

## ASP PROGRAMMERS GUIDE LESSON 1

File:	aspGuideL1.doc
Date Started:	Mar 16, 2002
Last Update:	June 15, 2003
ISBN:	0-9730824-3-7
Version:	0.0

### VB SCRIPT

**ASP** stands for **A**ctive **S**erver **P**age. ASP allows a Web Browser to interact with a Web server. A Web browser can request information from an ASP program running on the Web server and the ASP program can respond back with the required information. A good example is someone is buying items on the Internet, the ASP program can track the items being bought, display information about the bought items and display a bill of sale. **ASP** uses **HTML** and the **VB Script** Language. VB Script is very similar to Visual Basic. If you know Visual Basic than you can learn VB Script very easily. Before you can learn VB Script you need to know a little bit about HTML. For those who do not know HTML you can take our **HTML** course before proceeding with the **ASP** course.

### VBSCRIPT LANGUAGE FUNDAMENTALS

We will first study the **VBScript** programming language. A programming language uses **data** and **code**. A programming language needs to use data to represent values where code are the instructions telling the program what to do. The programming instructions are in the form of programming statements. VBScript uses different types of **data** known as **data types**. There is **numeric data** like 10.5 or **string data** like "hello". Here's a chart with examples of the different types of data used in VB Script

Data Type	example
string	"hello there"
integer	5
decimal	10.5
numbers	1234
dates	10/15/2002

Every piece of data will have value and a data type. The value can be stored as a **constant** or in a **variable**. Constants and variables are identified by a **label**, like **Max** or **x**. A label is just a sequence of letters used for identification. Labels are also known as **identifiers**. The difference between a constant and a variable is that the constant value cannot be changed but the variable value can be changed at any time.

#### constants

Constants store values, but the value, once set cannot be changed. Constants are defined using the **const keyword**. Keywords are VB Script language command words, that state what you want to do. Constants usual start with an UPPER case letter. You assign a value to the constant using the "=" assignment operator.

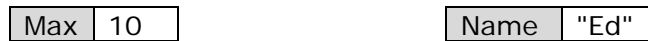
**Const Max = 10**

The value on the right hand side of the assignment operator is assigned to the constant label on the left hand side of the assignment operator.

The following example declares a numeric constant and a string constant. Notice strings are enclosed by double quotes " "



Think that a constant value is contained in a box represented by the constant label. If you know the constant name then you can get its value.

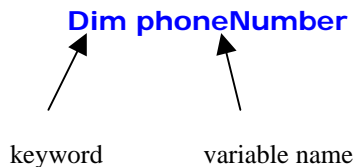


VBScript has constants already predefined for you. The following chart contains some VB script predefined constants, that you will be using soon.

<b>Empty</b>	indicates an uninitialized variable
<b>Nothing</b>	indicated a variable does not refer to any object
<b>Null</b>	indicates a variable contains invalid data
<b>True</b>	value of -1
<b>False</b>	a value of 0

## Variables

Variables store values and the value can be changed at any time. Before you can use a variable you have to declare it with the **Dim** keyword. Variables can hold any kind of data and have a **Variante** data type. Variante means represents any data type.



You may declare many variables at once using the Dim keyword.

**Dim name, address, phoneNumber.**

Once you declare your variable you can initialize it to a value.

**phoneNumber = "967-1111";**

Later you can give your variable a different value.

**phoneNumber = "765-8756"**

A variable or constant value can be assigned to another variable.

```
Dim x, y
y = Max
x = y
```

If Max is 10, what is the value of y? What is the value of x?

Again think as variables represented by a box having a name and a value. With a variable the value in the box can always change.



You don't always need to use the keyword Dim to declare a variable, but you should. To make sure you declare every variable before you use it you need to put the

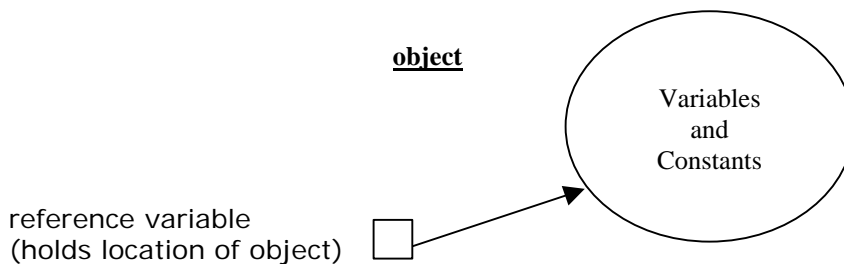
### Option Explicit

**directive** at the top of your VBScript program.

Variables do not represent a particular data type. Variables can represent any data type numeric, string etc. In VBScript variables are **variants** that represent any data type. When you use a variable the most appropriate data type is used.

## OBJECTS

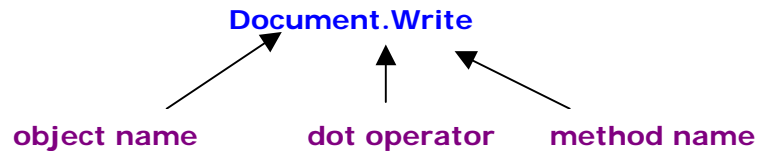
Many Variables and constants that have something in common may be grouped into an object. An object will contain many data values. Their data values are known as **properties**. Properties are used to describe an object.



Every object has a location in computer memory. The location where the object is, is identified by a variable. A variable that holds the location of an object is known as a **reference variable**. Objects are automatically supplied for you or they may be user defined. There are many Objects defined in VB Script that you can use right away.

## methods

An Object also has **methods** associated with them. A method contains **programming statements** that are instructions to tell the computer what to do. Methods can do many things. Methods are always associated with an object. When you want to use a method you state the object reference name, the dot operator and the method name.



For example you may want to print out a value of a variable on a web browser screen. To do this you need a method associated with an object that can print values to a web browser screen. A good example is the **write** method of the pre-defined **Document** object used to print out the value of a variable on a web browser screen.

**Document.Write Max**

**Document.Write phoneNumber.**

10

967-1111

## our first VB Script program

We can now write our first VB script program. The VB script is included within an HTML page. All VB scripts start with a `<script language="VBScript">` tag and end with a `</script>` tag. VB script comments start with a single quote like `' vbscript comment`. HTML Comments start with a `<!--` and end with a `-->` (notice the double hyphens). HTML comments and VB Script comments are quite different. Here is an HTML program containing a VBScript.

```

<!-- vartest.html -->
<html>
  <head>
    <title>My first VB Script program</title>
  </head>
  <body>
    <script language="VBScript">
      Dim name
      name = "hello" ' assign "hello" to variable name name
      Document.Write name
    </script>
  </body>
</html>

```

To run this script just type it into a text editor like Notepad, save as **vartest.html** and then open up using Internet Explorer.



```

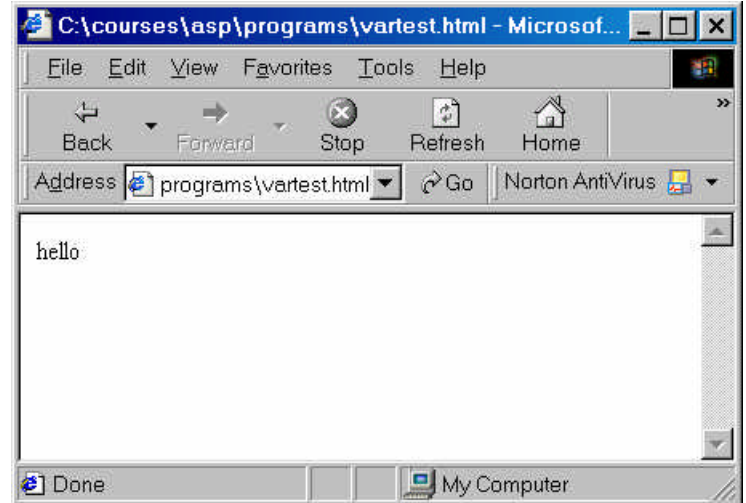
<!-- vartest.html -->
<html>
<head>
<title></title>
</head>

<body>
<script language="VBScript">

Dim name
name = "hello" ' set
Document.write name

</script>
</body>
</html>

```



This script runs on the client side. The client side means the Web browser is executing the HTML and the VBScript. It is not an ASP page rather than just an HTML file with a VBScript. ASP pages run on the server. ASP pages also use VBScript for its language.

If you want your output to start on a new line then you need to add a **<br>** tag in your script. You can use the **concatenation operator &** to do this.

#### Document.Write name & "<br>"

We have made a string out of the "**<br>**" tag. The **&** operator is known as a **concatenation** operator that joins two strings. The variable name and the HTML tag string "**<br>**". The only way to start a new line is to use the **<br>** tag. Why ? Because the string would be interpreted by the web browser as a line break. The web browser only knows what HTML tags mean. The page written in HTML and VB Script gets translated into HTML.

You can extend VBScript lines onto other lines in your program using the **&** operator and **\_** operator (you must leave a space before and after the **&** operator )

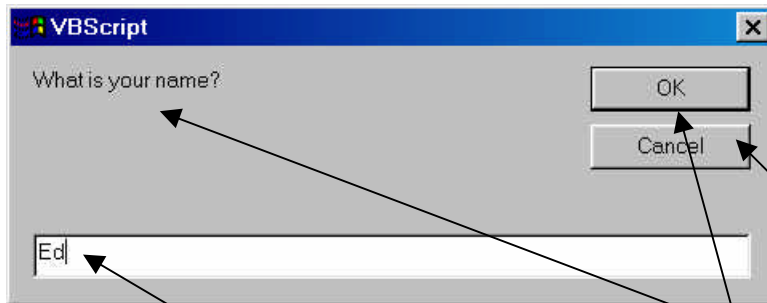
#### Document.Write name & \_ "<br>"

## ASP LESSON 1 EXERCISE 1

Write a VB Script that declares 2 constants and 2 variables. Use Document.write method to print the values to the screen. Assign the constants to the variables then print out the variable values to the screen using **Document.Write** . Call you file ASPL1Ex1.html

## INPUT BOX

Your VBScript program will need to get data values from the keyboard. A good way to get data from the keyboard is to use an Input Box.



These look very professional and impressive. Each Input Box gets a **prompt message** to tell the user what to do. Once they **enter the data** they can click on the **OK** button or the **Cancel** button.

The **InputBox** is a built in VBScript **function**. Built in functions are pre-defined code that you can use right away. Functions receive values and return values. Using built in functions let you program very fast. You need to know how to use built in functions. Every built in function has a **syntax** that specifies how to use it. Functions are different from method. Methods are associated with object and access the variables in the object. Functions are not associated with any objects and can be used without specifying an object reference.

The syntax for the Input Box function is a little overwhelming.

**InputBox (prompt[, title] [, default] [, xpos] [, ypos] [, helpfile, context])**

Every function gets arguments. Arguments are used to pass values to the function. The InputBox function needs the **prompt** message argument. The other arguments enclosed in square brackets [ ] are optional meaning you do not need to supply values for them. Here are the arguments and their descriptions. Don't be too concerned right now about all the arguments.

Argument	Description
<b>prompt</b>	Required. String expression displayed as the message in the dialog box. The maximum length of <b>prompt</b> is approximately 1024 characters, depending on the width of the characters used. If <b>prompt</b> consists of more than one line, you can separate the lines using a carriage return character ( <b>Chr(13)</b> ), a linefeed character ( <b>Chr(10)</b> ), or carriage return–linefeed character combination ( <b>Chr(13) &amp; Chr(10)</b> ) between each line.
<b>title</b>	Optional. String expression displayed in the title bar of the dialog box. If you omit <b>title</b> , the application name is placed in the title bar.
<b>default</b>	Optional. String expression displayed in the text box as the default response if no other input is provided. If you omit <b>default</b> , the text box is displayed empty.
<b>xpos</b>	Optional. Numeric expression that specifies, in twips, the horizontal distance of the left edge of the dialog box from the left edge of the screen. If <b>xpos</b> is omitted, the dialog box is horizontally centered.
<b>ypos</b>	Optional. Numeric expression that specifies, in twips, the vertical distance of the upper edge of the dialog box from the top of the screen. If <b>ypos</b> is omitted, the dialog box is vertically positioned approximately one-third of the way down the screen.

<b>helpfile</b>	Optional. String expression that identifies the Help file to use to provide context-sensitive Help for the dialog box. If <b>helpfile</b> is provided, <b>context</b> must also be provided.
<b>context</b>	Optional. Numeric expression that is the Help context number assigned to the appropriate Help topic by the Help author. If <b>context</b> is provided, <b>helpfile</b> must also be provided.

The following string constants can be used anywhere in your code in place of actual values:

Constant	Value	Description
<b>vbCr</b>	<b>Chr(13)</b>	Carriage return
<b>vbCrLf</b>	<b>Chr(13) &amp; Chr(10)</b>	Carriage return–linefeed combination
<b>vbFormFeed</b>	<b>Chr(12)</b>	Form feed; not useful in Microsoft Windows
<b>vbLf</b>	<b>Chr(10)</b>	Line feed
<b>vbNewLine</b>	<b>Chr(13) &amp; Chr(10)</b> or <b>Chr(10)</b>	Platform-specific newline character; whatever is appropriate for the platform
<b>vbNullChar</b>	<b>Chr(0)</b>	Character having the value 0
<b>vbNullString</b>	String having value 0	Not the same as a zero-length string (""); used for calling external procedures
<b>vbTab</b>	<b>Chr(9)</b>	Horizontal tab
<b>vbVerticalTab</b>	<b>Chr(11)</b>	Vertical tab; not useful in Microsoft Windows

### Using the Input Box

It's easy to use the **InputBox** function, all you have to do is assign the function to a variable. The variable gets the value that the user has entered into the input box

**Dim name**

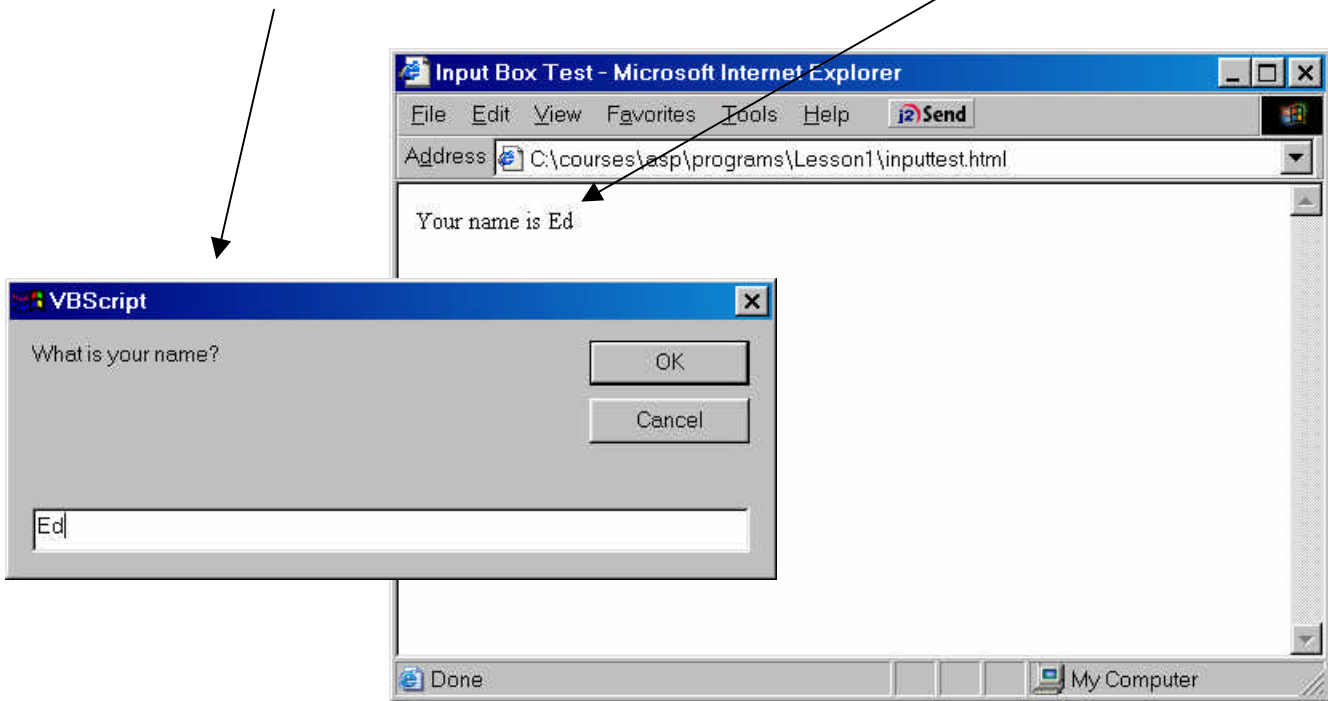
**name = InputBox("What is your name?")**

Here's a sample program that asks someone for their name and then prints their name to the web browser screen.

```
<html>
  <head>
    <title>Input Box Test</title>
    <script language="VBScript">
      Dim name
      name = InputBox("What is your name?")
      Document.Write "Your name is " & name
    </script>
  </head>
  <body>
  </body>
</html>
```

Here's the input box

Here's the VBScript output:

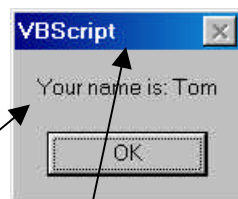


## ASP LESSON 1 EXERCISE 2

Write a VBScript that uses multiple input boxes to ask someone for their name, address and age, then print the answers on the web browser screen. Call your HTML file ASPL1Ex2.htm

## MSG BOX

A message box lets you display information in a window. The message box is displayed until the OK button is pressed.



A Message Box function has the following syntax:

**MsgBox(prompt[, buttons] [, title] [, helpfile, context])**

Again the optional supplied argument values are in square brackets[ ]. We just use **prompt** and **title**.

Here are the descriptions of the named arguments for the MsgBox:

Part	Description
<b>prompt</b>	Required. String expression displayed as the message in the dialog box. The maximum length of <b>prompt</b> is approximately 1024 characters, depending on the width of the characters used. If <b>prompt</b> consists of more than one line, you can separate the lines using a carriage return character ( <b>Chr(13)</b> ), a linefeed character ( <b>Chr(10)</b> ), or carriage return – linefeed character combination ( <b>Chr(13) &amp; Chr(10)</b> ) between each line.
<b>buttons</b>	Optional. Numeric expression that is the sum of values specifying the number and type of buttons to display, the icon style to use, the identity of the default button, and the modality of the message box. If omitted, the default value for <b>buttons</b> is 0.
<b>title</b>	Optional. String expression displayed in the title bar of the dialog box. If you omit <b>title</b> , the application name is placed in the title bar.
<b>helpfile</b>	Optional. String expression that identifies the Help file to use to provide context-sensitive Help for the dialog box. If <b>helpfile</b> is provided, <b>context</b> must also be provided.
<b>context</b>	Optional. Numeric expression that is the Help context number assigned to the appropriate Help topic by the Help author. If <b>context</b> is provided, <b>helpfile</b> must also be provided.

The **buttons** argument settings are:

Constant	Value	Description
<b>vbOKOnly</b>	0	Display <b>OK</b> button only.
<b>vbOKCancel</b>	1	Display <b>OK</b> and <b>Cancel</b> buttons.
<b>vbAbortRetryIgnore</b>	2	Display <b>Abort</b> , <b>Retry</b> , and <b>Ignore</b> buttons.
<b>vbYesNoCancel</b>	3	Display <b>Yes</b> , <b>No</b> , and <b>Cancel</b> buttons.
<b>vbYesNo</b>	4	Display <b>Yes</b> and <b>No</b> buttons.
<b>vbRetryCancel</b>	5	Display <b>Retry</b> and <b>Cancel</b> buttons.
<b>vbCritical</b>	16	Display <b>Critical Message</b> icon.
<b>vbQuestion</b>	32	Display <b>Warning Query</b> icon.
<b>vbExclamation</b>	48	Display <b>Warning Message</b> icon.
<b>vbInformation</b>	64	Display <b>Information Message</b> icon.
<b>vbDefaultButton1</b>	0	First button is default.
<b>vbDefaultButton2</b>	256	Second button is default.
<b>vbDefaultButton3</b>	512	Third button is default.
<b>vbDefaultButton4</b>	768	Fourth button is default.
<b>vbApplicationModal</b>	0	Application modal; the user must respond to the message box before continuing work in the current application.
<b>vbSystemModal</b>	4096	System modal; all applications are suspended until the user responds to the message box.
<b>vbMsgBoxHelpButton</b>	16384	Adds Help button to the message box
<b>VbMsgBoxSetForeground</b>	65536	Specifies the message box window as the foreground window
<b>vbMsgBoxRight</b>	524288	Text is right aligned
<b>vbMsgBoxRtIReading</b>	1048576	Specifies text should appear as right-to-left reading on Hebrew and Arabic systems

The first group of values (0–5) describes the number and type of buttons displayed in the dialog box; the second group (16, 32, 48, 64) describes the icon style; the third group (0, 256, 512) determines which button is the default; and the fourth group (0, 4096) determines the modality of the message box. When adding numbers to create a final value for the **buttons** argument, use only one number from each group. **Note** these constants are specified by Visual Basic for Applications. As a result, the names can be used anywhere in your code in place of the actual values.

Here are the Return Values from the MsgBox

Constant	Value	Description
<b>vbOK</b>	1	OK
<b>vbCancel</b>	2	Cancel
<b>vbAbort</b>	3	Abort
<b>vbRetry</b>	4	Retry
<b>vbIgnore</b>	5	Ignore
<b>vbYes</b>	6	Yes
<b>vbNo</b>	7	No

When both **helpfile** and **context** are provided, the user can press F1 to view the Help topic corresponding to the **context**. If the dialog box displays a **Cancel** button, pressing the ESC key has the same effect as clicking **Cancel**. If the dialog box contains a **Help** button, context-sensitive Help is provided for the dialog box. However, no value is returned until one of the other buttons is clicked. **Note** To specify more than the first named argument, you must use **MsgBox** in an expression. To omit some positional arguments, you must include the corresponding comma delimiter

Here's our example program using a MsgBox

```
<html>
<head>
<title>Msg Box Test</title>
<script language="VBScript">
name = MsgBox("Your name is: Tom")
</script>
</head>
<body>
</body>
</html>
```

### ASP LESSON1 EXERCISE 3

Change the previous exercise to use a **MsgBox**. Ask the user to enter their name, address and age using the **Input Box** and then display all three answers in a MsgBox. Use a line feed character **Chr(10)** or **VbCrLf** to separate lines in the MsgBox. Call your HTML file APSL1Ex3.htm

## ARRAYS

Arrays represent many variables all accessed by a common name.

variable	variable	variable	variable	variable
----------	----------	----------	----------	----------

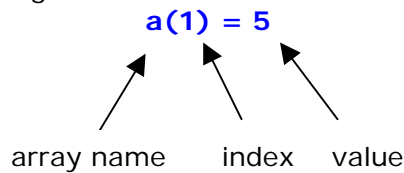
Each variable of the array is known as a cell or element. An example of using an array is to record the temperature of every hour for a certain day.

temperatures	45	89	56	34	23
--------------	----	----	----	----	----

It's easy to make an array you just declare a variable with the number of columns you want minus 1. The number of columns minus 1 is known as the upper bound. The lower bound is 0.

### Dim a(4)

Once you declare your array you can store values in it. You access the array cells by an index. Each cell is accessed using an index.



In VB Script All array start at index 0. Array **a** will have indexes from 0 to 4. The value 5 will be located in the array at index 1. Arrays have default value 0.

<b>a</b>	<b>0</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>
	0	5	0	0	0

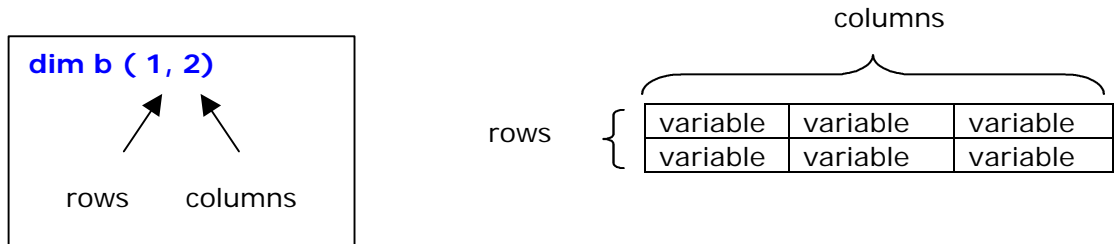
Arrays that have only one row, are known as 1 dimensional arrays.

## ASP LESSON 1 EXERCISE 4

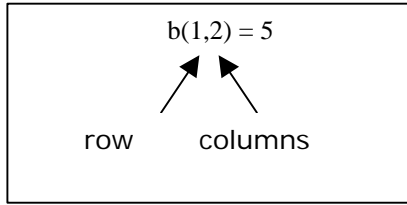
Declare a 1 dimensional array of 5 elements assign to each element the value of its index. Print out each array element using **Document.write**. Call you file ASPL1Ex4.html

## 2 dimensional arrays

An array may have more than 1 row. These arrays are known as 2 dimensional. To declare a two-dimensional array you must declare the number of rows and columns. Again we use minus 1. These are the upper bound indexes for the number of rows and columns. The lower bound is 0.



You assign values to the array cells by specifying row and column indexes. Arrays have default value starting index of 0.



		columns		
		0	1	2
rows	0	0	0	0
	1	0	0	5

**ASP LESSON 1 EXERCISE 5**

Declare a 2 dimensional array of 3 rows and 3 columns. Assign to each element the value of its row and column index. For example B(1,2) = 12 Print out each array element using [Document.Write](#) as the array would normally appear. ( 3 rows and 3 columns ). Call you file ASPL1Ex5.html

1	2	3
4	5	6
7	8	9

**BUILT IN FUNCTIONS**

The VBScript language come with **functions** that you can use right away to do things. Functions contain variables and programming statements. Functions differ from methods in that they are not associated with any objects. "Built in" means that someone has already coded it for you. So you can use them right away. Functions receive values, do a calculation and return the calculated value. There are many groups of functions available in VBScript . The first group of functions are used to check variables to determine what data type they represent.

**functions to check a variable for its data type**

function	description
<a href="#">IsArray(varname)</a>	returns true if variable is an array
<a href="#">IsDate(expression)</a>	returns true if expression can be converted to a date
<a href="#">IsEmpty(expression)</a>	returns true if variable has been initialized
<a href="#">IsNull(expression)</a>	returns true if expression returns no valid data
<a href="#">IsNumeric(expression)</a>	returns true if expression can be evaluated as a number
<a href="#">IsObject(expression)</a>	returns true if expression references an automated object
<a href="#">VarType(varname)</a>	returns a value representing the subtype

Example using:

A **varname** represents a variable name where an **expression** is any **value** represented by a constant or variable. We can use the **Document.Write** method to print out the results of our test.

```
Document.write ( IsArray(a) )
```



The output on the screen would be **True**.

The **VarType** function returns a **numeric value** identifying what the type of data a variable represents.

```
VarType(varname)
```

The *varname* argument can be any variable.

Example using:

```
Dim x
x = 5
Document.write ( VarType (x) )
```

The output on the screen would be 2.



(2 is the type for integer see following chart:)

subtypes returned by **VarType** function:

value	subtype
0	empty (uninitialized)
1	null (no valid data)
2	integer
3	long integer
4	single precision floating point number
5	double precision floating point number
6	currency
7	date
8	string
9	automation object
10	error
11	boolean
12	variant
13	non automated object
17	byte
8192+type	array

The **TypeName** function returns a string describing what data type a variable represents.

**TypeName**(*varname*)

The required *varname* argument can be any variable.

Example using:

```
Dim x
x = 5
Document.write ( TypeName ( x ) )
```

The output on the screen would be "Integer".

(see the following chart for all return string types)

Integer

The **TypeName** function has the following return values:

Value	Description
<b>Byte</b>	Byte value
<b>Integer</b>	Integer value
<b>Long</b>	Long integer value
<b>Single</b>	Single-precision floating-point value
<b>Double</b>	Double-precision floating-point value
<b>Currency</b>	Currency value
<b>Decimal</b>	Decimal value
<b>Date</b>	Date or time value
<b>String</b>	Character string value
<b>Boolean</b>	Boolean value; <b>True</b> or <b>False</b>
<b>Empty</b>	Unitialized
<b>Null</b>	No valid data
<b>&lt;object type&gt;</b>	Actual type name of an object
<b>Object</b>	Generic object
<b>Unknown</b>	Unknown object type
<b>Nothing</b>	Object variable that doesn't yet refer to an object instance
<b>Error</b>	Error

## ASP LESSON 1 EXERCISE 6

Make variables of different data types. Use the variable check functions to test if they are really what you assign them to be. Finally print out the **value type** and **string name** of each variable. Call your file ASPL1Ex6.htm

**FUNCTIONS TO DO OPERATIONS ON ARRAYS:**

Here are some function to do operations on arrays.:

<b>Redim [preserve] array_name(subscripts)</b>	<b>used to enlarge or reduce an array</b>
<b>preserve</b>	save existing data when array is resized
<b>varname</b>	name of array to resize
<b>subscripts</b>	size you want to change only the last dimension can be changed
	<b>Redim preserve a(10)</b>

<b>Erase array_name</b>	<b>clears the arrays memory</b>
	<b>Erase a</b>

<b>LBound arrayname[.dimension])</b>	<b>returns the lower bound of the specified dimension</b>
<b>arrayname</b>	name of array
<b>subscripts</b>	specifies which dimension you want 1 for first, 2 for second dimension 1 is default
	<b>Dim b(3,4)</b> <b>x = LBound(1)</b> ' x is 0 <b>x = LBound(2)</b> ' x is 0

<b>UBound arrayname[. dimension])</b>	<b>returns the upper bound of the specified dimension</b>
<b>arrayname</b>	name of array
<b>subscripts</b>	specifies which dimension you want 1 for first, 2 for second dimension (1 is default)
	<b>Dim b(3,4)</b> <b>x = UBound(1)</b> ' x is 2 <b>remember indexes start at 0</b> <b>x = UBound(2)</b> ' x is 3 <b>remember indexes start at 0</b>

**ASP LESSON 1 EXERCISE 7**

Declare an array2 by 3 array and fill it values of your choice. Print out the lower and upper bound of each dimension. Resize the array to 4 columns. Assign new values to this array. Print out the lower and upper bounds again. Finally erase the array. Call you file ASPL1Ex7.htm

**IMPORTANT**

You should use all the material in all the lessons to do the questions and exercises. If you do not know how to do something or have to use additional books or references to do the questions or exercises, please let us know immediately. We want to have all the required information in our lessons. By letting us know we can add the required information to the lessons. The lessons are updated on a daily bases. We call our lessons the "living lessons". Please let us keep our lessons alive.

**E-Mail all typos, unclear test, and additional information required to:**

[courses@cstutoring.com](mailto:courses@cstutoring.com)

**E-Mail all attached files of your completed exercises to:**

[students@cstutoring.com](mailto:students@cstutoring.com)

**Order your next Lesson from:**

[www.cscourses.com/asp.htm](http://www.cscourses.com/asp.htm)

This lesson is copyright (C) 1998-2003 by The Computer Science Tutoring Center "cstutoring"  
This document is not to be copied or reproduced in any form. For use of student only