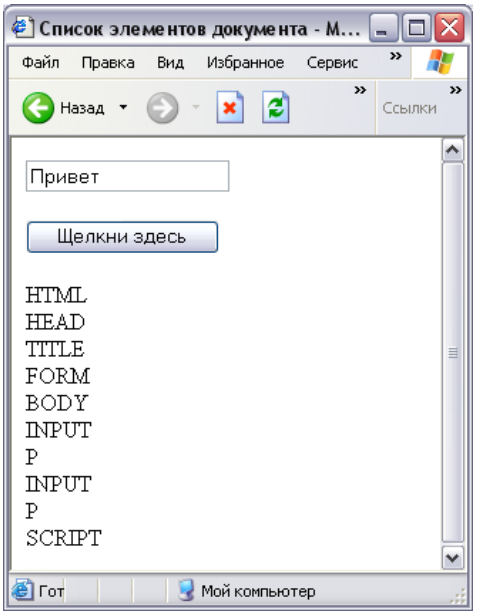

Рис. 13.5. Элементы формы и отображение имен всех тегов документа

Если необходимо прекратить выполнение оператора цикла до окончания перебора всех элементов заданного множества, то можно использовать оператор выхода из цикла `Exit For`.

Операторы *Do ... Loop*

Конструкция `Do...Loop` выполняет выражения кода до тех пор, пока условие истинно или, наоборот, ложно (в зависимости от конструкции). Возможно несколько вариантов этого оператора цикла:

- `Do While` *условие*

  выражения

  `Loop`

  Выражения в теле этого оператора выполняются до тех пор, пока истинно *условие* продолжения. Возможен случай, когда выражения в теле оператора цикла ни разу не выполнятся (условие продолжения сразу оказалось ложным). Если условие продолжения постоянно истинно, цикл будет повторяться бесконечно.

  Пример:

```
Do While strDayOfWeek <> "Суббота" And strDayOfWeek <> "Воскресенье"
  MsgBox ("Вставай, пора работать!")
  ...
Loop
```

- `Do`

  выражения

  `Loop While` *условие*

  Выражения в теле этого оператора цикла будут выполнены хотя бы один раз, после чего определяется, следует ли продолжать их выполнение. Работа цикла продолжится если условие истинно. Не исключен случай зацикливания (бесконечного повторения).

- `Do Until` *условие*

  выражения

  `Loop`

  Этот оператор аналогичен `Do While...Loop` за исключением того, что выражения будут выполняться, пока условие ложно (здесь `until` — до тех пор, пока не).

- `Do`

  выражения

  `Loop Until` *условие*

  Этот оператор аналогичен `Do...Loop While...` за исключением того, что выражения будут выполняться, пока условие продолжения ложно.

Оператор *While*

Оператор цикла `While` имеет следующий синтаксис:

```
While условие
  выражения
Wend
```

Пока условие истинно, выражения в теле цикла будут выполняться.

13.7.Функции и процедуры

Функции и процедуры представляют собой подпрограммы (блоки кода), которые можно вызвать для выполнения, обратившись к ним по имени. Функция может возвращать, а может и не возвращать некоторое значение. Если она возвращает значение, то может участвовать в выражениях с операторами. Процедура отличается от функции только тем, что не возвращает никаких значений и поэтому не может участвовать в других выражениях.

Вызовы функций и процедур осуществляются путем указания их имени. Если им передаются параметры, то они указываются через запятую после имени функции или процедуры. Особенность VBScript состоит в том, что возможны два варианта указания списка параметров:

- после имени через пробел, не заключая его в круглые скобки, например,

  ```
  myfunc prm1, prm2, prm3
  ```

- сразу после имени в круглых скобках, например,

```
myfunc(prm1, prm2, prm3)
```

Это же в полной мере относится и к использованию методов объектов. Например, следующие две записи эквивалентны:

```
document.write "<h1>Привет</h1>"
document.write("<h1>Привет</h1>")
```

Если функция возвращает значение и необходимо использовать ее в выражении, то обязательно применяется скобочный синтаксис, как в следующем примере:

```
x = 2 + sqr(9)      ' Переменной x присваивается число 5
```

13.7.1. Встроенные функции

В VBScript имеется большое количество встроенных функций, которые можно разделить на несколько категорий:

- функции ввода и вывода данных;
- функции преобразования;
- функции даты/времени;
- математические функции;
- функции управления объектами;
- функции инициализации интерпретатора кода сценария;
- строковые функции;
- функции проверки переменных.

Функции

В этом разделе ограничимся лишь перечислением имен и назначений функций.

- Функции преобразования (табл. 13.18).

  Эти функции используются для преобразования значений переменных разных типов.

Таблица 13.18. *Функции преобразования*

Функция	Описание
`Asc`	Возвращает кодовый номер ANSI первого символа в строке
`AscB`	То же, но используется для обработки однобайтовых данных. Возвращает результат для первого байта
`AscW`	То же, но используется для символов Unicode. Возвращает символ

	кода `Wide`, что позволяет преобразование из Unicode в ANSI
`Chr`	Возвращает строку символов с соответствующими номерами
`ChrB`	То же, но используется для обработки однобайтовых данных, содержащихся в строке. Возвращает всегда один байт
`ChrW`	То же, но используется для символов Unicode. Возвращает символ кода `Wide`, что позволяет преобразование из Unicode в ANSI
`Cbool`	Возвращает аргумент, преобразованный в логический тип (`Boolean`)
`Cbyte`	Возвращает аргумент, преобразованный в тип "байт" (`Byte`)
`Cdate`	Возвращает аргумент, преобразованный в тип даты (`Date`)
`CDbl`	Возвращает аргумент, преобразованный в числовой тип двойной точности (`Double`)
`Cint`	Возвращает аргумент, преобразованный в целочисленный тип (`Integer`)
`CLng`	Возвращает аргумент, преобразованный в тип длинного целого (`Long`)
`CSng`	Возвращает аргумент, преобразованный в числовой тип обычной точности (`Single`)
`CStr`	Возвращает аргумент, преобразованный в строковый тип (`String`)
`Fix`	Возвращает целую часть числа
`Hex`	Возвращает строку, представляющую собой аргумент в шестнадцатеричной системе счисления
`Int`	Возвращает целую часть числа
`Oct`	Возвращает строку, представляющую собой аргумент в восьмеричной системе счисления
`Round`	Возвращает число, округленное до заданного числа десятичных позиций
`Sgn`	Возвращает целое число, отражающее знак аргумента

- Функции даты/времени (табл. 13.19).

  Эти функции возвращают текущие дату и время или управляют существующими значениями.

Таблица 13.19. *Функции даты/времени*

Функция	Описание
`Date`	Возвращает текущую системную дату
`DateAdd`	Возвращает дату с добавленным заданным временны́м интервалом
`DateDiff`	Возвращает число дней, недель или лет между двумя заданными датами
`DatePart`	Возвращает только день, месяц или год заданной даты
`DateSerial`	Возвращает значение в формате `Date` для заданных года, месяца и дня
`DateValue`	Возвращает значение в формате `Date`
`Day`	Возвращает число от 1 до 31, отражающее день месяца
`Hour`	Возвращает число от 0 до 23, отражающее час дня
`Minute`	Возвращает число от 0 до 59, отражающее минуты
`Month`	Возвращает число от 1 до 12, отражающее месяц в году
`MonthName`	Возвращает название данного месяца в виде строки
`Now`	Возвращает текущие дату и время
`Second`	Возвращает число от 0 до 59, отражающее секунды
`Time`	Возвращает значение в формате `Date` для текущего системного времени
`TimeSerial`	Возвращает значение в формате `Date` для заданных часа, минуты и секунды
`TimeValue`	Возвращает значение в формате `Date`, содержащее время
`Weekday`	Возвращает число, отражающее день недели
`WeekdayName`	Возвращает название данного дня недели в виде строки
`Year`	Возвращает число, отражающее год

- Математические функции (табл. 13.20).

  Эти функции осуществляют математические операции над переменными, содержащими численные значения.

Таблица 13.20. *Математические функции*

Функция	Описание
`Atn`	Возвращает арктангенс числа
`Cos`	Возвращает косинус угла
`Exp`	Возвращает число e (~2.71828) в заданной степени (экспонента)
`Log`	Возвращает натуральный логарифм числа
`Randomize`	Инициализирует генератор случайных чисел
`Rnd`	Возвращает случайное число
`Sin`	Возвращает синус угла
`Sqr`	Возвращает квадратный корень из числа
`Tan`	Возвращает тангенс угла

Примечание

В JavaScript математические функции представлены как методы объекта `Math` и вызываются посредством выражения вида `Math.`*`функция`*`(`*`параметры`*`)`.

- Функции управления объектами (табл. 13.21).

  Эти функции используются для управления объектами там, где это нужно.

Таблица 13.21. *Функции управления объектами*

Функция	Описание
`CreateObject`	Создает и возвращает ссылку на объект ActiveX или OLE Automation
`GetObject`	Возвращает ссылку на объект ActiveX или OLE Automation
`LoadPicture`	Возвращает графический объект

- Функции инициализации интерпретатора кода сценария (табл. 13.22).

  Эти функции возвращают версию интерпретатора кода сценария.

Таблица 13.22. *Функции инициализации интерпретатора кода сценария*

Функция	Описание
`Script Engine`	Строка, содержащая основной, дополнительный номера версии и номер разработки интерпретатора
`ScriptEngineMajorVersion`	Основной номер версии интерпретатора
`Scri ptEngineMinorVersion`	Дополнительный номер версии интерпретатора

`ScriptEngineBundVersion`	Номер разработки интерпретатора

- Строковые функции (табл. 13.23).

  Эти функции используются для управления строковыми значениями переменных.

Таблица 13.23. *Строковые функции*

Функция	Описание
`Filter`	Возвращает массив, отобранный по заданному критерию из заданного массива строк
`FormatCurrency`	Возвращает строку, отформатированную для представления денежных сумм
`FormatDateTime`	Возвращает строку, отформатированную для представления даты и времени
`FormatNumber`	Возвращает строку, отформатированную как число
`FormatPercent`	Возвращает строку, отформатированную для представления процентного отношения
`InStr`	Возвращает место первого появления одной строки
`InStrB`	То же, но используется для работы с отдельными байтами. Возвращает положение байта, а не символа
`InStrRev`	То же, что и `InStr`, но начинается с конца строки
`Join`	Возвращает строку, образованную соединением всех строк массива
`Lcase`	Возвращает строку, все символы которой преобразованы в строчные
`Left`	Возвращает заданное число символов от левого конца строки
`LeftB`	То же, но для работы с байтами. Вместо числа символов — число байтов
`Len`	Возвращает длину строки или количество байтов, необходимое для переменной
`LenB`	То же, но для работы с байтами. Вместо числа символов — число байтов
`Ltrim`	Возвращает копию строки без начальных пробелов
`Mid`	Возвращает заданное количество символов из строки
`MidB`	То же, но для работы с байтами. Вместо числа символов — число байтов
`Replace`	Возвращает строку, в которой одна заданная последовательность заменена другой указанное число раз

Right	Возвращает заданное число символов от правого конца строки
RightB	То же, но для работы с байтами. Вместо числа символов — число байтов
Rtrim	Возвращает копию строки без завершающих пробелов
Space	Возвращает строку, состоящую из заданного количества пробелов
Split	Возвращает одномерный массив, состоящий из заданного числа подстрок
StrComp	Возвращает значение, отражающее результат сравнения строк
String	Возвращает строку заданной длины, образованную повторением одного символа
StrReverse	Возвращает "отраженную" строку, в которой порядок символов противоположен исходной
Trim	Возвращает копию строки без ограничивающих пробелов
Ucase	Возвращает строку, все символы которой преобразованы в прописные

- Функции проверки переменных (табл. 13.24).

  Эти функции используются для определения типа информации, находящейся в переменной.

Таблица 13.24. *Функции проверки переменных*

Функция	**Описание**
IsArray	Возвращает логическое значение (True/False), отражающее, является ли переменная массивом
IsDate	Возвращает логическое значение (True/False), отражающее, может ли значение переменной быть преобразовано в формат даты/времени
IsEmpty	Возвращает логическое значение (True/False), отражающее, инициализирована ли переменная
IsNull	Возвращает логическое значение (True/False), отражающее, содержит ли переменная некорректные данные
IsNumeric	Возвращает логическое значение (True/False), отражающее, является ли значение переменной числовым
IsObject	Возвращает логическое значение (True/False), отражающее, является ли переменная ссылкой на действующий объект ActiveX или OLE Automation
VarType	Возвращает число, отображающее номер типа переменной

- Обработка ошибок.
  - `On Error Resume Next` — указывает, что при появлении ошибки необходимо ее игнорировать и начать дальнейшую обработку кода со следующей строки.
  - `Err` — объект `error`, содержащий информацию об ошибках выполнения.

  Возможности для обработки ошибок в VBScript ограничены, и для выяснения, произошла ли ошибка, необходимо явно проверять объект `Err`.

13.5.2. Пользовательские функции и процедуры

Многократно используемые блоки программного кода целесообразно оформлять в виде функций или процедур.

Определение функции имеет следующий синтаксис:

```
Function имя_функции([параметры])
  выражения
  [имя_функции=значение]
End Function
```

Функция может не иметь параметров. Чтобы функция возвращала некоторое значение, используется оператор присваивания *имя_функции=значение*. Если он не задан, то функция ничего не будет возвращать.

Пример:

```
Function Srectangle(a, b)
  Srectangle=a*b
End Function
```

Определение процедуры имеет следующий синтаксис:

```
Sub имя_процедуры([параметры])
  выражения
End Sub
```

В отличие от функции процедура никогда ничего не возвращает, и поэтому для нее не предусмотрена конструкция определения возвращаемого значения.

Пример:

```
Sub ToDay(xdate)
  Dim xstr
  xstr="Сегодня "
  If xdate =Empty Then
    xstr=xstr + Cstr(Now()
  Else
    xstr=xstr + Cstr(xdate)
  End If
  MsgBox(xstr)
```

```
End Sub
```

Вызов функций и процедур осуществляется по имени. Список параметров, если он имеется, следует за именем. Он может заключаться в круглые скобки или следовать через пробел без скобок.

Примеры:

```
Srectangle(5, 8)     ' Возвращает 40
Srectangle 5, 7+3    ' Возвращает 50
ToDay(#10-11-2004#) ' Выводит окно с сообщением "Сегодня 10.11.2004"
ToDay #10-11-2004#  ' То же самое
ToDay("")           ' Выводит окно с сообщением о текущих дате и времени
```

Для вызова функций и процедур можно использовать оператор `Call`. Однако при этом за именем функции обязательно должна быть указана пара круглых скобок.

Примеры:

```
Call Srectangle(5, 8)
Call ToDay("")
```

Как уже отмечалось выше, вызов функции может участвовать в выражениях, если используется скобочный синтаксис (за именем следует пара круглых скобок) и функция возвращает какое-нибудь значение.

Пример:

```
Dim TotalS, a, b, x, y
a=2
b=5
x=10
y=8.5
TotalS=Srectangle(a, b) + Srectangle(x, y) ' Сумма площадей двух
                                           ' прямоугольников
```

Параметры функциям и процедурам могут передаваться двумя способами: по ссылке и по значению. При передаче по ссылке значение переменной, указанной в качестве параметра, может быть изменено кодом функции или процедуры так, что эта переменная будет иметь новое значение и во внешнем коде. При передаче по значению коду функции или процедуры передается одноименная копия переменной, существующая там как локальная переменная. В этом случае любые ее изменения не отразятся на одноименной переменной во внешнем коде.

Задать способ передачи переменных в качестве параметров можно по-разному. Так, если в вызове функции или процедуры используется скобочная запись, то переменные передаются по значению, а если бесскобочная запись, то по ссылке. В следующих двух примерах переменная `x=1` определена во внешнем по отношению к функции `myfunc` коде и передается ей в качестве параметра. Функция присваивает этой переменной новое значение. Далее во внешней программе происходит вызов функции `myfunc` из диалогового окна `MsgBox`, в котором отображается значение переменной `x`. Коды программ отличаются только способом вызова функции `myfunc`.

Однако именно это различие приводит к тому, что в одном случае переменная `x` во внешней программе изменяет свое значение, а во втором — нет.

- Передача переменной в качестве параметра функции по ссылке:

```
x=1
Function myfunc(x)
  x=2
End Function
myfunc x
MsgBox(x)    ' Выводит значение x, равное 2
```

- Передача переменной в качестве параметра функции по значению:

```
x=1
Function myfunc(x)
  x=2
End Function
myfunc(x)
MsgBox(x)    ' Выводит значение x, равное 1
```

Для передачи переменной в качестве параметра по значению при любой форме записи вызова, перед ее именем в определении функции или процедуры указывают ключевое слово `ByVal`.

Пример:

```
x=1
Function myfunc(ByVal x)
  x=2
End Function
myfunc(x)
MsgBox(x)    ' Выводит значение x, равное 1
myfunc x
MsgBox(x)    ' Выводит значение x, равное 1
```

> ***Примечание***
> В JavaScript способ передачи переменных в качестве параметров задается путем надлежащего использования ключевого слова `var` (оператора объявления переменных).

13.8. Вызов методов объектов и обработчиков событий

Методы объектов в VBScript вызываются так же, как и JavaScript:

объект`.`*метод*`(`*параметры*`)`

Однако возможна запись вызова метода без круглых скобок, подобно вызовам функций и процедур. Например, вызов метода `alert("Привет!")` в сценариях на VBScript возможен в двух вариантах:

```
window.alert("Привет!")
window.alert "Привет!"
```

Обработчики событий, определяемые как методы соответствующих объектов, должны записываться через дефис, а не через точку:

```
объект_событие()
```

В листинге 13.4 показаны определения обработчиков `onfocus()` для окна и `onclick` для кнопки в виде процедуры и функции соответственно.

Листинг 13.4. Использование обработчиков событий

```
<HTML>
  <HEAD><TITLE>Пример</TITLE></HEAD>
  <BUTTON ID=mybt>Щелкни здесь</BUTTON>
  <SCRIPT LANGUAGE=VBScript>
    Sub window_onfocus()
      window.alert("Окно в фокусе")
    End Sub
    Function mybt_onclick()
      MsgBox "Был щелчок на элементе id=" + document.all.mybt.id
    End Function
  </SCRIPT>
</HTML>
```

На рис. 13.6. показан вид этого HTML-документа в браузере после щелчка на кнопке.

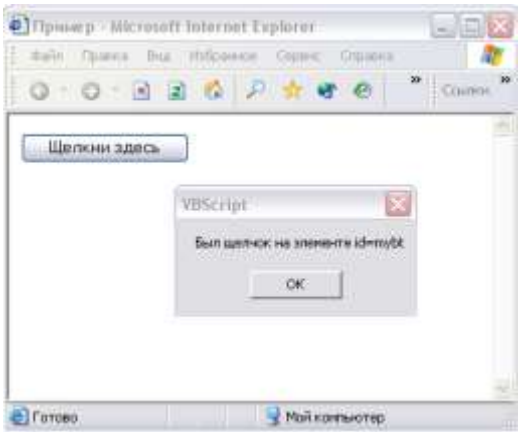

Рис. 13.6. Щелчок на кнопке обработан функцией `mybt_onclick`