Learning

VB Script

TRAININGTOOLS.COM

Share these FREE Courses!

Why stuff your friend's mailbox with a copy of this when we can do it for you!
Just e-mail them the link info – http://www.trainingtools.com
Make sure that you visit the site as well:

- MORE FREE COURSES
- Weekly Tool Tips
- Updated course versions
- New courses added regularly

So don't copy files or photocopy - Share!

End User License Agreement

Use of this package is governed by the following terms:

A. License

TrainingTools.com Inc, ("we", "us" or "our"), provides the Licensee ("you" or "your") with a set of digital files in electronic format (together called "the Package") and grants to you a license to use the Package in accordance with the terms of this Agreement. Use of the package includes the right to print a single copy for personal use.

B. Intellectual Property

Ownership of the copyright, trademark and all other rights, title and interest in the Package, as well as any copies, derivative works (if any are permitted) or merged portions made from the Package shall at all times remain with us or licensors to us. This Package is protected by local and international intellectual property laws, which apply but are not limited to our copyright and trademark rights, and by international treaty provisions.

C. Single-User License Restrictions

1. You may not make copies of the files provided in the Package
2. You may not translate and/or reproduce the files in digital or print format
3. You may not rent, lease, assign or transfer the Package or any portion thereof
4. You may not modify the courseware

Copyrights and Trademarks

Published by

XtraNet

Communications Inc.
180 Attwell Dr., Suite 130
Toronto, Ontario, Canada
M9W 6A9
Phone: 416-675-1881
Fax: 416-675-9217
E-mail: info@xnu.com

Printed in Canada
June 2000
First Edition
1 2 3 4 5 6 7 8

Table of Contents

Table of Contents

1

Introduction to VBScript Programming

This section will provide you with the basics of what VBScript is, and why you would use it.

Objectives

1. Interpreted programs versus Compiled programs
2. Why VBScript?
3. What you can use VBScript for
4. About VBScript

Interpreted programs versus Compiled programs

Before we start discussing the differences between interpreted and compiled, we have to define the term source code, more commonly referred to as code. The code is the plain text commands that the program is written in. All programming languages start out as source code; it is then either interpreted or compiled. The code that you will create in this course can be considered source code.

Interpreted programming languages tend to be simpler to program but slower to execute in general. Each time a program is run, it has to be interpreted (interrogated) line by line, based on the flow of execution (you will see later how branches and loops affect the flow of execution).

Compiled programming languages have a more complex syntax, and require more strict programming practices. With a compiled programming language, you first write the source code, then you feed it to a compiler (a special computer program), which produces an executable binary program. On the Windows platform, the output of the compiler usually ends in the ".exe" file extension. The program that comes out of the compilation process tends to be platform (operating system) specific. The key benefit for the programmer is that no other programmer can look at the source code once it is compiled. The other key factor is that the language used to write the source code becomes irrelevant once it has been compiled.

Visual Basic is a compiled language, whereas VBScript is an interpreted language.

Why Learn VBScript

VBScript is used to create Active Server Pages (ASPs), to create administration scripts for Windows 95/98/NT, to extend or enhance the functionality of the Microsoft Office products (like Word and Excel (macros)). It can also be used as a client side scripting language for Internet Explorer. Netscape does **NOT** support VBScript as a client side scripting language.

About VBScript

VBScript is an interpreted programming language that can be embedded into an HTML web page or used in server side scripting.

Client Side Scripting

VBScript code is executed/interpreted when an event is triggered. When the code is executed it is interpreted one line at a time. There are a number of events that will trigger the execution of a VBScript, like clicking on a form button, or the completion of a web page loading.

Note: Internet Explorer is the only browser that supports VBScript today.

Server Side Scripting

When the web server loads an .asp page from the disk into memory, it automatically knows to interpret the code in this document. Once the code has been interpreted, the resulting HTML page is sent to the browser (client) making the request.

Review Questions

1. (True or False) VBScript is an interpreted language.

2. Visual Basic is a _____ programming language.

3. (True or False) Visual Basic and VBScript were created by the same company.

4. Microsoft Internet Explorer supports the following scripting languages.
   a. VBScript
   b. Visual Basic
   c. BASIC
   d. JavaScript
   e. C++
   f. Perl

5. (True or False) VBScript is supported by a large number of browsers.

Summary

In this module you learned:

1. VBScript is Interpreted, and Visual Basic is Compiled
2. Why you would use VBScript
3. What you can use VBScript for
4. About the VBScript Language

2

VBScript Syntax

In this chapter you will learn about the peculiarities of the VBScript language. These are the details for writing a script that will help you avoid errors while you are creating your own scripts and learning the basics of the VBScript programming language.

Objectives

1. Placing VBScript in an HTML page and an Active Server Page
2. Case-sensitivity
3. Whitespace
4. Brackets
5. Comments
6. Variable and Function Names
7. Reserved Words

Inserting Client Side VBScript into an HTML Page

VBScript is added to an HTML page using the SCRIPT tag. The script tags should be placed inside the head tag of the document. They can appear anywhere in the document; but must be before the event that is going to trigger them. If an older browser looks at a page containing script tags it will ignore them, as older browsers are written to ignore tags they can't interpret.

VBScript code should also be placed inside an HTML Comment tag set.

E.g. **<!--** code **-->**

When used with VBScripts the ending comment tag will also start with two slashes REM which is the VBScript code for comment. This tells the VBScript interpreter to ignore that statement.

This is a standard way for adding VBScript to your HTML pages so that it works properly for browsers that are VBScript enabled and those that do not support VBScript.

```
<HTML>
<HEAD>
<TITLE>Web Page containing VBScript</TITLE>

<SCRIPT LANGUAGE="VBSCRIPT">
<!-- hide VBScript code from browsers that are not VBScript enabled
        .
        .       (VBScript Statements goes here)
        .
REM end hiding of VBScript code -->
</SCRIPT>
</HEAD>

<BODY>
        (HTML document goes here)
</BODY>
</HTML>
```

We may also put in a single line of code attached to an event. Events will be explained later. The general syntax for this structure is:

<HTML_TAG Attribute="option" **onEvent="VBScript code statements go here"**>stuff in between the opening and closing tag</HTML_TAG>

Inserting VBScript into an Active Server Page (.asp)

To create an Active Server Page, the file is normally stored with the .asp extension in a directory on a web server that can process Active Server Pages. You can blend VBScript with normal HTML when creating your Active Server Pages. See below:

```
<%@ Language=VBScript %>
<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
</HEAD>
<BODY>
<%
dim helloString
helloString = "Hello World!"
%>
<P><% Response.Write helloString %></P>
</BODY>
</HTML>
```

In the above helloworld.asp page, the first line instructs the server that the default scripting language used in this page will be VBScript. Internet Information Server (IIS) uses VBScript as the default scripting language if the language directive is omitted from a page.

To turn on the server side VBScript interpreter you use the less than followed by the percent “<%” characters. To turn off the server side VBScript interpreter you use the percent sign followed by the greater than sign “%>”. Both are indicated in bold in the above example.

Syntax and Conventions

Writing in any language follows rules and conventions. For example, the English language requires that each sentence must contain a subject and verb to be legitimate. You must also capitalize the first letter of a sentence, and end each sentence with punctuation such as a period or question mark. This is the syntax, or grammar of the English language. Each programming language has a similar set of rules commonly referred to as the syntax.

VBScript has several rules and conventions for its syntax:

Case-sensitivity:

> VBScript is a case-insensitive language, meaning that the language will treat these words the same: example, Example, EXAMPLE

White Space:

> VBScript, like HTML, ignores spaces, tabs that appear in statements. VBScript does, however recognize spaces, tabs, and newlines that are part of a string. We will talk more about strings later in the course.
>
> x=0 is the same as x = 0
>
> All of these examples will produce the same results. It is a good idea to put some spacing in your code to make it easier to read.
>
> You do need a space between a programming command/function and the data it is working on.

Strings and Quotes:

A string is a sequence of zero or more characters enclosed within single or double quotes ('single',. "double").

The double quotation mark can be found within strings that start, and end with (are delimited by) single quotes ('He said, "VBScript is an interesting language." ').

The single quotation mark can be used within a string delimited by double quotation marks. This syntax will be used quite often through out the book.

For example:

```
<INPUT TYPE="Button" VALUE="Click Me"
 onclick="window.alert('You Clicked me');">
```

In the example above we have a line of HTML code that uses double quotes to delimit the tag's attributes. So to create a popup window that displays the string "You Clicked me" we need to enclose the string within single quotes. This is done so that the entire VBScript statement is interpreted and the HTML syntax also remains valid.

Opening and Closing Brackets:

All brackets you open must be closed!
i.e. winpop = window.open('ex1.htm','popup','scrollbars=yes');

```
if ( x(0) == 10 ) then
        x(0) = 0
        x(1) = 0
end if
```

In the above example “x(0)=0” and “x(1)=0” are two different statements.

The round brackets ( ) are part of a special data structure called arrays. Arrays will be covered later in the course.

The curved brackets ( ) are also used to contain a function or a method’s arguments, multiple arguments are separated by commas. i.e. ('ex1.htm','popup','scrollbars=yes'). Functions and methods will be described shortly.

Comments:

You can create a comment using the REM command or a single quote, like this:

```
REM this is a comment
```

or

```
‘this is a comment
```

Continuation Character

You can continue a VBScript statement on the next line using “_”.

```
If x = 1 or _
  Y = 2 or _
  Z = 3 then
```

Multiple Statements

You can put multiple VBScript statements on one line if they are separated by full colons “:”.

```
X = 0 : y = 1
```

Variable, Subroutine and Function Names

In the next chapter you will be introduced to variables, subroutines and functions. As the programmer you get to choose and assign the names. The names of the variables and functions must follow these simple rules.

1. The first character must be a letter of the alphabet (lowercase or uppercase).
2. The name can contain an underscore "_", but NOT start with an underscore.
3. You CANNOT use a number as the first character of the name.
4. Names CANNOT contain spaces.
5. Names CANNOT match any of the reserved words.

The following are examples of valid names:

x
add_two_num
x13

We recommend that you use descriptive names for your variables and your functions and that you adopt a standard way of naming things. The two formats that are common are; using the underscore to replace spaces, or capitalizing the first letter of complete words after the first word in the name. For example:

add_two_num
addTwoNumbers

Reserved Words

There are a number of words that make up the components of the VBScript language. These words cannot be used for variable or function names because the program interpreter would be unable to distinguish between a default VBScript command and your variable or function name.

ABS	Erase	Lset	Sgn
and	Error	Ltrim	Sin
Array	Exit	Mid	Space
ASC	Exp	Minute	Sqr
ATN	Fix	mod	Static
Call	For	Month	Step
Cbool	Function	Next	Str
Cbyte	Hex	not	StrComp
Cdate	Hour	Nothing	String
CDbl	If	Now	Sub
Chr	imp	Null	Tan
Cint	instr	Oct	Then
Clng	Int	On	Time
Cos	is	or	Timer
Csng	IsArray	Preserve	TimeSerial
Cstr	IsDate	Private	TimeValue
CVErr	IsEmpty	Public	Trim
Date	IsNull	Randomize	Ubound
Dateserial	IsNumeric	ReDim	Ucase
Datevalue	IsObject	Rem	Until
Day	Lbound	Resume	Val
Dim	Lcase	Right	VarType
Do	Left	Rnd	WeekDay
Else	Len	Rset	Wend
Empty	Let	Rtrim	While
End	Log	Second	Xor
eqv	Loop	Set	Year
			FALSE
			TRUE

Review Questions

1. Which of the following are valid variable or function names:
   a. y
   b. 100percent
   c. a big number
   d. public
   e. subtractTwoNumbers
   f. First_Name

2. True or False. VBScript is a case insensitive language.

3. True or False. It is a good idea to add comments to your program code.

Summary

1. VBScript is placed within the <SCRIPT> tags
2. VBScript is case-insensitive
3. VBScript ignores whitespace
4. How and why you should put comments in your program code
5. What names you can use for variables and function names
6. What words are reserved as part of the VBScript language

3

Basic Programming Constructs

In this chapter you will learn the basics constructs of programming. These constructs are similar in a number of programming languages, and will be used in a number of our scripts in later chapters.

Objectives

1. Declaring Variables
2. Using Operators
3. Creating Control Structures (Branches and Loops)
4. Functions (Built-in and programmer-created)

Declaring Your Variables

A variable is a name assigned to a location in a computer's memory to store data. Before you use a variable in a VBScript program, you should declare its name. Variables are declared with the **dim** keyword, like this:

```
dim x
dim y
dim sum
```

You can also declare multiple variables with the same **dim** keyword.

```
dim x, y, sum
```

The initial value of the variable is **empty**, a special value in VBScript.

Dim is short for dimension. When you declare a variable the interpreter needs to allocate memory to hold the contents for you. In strongly typed languages (which VBScript is not) the interpreter or compiler will know what the dimension (size of memory) is required to hold the value is based on the data type. In VBScript no memory is allocated until you store a value in the variable.

Types of Variables

A big difference between VBScript and other languages like Visual Basic and C is that VBScript is *untyped*. This means that a VBScript variable can hold a value of any data type, and its data type does not have to be set when declaring the variable. This allows you to change the data type of a variable during the execution of your program, for example:

```
dim x
x = 10
x = "ten"
```

In this example the variable x is first assigned the integer value of 10, and then the string value of the word ten.

The following is a list of data types supported by VBScript

Datatype	Description
Empty	**Variant** is uninitialized. Value is either 0 for numeric variables or a zero-length string ("") for string variables.
Null	**Variant** intentionally contains no valid data.
Boolean	Contains either **True** or **False**.
Byte	Contains integer in the range 0 to 255.
Integer	Contains integer in the range -32,768 to 32,767.
Currency	-922,337,203,685,477.5808 to 922,337,203,685,477.5807.
Long	Contains integer in the range -2,147,483,648 to 2,147,483,647.
Single	Contains a single-precision, floating-point number in the range -3.402823E38 to -1.401298E-45 for negative values; 1.401298E-45 to 3.402823E38 for positive values.
Double	Contains a double-precision, floating-point number in the range -1.79769313486232E308 to -4.94065645841247E-324 for negative values; 4.94065645841247E-324 to 1.79769313486232E308 for positive values.
Date (Time)	Contains a number that represents a date between January 1, 100 to December 31, 9999.
String	Contains a variable-length string that can be up to approximately 2 billion characters in length. ie. "The quick brown fox jumped over the lazy brown dog."
Object	Contains an object.
Error	Contains an error number.

Using Operators

Operators are the things that act on variables. We have already seen an operator used in the previous example, where we were assigning values to our variables. The example used one of the most common operators, "=" or the assignment operator. Another operator would be the addition operator "+".

```
dim x, y, sum
x = 1: y = 3: sum = 0
sum = x + y
```

This small chunk of VBScript code will declare the variables x, y and sum and assign the number 1 to x, 3 to y and 0 to sum. The next line of the script will add x to y and assign it to sum. The value of the sum variable will be 4.

Other operators are used to compare things, i.e. "=" equality, ">" greater than. For example,

```
if ( sum = 0 ) then
        sum = x + y
end if
```

This bit of code first checks to see if sum is equal to zero, and if so then it adds x and y together and assigns the result to sum. The "if" statement is an example of a control structure which we will examine shortly.

VBScript Operators

Computational

These are probably the most common operators. They are used for common mathematical operations.

Multiplication (*)
Division (/)
Modulo arithmetic (Mod)
Addition (+)
Subtraction (-)

Logical

These operators are very commonly used in conditional statements like “if” and “switch” that you will be learning about shortly.

Logical NOT (NOT)
Less than (<)
Greater than (>)
Less than or equal to (<=)
Greater than or equal to (>=)
Equality (=)
Inequality (<>)
Logical AND (and)
Logical OR (or)
Logical XOR (Xor)

Assignment

It is important to note that the single equal sign is used for assignment and for testing equality

Assignment (=)

Control Structures (Loops and Branches)

Branches

if

The "if" statement is a fundamental control statement. It allows your program to perform a test, and act based on the results of that test. For example:

```
if ( (x  = 1) and (y = 3) ) then
        sum = y - x
end if
```

In this statement the value of the x variable is compared to 1 to see if it is equal, and the value of the y variable is compared with 3 to see if it is equal. The use of the “and” operator, adds an additional logical condition that says that the first comparison must be true **and** the second comparison must be true for the overall result of the test to be true. If the test results in an overall true condition then the statements that follow the if statement will be executed. If the test results are false nothing will occur.

An additional clause you can add to the "if" statement is the "else", an example:

```
if (sum = 0) then
        sum = x + y
else
        subtotal = sum
end if
```

This statement is read as: if sum equals 0 then sum equals x plus y, or else subtotal equals sum.

ElseIf

ElseIf can be used to refine your code so that you can reduce the number of statements and make your program code easier to read.

```
If (sum > 10) then
        Alert( “Sum is greater than ten”)
ElseIf (sum > 5 ) then
        Alert( “Sum is greater than five”)
End If
```

Nested Ifs

You can also nest your If statements. Nesting is where one statement is contained within another one. You would do this when you need to meet one condition before you test for the second one.

```
If (x>10) then
        If (y>5) then
                Rem do something
        ElseIf (y>10) then
                Rem do something else
        End if
End if
```

Select Case

The select case statement is handy when a variable may take on a number of values and you want to test for some of those values. The use of “select case” is shorter and easier to read than a number of "if" statements.

```
Select case n
        case 1
                REM start here if n equals 1.
                REM place code here

        case 2
                REM start here if n equals 2.
                REM place code here

        case 3
                REM start here if n equals 3.
                REM place code here
        Case else
                REM if n is not equal to 1, 2 or 3
                REM case else is optional
End Select
```

A case can also take multiple values:

```
Case 5, 10, 15
```

A case can also take strings instead of numeric values:

```
Case “red”
```

Loops

A loop is a programming structure that forces the statements contained within its delimiters to execute over and over again until a condition is met at which point the loop ends.

Do while … Loop and do until … Loop

While a condition is true, execute one or more statements. “While loops” are especially useful when you do not know how many times you have to loop, but you know you should stop when you meet the condition.

```
dim x
x = 1
do while x <= 10 : REM loop until x is greater than 10

        until x is greaterthan 10

        x = x + 1 : REM add one to the value of x
loop

x = 1
do until x = 10 : REM loop until x is greater than 10

        until x is greaterthan 10

        x = x + 1 : REM add one to the value of x
loop
```

Loops that execute at least once.

```
x = 1
do

        until x is greater than 10

        x = x + 1 : REM add one to the value of x
loop until x = 10 : REM loop until x is greater than 10
```

For … Next

"For loops" are useful when you know exactly how many times you want the loop to execute.

```
for x = 1 to 10 : REM loop while x is <= 10

        do something ten times

Next
```

The loops normally increment by 1, you can alter increment using the "step" statement. You can change the step to be positive or negative.

i.e. **for** x = 0 **to** 10 **step** 2
i.e. **for** x = 10 **to** 0 **step** –2

Remember that your step value should increment and decrement to the final value.

Exit For

You can also exit out of a for loop before you have reached the end of the loop. The "Exit For" statement can be used with a condition, like below.

```
for x = 1 to 10 : REM loop while x is <= 10

        REM do something ten times

        If (sum>10) then
                Exit For
        End if
Next
```

For Each … Next

The For Each … Next statement makes working with collections easier. An easy way to think about it is; loop through all the items in a collection. It is exceptionally handy because you don't need to determine the beginning and ending points of the collection to construct your loop. In the example below rs is the object and fields is a collection contained in that object. We will discuss collections in detail when we cover objects.

```
Dim customer_field
For Each customer_field in rs.fields
        Rem Display the field
Next
```

While … Wend

While … Wend is an older loop construct. Microsoft recommends using the Do … Loop because it affords more flexibility.

```
Dim X
X = 1
While ( X < 10 )

        Rem do something while C < 10

        X=X+1
Wend
```

Functions and Procedures (Subroutines)

Functions are an important part of programming as they allow you to create chunks of code that perform a specific task. VBScript has a number of built-in functions that are part of the language. VBScript also gives you the ability to create your own functions. Your VBScript programs may be made up of one function or several functions.

Built-in:

The built in functions are there to perform a number of actions that programmers expect to be part of a programming language. Some of the built in functions include: CLng(*string value*), CInt(*string value*), Cdate (*value*), etc.. We will use these functions later in the course.

Programmer Created:

Subroutines that you create start with the command “sub” and are followed by the name for the subroutine. For example:

```
Sub subroutine_name( argument1, argument2, … )
        .
        .
        VBScript statements go here
        .
        .
End Sub
```

Usually the name is an indicator of the action the function is going to provide such as “checkForm”. The name is followed by a set of parenthesis and inside the parenthesis there can be a number of arguments. Arguments are the variable names that will be used for values (data) being passed to the subroutine.

Two ways to invoke a Subroutine

There are two ways to invoke a subroutine, the later is the preferred method as it makes the code more readable:

1. use the subroutine name followed by the arguments.
   I.e. showAlertBox arg1
2. use the call statement followed by the subroutine name with the arguments in brackets.
   I.e. Call showAlertBox(arg1)

Function … End Function

All functions have the ability to pass back a single value. Another way of saying this is you can pass data into a function using its arguments, and get the results out with a returned value. Functions can only return **one** value. In the example below the sum of two numbers is returned by the add_two_num() function. The trick to getting the function to return a value is to assign the value to a function name, see the second line in the first function below.

```
dim globalVar
globalVar = 1999

function add_two_num( a, b)
        add_two_num = a + b
end function

function mainProgram()
        dim x, y, total
        x = 5: y = 3: total = 0
        total = add_two_num( x , y )
        alert(total)
end function
```

In the above example:

1. The function mainProgram declares variables and assigns their initial values as follows - “x” equal to 5, “y” equal to 3 and “total” equal to 0
2. Then it calls the add_two_num function and passes in the values of x and y.
3. In the add_two_num function the values are added together and stored in a variable named sum.
4. The value of sum is returned back to the mainProgram function and stored in the variable named total.
5. The value of total is then displayed to user in an alert box.

Variables declared in a function are only available while within that. These are commonly referred to as local variables. Another type of variable is a global variable. Global variables are available to all functions, and should be used with caution, as it is easy to assign the wrong value to a global variable. In the above example globalVar is a global variable because it is declared outside of all of the functions.

Review Questions

1. True or False. Variables in VBScript are strongly typed.

2. True or False. You can declare a number of variables all at once.

3. The _____ keyword is used to declare variables.

4. Match the operator with its function.

Column A	Answer	Column B
a. <>		Assignment
b. =		Addition
c. +		In-equality

5. Create an if structure that will make sure the value of the p variable is greater than or equal to 99, and if so set the total equal to the value of p.

6. Create a function that will multiply two numbers together and return the result.

Summary

1. Variables are declared using the keyword dim.
2. What operator to use
3. Creating control structures, branches and loops
4. The two types of functions, built-in and programmer-created

4

Objects, Events, and the Document Object Model

In this module you will be introduced to the concepts of Objects, the DOM, and events and how you can use them.

Objectives

1. Arrays
2. Objects
3. The "set" command
4. The Document Object Model
5. Events
   i. onClick
   ii. onSubmit
   iii. onMouseOver
   iv. onMouseOut
   v. onFocus
   vi. onChange
   vii. onBlur
   viii. onLoad
   ix. onUnload

Arrays

VBScript, similar to other programming languages, has the capabilities to use a data structure called an array. An array is a collection of data where each piece of data is held in a numbered position within the array. A one-dimensional array would be similar to a column in a spreadsheet.

Each position in the array is assigned an index number beginning with 0 for the first position, 1 for the second position, and so on. This allows any position within the array, or “element” of the array, to be referenced by its index number.

The number contained within round brackets indicates the index of the array. An array can be created using the “dim” statement.

```
Dim a(2) : REM creates an array called “a”
a(0) = 1.2 :  REM sets the first element
a(1) = "VBScript" :  REM sets the second element
```

Arrays are important to understand because a number of components of the Document Object Model (DOM) are implemented as arrays, like forms, images, and elements.

Dim is short for dimension. When you declare a variable the interpreter needs to allocate memory to hold the contents for you. In strongly typed languages (which VBScript is not) the interpreter or compiler will know what the dimension (size of memory) is required to hold the value, based on the data type. In VBScript no memory is allocated until you store a value in the variable.

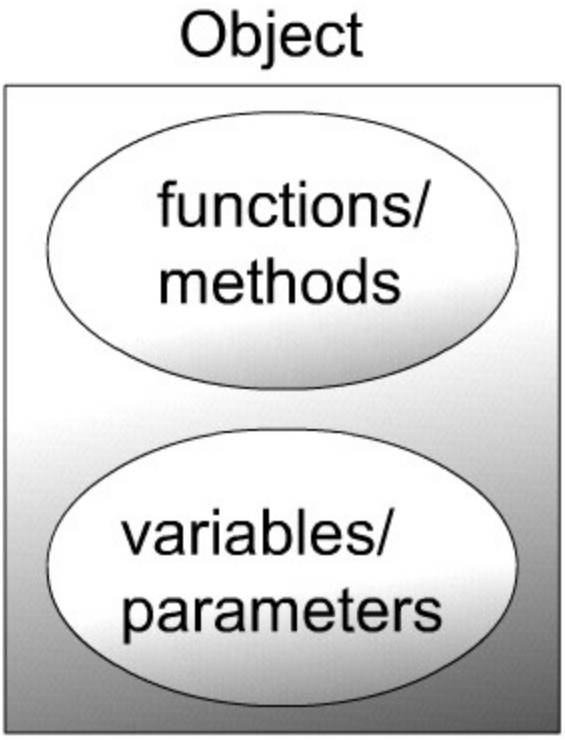
When dimensioning an array it is important to tell the interpreter how many elements are going to be in the array. If you need to increase the number of elements you can redim(ension) the array. You have to be careful when redimensioning arrays because the values of the array will be dropped. To prevent the values from being dropped use the keyword “preserve” to keep the existing values. For example:

```
ReDim Preserve a(10)
```

The ReDim statement actually creates a new array and copies the values from the first smaller array into the bigger array. If the new array is smaller than the existing array some data will be lost.

Objects

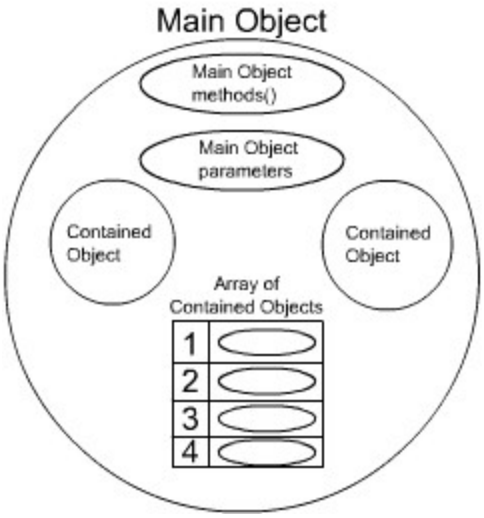
Object oriented programming was created to reduce the amount of time and effort that goes into creating new programs. Objects can be thought of as a level of organization or collection of like functions and variables into a single entity.



An object is a collection of variables (parameters) and functions (methods). The syntax for using an object is: object.parameter, or object.method(). This is the basic level of Object Oriented Programming.

A collection is an array of variables/parameters held within an object.

At the next level Objects can contain other objects. Each contained object has it's own properties and methods. Objects can also contain arrays of sub objects.



Unlike variables that are declared, objects are instantiated. When a variable is declared using dim, it is empty; this is not the case when an object is instantiated. When an object is instantiated with the keyword "set", the variable is a copy of the original object with all of its properties and methods, and contained objects. Normally when objects are created they will have default values for any of the critical properties, so that you won't have to worry about them when you create a copy.

Syntax: **Set** variable = object

Example: **Set** FormObject = document.form1

The Document Object Model (DOM)

If your script is executing in the browser, then the browser will provides us with a series of objects. The browser window that the page is displayed in is known as the window object. The HTML page displayed by your browser is known as the document object. The document object is probably the most commonly used object in client-side VBScript.

The HTML elements that you add to a page also extend the object hierarchy. An example is the FORM element and the elements that reside inside the form. This means that you can reference these objects, as illustrated in the HTML page below:

window.document.forms(0)
: Refers to the first form in the document. Forms are implemented as arrays in the DOM. If there is more than one form on the page the numbers will start at zero and go up.

window.document.Form1
: Refers to the form by name Form1

window.document.Form1.FirstName.value
: Refers to the value typed into the textbox named FirstName by the client, in the form named Form1

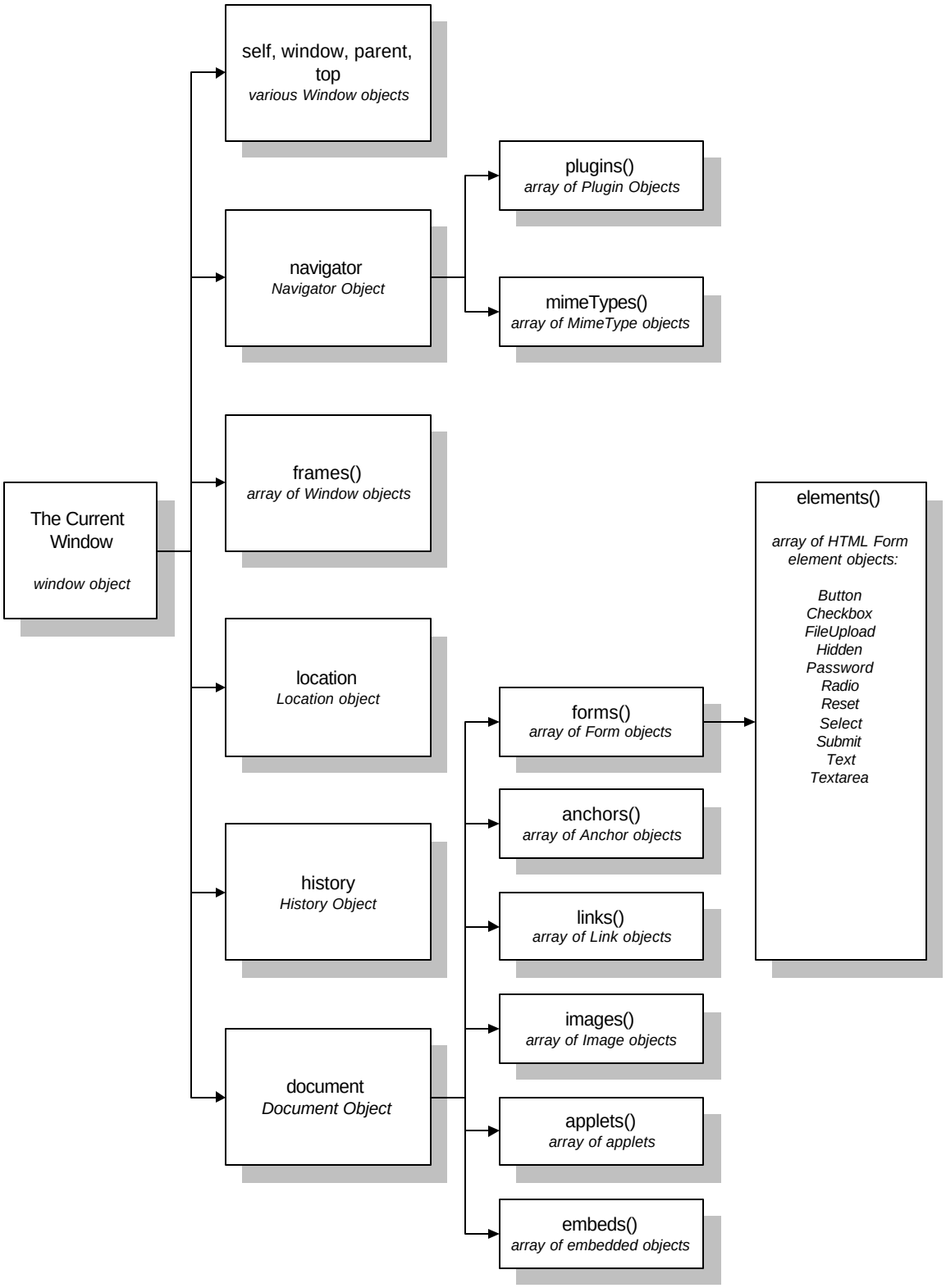
```
<HTML>
<HEAD>
    <TITLE>Simple Form</TITLE>
</HEAD>
<BODY>

<FORM NAME="Form1">
Name: <INPUT TYPE="TEXT" NAME="FirstName"><BR>
<INPUT TYPE="Button" VALUE="Submit Info" >
</FORM>

<FORM NAME="Form2">
Name: <INPUT TYPE="TEXT" NAME="LastName"><BR>
<INPUT TYPE="Button" VALUE="Submit Info" >
</FORM>

</BODY>
</HTML>
```

Objects located in the current document, in the current window can drop the reference to those two objects. For example:

forms(0)
    refers to the first form within this document
Form1
    refers to the form named Form1 in this document
Form1.FirstName.value
    refers to the value typed, in the browser by the client, into the textbox named FirstName, in the form named Form1, in this document

We recommend that you use the NAME attribute of any HTML tag that you are going to script, as in the above example the form is named Form1. This practice will simplify object naming for you.

The diagram below illustrates the HTML Document Object Model (DOM).



Events

Events are the triggers that call (start) one of your functions. Your client-side VBScript programs will not execute (run / be interpreted) unless started by an event. An event could be an action such as clicking on a button or placing your mouse over an image. We will use the onClick event for starting our form validation scripts, and the onMouseOver event for creating graphic images that change when you place your cursor over them.

The available event handlers in VBScript are:

onClick( )

A click event occurs when a button, checkbox, radio button, reset button, or submit button is clicked. This event is regularly used with a button to start script execution.

```
<INPUT TYPE="Button" VALUE="Click Me"
 onClick="window.alert('You Clicked me');">
```

In the above example when you click on the button "Click Me" it will execute the VBScript statement "window.alert('You Clicked me');". You can call any function or method this way.

onSubmit( )

A submit event occurs when the user submits a form. This event is regularly used with a form and a submit button to start the form validation script.

```
<FORM action="http://www.xnu.com/formtest.asp"
 onSubmit="return checkform();">
```

In the above example when the user clicks on the submit button, it will execute the function "checkform()". If the form passes all the validation tests, a true value is returned and the data will be passed to the "formtest.asp" CGI program. If the form contents do not pass the validation tests, it will not send the data to the "formtest.asp" CGI program.

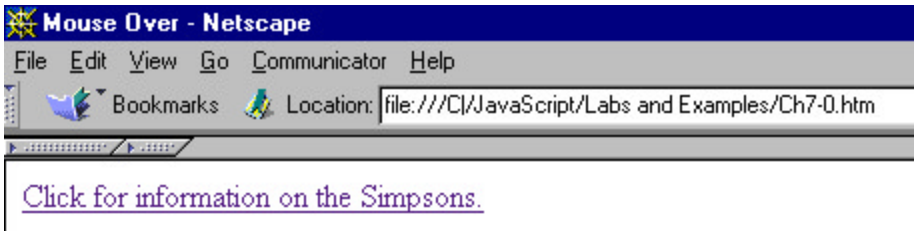
Note: We will be using these events in our discussion of Alerts, Prompts, and Confirmations; as well as, for Form Validation.

onMouseOver( )

An onMouseOver event occurs when the user positions their mouse over a hyperlink, or a linked region of a client-side image map.

<a href="http://synergy.simplenet.com/simpsons/" onMouseOver="window.status='The best Simpsons Webpage on the NET!'; return true;">Click for information on the Simpsons.</A>

In the above example an HTML link to a web site is created using the anchor tag. The onMouseOver() event is added to the anchor tag, and in this case the script will put a message in the status line of the browser window. Take a look at the results below.



Note the message in the status line of the browser window.

onMouseOut( )

An onMouseOut event occurs when the user moves their mouse off of a hyperlink, or a linked region of a client-side image map.

<a href="http://synergy.simplenet.com/simpsons/" onMouseOut="window.status='The best Simpsons Webpage on the NET!'; return true;">Click for information on the Simpsons.</A>

This example is the similar to the one above, except when the mouse is over the link, the hyperlink URL is shown in the status line, and when the mouse is moved off the hyperlink the message is displayed.

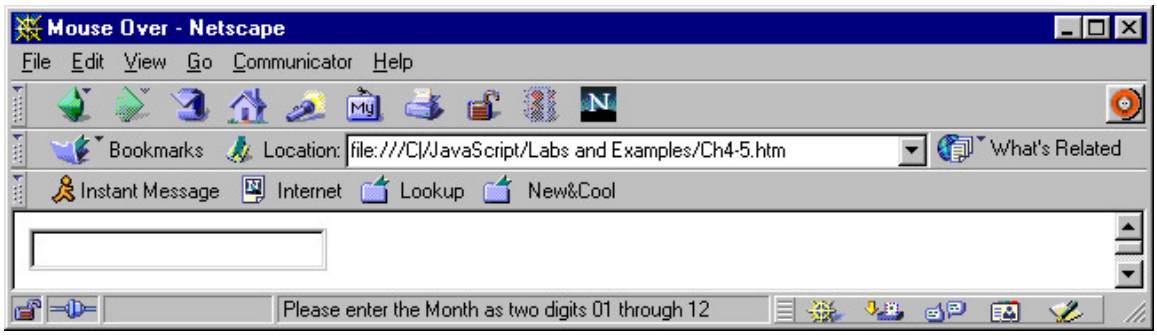
Note: We will be using these events in the Chapter on using Mouse Roll-overs.

onFocus( )

This event occurs when a user tabs into or clicks on a password field, a text field, a textarea, or a FileUpload field in an HTML form. If a user clicks on one of these elements in a form, it is receiving the user's focus.

```
<INPUT TYPE="TEXT" NAME="Month"
onFocus="window.status=('Please enter the Month as two digits 01 through 12'); return true;">
```

In this example when the user clicks on the month box or tabs into it a message is displayed in the status line of the browser that indicates what the user should type in.



onChange( )

The change event happens when the user leaves a password, text, textarea or FileUpload field in an HTML form, and its value has changed.

```
<INPUT TYPE="TEXT" NAME="Month"
onChange="window.status=('The value of the Month Changed!!!!'); return true;" >
<INPUT TYPE="TEXT" NAME="Year">
```

onBlur( )

The blur event triggers when the user leaves a password, text, textarea or FileUpload field in an HTML form.

```
<INPUT TYPE="TEXT" NAME="Month"
onBlur="window.status=('Do you not care about the value of the Month!'); return true;">
<INPUT TYPE="TEXT" NAME="Year">
```

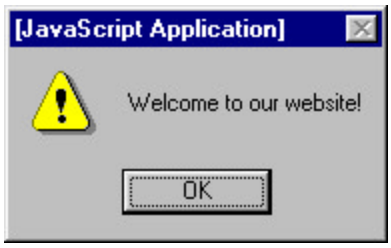
Note: It is usually a good idea to use either onChange or onBlur, and not both at the same time as they will appear to conflict with one another.

onLoad( )

The load event triggers when the browser finishes loading a document or all the frame pages within a <FRAMESET>.

```
<BODY onLoad="alert('Welcome to our website!');">
```

In this example after the page has completed loading into the browser the alert message below is popped up.

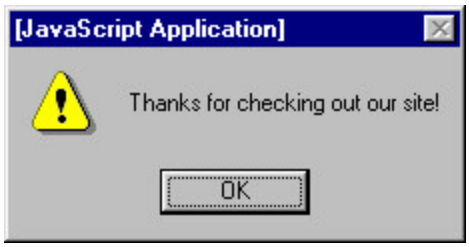


onUnload( )

The unload event occurs when you move to a new document. For example if you use the back button, or click on a link to another page the unload event will occur.

```
<BODY onUnload="alert('Thanks for checking out our site!');">
```

This example will popup the following message when someone moves off of the current page.



Note: It is possible to specify multiple events on a single HTML tag. For example,

```
<a href="http://synergy.simplenet.com/simpsons/"
onMouseOver="mouse1.src='gifs/mouse_over.gif'; "
onMouseOut="mouse1.src='gifs/mouse_out.gif'; ">
```

We will use this feature in the mouse over section.

Review Questions

1. An object is a collection of _____ and _____.

2. (True or False) The DOM is a collection of objects added to the VBScript language by the browser.

3. (True or False) Events are key to beginning the execution of your scripts.

4. Which event would you use to start a script after a page has been fully loaded by the browser?
   a. onClick()
   b. onSubmit()
   c. onLoad()
   d. onMouseOver()
   e. onload()

5. (True or False) Events are tied to specific HTML elements.

Summary

1. Describe an object

2. Use the new operator

3. Understand Document Object Model

4. Create and use an array

5. Use the built-in events
    i. onClick
    ii. onSubmit
    iii. onMouseOver
    iv. onMouseOut
    v. onFocus
    vi. onChange
    vii. onBlur
    viii. onLoad
    ix. onUnload

G

Glossary

Array(s):

An array is a collection of data that is associated with a single variable name. Each piece of data is considered a value and is indexed numerically within the array.

Comment(s):

Comments are often used by programmers as a means of documenting their program logic so that when they return to update it, or someone else needs to edit it, they can understand what the programmer was doing at the time.

Delimiter(s):

These are the brackets, quotations etc. that surround pieces of code. Examples of delimiters are " ", { }, ( ). A start delimiter such as (must be accompanied by a closing delimiter such as).

Event(s):

These are the 'triggers' that initiate (start) one of your functions. Client-Side VBScript will not run unless it is 'called on' by an event. Examples of events include clicking on a form's submit button or moving your cursor over an image that is a hyperlink.

Expressions:

Expressions are a set of statements that, as a group, evaluate to a single value. VBScript then categorizes this resulting value as one of the five data types: number, string, logical, function, or object.

Function(s):

Functions are an important part of programming as they enable you to create chunks of code that perform specific tasks. In VBScript, functions are referred to as 'subroutines' or 'procedures.' VBScript has several built-in functions, but you can also create your own functions for your programs.

Literals:

Literals are data comprised of numbers or strings used to represent fixed values in VBScript. They are values that do not change during the execution of your scripts.

Method(s):

A method is a function that has been encapsulated in an object.

Object(s):

An object is a collection of variables and functions. VBScript has four built-in objects including String, Date, Math and Array.

Operator(s):

These are the things that act on variables. There are three types of operators used in the VBScript language: Computational, Logical and Bitwise. Computational operators perform mathematical tasks such as addition, subtraction, multiplication etc. and are the most commonly used. Logical operators are often used in condition statements like “if” and “switch statements. Bitwise operators work with bits and bytes (or 1s and 0s) and are rarely used.

Statement(s):

A statement is a complete line of VBScript code.

Syntax:

A set of rules and guidelines used for writing in any language. Each programming language has its own set of rules and conventions. If you do not comply with the language’s syntax, your programs or scripts will not work correctly – if at all.

Variable(s):

A variable is a name assigned to a location in a computer’s memory to store data. Variables must be declared in your programs and can be identified by the fact that they are proceeded by the keyword ‘var’. VBScript, unlike other programming languages, can hold different data types as variables without declaring the type.

Local Variable(s):

These are variables that are only available within a specific function’s braces.

Global Variable(s):

These are variables that are available to all functions. Global variables should be used with caution, as it’s easy to assign the wrong value to a global variable.

A

Answer Appendix

Chapter 1 - Answers

1. (**True** or False) VBScript is an interpreted language.

2. VisualBasic is a **compiled** programming language.

3. (**True** or False) Visual Basic and VBScript were created by the same company.

4. Microsoft Internet Explorer supports the following scripting languages.
   a. VBScript
   b. Visual Basic
   c. BASIC
   d. JavaScript
   e. C++
   f. Perl

5. (True or **False**) VBScript is supported by a large number of browsers.

Chapter 2 - Answers

1. Which of the following are valid variable or function names:
   a. y
   b. 100percent
   c. a big number
   d. break
   e. subtractTwoNumbers
   f. First_Name

2. **True** or False. VBScript is a case insensitive language.

3. **True** or False. It is a good idea to add comments to your program code.

Chapter 3 - Answers

1. True or **False**. Variables in VBScript are strongly typed.

2. **True** or False. You can declare a number of variables all at once.

3. The **dim** keyword is used to declare variables.

4. Match the operator with its function.

Column A	Answer	Column B
a. <>	a	assignment
b. =	c	addition
c. +	b	In-equality

5. Create an if structure that will make sure the value of the p variable is greater than or equal to 99, and if so set the total equal to the value of p.

```
if ( p >= 99 ) then
    total = p
end if
```

6. Create a function that will multiply two numbers together and return the result.

```
function multiplyTwoNum( m, n )
    dim total
    total = m * n
    multiplyTwoNum = total
end function
```

or

```
function multiplyTwoNum( m, n)
     multiplyTwoNum m * n
end function
```

Chapter 4 - Answers

1. An object is a collection of **methods** and **parameters**.

2. (**True** or False) The DOM is a collection of objects added to the VBScript language by the browser.

3. (**True** or False) Events are key to beginning the execution of your scripts.

4. Which event would you use to start a script after a page has been fully loaded by the browser?
   a. onClick()
   b. onSubmit()
   c. onLoad()
   d. onMouseOver()
   e. onload()

5. (**True** or False) Events are tied to specific HTML elements.

Index