<https://www.softwaretestinghelp.com/vbscript-tutorial-1/>

**VBScript Tutorials: Learn VBScript From Scratch (15+ In-Depth Tutorials)**

**==> Learn VBScript with these 15 Tutorials  <==**

**Tutorial #1:**[Introduction to VBScript](https://www.softwaretestinghelp.com/vbscript-tutorial-1/)  
**Tutorial #2:**[Declaring and Using Variables in VBScript](https://www.softwaretestinghelp.com/vbscript-variables-tutorial-2/)  
**Tutorial #3:**[Operators, Operator Precedence and Constants in VBScript](https://www.softwaretestinghelp.com/vbscript-constants-and-operators-tutorial-3/)  
**Tutorial #4:**[Using Conditional Statements in VBScript](https://www.softwaretestinghelp.com/vbscript-conditional-statements-tutorial-4/)  
**Tutorial #5:**[Loops in VBScript](https://www.softwaretestinghelp.com/vbscript-loops-tutorial-5/)  and also [Part 2 here](https://www.softwaretestinghelp.com/qtp-tutorial-10-vb-script-basics-part2/)  
**Tutorial #6:**[Using Procedures and Functions in VBScript](https://www.softwaretestinghelp.com/vbscript-procedures-functions-tutorial-6/)  
**Tutorial #7:**[Arrays in VBScript](https://www.softwaretestinghelp.com/vbscript-arrays-tutorial-7/)  
**Tutorial #8:**[Date Functions in VBScript](https://www.softwaretestinghelp.com/vbscript-date-functions-tutorial-8/)  
**Tutorial #9:**[Working with Strings and Cookies in VBScript](https://www.softwaretestinghelp.com/vbscript-string-functions-instr-replace-tutorial-9/)  
**Tutorial #10:**[Working with Events in VBScript](https://www.softwaretestinghelp.com/vbscript-events-tutorial-10/)  
**Tutorial #11:**[Working with Excel Objects in VBScript](https://www.softwaretestinghelp.com/vbscript-excel-tutorial-11/)  
**Tutorial #12:**[Working with Connection Objects in VBScript](https://www.softwaretestinghelp.com/vbscript-connection-objects-tutorial-12/)  
**Tutorial #13:** [Working with Files in VBScript](https://www.softwaretestinghelp.com/vbscript-files-tutorial-13/)  
**Tutorial #14:**[Error Handling in VBScript](https://www.softwaretestinghelp.com/vbscript-error-handling-tutorial-14/)  
**Tutorial #15:**[VBScript Interview Questions](https://www.softwaretestinghelp.com/popular-vbscript-interview-questions-answers/)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Initially, to start with I have chosen the first topic as **‘Introduction to VBScript’.**

***In this tutorial, I will discuss the basics of VBScript, thereby focusing more on its features, data types supported by it and coding methodologies along with the procedure to handle comments and formats in scripts***.

**What You Will Learn:**[[hide](https://www.softwaretestinghelp.com/vbscript-tutorial-1/)]

* [What is VBScript?](https://www.softwaretestinghelp.com/vbscript-tutorial-1/#What_is_VBScript)
* [Basic of VB Scripting Concepts](https://www.softwaretestinghelp.com/vbscript-tutorial-1/#Basic_of_VB_Scripting_Concepts)
  + [Data types](https://www.softwaretestinghelp.com/vbscript-tutorial-1/#Data_types)
  + [Variables](https://www.softwaretestinghelp.com/vbscript-tutorial-1/#Variables)
  + [Constants](https://www.softwaretestinghelp.com/vbscript-tutorial-1/#Constants)
  + [Operators](https://www.softwaretestinghelp.com/vbscript-tutorial-1/#Operators)
* [Environments Supporting VBScript](https://www.softwaretestinghelp.com/vbscript-tutorial-1/#Environments_Supporting_VBScript)
* [Data Types in VBScript](https://www.softwaretestinghelp.com/vbscript-tutorial-1/#Data_Types_in_VBScript)
* [How to Create a Simple VBScript?](https://www.softwaretestinghelp.com/vbscript-tutorial-1/#How_to_Create_a_Simple_VBScript)
* [Where to Insert Scripts in an HTML Page?](https://www.softwaretestinghelp.com/vbscript-tutorial-1/#Where_to_Insert_Scripts_in_an_HTML_Page)
* [How Comments are Handled in VBScript](https://www.softwaretestinghelp.com/vbscript-tutorial-1/#How_Comments_are_Handled_in_VBScript)
* [Reserved Keywords](https://www.softwaretestinghelp.com/vbscript-tutorial-1/#Reserved_Keywords)
* [Conclusion](https://www.softwaretestinghelp.com/vbscript-tutorial-1/#Conclusion)
* [Recommended Reading](https://www.softwaretestinghelp.com/vbscript-tutorial-1/#Recommended_Reading)

### What is VBScript?

As the name itself explains, **VBScript is a ‘Scripting Language’**. It is a lightweight case insensitive programming language developed by Microsoft. It is a subset of ‘Visual Basic’ or we may also say it as a lighter version of Microsoft’s programming language Visual Basic.

Most of us would have used Visual Basic during our course curriculum in our school or college. Visual Basic is an event-driven programming language and an Integrated Development Environment from Microsoft.

VBScript language is used in QTP for coding and running Automated Test Scripts. This is not a very difficult language to learn and with a little knowledge of basic programming skills and passion for writing code, anyone can learn this easily. For those who know Visual Basic, it is an added advantage.

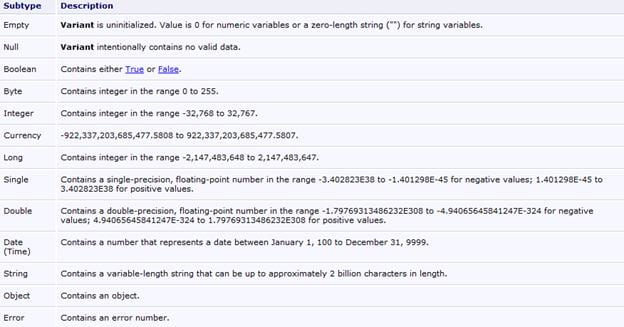
Automation Testers, who want to create, maintain and execute the tests in QTP need to have basic programming skills using VBScript.

### Basic of VB Scripting Concepts

**Now let’s move on to some basics topics that are revolving around VBScript to enable clear understanding and knowledge about VBScript.**

#### **Data types**

1) There is only one data type: **Variant**. It can store different kinds of information based on the context in which it is used.  
2) If used in a numeric context it is a number or a string if used in a string expression.  
3) If a number has to behave as a string we could enclose it within “ “.  
4) There are various subtypes to a variant. You can explicitly specify these subtypes to achieve a clear definition for your data. The below is a screenshot from the VB User guide that shows all the subtypes of data that can be used:  
(click on image to enlarge)

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2013/04/VB-script-data-types.jpg)

5) Conversion functions can be used to convert one subtype of data into another.  
6) Since it is the only data type available, all the return values from a function are variants.

Here are different VBScripting [examples](https://www.w3schools.com/asp/asp_examples.asp) you can try on your own.

#### Variables

**1)** A variable is nothing but a space in the computer’s memory that can store certain information. This information is bound to change from time to time. Where the information goes physically is immaterial but when needed, it can be accessed or changed by addressing the name of the variable.

E.g: If there is a statement that you want to run several times, you could use a variable to contain that count. Say X. X is a variable that can be used to store, change and use the space in the memory where we want to keep the count.

**2)** All variables are of the datatype Variant.

**3)** Declaring a variable before its use is optional, although it’s a good practice to do so.

**4)** To make the declaration mandatory there is an “**Option Explicit”**Statement available. To declare variables:

Dim x – This declares  x  
Dim x, y, z – This declares multiple variables  
X=10 – This is how a value is assigned. As a general rule, the variable is the left-hand side component and the right is its value.  
X=”Swati” – this is the way a string value is assigned.

To make declarations mandatory this is how the code has to be written:  
*Option Explicit*  
*Dim x, stri*

If Option explicit statement was not used, we could have directly written:  
*x=100*  
*stri=”Swati”*  
and it would not have thrown an error.

**5)** **Naming convention**: Names must start with an alphabetic character, must be unique, cannot contain an embedded period and cannot exceed 255 chars.

**6)** A variable containing a single value is a scalar variable and the one that has more than one is an array.

**7)** A one dimensional Array can be declared as Dim A(10). All the arrays in VB Script are zero-based that means the array index starts from 0 through the number declared. That means, our array A has 11 elements. Starting from 0 to 10.

**8)** To declare a 2-dimensional array simply separate the row count and column count by a comma. Eg: Dim A(5, 3). This means it has 6 rows and 4 columns. The first number is always row and the second a comma.

**9)** There is also a dynamic array whose size can change during runtime.  These arrays can be declared using dim or redim statements.

If an array is declared as Dim A(10) and during runtime, if we need more space we can do the same by using the statement: redim A(10). There is a “Preserve” statement that can be used in conjunction with the redim statement.

Dim A(10,10)  
……  
….  
Redim preserve A(10,20)

This piece of code shows how we do it. Initially, A is a 11 by 11 array. Then we are resizing it to be an 11 by 21 array and the preserve statement will make sure that the data that is previously contained in the array is not lost.

#### Constants

1. As the name implies a constant is nothing but an unchanging value in a program that is assigned a name.
2. They can be declared by prefixing “Const” to a name.
3. Eg: Const a=”10” or Const Astr=”Swati”.
4. This value cannot be changed accidentally while the script is running.

#### Operators

Some of the important operators that are most commonly used are:

1. String concatenation: & (Eg: Dim x=”good”&”day”, so x contains “goodday”
2. Addition (+)
3. Subtraction (-)
4. Multiplication (\*)
5. Division(/)
6. Logical negation (Not)
7. Logical conjunction (And)
8. Logical disjunction ( Or)
9. Equality(=)
10. Inequality (<>)
11. Less than (<)
12. Greater than(>)
13. Less than or equal to(<=)
14. Greater than or equal to (>=)
15. Object equivalence(Is)

It is important to note that the list is not complete but merely a subset containing the most commonly used operators.

**The operator precedence rules are:**

1. Multiplication or Division take precedence over addition or subtraction
2. If multiplication and division exist in the same expression, then left to right order is considered
3. If Addition and subtraction occur in the same expression, then too, left and right order is taken into consideration.
4. The order can be overridden by using parenthesis. In this case, the expression within the parenthesis is executed first.
5. & operator takes precedence after all arithmetic operators and before all logical operators.

### Environments Supporting VBScript

Primarily, there are 3 Environments where VBScript can be run.

**They include:**

**#1) IIS (Internet Information Server):** **I**nternet **I**nformation **S**erver is Microsoft’s Web Server.

**#2) WSH (Windows Script Host): W**indows **S**cript **H**ost is the hosting environment of the Windows Operating System.

**#3) IE (Internet Explorer): I**nternet **E**xplorer is a simple hosting environment that is most frequently used to run scripts.

### Data Types in VBScript

Unlike other languages, VBScript has only 1 data type called **Variant**.

As this is the only data type that is used in VBScript, it’s the only data type that is returned by all the functions in the VBScript.

A variant data type can contain different kinds of information, depending on how it is used. ***For Example,*** If we use this data type in String context then this will behave like a String and if we use this in the Numeric context then this will behave like a Number. This is the specialty of a Variant data type.

A Variant data type can contain several subtypes. Now, let’s take a look at what all values/data will be returned if a particular subtype is used.

**Subtypes include:**

**#1) Empty:** This subtype indicates that the value will be 0 in case of Numeric Variables and “” for String Variables.

**#2) Null:**This subtype indicates that there is no valid data.

**#3) Boolean:**This subtype indicates that the resultant value will be either true or false.

**#4) Byte:**This subtype exhibits that the resultant value will lie in the range between 0 to 255 i.e. the result will be from any value ranging from 0 to 255.

**#5) Integer:**This subtype shows that the resultant value will lie in the range between -32768 to 32767 i.e. the result will be from any value ranging from -32768 to 32767

**#6) Currency:**This subtype indicates that the resultant value will lie in the range between -922,337,203,685,477.5808 to 922,337,203,685,477.5807 i.e. the result will be from any value ranging from -327-922,337,203,685,477.5808 to 922,337,203,685,477.5807.

**#7) Long:**This subtype shows that the resultant value will lie in the range from -2,147,483,648 to 2,147,483,647 i.e. result will be from any value in between -2,147,483,648 to 2,147,483,647.

**#8) Single:**This subtype exhibits that the resultant value will be from any value in between -3.402823E38 to -1.401298E-45 in case of negative values.

And for positive values, the result will be from any value in between 1.401298E-45 to 3.402823E38.

**#9) Double:**This subtype indicates that the resultant value will be from any value in between -1.79769313486232E308 to 4.94065645841247E-324 in case of negative values.

And for positive values, the result will be from any value in between 4.94065645841247E-324 to 1.79769313486232E308.

**#10) Date (Time):**This subtype will return a number which will represent a date value in between January 1, 100 to December 31, 9999

**#11) String:**This subtype will return a variable-length string value which can approximately be up to 2 billion characters in length.

**#12) Object:**This subtype will return an object.

**#13) Error:**This subtype will return an error number.

### How to Create a Simple VBScript?

To create a VBScript, there are only 2 things required.

**They are:**

* **Text Editors** like Notepad++ or even Notepad to write the VBScript Code.
* **IE** (good to have IE6 or above) to run the VBScript Code.

Now, let’s see a few VBScript Code for clarity purpose but before that, it is important to know where can the Scripts be inserted in an HTML Page.

### Where to Insert Scripts in an HTML Page?

VBScript provides you the liberty to place a code in any of the following sections:

* **Within the Header Tags** i.e. in between <head> and </head>.
* **Within the Document’s Body** i.e. between <body> and </body> tags.

**First VBScript Code in HTML:**

***Now, let’s take a simple example to understand how VBScript code can be written inside HTML tags.***

<html>

<head>

<title> Testing VBScript Skills </title>

</head>

<body>

<script type=”text/vbscript”>

variable1 = 1

variable2 = 2

output = (variable1 + variable2) / 1

document.write (“resultant from the above equation is ” & output)

</script>

</body>

</html>

**Note**: Whatever is put inside the brackets of ‘document.write’, will be displayed as an output on the display page.

**The Output** of this program is: **resultant from the above equation is 3**

After completing the code, you can save this in a file and give a file name as anyfilename.html.

**To run**, just open this file in IE.

**Important to Know:**

We have just seen the implementation of VBScript code in the HTML file. **However, VBScript in QTP is not placed inside the HTML tags. It is saved with an extension ‘.vbs’ and is executed by the QTP Execution Engine.**

To understand the practical implementation of VBScript in terms of QTP, you must know variables, constants, etc. and I will cover that in my upcoming tutorials and for time being, I just want to show you the VBScript code with the concept of an external file.

**VBScript in External File:**

<html>

<head>

<script type=”text/vbscript” src=”nameofthefile.vbs”> </script>

</head>

<body>

variable1 = 22

variable2 = 21

subtraction = variable1 - variable2

document.write (“subtraction of 2 numbers is” & subtraction)

</body>

</html>

**To access this code from an external source, save this code in a text file with an extension “.vbs”.**

### How Comments are Handled in VBScript

It is considered to be a good programming practice to include comments in the Scripts for better readability and understanding purposes.

**There are 2 ways in which Comments can be handled in a VBScript:**

**#1) Any Statement that starts with a Single Quote (‘) is treated as a comment:**

<html>

<head>

<script type=”text/vbscript”</script>

</head>

<body>

‘let’s do subtraction of 2 numbers

variable1 = 11

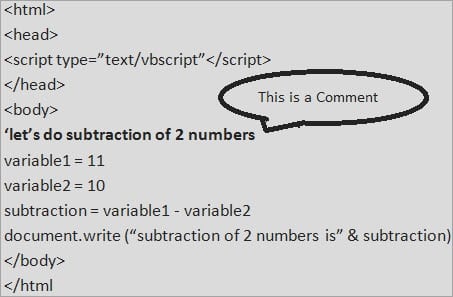
variable2 = 10

subtraction = variable1 - variable2

document.write (“subtraction of 2 numbers is” & subtraction)

</body>

</html

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/01/Single-Quote.jpg)

**#2)** **Any Statements that start with the keyword REM are treated as Comments.**

<html>

<head>

<script type=”text/vbscript”></script>

</head>

<body>

**REM let’s do subtraction of 2 numbers**

variable1 = 11

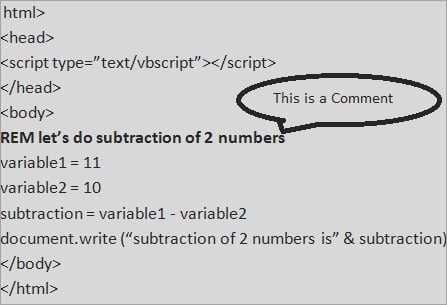
variable2 = 10

subtraction = variable1 - variable2

document.write (“subtraction of 2 numbers is” & subtraction)

</body>

</html>

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/01/REM-1.jpg)

**Formatting Tips:**

**#1)** No Semicolon is required to end the particular statement in VBScript.

**#2)** If 2 or more lines are written in the same line in VBScript then**Colons (:) act as a line separator**.

**Let’s understand this with the help of an Example:**

<html>

<head>

<script type=”text/vbscript”></script>

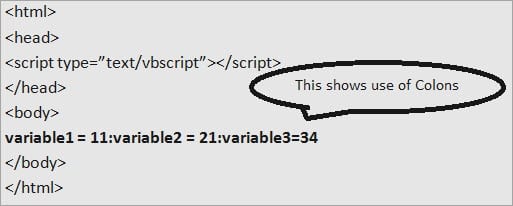
</head>

<body>

**variable1 = 11:variable2 = 21:variable3=34**

</body>

</html>

**[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/01/Formatting-Tips.jpg)  
#3)** If a statement is lengthy and required to break into multiple statements then you can use **underscore** “\_”.

***Let’s see its Example:***

<html>

<head>

<script type=”text/vbscript”></script>

</head>

<body>

variable1 = 11

variable2 = 10

output = (variable1 - variable2) \* 10

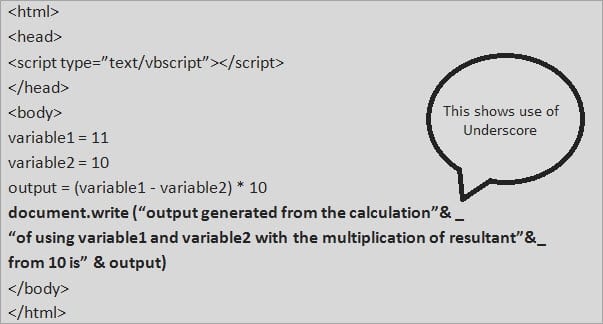
document.write (“output generated from the calculation”& \_

“of using variable1 and variable2 with the multiplication of resultant”&\_

from 10 is” & output)

</body>

</html>

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/01/Underscore.jpg)

### Reserved Keywords

In any language, there are a set of words which work as Reserved Words and they cannot be used as Variable names, Constant names or any other Identifier names.

**Following is the list of Reserved Keywords in VBScript:**

| **Loop** | **New** | **Null** | **ParamArray** |
| --- | --- | --- | --- |
| LSet | Next | On | Preserve |
| Me | Nothing | Option | Private |
| Mod | Not | Optional | Public |
| RaiseEvent | ReDim | Dem | Rem |
| Resume | RSet | Select | Set |
| Shared | Single | Static | Stop |
| Sub | Then | To | True |
| Type | And | As | Boolean |
| Case | Class | Const | Currency |
| Debug | Dim | Do | Double |
| Each | Else | Elself | Empty |
| Event | Exit | False | For |
| Function | GoTo | If | Imp |
| Implements | In | Integer | Is |
| Until | Variant | Wend | While |
| With | Xor | Eval | Execute |
| MsgBox | Erase | ExecuteGlobal | Option |
| OptionExplicit | Randomize | SendKeys | Let |
| Let | Like | Long | Type |
| End | EndIf | Enum | Eqv |

### Conclusion

That’s it! This is all about the basic concepts that are involved in VBScript.

I hope that this VBScript tutorial would have a given you a brief overview and clear understanding of this scripting language with easy examples.

***About the author:*** Thanks to STH Team member Varsha for helping us to compile this series. She is in the Software Testing field with core VBScripting skills and received many internal excellence awards.

[***NEXT Tutorial #2***](https://www.softwaretestinghelp.com/vbscript-variables-tutorial-2/)***=> In my upcoming VBS tutorial, I will cover about Variables in VBScript.***

# VBScript Variables: How To Declare And Use Variables – VBScript Dim

Variables are like Containers where the values can be stored and changed as well during the **Script Execution**.

These are named as memory locations which can be accessed by giving some names i.e. we can refer any variable by its name to see what value it contains or if we need to change any value. Variable is the key aspect of learning any language easily and efficiently.

**Suggested reading =>>**[**VBA Variables and Option Explicit**](https://www.softwaretestinghelp.com/vba-variables/)

**What You Will Learn:**[[hide](https://www.softwaretestinghelp.com/vbscript-variables-tutorial-2/)]

* [Rules for Naming Variables/Naming Conventions](https://www.softwaretestinghelp.com/vbscript-variables-tutorial-2/#Rules_for_Naming_VariablesNaming_Conventions)
* [Variable Declaration & Its Types](https://www.softwaretestinghelp.com/vbscript-variables-tutorial-2/#Variable_Declaration_Its_Types)
  + [#1) Implicit Declaration](https://www.softwaretestinghelp.com/vbscript-variables-tutorial-2/#1_Implicit_Declaration)
  + [#2) Explicit Declaration](https://www.softwaretestinghelp.com/vbscript-variables-tutorial-2/#2_Explicit_Declaration)
* [Assigning Values to Variables in VBScript](https://www.softwaretestinghelp.com/vbscript-variables-tutorial-2/#Assigning_Values_to_Variables_in_VBScript)
* [Scope of the Variables](https://www.softwaretestinghelp.com/vbscript-variables-tutorial-2/#Scope_of_the_Variables)
* [Declaring & Assigning Values to Arrays](https://www.softwaretestinghelp.com/vbscript-variables-tutorial-2/#Declaring_Assigning_Values_to_Arrays)
* [Conclusion](https://www.softwaretestinghelp.com/vbscript-variables-tutorial-2/#Conclusion)
* [Recommended Reading](https://www.softwaretestinghelp.com/vbscript-variables-tutorial-2/#Recommended_Reading)

### Rules for Naming Variables/Naming Conventions

There are certain points to be kept in mind, rather we can say that there are certain rules to be followed before naming a variable.

**They are:**

* Variable Name must always start with an alphabet. **E.g:** output, name, etc. And Variable Name should not start with a number or any special character like \_va123, 12non, etc.
* Variable Name cannot exceed a limit of 255 characters
* Variable Name should never contain a period (.)

These rules should be strictly followed by a good programming practice.

Now, it is important to know how variables are declared.

### Variable Declaration & Its Types

Any variable must be declared first, before using. In various languages, different keywords are used to declare a variable.

**Let’s see how a variable is declared in the VBScript.**

It is a good programming practice to declare the variable first and then starts using it to avoid any problems/confusions arising in the remaining code. I will brief on this point in a while.

**We can declare variables using Public, Private or Dim Statements**.

**Example:**

If you want to declare a variable ‘age’ then it can be declared as follows:

**Dim age**

You can declare multiple variables in a single line by separating it with Comma (,) as shown below:

**Dim employee, name, school, class**

In this way, a variable can be created.

**Types of Variable Declaration:**

There are 2 ways in which a variable can be declared. They are as given below.

#### **#1) Implicit Declaration**

When variables are used directly without declaration, it is termed as Implicit Declaration. However, it’s not a good practice because if at times a variable name is not spelled correctly in the script then it can produce weird results while running and sometimes, it will not be easy as well to detect this by the user.

**Let’s understand this with a simple Example:**

age = 10

In this case, if you misspell “age” variable for “aeg”, the script will automatically create a new variable with the name ‘aeg’.

To overcome this, you can use ‘**Option Explicit’**, which is discussed below.

#### **#2) Explicit Declaration**

Declaring variables before using them is called an Explicit Declaration of variables. This is what we have done above.

**Eg**: Dim age

So, before moving to the next topic, let me explain a bit about Option Explicit.

**Option Explicit:**

This provides a mechanism where the user has to declare all the variables using Dim, Public or Private Statements before using them in the Script.

In option explicit, if a user tries to use the variables which are not declared in case of Option Explicit then an error occurs. It is always recommended to use ‘Option Explicit’ at the top of the code so that even if unintentionally you used a wrong name of the variable then you can correct it immediately without any confusion.

### Assigning Values to Variables in VBScript

Assigning values to the variables in VBScript is done in the same manner as in other programming languages. After the declaration of a variable, a value is assigned to it with the help of an **Equal (=)**Operator.

Name of the Variable comes on the left and the value which is assigned to the Variable is on the Right Hand Side of the ‘=’ Operator.

**Let’s understand this with a few simple Examples:**

***Example 1:***

If a variable to which a value is to be assigned is of a **String** type then it can be assigned using double quotes (“”).

<html>

<head>

<title>Let’s learn assigning values to variables</title>

</head>

<body>

<script type=”text/vbscript”>

Dim msg

msg = “Hello Everyone”

Msgbox msg ‘this will show Hello Everyone message in the message box

</script>

</body>

</html>

***Example 2:***

If a variable to which a value is to be assigned is of a **Numeric** type then it can be assigned without using double-quotes.

<html>

<head>

<title>Let’s learn assigning values to variables</title>

</head>

<body>

<script type=”text/vbscript”>

Dim val

val = 10

Msgbox val ‘this will show 10 in the message box

</script>

</body>

</html>

Now, let’s understand the scope of the variables.

### Scope of the Variables

The scope of the Variables provides an easy to see the visibility of a variable to the point to which it can be accessible. Let’s discuss it in detail to understand in a better way.

A variable can be declared by using any of the 3 keywords – ***Dim, Public or Private***.

The question which arises now is “what is the difference in their usage and when is a particular one used?” I will try to explain each of these keywords with the help of examples.

***Dim –*** If a variable is declared using Dim keyword inside the function then its scope will be limited to the function level only i.e. this variable cannot be accessed once the function ends.

**Following is an Example for Dim:**

<html>

<head>

<title>Let’s learn about scope of the variables</title>

</head>

<body>

<script type=”text/vbscript”>

Dim val1

Call output()

Function output()

val1 = 11

Dim val2

val2 = 12

Dim val3

val3 = (val1 + val2) \* 10

document.write(val1) ‘this will produce 11 as output

document.write(val2) ‘this will produce 12 as output

document.write(val3) ‘this will produce 230 as output

End Function

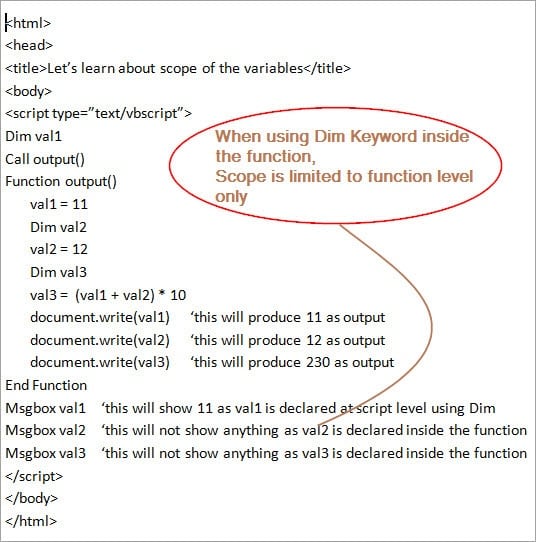
Msgbox val1 ‘this will show 11 as val1 is declared at script level using Dim Msgbox val2

‘this will not show anything as val2 is declared inside the function

Msgbox val3 ‘this will not show anything as val3 is declared inside the function </script>

</body>

</html>

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/01/Dim-Keyword.jpg)

***Private –***If a variable is declared using Private keyword inside the function then its scope will not be limited to function level alone, but it can be accessed everywhere in that particular script. Here, the scope of a variable is wider than in the case of Dim as it’s not limited to the function level but a particular script.

**Following is an Example for Private:**

<html>

<head>

<title>Let’s learn about scope of the variables</title>

</head>

<body>

<script type=”text/vbscript”>

Dim val1

Call output()

Function output()

val1 = 11

Private val2

val2 = 12

Private val3

val3 = (val1 + val2) \* 10

document.write(val1) ‘this will produce 11 as output

document.write(val2) ‘this will produce 12 as output

document.write(val3) ‘this will produce 230 as output

End Function

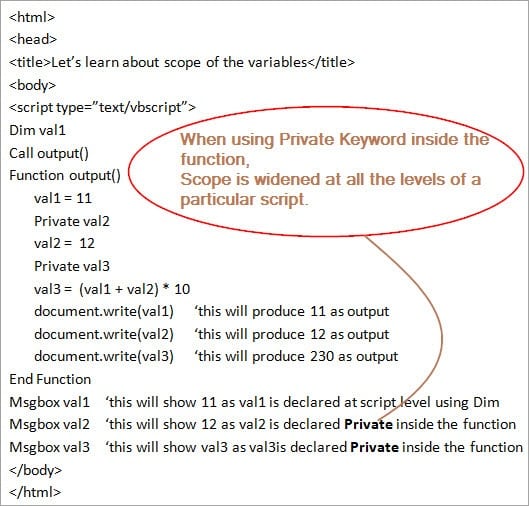
Msgbox val1 ‘this will show 11 as val1 is declared at script level using Dim Msgbox val2 ‘this will show 12 as val2 is declared Private inside the function

Msgbox val3 ‘this will show val3 as val3is declared Private inside the function

</script>

</body>

</html>

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/01/Private-Keyword.jpg)

***Public –*** If a variable is declared using Public keyword inside the function then its scope will not be limited till function level, but it can be accessed everywhere in that particular script and in other scripts too. Here, the scope of a variable is much wider than that in case of Dim and Private.

**Following is an Example for Public:**

<html>

<head>

<title>Let’s learn about scope of the variables</title>

</head>

<body>

<script type=”text/vbscript”>

Dim val1

Call output()

Function output()

val1 = 11

Public val2

val2 = 12

Public val3

val3 = (val1 + val2) \* 10

document.write(val1) ‘this will produce 11 as output

document.write(val2) ‘this will produce 12 as output

document.write(val3) ‘this will produce 230 as output

End Function

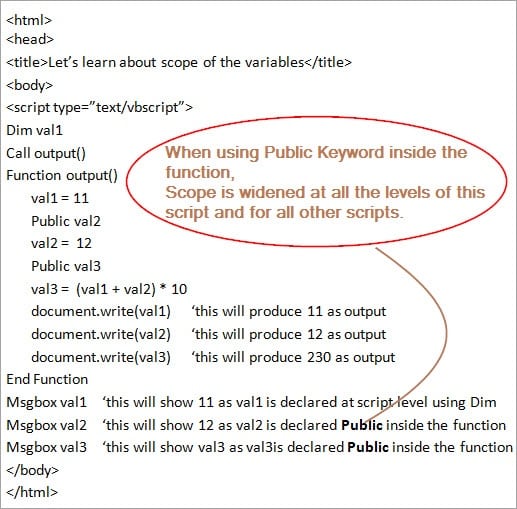
Msgbox val1 ‘this will show 11 as val1 is declared at script level using Dim Msgbox val2 ‘this will show 12 as val2 is declared Public inside the function

Msgbox val3 ‘this will show val3 as val3is declared Public inside the function

</script>

</body>

</html>

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/01/Public-Keyword.jpg)

Let’s move ahead towards the important topic “Declaring and Assigning Values to Arrays”.

### Declaring & Assigning Values to Arrays

Arrays itself is a very important topic to understand. Hence will discuss Arrays in detail in one of my coming tutorials but right now I just wish to show you the declaration of array variables.

**The need for an Array arises when you want to have a variable which can carry a series of values in one memory location.**

**Following is the procedure to declare an Array:**

**Dim Arraycount (4)** – This means that this is an array having 5 values starting from index ‘0’.

To assign values to this array, you can give any numeric or string type values for each index. Use double quotes for assigning string values.

array count (0) = 1  
array count (1) = 2  
array count (2) = 3  
array count (3) = 4  
array count (4) = 5

**Multidimensional Array**: When there is more than one array then it is known as a Multidimensional array. Let’s assume that there is an array of 2 dimensions i.e. there are 6 rows and 6 columns, hence it can be declared as follows:

Dim arrayage (6, 6)

### Conclusion

This was all about Variables, their usage, and scope in VBScript Programming Language.

I hope that this tutorial will be easier for you to proceed further with other subsequent topics of this series.

[**Next Tutorial #3**](https://www.softwaretestinghelp.com/vbscript-constants-and-operators-tutorial-3/)**:** We will discuss more on Constants, Operators and Operator precedence in the next tutorial.

# Constants, Operators, And Precedence Of Operators In VBScript

**What You Will Learn:**[[hide](https://www.softwaretestinghelp.com/vbscript-constants-and-operators-tutorial-3/)]

* [Constants, Operators & Operators Precedence in VBScript](https://www.softwaretestinghelp.com/vbscript-constants-and-operators-tutorial-3/#Constants_Operators_Operators_Precedence_in_VBScript)
* [Declaring and Assigning Values to Constants](https://www.softwaretestinghelp.com/vbscript-constants-and-operators-tutorial-3/#Declaring_and_Assigning_Values_to_Constants)
* [VBScript Data Type Constants](https://www.softwaretestinghelp.com/vbscript-constants-and-operators-tutorial-3/#VBScript_Data_Type_Constants)
* [VBScript String Constants](https://www.softwaretestinghelp.com/vbscript-constants-and-operators-tutorial-3/#VBScript_String_Constants)
* [VBScript Date and Time Constants](https://www.softwaretestinghelp.com/vbscript-constants-and-operators-tutorial-3/#VBScript_Date_and_Time_Constants)
* [Different Types of Operators](https://www.softwaretestinghelp.com/vbscript-constants-and-operators-tutorial-3/#Different_Types_of_Operators)
  + [#1) Arithmetic Operators](https://www.softwaretestinghelp.com/vbscript-constants-and-operators-tutorial-3/#1_Arithmetic_Operators)
  + [#2) Comparison Operators](https://www.softwaretestinghelp.com/vbscript-constants-and-operators-tutorial-3/#2_Comparison_Operators)
  + [#3) Logical Operators](https://www.softwaretestinghelp.com/vbscript-constants-and-operators-tutorial-3/#3_Logical_Operators)
  + [#4) Concatenation Operators](https://www.softwaretestinghelp.com/vbscript-constants-and-operators-tutorial-3/#4_Concatenation_Operators)
* [Operators Precedence](https://www.softwaretestinghelp.com/vbscript-constants-and-operators-tutorial-3/#Operators_Precedence)
* [Conclusion](https://www.softwaretestinghelp.com/vbscript-constants-and-operators-tutorial-3/#Conclusion)
* [Recommended Reading](https://www.softwaretestinghelp.com/vbscript-constants-and-operators-tutorial-3/#Recommended_Reading)

### Constants, Operators & Operators Precedence in VBScript

**Constants** are named memory locations within a program that never changes their values during the execution of the script.

**Operators,**as the name suggests, are used for performing some operations on values or we can say these are used to change the variables and values.

When working with an expression, there are some rules that are to be followed to in order to evaluate each part of the expression in the VBScript language. And any predetermined order that is followed to evaluate the various operations that are performed is known as **Operator Precedence**.

Moving forward we will discuss them all in detail.

### Declaring and Assigning Values to Constants

Constants are declared in the same manner as Variables, but with a small difference that the value of the constant remains same throughout its lifetime i.e., you cannot change its value, unlike variable.

You can use **‘const’** keyword to declare Constants in your script. Constants can be declared as public or private depending on its usage. If it’s declared **Public** then it can be used everywhere in a particular script and also in all the other scripts and procedures whereas if it’s declared as **Private** then it can be used only in one particular script in which you are working including its procedures and classes.

Using Const keyword with some meaningful names, you can create numeric, string and date type constants and also assign values to them.

***Let’s understand the declaration and usage of a Constant with the help of a Simple Example:***

<html>

<head>

<title>Let’s learn assigning values to constants</title>

</head>

<body>

<script language=”vbscript” type=”text/vbscript”>

const val = 10

const val1 = “Hello Everyone”

const val2 = #09/09/1987#

Msgbox val ‘this will show 10 in the message box

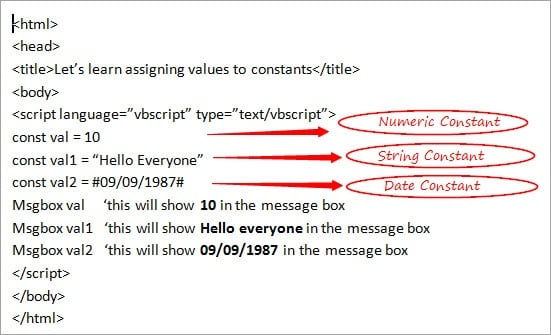
Msgbox val1 ‘this will show Hello everyone in the message box

Msgbox val2 ‘this will show 09/09/1987 in the message box

</script>

</body>

</html>



These are User Defined Constants. There are various predefined Constants that are provided by the VBScript in order to help the users to directly use them in the scripts without any declaration as they are already defined in the VBScript.

Let’s take a look at some of the important VBScript Constants.

### VBScript Data Type Constants

Given below are some of the Data Type Constants in the VBScript.

* **vbEmpty**: This is used for a data type which is not initialized with the value as 0.
* **vbNull**: This is used when there is no valid data with the value as 1.
* **vbBoolean**: This is used for a Boolean data type with the value as 11.
* **vbByte**: This is used for byte data type with the value as 17.
* **vbInteger**: This is used for integer data type with the value as 2.
* **vbLong**: This is used for the long data type with the value as 3.
* **vbSingle**: This is used for a single data type with the value as 4.
* **vbDouble**: This is used for a double data type with the value as 5.
* **vbDate:** This is used for date data type with the value as 7.
* **vbString**: This is used for string data type with the value as 8.
* **vbObject**: This is used for object data type with the value as 9.
* **vbArray**: This is used for array data type with the value as 8192.

### VBScript String Constants

Given below are some of the String Constants in the VBScript.

* **vbCr**: This is used for print and display functions representing a carriage return character, basically for the purpose of returning to the beginning of the line. Value of this is Chr(13).
* **vbCrLf**: This is also used for print and display functions representing a carriage return with line feed character having values as Chr(13) & Chr(10). This works in the same manner as in the above case of pressing the Enter key.
* **vbformfeed**: This is used for ‘form feed’ which means for advancing downwards to the next page or in more common terms, works as a page separator. This has the value of Chr(12) and it does not work in OS Windows.
* **vbLf**: This is used for advancing downwards to the next line i.e. to go to the next line. This represents a line feed character with the value of Chr(10).
* **vbNewLine**: This is used at times in the combination of vbCrLf and vbLf i.e. Chr(13) & Chr(10) and sometimes only vbLf works i.e. Chr(10). This is a platform-dependent one.
* **vbNullChar**: This represents the character which is having the value as 0. This has its value as ‘0’.
* **vbNullString**: This represents a string which is having value 0. This is not equivalent to zero length string
* **vbTab**: This is used for providing horizontal tab, having its value as Chr(9).

### VBScript Date and Time Constants

Given below are some of the Date and Time Constants in the VBScript.

* **vbSunday**: This represents the week of the day ‘Sunday’ holding the value as 1. If you want to work in the script having the usage of Sunday then can use this constant.
* **vbMonday**: This represents the week of the day ‘Monday’ holding the value as 2. If you want to work in the script having the usage of Monday then can use this constant.
* **vbTuesday**: This represents the week of the day ‘Tuesday’ holding the value as 3. If you want to work in the script having the usage of Tuesday then can use this constant.
* **vbWednesday**: This represents the week of the day ‘Wednesday’ holding the value as 4. If you want to work in the script having the usage of Wednesday then can use this.
* **vbThursday**: This represents the week of the day ‘Thursday’ holding the value as 5. If you want to work in the script having the usage of Thursday then can use this constant.
* **vbFriday**: This represents the week of the day ‘Friday’ holding the value as 6. If you want to work in the script having the usage of Friday then can use this constant.
* **vbSaturday**: This represents the week of the day ‘Saturday’ holding the value as 7. If you want to work in the script having the usage of Saturday then can use this constant.
* **vbFirstFourDays**: This makes use of the week that has at least 4 days in the new year, holding the value as 2.

***Let’s see a simple Example using some of the user-defined constants that we have discussed above.***

<html>

<head>

<title>Let’s see the usage of user defined constants</title>

</head>

<body>

<script language=”vbscript” type=”text/vbscript”>

Dim val, val1, val2

val = “abcd”

val1=”hello” & vbNewLine & “how are you?”

val2 =WeekdayName(1)

Select Case VarType(val)

Case vbEmpty

Msgbox “this is an empty variable”

Case vbNull

Msgbox “this is a null variable”

Case vbString

Msgbox “this is a string variable”

Case vbByte

Msgbox “this is a variable of byte type”

End Select

Msgbox val1 ‘this will show result as follows as vbNewLine is used:

‘hello

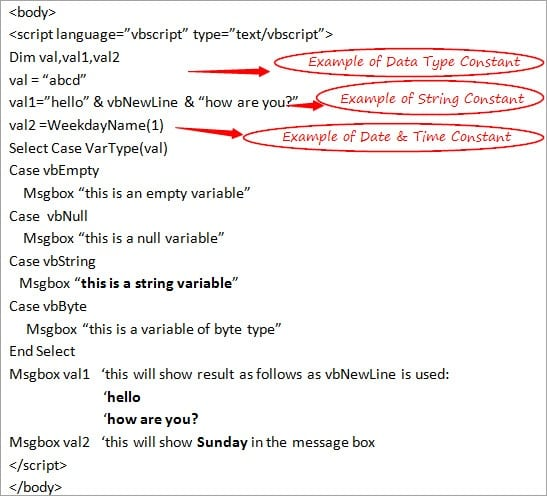
‘how are you?

Msgbox val2 ‘this will show Sunday in the message box

</script>

</body>

</html>



**Note**: In the above example in val2, I have passed ‘1’ in weekdayname which is same as passing as vbSunday. ‘VarType’ is a function which returns a subtype of the variable. I will discuss this in functions in one of my next tutorials.

Let’s move on to Operators now.

### Different Types of Operators

Operators are very important in any programming language as without them the tasks cannot be performed.

If you want to perform a simple operation of multiple then the following is the way to do so:

multiply = a \* b

Here, a, b and multiply are the **Operands** while ‘=’ and ‘\*’ are the **Operators**.

***There are mainly 4 types of Operators in the VBScript language.***

***Let’s discuss them in detail by taking 2 operands as 1 and 2.***

#### **#1) Arithmetic Operators**

All of us are already aware of arithmetical operators in Mathematics, same applies here too.

**Different arithmetic operations are:**

* **Addition**: This performs addition of 2 operands using ‘+’ sign. In our case, addition will be 1 + 2 = 3.
* **Subtraction**: This performs subtraction among 2 operands using ‘-‘sign and hence in our case, the result of the subtraction will be 1-2 = -1.
* **Multiplication**: This performs multiplication using ‘\*’ among the 2 operands. Here, it is 1 \* 2 = 2.
* **Division**: This performs division of the numerator by the denominator using ‘/’. In our case, if we assume numerator is 2 and denominator is 1 then the result will be 2/1 = 2.
* **Modulus**: This is not used very often but it’s good to know about this. This provides ‘remainder’ as a result of division using ‘%’ operator i.e. after dividing 2 by 1, the remainder will be 0 and hence this will give the result as 0.
* **Exponential:**This is used to calculate exponential using ‘^’ operator. In our case, 2^1 will give the exponential result as 2.

#### **#2) Comparison Operators**

We are all already aware of some of the comparison operators in Mathematics, let’s take a look at the various comparison operators in the VBScript language using values as 1 and 2 respectively.

**Various Comparison Operators in the VBScript:**

* **Equal:** This is used to compare if the values of any 2 operands are equal or not. It produces true if the value of both matches otherwise false. Uses ‘==’ notation i.e. in our case, if we check 1 == 2 then it will give False.
* **Not Equal**: This is the reverse of above one i.e. this checks if the values of both the operands do not match and produces true if the condition satisfies using ‘<>’ symbol. In our case, if we match 1<> 2 then it will give True.
* **Greater than:**This checks which of the value out of any of the 2 operands is greater than the other one. This uses ‘>’ symbol and verifies if the left-hand side operand is greater than the right-hand side and produces True if this happens. In our case, if we verify 1>2 then this will produce False.
* **Less than:**This verifies if the operand on the right-hand side is greater than the one on the left-hand side and produces True if this happens. This uses ‘<’ symbol. In our case, if we verify 1<2 then this will produce True.
* **Greater than, Equal to:**This verifies if the operator on the left-hand side is either greater than or equal to the operand on the right-hand side and if this happens, this produces True. ‘>=’ symbol is used for this. In our case, if we check 1 >=2 then this will produce False as neither 1> 2 nor 1=2.
* **Less than, Equal to:**This verifies if the operator on the right-hand side is either greater than or equal to the operand on the left-hand side and if this happens, this produces True. ‘<=’ symbol is used for this. In our case, if we check 1 <=2 then this will produce True as 1 < 2 even if 1 is not equal to 2.

#### **#3) Logical Operators**

Most of us are already aware of most of the logical operators like AND, OR, etc. and we have also used these in many programming languages. Let’s see the list of various logical operators in VBScript language using 2 variables as x and y with values as 1 and 2 respectively.

**Logical Operators include:**

* **AND**: This is the logical AND operator and this produces True only when both the conditions are true or when both the conditions are satisfied i.e. if in our case, I take one condition as **(x>0) AND (x==y)** then this will produce False as both the conditions are not true. 1> 0 is true but 1=2 is not true, hence the result is **False**.
* **OR**: This is the logical OR operator and this produces True even if one of the conditions is true or when either of the conditions is satisfied i.e. if in our case, I take one condition as **(x>0) OR (x==y)** then this will produce True as one of the condition is true . 1> 0 is true, hence the result is **True**.
* **NOT**: This is the logical NOT operator and this works to change the value of the operand or I can say this changes the state of the condition i.e. if the condition is True then this will change it to False and vice versa. If the condition is **NOT (x>0)** then this will produce **False** as 1>0 which is true and NOT of true is false.

#### **#4) Concatenation Operators**

There are 2 Concatenation operators in the VBScript. As the name itself suggests, concatenation means joining. This works for numeric and string values.

**Let’s see its operators as follows:**

* **+:**This is our arithmetic operator ‘+’ only. If we take 2 strings as “Good” and “Bad” then this operator will concatenate this to Good + Bad = GoodBad and if we take 2 numeric values as 1 and 2 so concatenation in this case using ‘+’ will be **1 + 2 = 3.**
* **&:** This is our normal ‘&’ symbol. This concatenates 2 values using this symbol. If we consider the above example itself, then String concatenation will be Good & Bad = GoodBad and Numeric concatenation will be **1 & 2 = 12**

***Let’s clearly understand working with the Operators with the help of a Simple Example:***

<html>

<head>

<title>Let’s see usage of different type of operators in the VBScript</title>

</head>

<body>

<script language=”vbscript” type=”text/vbscript”>

Dim val, val1, val2, val3

Const a = 5

Const b = 10

Const c = “hey”

val = a + b

val1 = (a == b) OR (a<=10)

val2 = a & c

val3 = a & b

Msgbox val ‘this will give result as 15

Msgbox val1 ‘this will give result as True

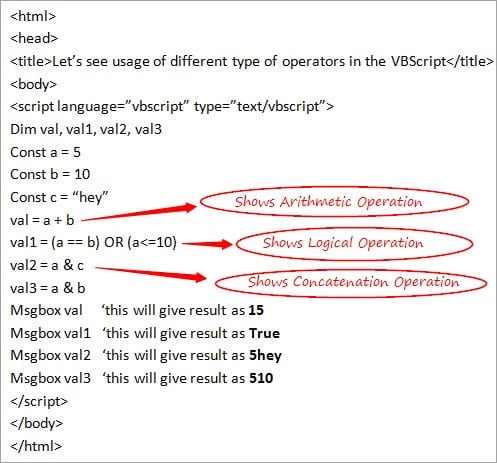
Msgbox val2 ‘this will give result as 5hey

Msgbox val3 ‘this will give result as 510

</script>

</body>

</html>

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/01/Operations.jpg)

We have seen all about the Operators. Next, let’s discuss Precedence of Operators in the VBScript.

### Operators Precedence

It’s become very important to follow the precedence of operators when there are various operators in a single expression. Hence there is a need of some predetermined order on the basis of which expression can be evaluated easily and quickly.

We have seen regarding the various types of operators, and as per precedence law,

* **Arithmetic Operators** are evaluated firstly as compared to the other operators.
* Next comes the turn of **Comparison Operators** for evaluation.
* Lastly, **Logical Operators** are evaluated.

There is a fixed order for arithmetic operators in which they are evaluated.

**Order of Arithmetic Operators**

* Exponentiation
* Multiplication
* Division
* Modulus
* Addition and Subtraction
* Concatenation

**Logical Operators are Evaluated as**

* NOT
* AND
* OR
* XOR

***Note***: If you use parenthesis in an expression then its precedence is above all of these and the expression which is held inside the bracket is evaluated first.

**I will explain this concept with help of a Simple Example**

<html>

<head>

<title>Let’s understand about precedence of operators</title>

</head>

<body>

<script language=”vbscript” type=”text/vbscript”>

Dim a

Dim b

a=4+8/8\*2^1

b= (4+8)/ (8\*2) ^1

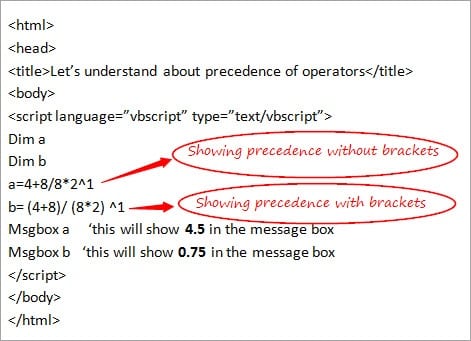
Msgbox a ‘this will show 4.5 in the message box

Msgbox b ‘this will show 0.75 in the message box

</script>

</body>

</html>

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/01/Precedence.jpg)

### Conclusion

Through this tutorial, we have discussed all about Constants, Operators, and Precedence of Operators in the VBScript. I tried to cover all the main topics that are involved in it. And I am sure that this would have given you a great knowledge of the concept.

[***Next Tutorial #4***](https://www.softwaretestinghelp.com/vbscript-conditional-statements-tutorial-4/)***: We will discuss Conditional Statements in VBScript in my next tutorial.***

# VBScript Conditional Statements: VBScript If, ElseIf, Select Case

**What You Will Learn:**[[hide](https://www.softwaretestinghelp.com/vbscript-conditional-statements-tutorial-4/)]

* [Conditional Statements](https://www.softwaretestinghelp.com/vbscript-conditional-statements-tutorial-4/#Conditional_Statements)
* [Various Conditional Statements in the VBScript](https://www.softwaretestinghelp.com/vbscript-conditional-statements-tutorial-4/#Various_Conditional_Statements_in_the_VBScript)
  + [#1) VBScript If…End If Statement](https://www.softwaretestinghelp.com/vbscript-conditional-statements-tutorial-4/#1_VBScript_IfEnd_If_Statement)
  + [#2) VBScript If…Else…End If Statement](https://www.softwaretestinghelp.com/vbscript-conditional-statements-tutorial-4/#2_VBScript_IfElseEnd_If_Statement)
  + [#3) VBScript If…ElseIf…Else…End If Statement](https://www.softwaretestinghelp.com/vbscript-conditional-statements-tutorial-4/#3_VBScript_IfElseIf8230ElseEnd_If_Statement)
  + [#4) VBScript Select Case…End Select Statement](https://www.softwaretestinghelp.com/vbscript-conditional-statements-tutorial-4/#4_VBScript_Select_CaseEnd_Select_Statement)
* [Conclusion](https://www.softwaretestinghelp.com/vbscript-conditional-statements-tutorial-4/#Conclusion)
* [Recommended Reading](https://www.softwaretestinghelp.com/vbscript-conditional-statements-tutorial-4/#Recommended_Reading)

### Conditional Statements

As the name itself explains, **Conditional Statements** refers to a set of statements which are written to meet a particular condition. In simple terms, these are such statements which help in the decision-making process by enabling one to check if any specific condition is met or not as per the requirement of a program.

At times a situation may occur when in a piece of code, some actions are required to be performed for taking some decisions and Conditional Statements satisfy this criterion. These statements are widely used during coding, hence they form the basis of a programming language.

Let me quote a simple **Example** from our day to day life to make you understand this more clearly. Let’s consider a scenario that ‘you have to go to some party today’ but there are some conditions for that. If the cab comes then you will able to go to the party or else you will have to miss the party.

***In such a case, conditional statements can be written as follows:***

If cab comes Then  
   you will go in the party  
Else  
   you will miss the party  
End If

Let’s discuss the various types of Conditional Statements in the VBScript.

**Suggested reading =>>**[**Conditional statements in VBA**](https://www.softwaretestinghelp.com/conditional-statements-in-vba/)

### Various Conditional Statements in the VBScript

VBScript language supports 4 kinds of Conditional Statements.

Most of us must already be aware of these statements while working in other programming languages. Out of these statements, any one statement can be used based on your requirement and it’s up to you to decide and pick the best option for a specific scenario.

***Following is the list of Conditional Statements:***

* VBScript If…End If Statement
* VBScript If…Else…End If Statement
* VBScript If…ElseIf…Else…End If Statement
* VBScript Select Case…End Select Statement

***Let’s discuss each of these statements in detail.***

#### **#1) VBScript If…End If Statement**

This Conditional Statement is the most basic and widely used one out of all the 4 Conditional Statements available. Also, it is very easy to use and understand.

This is used in those scenarios where there is any specific condition to be met and if that condition is satisfied then some specific lines of statements that are written inside the ‘If…End If’ will be executed.

**Let’s understand this with the help of a simple example.**

***Example:***

If you want to display some specific message based on the value of some variable then If…End If Statement can be used in the following manner.

<html>

<head>

<title>Let’s see implementation of If End If Statement</title>

</head>

<body>

<script language=”vbscript” type=”text/vbscript”>

Dim val = “Wow”

If val = “Wow” Then

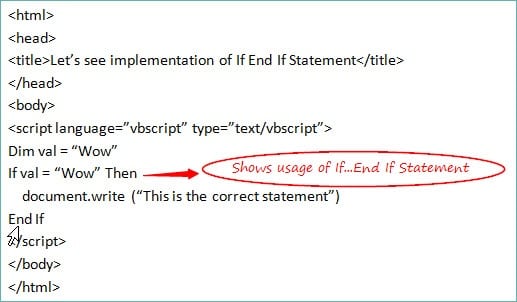
document.write (“This is the correct statement”)

End If

</script>

</body>

</html>

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/01/If-Statement.jpg)

#### **#2) VBScript If…Else…End If Statement**

**This one is the expansion of the If…End If statement.** Under this type, the block of statements is divided into 2 parts – inside the ‘If’ part and inside the ‘Else’ part respectively. Either of the 2 parts is executed immediately depending upon which condition is satisfied.

**Let’s understand this with the help of a simple example.**

***Example:***

If there is an expression which will evaluate either True or False. Then in case of True, some specific message will be displayed and in case of False, some other message will be displayed.

In such a scenario, you can make use of If…Else…End If Statement in the following way.

<html>

<head>

<title>Let’s see implementation of If Else Statement</title>

</head>

<body>

<script language=”vbscript” type=”text/vbscript”>

Dim val1, a, b

a=5

b=10

val1 = (a == b) OR (a<=10)

If val1 = True Then

document.write (“Result of the expression is true”)

Else

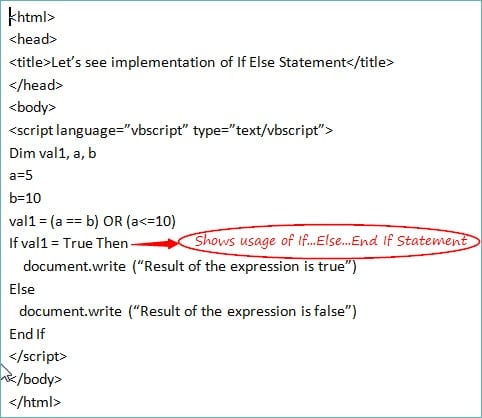
document.write (“Result of the expression is false”)

End If

</script>

</body>

</html>

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/01/If-else-statement.jpg)

#### **#3) VBScript If…ElseIf…Else…End If Statement**

**This method is an extension of the earlier one i.e. If…Else…End If Statement.** When a set of statements are divided into multiple blocks along with the If and Else part and if some other parts are also required in a code then you can use this Conditional Statement.

At times, a situation may arise when a condition is not satisfied in the ‘If’ part nor in the ‘Else’ part and some other conditions are required to be checked.

In the above-mentioned cases **If…ElseIf…Else…End If Statement** can be used.

**Let’s understand this clearly with the help of an Example.**

***Example:***

Let’s assume a case where you need to check the data type of a variable as whether it’s of Boolean type, string type, etc. Then in such a scenario you have to check multiple conditions and must move in various blocks one by one and if nothing satisfies the condition then finally, you will move in the Else part of the code.

**Let’s take a look at its implementation.**

<html>

<head>

<title>Let’s see implementation of If ElseIf Statement</title>

</head>

<body>

<script language=”vbscript” type=”text/vbscript”>

Dim val

val = “abcd”

val1 = VarType(val)

If val1 = vbEmpty Then

document.write (“This is an Empty variable”)

ElseIf val1=vbNull

document.write (“This is a Null variable”)

ElseIf val1=vbString

document.write (“This is a variable of String data type”)

ElseIf val1=vbBoolean

document.write (“This is a variable of Boolean data type”)

Else

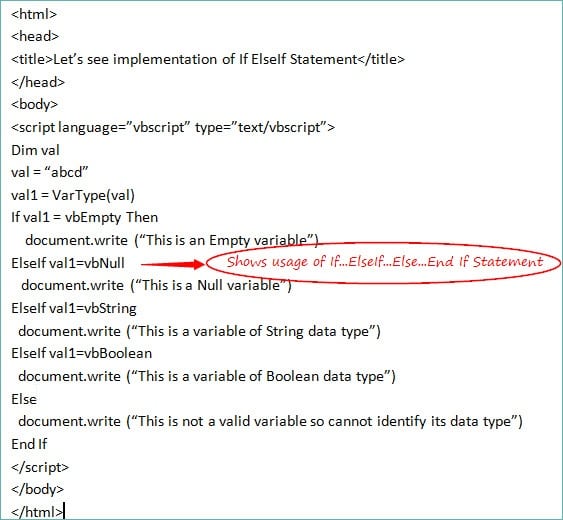
document.write (“This is not a valid variable so cannot identify its data type”)

End If

</script>

</body>

</html>

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/01/If-Else-if-statement.jpg)

#### **#4) VBScript Select Case…End Select Statement**

This Conditional Statement works in the same manner as the above one i.e. If…ElseIf…Else…End If Statement but as per the observation Select Case is the better option when it comes to working with multiple sets of conditions in a program.

When you work with ‘If…ElseIf…Else…End If Statement’ you have to move each condition or block one by one until you reach the expected one and this consumes a lot of time and effort. Whereas in case of Select Statement, you can directly move to the exact case without wasting time by going to each condition one by one.

**Let’s see the implementation of the Select Case by making reference to the above example itself.**

***Example:***

<html>

<head>

<title>Let’s see implementation of Select Case Statement</title>

</head>

<body>

<script language=”vbscript” type=”text/vbscript”>

Dim val

val = “abcd”

Select Case VarType(val)

Case vbEmpty

document.write (“This is an Empty variable”)

Case vbNull

document.write (“This is a Null variable”)

Case vbString

document.write (“This is a variable of String data type”)

Case vbBoolean

document.write (“This is a variable of Boolean data type”)

Case Else

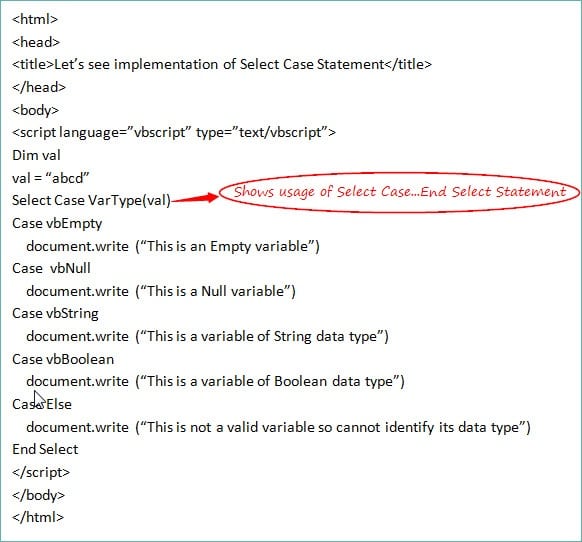
document.write (“This is not a valid variable so cannot identify its data type”)

End Select

</script>

</body>

</html>

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/01/Select-Statement.jpg)

### Conclusion

I hope that by the end of this tutorial of [VBScript Series](https://www.softwaretestinghelp.com/vbscript-tutorial-1/) you will be able to understand the subsequent tutorials of this series with ease and comfort.

[**Next Tutorial #5**](https://www.softwaretestinghelp.com/vbscript-loops-tutorial-5/)**:** I will brief about the various types of loops in the VBScript language in my upcoming tutorials.

# VBScript Loops: For Loop, Do Loop, And While Loop

**What You Will Learn:**[[hide](https://www.softwaretestinghelp.com/vbscript-loops-tutorial-5/)]

* [What are Loops?](https://www.softwaretestinghelp.com/vbscript-loops-tutorial-5/#What_are_Loops)
* [Different types of Loops in the VBScript](https://www.softwaretestinghelp.com/vbscript-loops-tutorial-5/#Different_types_of_Loops_in_the_VBScript)
  + [#1) For Loop](https://www.softwaretestinghelp.com/vbscript-loops-tutorial-5/#1_For_Loop)
  + [#2) Do Loop](https://www.softwaretestinghelp.com/vbscript-loops-tutorial-5/#2_Do_Loop)
  + [#3) While Loop](https://www.softwaretestinghelp.com/vbscript-loops-tutorial-5/#3_While_Loop)
* [Conclusion](https://www.softwaretestinghelp.com/vbscript-loops-tutorial-5/#Conclusion)
* [Recommended Reading](https://www.softwaretestinghelp.com/vbscript-loops-tutorial-5/#Recommended_Reading)

### What are Loops?

Generally, Loop means to repeat something several times. In the same way, Loops in the VBScript means those statements in the code which can be repeated several times until any particular condition reaches to an end.

A sequence is followed while using a loop and the statement which comes at the beginning of the code is executed first and so on. Whenever repetitions of some particular statements are required in the code then loops are used until the condition is fulfilled.

***Let me take a simple example to explain the concept easily.***

***Example:***

If you want to send an invitation to 10 people with the same message then you can use ‘for loop’ in this case as a counter is fixed and you know the message which is to be repeated 10 times.

***The syntax of the loop will be as follows:***

For i = 1 to 10  
Msgbox “Please do come to my party”  
Next

Let’s move to the different types of loops that are supported by VBScript.

### Different types of Loops in the VBScript

There are several types of Loops in the VBScript which can be used under various scenarios based on the requirements of a code.

***Broadly, there are 3 types of loops in the VBScript, which are as follows:***

* **For Loop**
* **Do Loop**
* **While Loop**

These types further include some other loops as well. Let’s discuss them one-by-one.

#### **#1) For Loop**

This is the most basic and widely used loop. This is used in those scenarios where you know the fixed number of times for executing the statements in a code i.e. there is some **fixed number of times/fixed iteration count** to perform a condition.

***Also read =>>***[***For Loop in Java***](https://www.softwaretestinghelp.com/java-for-loop-tutorial/)

***Example to show the usage of ‘For Loop’ is as follows:***

<html>

<head>

<title>Let’s see implementation of For Loop</title>

</head>

<body>

<script language=”vbscript” type=”text/vbscript”>

Dim val

For val = 1 to 4

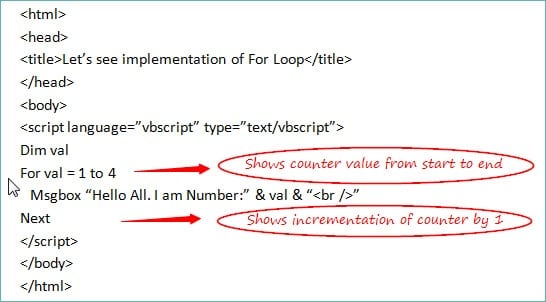
Msgbox “Hello All. I am Number:” & val & “<br />”

Next

</script>

</body>

</html>

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/01/Counter-and-Increment.jpg)

**The output** of this is:

Hello All. I am Number:1  
Hello All. I am Number:2  
Hello All. I am Number:3  
Hello All. I am Number:4

***Let’s understand the working of the code:***

* ‘For Loop’ is starting with a counter value (which we are defining with the variable name ‘var’) of 1 and this will repeat 4 times as the counter is from 1 to 4.
* The statement inside the loop is executed adjoining with the value of the variable.
* The counter will be increased by 1 using ‘Next’ keyword.
* Again the same process will go on and this will last for 4 times as the range is from 1 to 4.

**For Each Loop**

For Each Loop is an extension of For Loop. This is used in case of**‘Arrays’**. When you want to repeat the code for each index value of an array then you can use ‘For Each Loop’. This works in the same manner as the above but the implementation is slightly different.

***Let’s see its usage with the help of a Simple Example:***

<html>

<head>

<title>Let’s see implementation of For Each Loop</title>

</head>

<body>

<script language=”vbscript” type=”text/vbscript”>

Dim array(3)

array(0) = 10

array(1) = 20

array(2) = 30

array(3) = 40

For Each val in array

Msgbox “Hello All. I am Number:” & val & “<br />”

Next

</script>

</body>

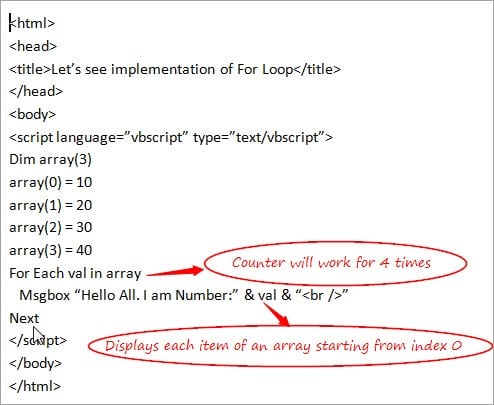
</html>

**The output** of this is:

Hello All. I am Number:10  
Hello All. I am Number:20  
Hello All. I am Number:30  
Hello All. I am Number:40

***Let’s understand the working of the code:***

* An array is defined by the name ‘array’ with the index values ranging from 0 to 3.
* ‘For each loop’ will start from 0 indexes of an array and go on till it reaches to 3 i.e. loop will go 4 times.
* Code written inside the loop will be executed 4 times with value of ‘val’ variable changing as per the index values of an array.
* When all the index values are executed, the loop will come to an end and the cursor will move to the next statement of the loop.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/01/Counter.jpg)

**For Loop with ‘Step’ keyword and ‘Exit For’ Statement**

In the case of ‘For Loop’, the counter is incremented by 1 when it comes to ‘Next’ keyword. But if you wish to change this value and if you want to specify the counter value by yourself then you can do so with the help of ‘**Step**’ keyword. It can be a **positive** or **negative** value depending on the requirement and accordingly it will increase or decrease the counter value.

***Let’s understand the usage of Step Keyword with the help of a Simple Example:***

<html>

<head>

<title>Let’s see implementation of For Loop with Step keyword</title>

</head>

<body>

<script language=”vbscript” type=”text/vbscript”>

Dim val

For val = 1 to 4 Step 2

Msgbox “Hello All. I am Number:” & val & “<br />”

Next

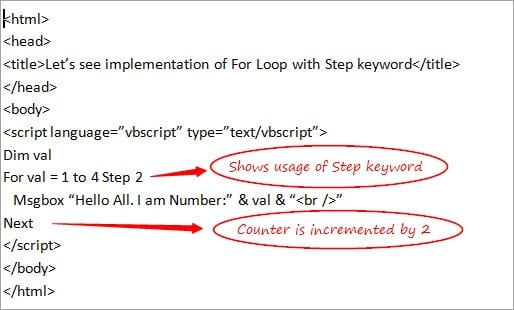
</script>

</body>

</html>

**The output** of this is:

Hello All. I am Number:1  
Hello All. I am Number:3

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/01/Step-Keyword.jpg)

***Let’s see the usage of ‘Exit For’ Statement by taking the reference from the above Example:***

<html>

<head>

<title>Let’s see usage of For Loop with Step keyword and Exit For</title>

</head>

<body>

<script language=”vbscript” type=”text/vbscript”>

Dim val

For val = 1 to 4 Step 2

Msgbox “Hello All. I am Number:” & val & “<br />”

If val = 3 Then

Exit For

End If

Next

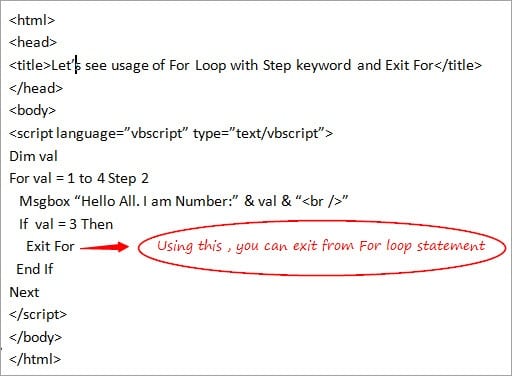
</script>

</body>

</html>

**The output** of this is:

Hello All. I am Number:1

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/01/Exit-Loop-Statement.jpg)

**‘Exit For’** is used to exit from the ‘For Loop’ block of the code. If anytime, in between the loop you wish to exit, then you can do so using the ‘Exit For’ Statement. In the above example, ‘For Loop’ is terminated when a value is equal to 3 and hence, the message is displayed only once.

Let’s take a look at the next type of loop.

#### **#2) Do Loop**

Do Loops are used when you are not sure about the number of **repetitions** (unlike in case of For Loop) that can take place in the code on the basis of some conditions.

There are 2 types of Do Loops in the VBScript.

**They are:**

* **Do While Loop**
* **Do Until Loop**

Let’s discuss each of them in detail.

**Do While Loop**

This makes use of keywords ‘Do’ and ‘While’. This can further be divided into **2 cases** depending upon the placement of the ‘Do’ and ‘While’ keywords. In the first case, Do and While are used in the beginning of the loop and in other cases, Do is used in the beginning of the Loop whereas While is used at the end of the loop.

***Let’s see the implementation of both with the help of some Simple Examples:***

**Case 1: Do While….Loop**

<html>

<head>

<title>Let’s see usage of Do While Loop with Exit Do Statement</title>

</head>

<body>

<script language=”vbscript” type=”text/vbscript”>

Dim val

val = 1

Do While val <= 6

Msgbox “This is value “& val

If val = 4 Then

Exit Do

End If

val = val \* 2

Loop

</script>

</body>

</html>

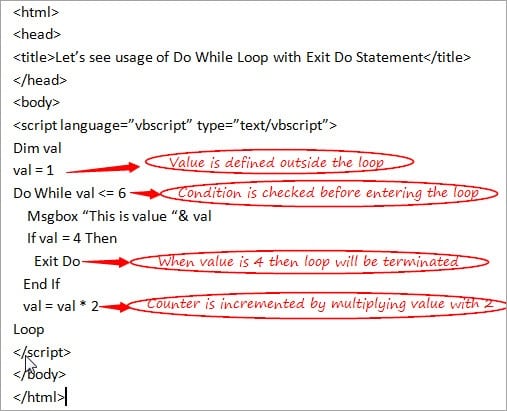
**The output**of this is**:**

This is value 1  
This is value 2  
This is value 4

***Let’s understand the working of the code:***

* Value of a variable (val) is declared and defined explicitly outside the loop unlike in the case of For Loop where it is declared in the For Loop statement only.
* Do while Loop starts with the checking of the condition if the value of a variable is less than or equal to 6.
* The message written inside the loop displays when the condition gets satisfied.
* If the value of a variable is equal to 4 then the loop is terminated as Exit Do statement is used at this point and the cursor will move to the next statement of Do While Loop. Hence no output is produced after the value of the variable becomes equal to 4.
* The counter is then incremented on the basis of the increment condition that is assigned i.e. val \* 2 **unlike** in the case of ‘For Loop’ where the counter is automatically incremented by 1 with the use of ‘Next’ keyword.

**Note**: If the value of a variable is declared as 10 i.e. val = 10 in the above example then Do While Loop cannot be executed even at once as condition val <=6 can never become true.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/01/Loop.jpg)

**Case 2: Do….Loop While**

As I mentioned in above note that Do While maybe not able to execute even at once when the condition is not satisfied at all. Do….While solves this issue and in this case even if the condition is not satisfied but at least**one-time** loop can be executed.

***Let’s understand this concept by taking the reference from the above Example:***

<html>

<head>

<title>Let’s see usage of Do….While Loop </title>

</head>

<body>

<script language=”vbscript” type=”text/vbscript”>

Dim val

val = 10

Do

Msgbox “This is value “& val

val = val \* 2

Loop While val <= 6

</script>

</body>

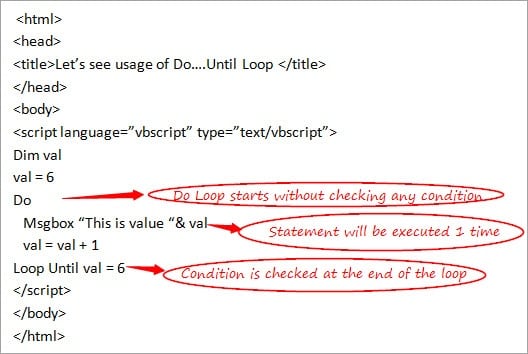
</html>

**The output**of this is**:**

This is value 10

***Let’s understand the working of the code:***

* Value of a variable (val) is declared and defined explicitly outside the loop i.e. val = 10.
* Do Loop starts without the checking of the condition (value of a variable is less than or equal to 6) and the Message written inside the loop will be executed i.e. loop will execute at least once.
* The counter is then incremented on the basis of the increment condition that is assigned i.e. val \* 2 i.e. 10 \* 2 = 20.
* Finally, the condition is checked at the end of the loop which will fail as val = 10 which is not less than 6. Hence, Do While Loop will get terminated here.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/01/Do-Loop-Starts.jpg)

**Do Until Loop**

This works in the same manner as ‘Do While’ Loops but with a difference that the **Do While** loop initially checks the condition and if it is**true** only after that the statements are executed and in the case of **Do Until**, the loop will be executed until the condition becomes**false**. This is used when you are not sure about the number of times when the loop can be executed.

Do Until Loop is also divided into 2 cases like in the case of Do While.

***Let’s take a look at their usage with the help of simple Examples:***

**Case 1: Do Until….Loop**

<html>

<head>

<title>Let’s see usage of Do Until Loop</title>

</head>

<body>

<script language=”vbscript” type=”text/vbscript”>

Dim val

val = 1

Do Until val = 6

Msgbox “This is value “& val

val = val + 1

Loop

</script>

</body>

</html>

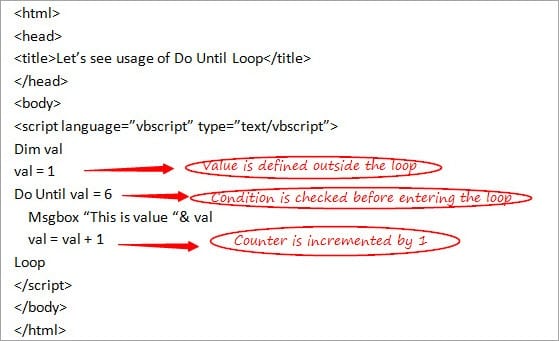
**The output**of this is**:**

This is value 1  
This is value 2  
This is value 3  
This is value 4  
This is value 5

***Let’s understand the working of the code:***

* Value of a variable (val) is declared and defined explicitly outside the loop i.e. val = 1.
* ‘Do Until’ Loop starts with the checking of the condition that the value of a variable should not be equal to 6.
* The message written inside the loop displays when the condition is satisfied.
* Counter is then incremented on the basis of the increment condition that is assigned i.e. here it is incrementing by 1 i.e. val = val + 1
* Loop will work till val = 5 as when the val becomes 6 then the condition becomes false and the loop will come to an end.

**Note**: If the value of a variable is declared as 6 (val = 6) in the above example then ‘Do Until’ Loop cannot be executed even at once as when val =6, the condition becomes false and a loop cannot be executed at all.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/01/Condition-Check-1.jpg)

**Case 2: Do….Loop Until**

As mentioned in the above note that ‘Do Until’ loop may not be able to execute even at once when the condition is not satisfied at all; Do….Until solves this issue and in this case even if the condition is not satisfied, at least **one-time** loop can be executed.

***Let’s understand this concept by taking the reference from the above Example:***

<html>

<head>

<title>Let’s see usage of Do….Until Loop </title>

</head>

<body>

<script language=”vbscript” type=”text/vbscript”>

Dim val

val = 5

Do

Msgbox “This is value “& val

val = val + 1

Loop Until val = 6

</script>

</body>

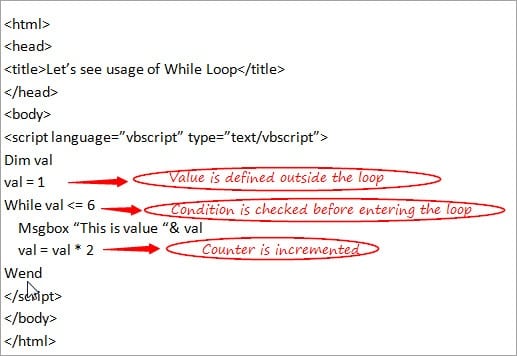
</html>

**The output**of this is**:**

This is value 5

***Let’s understand the working of the code:***

* Value of a variable (val) is declared and defined explicitly outside the loop i.e. val = 6.
* ‘Do’ Loop starts without checking the condition if the value of a variable is less than 6 and the Message written inside the loop will be executed i.e. loop will execute at least once.
* Counter is then incremented on the basis of the increment condition that is assigned i.e. val + 1 i.e. 6 + 1 = 7.
* Finally, the condition is checked at the end of the loop which will fail as the val is equal to 6 and hence ‘Do Until’ Loop will be terminated.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/01/Conter-Incremented.jpg)

#### **#3) While Loop**

However, this is same as the ‘Do While’ loop which we discussed just now but as it’s good to know about all the types of loops, let’s see about this too. This is also used when you are not sure about the number of **repetitions** in a loop. This tests the condition before entering the loop.

***Let’s understand this loop with the help of a Simple Example:***

<html>

<head>

<title>Let’s see usage of While Loop</title>

</head>

<body>

<script language=”vbscript” type=”text/vbscript”>

Dim val

val = 1

While val <= 6

Msgbox “This is value “& val

val = val \* 2

Wend

</script>

</body>

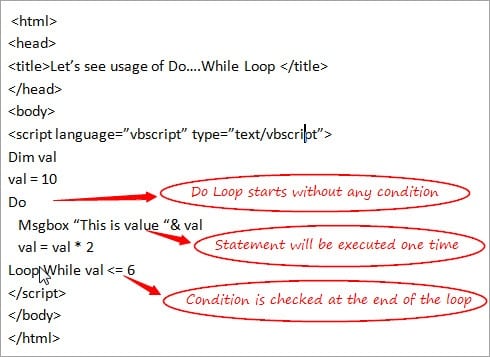
</html>

**The output**of this is**:**

This is value 1  
This is value 2  
This is value 4

***Let’s understand the working of the code:***

* Value of a variable (val) is declared and defined explicitly outside the loop i.e. val = 1.
* ‘While’ Loop starts with checking of the condition if the value of a variable is less than or equal to 6
* Message written inside the loop displays when the condition is satisfied
* The counter is then incremented on the basis of the increment condition that is assigned i.e. val will be multiplied by 2 every time when the condition satisfies.
* When the value of a variable becomes more than 6, the loop will come to an end and the statements written after ‘Wend’ keyword will be executed.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/01/Do-Loop.jpg)

### Conclusion

I hope that you must have gained good knowledge about the meaning and different types of loops in the VBScript through this tutorial. This, in turn, will help you in proceeding with the upcoming tutorials of the series.

[***Next Tutorial #6***](https://www.softwaretestinghelp.com/vbscript-procedures-functions-tutorial-6/)***: We will discuss ‘Procedures and Functions’ in the VBScript in my next tutorial.***

# VBScript Functions And Procedures

**What You Will Learn:**[[hide](https://www.softwaretestinghelp.com/vbscript-procedures-functions-tutorial-6/)]

* [Overview of Functions And Procedures](https://www.softwaretestinghelp.com/vbscript-procedures-functions-tutorial-6/#Overview_of_Functions_And_Procedures)
* [Types of Procedures in the VBScript](https://www.softwaretestinghelp.com/vbscript-procedures-functions-tutorial-6/#Types_of_Procedures_in_the_VBScript)
  + [#1) Sub Procedures](https://www.softwaretestinghelp.com/vbscript-procedures-functions-tutorial-6/#1_Sub_Procedures)
  + [#2) Function Procedures](https://www.softwaretestinghelp.com/vbscript-procedures-functions-tutorial-6/#2_Function_Procedures)
* [Difference Between Sub Procedure and Function Procedure](https://www.softwaretestinghelp.com/vbscript-procedures-functions-tutorial-6/#Difference_Between_Sub_Procedure_and_Function_Procedure)
* [Ways to Pass a Value in the Function](https://www.softwaretestinghelp.com/vbscript-procedures-functions-tutorial-6/#Ways_to_Pass_a_Value_in_the_Function)
  + [#1) Pass by Value:](https://www.softwaretestinghelp.com/vbscript-procedures-functions-tutorial-6/#1_Pass_by_Value)
  + [#2) Pass by Reference:](https://www.softwaretestinghelp.com/vbscript-procedures-functions-tutorial-6/#2_Pass_by_Reference)
* [Inbuilt Functions in the VBScript](https://www.softwaretestinghelp.com/vbscript-procedures-functions-tutorial-6/#Inbuilt_Functions_in_the_VBScript)
  + [#1) Format Functions](https://www.softwaretestinghelp.com/vbscript-procedures-functions-tutorial-6/#1_Format_Functions)
  + [#2) Math and Conversion Functions](https://www.softwaretestinghelp.com/vbscript-procedures-functions-tutorial-6/#2_Math_and_Conversion_Functions)
* [Conclusion](https://www.softwaretestinghelp.com/vbscript-procedures-functions-tutorial-6/#Conclusion)
* [Recommended Reading](https://www.softwaretestinghelp.com/vbscript-procedures-functions-tutorial-6/#Recommended_Reading)

### Overview of Functions And Procedures

When a need arises to accomplish a particular task then a piece of code can be written including several statements under a named section inside a program which is known as **Function/Procedure** in terms of programming language.

Functions and Procedures are mainly used to provide assistance for arranging the code in a program in an organizable way. Using Function/Procedure, the same code can be used multiple times by calling them and this reduces the pain of writing the same code again and again at the Script level.

This is a good programming practice to follow some guidelines while creating the scripts and by working with the functions and procedures you can ensure this to happen. Instead of writing the code in a linear manner in the scripts, it is advisable to divide the scripts into Procedures to provide better readability and understanding of the Scripts.

Dividing the scripts is an important part while dealing with the creation of frameworks in QTP/UFT. As they help in making the code manageable, it is easy to debug and less complex to use.

This is just an overview of Functions and Procedures.

Let’s move on to the next topics to gain more knowledge about Functions and Procedures.

### Types of Procedures in the VBScript

Basically, there are two different types of Procedures in the VBScript.

* **Sub Procedures**
* **Function Procedures**

Both are same in usage as both work for providing reusability of code but with few differences, let’s discuss each of them in detail along with some examples.

#### **#1) Sub Procedures**

This is a type of procedure which includes a set of statements inside the block of code and after execution, it **does not return** any value.

This can be defined by making use of keywords like ‘Sub’ and ‘End Sub’ respectively. It may or may not take an input.

This can be parameterized i.e. can take arguments inside the brackets if required.

***Below is an Example to show the usage of Sub Procedure:***

<html>

<head>

<title>Let’s see implementation of Sub Procedure</title>

</head>

<body>

<script language=”vbscript” type=”text/vbscript”>

Sub Returnvalue()

Msgbox “Let’s get back to work!!”

End Sub

Call Returnvalue()

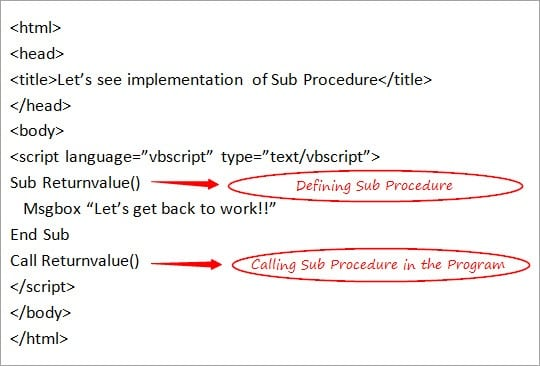
</script>

</body>

</html>

**The output** **is**: Let’s get back to work!!

In the above example, Sub Procedure is created without arguments and is called in a program using **‘Call’** keyword following the name of the Sub procedure. If you do not want to use ‘Call’ keyword while calling Sub procedure then you can simply call it by just writing the name of the Sub Procedure without making use of the Call keyword.



Next, let’s see about Function Procedures.

#### **#2) Function Procedures**

This is a type of procedure which includes a set of statements inside the block of the code and after execution **may return** value also. This can take an input if required, depending upon the situations.

This can be defined by making use of keywords like ‘Function’ and ‘End Function’ respectively.

Function Procedures when used with arguments i.e. by taking parameters inside the brackets, will **return a value** in such a case.

***Below is an Example to show the usage of Function Procedure:***

<html>

<head>

<title>Let’s see implementation of Function Procedure</title>

</head>

<body>

<script language=”vbscript” type=”text/vbscript”>

Function addition(a,b)

Dim result

result = a + b

addition = result ‘storing value of result in the name of the function

End Function

Dim a, b, output

a= 10

b=20

output = addition(10,20)

Msgbox(output)

</script>

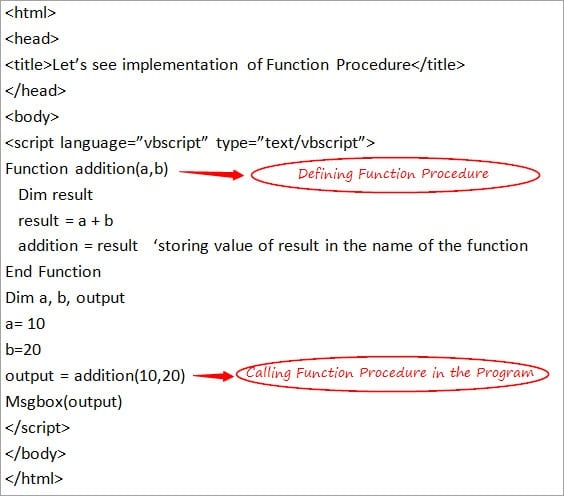
</body>

</html>

**The Output** **is**: 30

In the above example, Function Procedure is created using arguments and is called in a program by using the name of the Function procedure. This is the case when the value is returned. Value of a resultant from the function is stored in another variable ‘output’ and its value is displayed in the message box.

Function Procedure without argument works in the same way as shown in the Example of Sub Procedure.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/01/Function-Procedure.jpg)

Let’s summarize few basic differences between these 2 types of procedures.

### Difference Between Sub Procedure and Function Procedure

**The Differences include:**

* Sub Procedure never takes an input while Function Procedure may take an input if required.
* Sub Procedure starts and ends with using Sub and End Sub respectively while Function Procedure starts and ends with Function and End Function respectively.
* The most **important** difference is Sub Procedure never returns a value while the Function Procedure may return a value (as shown above).

### Ways to Pass a Value in the Function

There are 2 ways to pass a value in the function.

**They are:**

#### **#1) Pass by Value:**

While passing an argument, the Changes that take place in the called procedure and if the value of a variable does not remain to persist, then it means it is passing by value.

Keyword used in this case is **ByVal**.

***Let’s understand this with the help of a simple Example:***

<html>

<head>

<title>Let’s see implementation of Pass by Value </title>

</head>

<body>

<script language=”vbscript” type=”text/vbscript”>

Function output1(ByVal a)

Dim result

result = a / 5

output1 = result ‘storing value of result in the name of the function

End Function

Dim a, val

a= 10

val = output1(10)

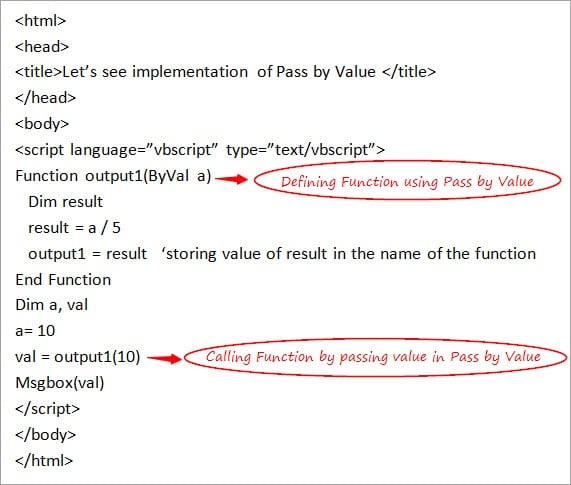
Msgbox(val)

</script>

</body>

</html>

**Output** **is:**10

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/01/Pass-By-Value.jpg)

#### **#2) Pass by Reference:**

While passing an argument, the Changes that take place in the called procedure and if the value of a variable remains to persist then it means it is passing by reference.

Keyword used in this case is **ByRef.**

**Let’s understand this with the help of a simple Example:**

<html>

<head>

<title>Let’s see implementation of Pass by Reference</title>

</head>

<body>

<script language=”vbscript” type=”text/vbscript”>

Function output1(ByRef a)

Dim result

result = a / 5

output1 = result ‘storing value of result in the name of the function

End Function

Dim a, val

a= 10

val = output1(10)

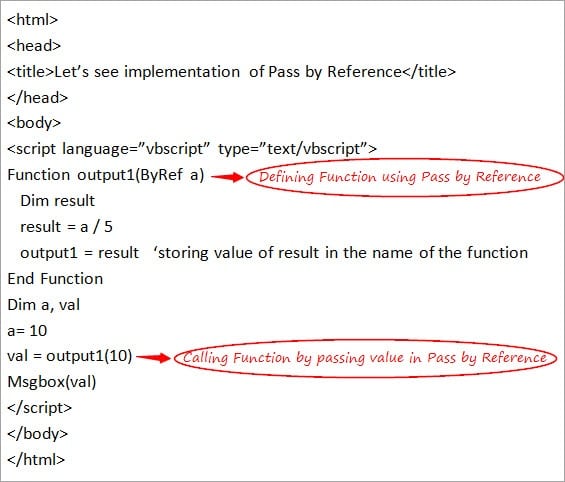
Msgbox(val)

</script>

</body>

</html>

**Output is:** 2



There are several types of In-Built Functions in the VBScript.

Let’s have a look at them in detail.

### Inbuilt Functions in the VBScript

Some important inbuilt functions that can be used directly in the script are explained below in detail.

#### **#1) Format Functions**

There are various inbuilt functions available for formatting the expression and they can be used directly in the script without any need for writing the code explicitly before using them in the script.

* **FormatCurrency:**This is used to convert a specified expression in the form of a Currency value.
* **FormatDateTime:**This is used to convert a specified expression in the form of a Date and Time value.
* **FormatNumber:**This is the most **important** and widely used Format Function and is used to convert the specified expression in the form of a Number.
* **FormatPercent:**This is used to convert the specified expression in the form of a Percentage value.

#### **#2) Math and Conversion Functions**

There are various inbuilt functions for performing mathematical operations and conversion purposes and can be used directly in the script without any need of writing the code explicitly before using them in the script.

* **Abs**: This is the **frequently** used Math function for the purpose of returning the Absolute value for a related number which is specified as a parameter.
* **Int:** This is Math Function and is used in those cases where it is required to fetch an integer part from a given expression/ number.
* **cDate**: This is one of the **frequently** used Conversion Function for converting the expression which includes the Date or Time parameters into Date subtype.
* **cStr:** This is the Conversion function which is used for converting the expression into String subtype.

***Let’s see a simple example using these inbuilt functions***

<html>

<head>

<title>Let’s see implementation of Inbuilt Functions</title>

</head>

<body>

<script language=”vbscript” type=”text/vbscript”>

Dim a , b

a = Abs (-10)

b = CStr(#10-10-17#)

Msgbox(a)

Msgbox(b)

</script>

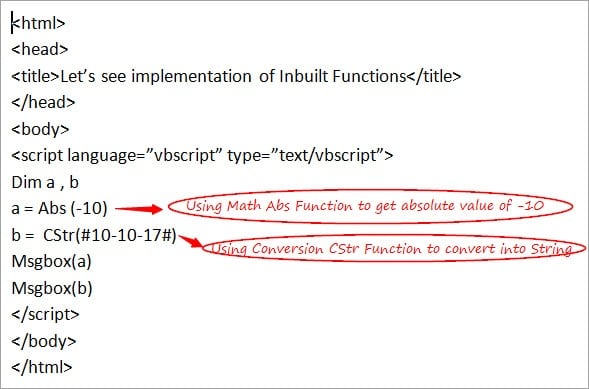
</body>

</html>

**Output** **is**:

10

10/10/2017

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/01/Conversion-Function.jpg)

### Conclusion

I hope that this tutorial would have provided an insight regarding the importance and effectiveness of these reusable statements and this, in turn, will help you in proceeding with subsequent tutorials in an easy manner.

[**Next Tutorial #7**](https://www.softwaretestinghelp.com/vbscript-arrays-tutorial-7/)**: I will cover about ‘VBScript Arrays’ in my next tutorial.**

# VBScript Arrays: Using DIM, REDIM, Split, And Ubound Array Functions

**What You Will Learn:**[[hide](https://www.softwaretestinghelp.com/vbscript-arrays-tutorial-7/)]

* [What is an Array?](https://www.softwaretestinghelp.com/vbscript-arrays-tutorial-7/#What_is_an_Array)
* [Declaration of Arrays in VBScript](https://www.softwaretestinghelp.com/vbscript-arrays-tutorial-7/#Declaration_of_Arrays_in_VBScript)
  + [#1) Way 1: Dim array1()](https://www.softwaretestinghelp.com/vbscript-arrays-tutorial-7/#1_Way_1_Dim_array1)
  + [#2) Way 2: Dim array1(5)](https://www.softwaretestinghelp.com/vbscript-arrays-tutorial-7/#2_Way_2_Dim_array15)
  + [#3) Way 3 : array1 = Array(1,2,3,4,5,6)](https://www.softwaretestinghelp.com/vbscript-arrays-tutorial-7/#3_Way_3_array1_Array123456)
* [Assignment of Values Inside an Array](https://www.softwaretestinghelp.com/vbscript-arrays-tutorial-7/#Assignment_of_Values_Inside_an_Array)
* [Types of Arrays](https://www.softwaretestinghelp.com/vbscript-arrays-tutorial-7/#Types_of_Arrays)
  + [#1) Single Dimensional Array:](https://www.softwaretestinghelp.com/vbscript-arrays-tutorial-7/#1_Single_Dimensional_Array)
  + [#2) Multi-Dimensional Array:](https://www.softwaretestinghelp.com/vbscript-arrays-tutorial-7/#2_Multi-Dimensional_Array)
* [Usage of REDIM Statement and PRESERVE Keyword in an Array](https://www.softwaretestinghelp.com/vbscript-arrays-tutorial-7/#Usage_of_REDIM_Statement_and_PRESERVE_Keyword_in_an_Array)
* [In-Built Array Functions](https://www.softwaretestinghelp.com/vbscript-arrays-tutorial-7/#In-Built_Array_Functions)
* [Conclusion](https://www.softwaretestinghelp.com/vbscript-arrays-tutorial-7/#Conclusion)
* [Recommended Reading](https://www.softwaretestinghelp.com/vbscript-arrays-tutorial-7/#Recommended_Reading)

**What is an Array?**

**An Array** is a variable having named memory location which serves as a **Container** and can hold multiple values in a single location.

*In short, Arrays group different type of elements together*in*one place.*

Let’s take a real-life **example** to understand this better. If you want to store the names of different students at a single place then Array of string type can be used starting at index 0. If you want to fetch the name of the first student then you can pick the value present at index 0 and so on.

**Suggested reading =>>**[**Arrays in VBA**](https://www.softwaretestinghelp.com/vba-array-tutorial/)

Now, let’s move ahead to the next topics to learn how Arrays are actually declared and used in the script.

**Declaration of Arrays in VBScript**

Declaration of an Array can be done in the same manner in which Variables are declared but with the difference that array variable is declared by using **parenthesis** ‘()’.

**The Dim** keyword is used to declare an Array.

**Ways to declare an Array:**

There are 3 ways in which an Array can be declared.

**They are as follows:**

#1) Way 1: Dim array1()

Here, array1 is the name of an array and as parenthesis is empty it means that the size of an array is not defined here.

If you want to declare an array by mentioning its size then it can be done in the following way.

#2) Way 2: Dim array1(5)

In this, array1 is declared with the size as 5 which states it holds 6 values considering that the index of an array always starts from 0. These 5 values can be of integer type, string or character types.

#3) Way 3 : array1 = Array(1,2,3,4,5,6)

Here, Array Function is used to declare an array with a list of arguments inside the parenthesis and all integer values are passed directly inside the parenthesis without any need of mentioning the size of an array.

**Note**: Index value of an Array can never be a negative value.

Next, let’s discuss how to assign values to an array.

**Assignment of Values Inside an Array**

Once an Array is declared, values are assigned to an Array Variable. To assign values, each specific index location is accessed as values are assigned specifically to the index values in an Array.

Taking the reference to the second way of declaring an array (as discussed above), let’s see how to assign values to such an array.

**Dim array1(5)**

Here, the size of an array is 6, which means you have to assign 6 values to an array starting at index 0 and ending at 5.

**So, following is the way to do so:**

array1(0) = “hello”  
array1(1) = 12  
array1(2) = 13  
array1(3) = 14  
array1(4) = 15  
array1(5) = 16

Each index has one specific value.

**Following is an example to show the usage of an Array:**

<html>

<head>

<title>Let’s see implementation of an Array</title>

</head>

<body>

<script language=”vbscript” type=”text/vbscript”>

Dim array1(5)

array1(0) = “hello”

array1(1) = 12

array1(2) = 13

array1(3) = “how are you”

array1(4) = 15

array1(5) = 16

For i = 0 to ubound(array1)

Msgbox “Value present at index ” & i & ” is “ & array1(i) & “<br />”

Next

</script>

</body>

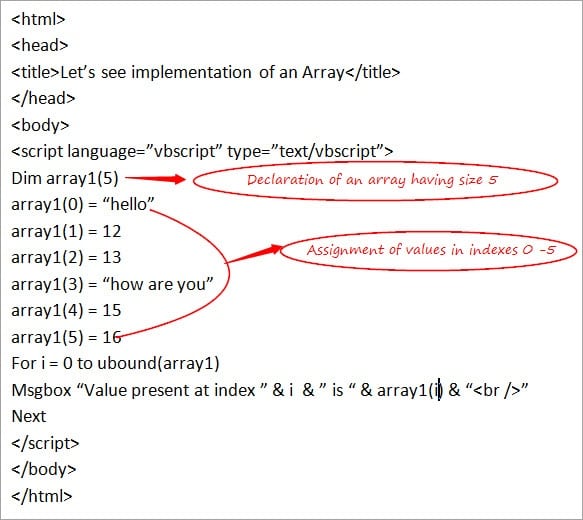
</html>

**Output is:**

Value present at index 0 is hello  
Value present at index 1 is 12  
Value present at index 2 is 13  
Value present at index 3 is how are you  
Value present at index 4 is 15  
Value present at index 5 is 16

In the above example, an array of size ‘5’ is declared and the values are assigned to each index which is the combination of integer and string values. Next, using ‘For loop’, the value present at each index is displayed with the help of a message box. Loop will start from 0 and will go till the unbound i.e. upper bound which is the maximum subscript(5 in this case) of an array.

I will discuss unbound later in this tutorial.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/01/Declaration-Of-Array.jpg)

**Types of Arrays**

There are basically 2 types of Arrays that are used in the VBScript.

**They are:**

#1) Single Dimensional Array:

This is a simple type of array which is used more often in the scripts, the one which is discussed above

#2) Multi-Dimensional Array:

When an array has more than 1 dimension then it is known as a multi-dimensional array. Normally, a **2-Dimensional Array** is the one which is used most of the times i.e. there will be rows and columns in an array. The maximum dimension of an array can reach up to 60.

***Let’s understand the working of a 2 Dimensional Array with the help of a simple example.***

***Example:***

<html>

<head>

<title>Let’s see implementation of a 2 Dimensional Array</title>

</head>

<body>

<script language=”vbscript” type=”text/vbscript”>

Dim array1(1,1)

array1(0,0) = “hello”

array1(0,1) = 12

array1(1,0) = “how are you”

array1(1,1) = 14

Msgbox “Value present at index 0,0” & ” is “ & array1(0,0) & “<br />”

Msgbox “Value present at index 0,1” & ” is “ & array1(0,1) & “<br />”

Msgbox “Value present at index 1,0” & ” is “ & array1(1,0) & “<br />”

Msgbox “Value present at index 1,1” & ” is “ & array1(1,1)

</script>

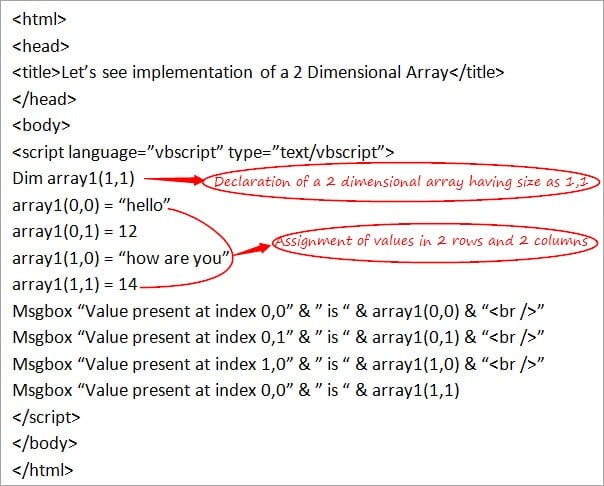
</body>

</html>

**Output is:**

Value present at index 0,0 is hello  
Value present at index 0,1 is 12  
Value present at index 1,0 is how are you  
Value present at index 1,1 is 14

In the above example, an array having 2 rows and 2 columns are declared with the size as (1,1) representing values present at the indexes 0 and 1 for both row and the column.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/01/Two-Dimentional-Array.jpg)

Next, let’s understand some of the frequently used concepts of an Array.

**Usage of REDIM Statement and PRESERVE Keyword in an Array**

**Redim**Statement is used to re-define the size of an Array. When the array is declared without any size, then it can be declared again using Redim with the feasibility of specifying the size of an array.

**Preserve** keyword is used to preserve the contents of a current array when the size of an array gets changed.

***Let’s understand the usage of these keywords with the help of a simple example.***

***Example:***

<html>

<head>

<title>Let’s see implementation of Redim and Preserve</title>

</head>

<body>

<script language=”vbscript” type=”text/vbscript”>

Dim array1()

REDIM array1(3)

array1(0) = “hello”

array1(1) = 12

array1(2) = 13

array1(3) = “how are you”

REDIM PRESERVE array1(5)

array1(4) = 15

array1(5) = 16

For i = 0 to ubound(array1)

Msgbox “Value present at index ” & i & ” is “ & array1(i) & “<br />”

Next

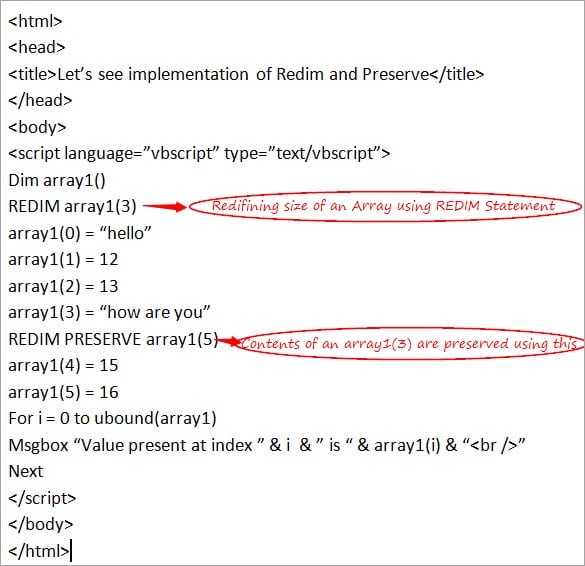
</script>

</body>

</html>

**Output is:**

Value present at index 0 is hello  
Value present at index 1 is 12  
Value present at index 2 is 13  
Value present at index 3 is how are you  
Value present at index 4 is 15  
Value present at index 5 is 16

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/01/Redim-Statement.jpg)

**In-Built Array Functions**

There are various inbuilt functions related to an Arras that is supported by the VBScript.

**Following is the list:**

**#1) lbound**:

This is the opposite of ubound (used above). This returns the smallest integer index value of an array i.e. the smallest subscript of an array.

**Example:** In the above example, the size of an array is 5. Hence, lbound will be 0 as this is the smallest subscript of an array.

**#2) ubound**:

This is already used above. This returns the largest subscript of a defined array.

**Example:**In the above example, the size of an array is 5. Hence, in this case, unbound is 5

**#3) Split**:

This returns an array consisting of a number of sub-strings and can be split using some delimiter. The syntax of this is: *Split(expression,[delimiter])*

Using a delimiter is an optional condition.

**#4) Join**:

This is the opposite of the Split function. Here, String is returned which includes various substrings in an array and thus joins all the sub-strings into one string.

The **syntax** of this is: *Join(array,[delimiter]. Using a delimiter is an optional condition.*

**#5) IsArray**:

This returns True/False on the basis of a specified variable. If the variable is passed is an Array then True is returned else False.

The **syntax** is: *IsArray( array variable)*

**#6) Filter**:

This returns a subset of an array based on the filter condition i.e. data is filtered on the basis of some condition.

The**syntax** is: *Filter(array, filter condition)*

***Let’s see the implementation of these functions with the help of a simple Example.***

***Example:***

<html>

<head>

<title>Let’s see implementation of In-Built Array Functions</title>

</head>

<body>

<script language=”vbscript” type=”text/vbscript”>

Dim array1 = Array(“January”, ”February”, ”March”, ”April”)

Dim a , b , c , d , e , f

a = lbound(array1)

b = ubound(array1)

c = Split(array1,”,”)

d = Join(array1,” $ “)

e = IsArray(array1)

f = Filter(array1,”J”)

Msgbox(a) & “<br />”

Msgbox(b) & “<br />”

Msgbox(c) & “<br />”

Msgbox(d) & “<br />”

Msgbox(e) & “<br />”

Msgbox(f)

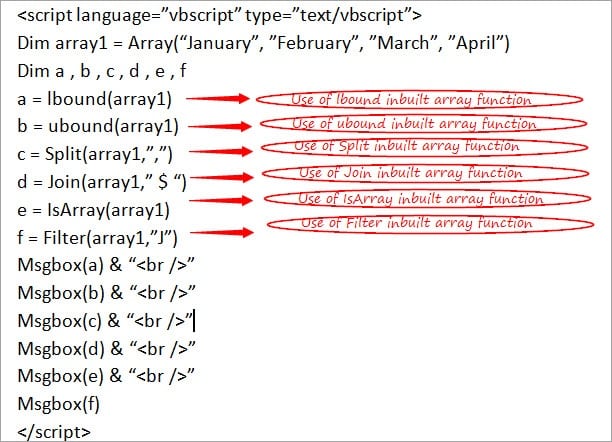
</script>

</body>

</html>

**Output** **is**:

0  
3  
January February March April  
January $ February $ March $ April  
True  
January

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/01/Array-Function.jpg)

**Conclusion**

I hope that this tutorial would have given you a brief overview about Arrays in VBScript. The simple practical examples covered must have made you understand about arrays in a better way.

[***Next VBScript Tutorial #8***](https://www.softwaretestinghelp.com/vbscript-date-functions-tutorial-8/)***: Our next tutorial will cover ‘Date Functions’ in the VBScript.***

# VBScript Date Functions: Date Format, DateAdd, And CDate Functions

**What You Will Learn:**[[hide](https://www.softwaretestinghelp.com/vbscript-date-functions-tutorial-8/)]

* [Date Functions](https://www.softwaretestinghelp.com/vbscript-date-functions-tutorial-8/#Date_Functions)
* [Different Date Functions used in the VBScript](https://www.softwaretestinghelp.com/vbscript-date-functions-tutorial-8/#Different_Date_Functions_used_in_the_VBScript)
  + [#1) cDate:](https://www.softwaretestinghelp.com/vbscript-date-functions-tutorial-8/#1_cDate)
  + [#2) IsDate:](https://www.softwaretestinghelp.com/vbscript-date-functions-tutorial-8/#2_IsDate)
  + [#3) Day:](https://www.softwaretestinghelp.com/vbscript-date-functions-tutorial-8/#3_Day)
  + [#4) Month:](https://www.softwaretestinghelp.com/vbscript-date-functions-tutorial-8/#4_Month)
  + [#5) Year:](https://www.softwaretestinghelp.com/vbscript-date-functions-tutorial-8/#5_Year)
  + [#6) Now:](https://www.softwaretestinghelp.com/vbscript-date-functions-tutorial-8/#6_Now)
  + [#7) DateAdd:](https://www.softwaretestinghelp.com/vbscript-date-functions-tutorial-8/#7_DateAdd)
  + [#8) DateDiff:](https://www.softwaretestinghelp.com/vbscript-date-functions-tutorial-8/#8_DateDiff)
  + [#9) DatePart:](https://www.softwaretestinghelp.com/vbscript-date-functions-tutorial-8/#9_DatePart)
  + [#10) MonthName:](https://www.softwaretestinghelp.com/vbscript-date-functions-tutorial-8/#10_MonthName)
  + [#11) FormatDateTime:](https://www.softwaretestinghelp.com/vbscript-date-functions-tutorial-8/#11_FormatDateTime)
* [Conclusion](https://www.softwaretestinghelp.com/vbscript-date-functions-tutorial-8/#Conclusion)
* [Recommended Reading](https://www.softwaretestinghelp.com/vbscript-date-functions-tutorial-8/#Recommended_Reading)

### Date Functions

In normal scenarios, Date function is used to display the current system date while working on the script and is the most basic and widely used function which is used while working with the Dates. There are various Date format functions available for converting the Date into different formats.

In some of my earlier tutorials of this series, Dates are used in one or more examples. Date function like cDate is also explained in one of my earlier tutorials.

**Let’s take a simple example to understand the usage of this basic Date Function as given below.**

***Example:***

<html>

<head>

<title>Let’s see implementation of a Date Function </title>

</head>

<body>

<script language=”vbscript” type=”text/vbscript”>

Dim val

val = Date

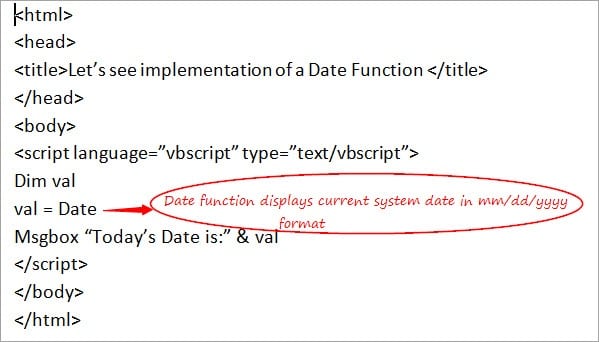
Msgbox “Today’s Date is:” & val

</script>

</body>

</html>

**The output** is: Today’s Date is: 10/24/2017

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/01/Date-Function.jpg)

Let’s take a look at the various types of Date Functions supported by VBScript.

### Different Date Functions used in the VBScript

There are multiple Date Functions which are used to perform operations on Dates and some Format functions which help the coders to convert date from one format to another.

***Following is the list of various Date Functions:***

#### **#1) cDate:**

cDate is used to convert a valid expression into a Date type value. The syntax of this is **cDate(date)**i.e. Any valid Date/Time expression will be converted into a particular Date.

#### **#2) IsDate:**

IsDate is used to check whether a specified expression is of Date type or not. This returns a Boolean value as True in case it is a Date else False. The syntax of this is **IsDate(any expression).**

#### **#3) Day:**

This Date function is used to fetch the day of the month of the date that is specified as a parameter i.e. any value in-between 1-31 as there are 30-31 days in a month. The syntax of this is **Day(Date).**

#### **#4) Month:**

This is used to fetch the month of the year. This returns a number in-between 1-12 for the specified date which is passed as a parameter. The syntax of this is **Month(Date)**.

#### **#5) Year:**

This function is used to fetch the year of the specified Date. The syntax of this is **Year(Date).**

#### **#6) Now:**

This function is used very frequently and works like the Date Function that is discussed above. This returns both the current system date as well as**time. The syntax** of this is Now. By writing simply Now, the current system date and time will be displayed

#### **#7) DateAdd:**

This function is used to fetch a Date value after the addition of some specified time interval which is specified as a parameter. The syntax of this is **DateAdd(Interval, Number, Date).**

**Here, the interval can be any of the following values:**

* **d**: This is passed if you want to pass **days** as an interval and then based on the ‘Number’ which is passed, days are either added or subtracted from the Date that is passed as the last parameter in the above function.
* **m**: This is passed if you want to pass **month** as an interval and then based on the ‘Number’ which is passed, months are either added or subtracted from the Date that is passed as the last parameter in the above function.
* **y**: This is passed if you want to pass the day of the **year** as an interval and then based on the ‘Number’ which is passed, days are either added or subtracted from the Date that is passed as the last parameter in the above function. This is same as d.
* **yyyy**: This is passed if you want to pass the **year** as an interval and then based on the ‘Number’ which is passed, years are either added or subtracted from the Date that is passed as the last parameter in the above function.
* **q**: This is passed if you want to pass the **quarter** as an interval and then based on the ‘Number’ which is passed, a quarter is either added or subtracted from the Date that is passed as the last parameter in the above function.
* **w**: This is passed if you want to pass the **weekday** as an interval and then based on the ‘Number’ which is passed, weekdays are either added or subtracted from the Date that is passed as the last parameter in the above function.
* **ww**: This is passed if you want to pass the **week** of the year as an interval and then based on the ‘Number’ which is passed, weeks are either added or subtracted from the Date that is passed as the last parameter in the above function.
* **h**: This is passed if you want to pass the **hour** as an interval and then based on the ‘Number’ which is passed, hours are either added or subtracted from the Date that is passed as the last parameter in the above function
* **m**: This is passed if you want to pass the **minute** as an interval and then based on the ‘Number’ which is passed, minutes are either added or subtracted from the Date that is passed as the last parameter in the above function.
* **s**: This is passed if you want to pass the **second** as an interval and then based on the ‘Number’ which is passed, seconds are either added or subtracted from the Date that is passed as the last parameter in the above function.

#### **#8) DateDiff:**

This function is used to fetch the difference between the 2 dates that are specified as parameters based on the interval specified. The syntax of this is **DateDiff(Interval,Date1,Date2)**. Value of interval is the same as discussed above in the DateAdd function.

#### **#9) DatePart:**

This is used to fetch some specific part of the date which is specified as a parameter. The syntax of this is **DatePart(Interval, Date)**. Value of the interval is the same as discussed above in the DateAdd function.

#### **#10) MonthName:**

This is used to fetch the name of the specified Month which is passed as a parameter inside the brackets. The syntax of this is **MonthName(Month value).**

#### **#11) FormatDateTime:**

This is a format function which is used to convert the Date to some specific format based on the parameters that are supplied to the function. The syntax of this is **FormatDateTime(Date,Format)**. This is a widely used format function

***The Format is an optional parameter but as this is widely used, it is good to know about the different format parameter values.***

**Various Format Parameter values are as follows:**

**0**: If a format value is passed as **0** then it will return the Date in mm/dd/yyyy format along with the time if it is specified in Date parameter. This is the default value.

**1**: If a format value is passed as **1** then it will return the Date in Weekday, Month Name, Year format.

**2**: If a format value is passed as **2** then it will return the Date in mm/dd/yyyy format.

**3**: If a format value is passed as **3** then it will return the Date in hh:mm: ss PM/AM format if time is specified in Date parameter.

**4**: If a format value is passed as **4** then it will return the Date in hh: mm format if time is specified in Date parameter.

**Note**: WeekDay and WeekDayName functions are not directly related to Date functions and already covered in some of the earlier tutorials so I am not covering them again.

**Now, let’s understand the usage of these functions with the help of an Example.**

***Example:***

<html>

<head>

<title>Let’s see implementation of various Date Functions </title>

</head>

<body>

<script language=”vbscript” type=”text/vbscript”>

Dim val,val1,val2,val3,val4,val5,val6,val7,val8,val9,val10,val11

val = “October 25 , 2017”

valnew = 10/25/2018

val1=CDate(val)

val2=IsDate(val)

val3=Day(val1)

val4=Month(val1)

val5=Year(val1)

val6=Now

val7=DateAdd(“d”,2,val1)

val8=DateDiff(“yyyy”,valnew,val1)

val9=DatePart(“m”,val1)

val10=MonthName(val4)

val11=FormatDateTime(val,2)

Msgbox “Converted Date Value is:” & val1 & “<br />”

Msgbox “Checking if it is Date Value:” & val2 & “<br />”

Msgbox “Day fetched from Date is:” & val3 & “<br />”

Msgbox “Month fetched from Date is:” & val4 & “<br />”

Msgbox “Year fetched from Date is:” & val5 & “<br />”

Msgbox “Current Date Value is:” & val6 & “<br />”

Msgbox “Date Value after addition is:” & val7 & “<br />”

Msgbox “Date Value Difference is:” & val8 & “<br />”

Msgbox “Part fetched from Date Value is:” & val9 & “<br />”

Msgbox “Month Name fetched from Date is:” & val10 & “<br />”

Msgbox “The new format of Date is:” & val11

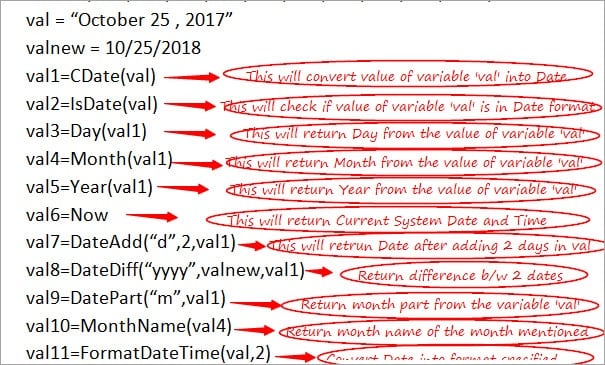
</script>

</body>

</html>

**Output** is:

Converted Date Value is:10/25/2017  
Checking if it is Date Value: True  
Day fetched from Date is:25  
Month fetched from Date is:10  
Year fetched from Date is:2017  
Current Date Value is:10/25/2017 1:48:29 AM  
Date Value after addition is: 10/27/2017  
Date Value Difference is:1  
Part fetched from Date Value is:10  
Month Name fetched from Date is:October  
The new format of Date is:10/25/2017

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/01/Conversion.jpg)

### **Conclusion**

I am sure that this tutorial would have enabled each of you to gain good knowledge about the Date Functions that are used in the VBScript and this, in turn, will help you in proceeding with the next tutorials of the VBScript series.

[**Next VBScript Tutorial #9**](https://www.softwaretestinghelp.com/vbscript-string-functions-instr-replace-tutorial-9/)**: I will discuss ‘Strings and Cookies’ in the next VBScript tutorial.**

# VBScript String Functions: VBScript InStr, Replace, Mid, And Trim Functions

**What You Will Learn:**[[hide](https://www.softwaretestinghelp.com/vbscript-string-functions-instr-replace-tutorial-9/)]

* [Strings & Cookies](https://www.softwaretestinghelp.com/vbscript-string-functions-instr-replace-tutorial-9/#Strings_Cookies)
* [Using Strings in the VBScript](https://www.softwaretestinghelp.com/vbscript-string-functions-instr-replace-tutorial-9/#Using_Strings_in_the_VBScript)
* [Concatenation of Strings in the VBScript](https://www.softwaretestinghelp.com/vbscript-string-functions-instr-replace-tutorial-9/#Concatenation_of_Strings_in_the_VBScript)
* [String Functions in the VBScript](https://www.softwaretestinghelp.com/vbscript-string-functions-instr-replace-tutorial-9/#String_Functions_in_the_VBScript)
  + [#1) InStr](https://www.softwaretestinghelp.com/vbscript-string-functions-instr-replace-tutorial-9/#1_InStr)
  + [#2) InStrRev](https://www.softwaretestinghelp.com/vbscript-string-functions-instr-replace-tutorial-9/#2_InStrRev)
  + [#3) LCase](https://www.softwaretestinghelp.com/vbscript-string-functions-instr-replace-tutorial-9/#3_LCase)
  + [#4) UCase](https://www.softwaretestinghelp.com/vbscript-string-functions-instr-replace-tutorial-9/#4_UCase)
  + [#5) Left](https://www.softwaretestinghelp.com/vbscript-string-functions-instr-replace-tutorial-9/#5_Left)
  + [#6) Len](https://www.softwaretestinghelp.com/vbscript-string-functions-instr-replace-tutorial-9/#6_Len)
  + [#7) StrReverse](https://www.softwaretestinghelp.com/vbscript-string-functions-instr-replace-tutorial-9/#7_StrReverse)
  + [#8) LTrim](https://www.softwaretestinghelp.com/vbscript-string-functions-instr-replace-tutorial-9/#8_LTrim)
  + [#9) Trim](https://www.softwaretestinghelp.com/vbscript-string-functions-instr-replace-tutorial-9/#9_Trim)
  + [#10) Right](https://www.softwaretestinghelp.com/vbscript-string-functions-instr-replace-tutorial-9/#10_Right)
  + [#11) RTrim](https://www.softwaretestinghelp.com/vbscript-string-functions-instr-replace-tutorial-9/#11_RTrim)
  + [#12) Mid](https://www.softwaretestinghelp.com/vbscript-string-functions-instr-replace-tutorial-9/#12_Mid)
  + [#13) Space](https://www.softwaretestinghelp.com/vbscript-string-functions-instr-replace-tutorial-9/#13_Space)
  + [#14) Replace](https://www.softwaretestinghelp.com/vbscript-string-functions-instr-replace-tutorial-9/#14_Replace)
  + [#15) StrComp](https://www.softwaretestinghelp.com/vbscript-string-functions-instr-replace-tutorial-9/#15_StrComp)
* [Working with the Cookies](https://www.softwaretestinghelp.com/vbscript-string-functions-instr-replace-tutorial-9/#Working_with_the_Cookies)
* [Reading and Writing Cookies](https://www.softwaretestinghelp.com/vbscript-string-functions-instr-replace-tutorial-9/#Reading_and_Writing_Cookies)
* [Conclusion](https://www.softwaretestinghelp.com/vbscript-string-functions-instr-replace-tutorial-9/#Conclusion)
* [Recommended Reading](https://www.softwaretestinghelp.com/vbscript-string-functions-instr-replace-tutorial-9/#Recommended_Reading)

### Strings & Cookies

**The string** is one of the different types of data types in the VBScript and is most frequently used while working with the coding part in the scripts. In simple terms, String is a collection of alphanumeric characters that are clubbed together. Strings can either consist of numbers, characters/special characters or a combination of all of them.

**The cookie**is a normal/plain piece of a text which the server uses to send data to the browser if a user visits the site. This is mainly used to keep a record or to maintain the information about the **session** of a user in the browser.

### Using Strings in the VBScript

A string is defined with the help of **double quotes (“”)** in the script. Any variable that is enclosed within the “” is referred as a String in the script. In simple terms, this is the way to identify a string.

**The syntax for this is as follows:**

**strvar**= “Hello”  
**strvar1**=”123456”  
**strvar2**=”%^&\*“  
**strvar3**=”H12ab$”

These are all String variables which have a different set of values assigned to them. **strvar** has all alphabetical values, **strvar1** has all numeric values, **strvar2** has all special characters and finally, **strvar3** includes a combination of all these.

So, a String can be defined in any of the above ways.

**Let’s see a Simple Example to understand the usage of Strings in the below script:**

<html>

<head>

<title>Let’s see implementation of a String</title>

</head>

<body>

<script language=”vbscript” type=”text/vbscript”>

Dim strname

strname = “My name is Ram”

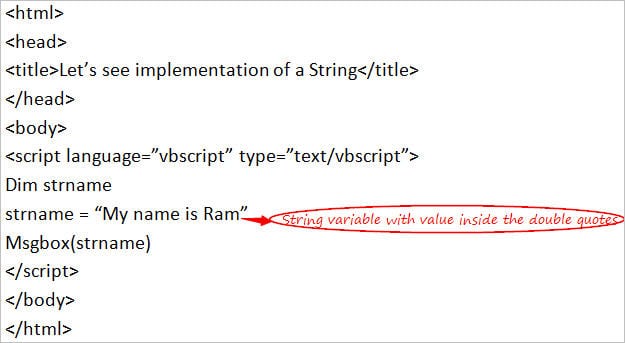
Msgbox(strname)

</script>

</body>

</html>

**The output is:** My name is Ram

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/01/simple-example.jpg)

### Concatenation of Strings in the VBScript

At times, there may be an occasion when 2 or more strings are required to be joined together. This can be achieved with the help of a concatenation operator “&” in the VBScript. With the usage of **‘&’**operator, any number of strings can be joined together by putting this operator in between the strings to form a big final string as a result.

This is considered to be a very useful operator while working with the Strings.

**Let’s see a Simple Example to understand the usage of String Concatenation Operation in the below script:**

<html>

<head>

<title>Let’s see implementation of a String Concatenation</title>

</head>

<body>

<script language=”vbscript” type=”text/vbscript”>

Dim strname,strname1,strname2,strname3

strname = “My name is Ram”

strname1 = “Hey!! ” & strname

strname2 = strname1 & “ and I like”

strname3 = strname2 & “ to play cricket!!”

Msgbox(strname3)

</script>

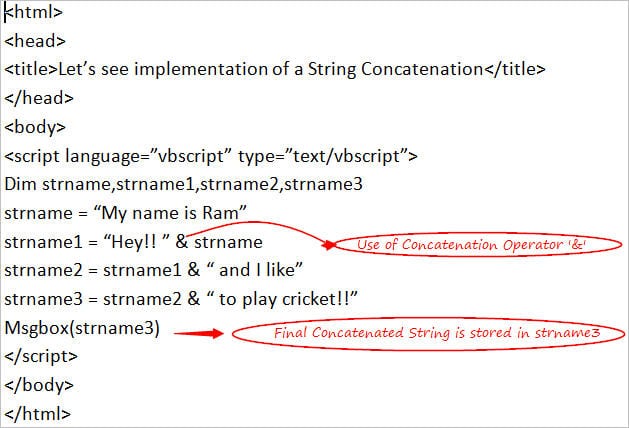
</body>

</html>

**The output is:** Hey!! My name is Ram and I like to play cricket!!

As shown below,**strname3** stores the concatenation of all the strings and displays the final output.

In this way, strings can be joined together.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/01/Example-2.jpg)

### String Functions in the VBScript

There are various string functions that are used in the script to perform the different type of operations on the Strings.

**Following is a list of the String functions:**

#### **#1) InStr**

**VBS InStr** is used to find the position value of a substring at its first occurrence inside the main string. This function requires 2 strings to be specified to perform this search operation and the search operation starts right from the first character.

***The syntax of this function is:***InStr(name of string1, name of string2)

If the name of string1 or string2 is null or “” then this function will return null and 0 respectively. In case, if the string is not found then the value of this function will be >=1 and 0.

#### **#2) InStrRev**

InStrRev is just the reverse of the above function. This is also used to find the position value of a substring at its first occurrence inside the main string. This function requires 2 strings to be specified to perform this search operation but with a minor difference that the search operation starts from the last character and even the position count is starts from the beginning character only.

***The syntax of this function is:***InStrRev(name of string1, name of string2)

If the name of string1 or string2 is null or “” then this function will return null and 0 respectively.In case, if the string is not found then the value of this function will be >=1 and 0.

#### **#3) LCase**

LCase is used to convert the specified string into a lower case.

***The syntax of this is:***LCase(name of the string)

#### **#4) UCase**

UCase is used to convert the specified string into an upper case.

***The syntax of this is:*** UCase(name of the string)

#### **#5) Left**

Left is used to fetch/get the mentioned number of characters(as per length parameter) from the left-hand side of the specified String.

***The syntax of this is:***Left(name of the string, length)

#### **#6) Len**

Len is used to get the length of a specified String i.e. the total number of characters of a specified String.

***The syntax of this is:*** Len(name of the string)

#### **#7) StrReverse**

StrReverse is used to reverse the specified string i.e. this will return the characters of a specified string in a reverse order starting from end to the beginning.

***The syntax of this is:*** StrReverse(name of the string)

#### **#8) LTrim**

LTrim is used to trim/remove the spaces from the left-hand side of the specified String.

***The syntax of this is:*** LTrim(name of the string)

#### **#9) Trim**

Trim is used to trim/remove the spaces from both the sides of the specified String.

***The syntax of this is:***Trim(name of the string)

#### **#10) Right**

Right is used to fetch/get the mentioned number of characters(as per length parameter) from the right-hand side of the specified String.

**The syntax of this is:** Right(name of the string, length)

#### **#11) RTrim**

RTrim is used to trim/remove the spaces from the right-hand side of the specified String.

***The syntax of this is:*** RTrim(name of the string)

#### **#12) Mid**

Mid is used to fetch the mentioned number of characters from the string by specifying the starting position.

***The syntax of this is:*** Mid(name of the string, starting position)

#### **#13) Space**

Space is used to fetch the String containing the required number of spaces as specified inside the parenthesis.

**The syntax of this is:** Space(number of spaces)

#### **#14) Replace**

Replace is used to replace the specified portion of a string with some other text as specified.

***The syntax of this is:*** Replace(name of the string, name of the string to be replaced, name of the new replaced string)

#### **#15) StrComp**

StrComp is used to compare the 2 strings and return values on the basis of comparison. This returns 0 if string1 = string2,-1 if string1<string2,1 if string1>string2 and null if any of the strings is null.

***The syntax of this is:*** Replace(name of the string1, name of the string2)

**Let’s understand the use of these String Functions with the help of a Simple Example.**

<html>

<head>

<title>Let’s see implementation of String Functions</title>

</head>

<body>

<script language=”vbscript” type=”text/vbscript”>

Dim strval,strval1,var1,var2,var3,var4,var5,strval2,var6,var7,var8,var9

strval = “My name is Ram”

strval1=” name ”

strval2=”My name is Ram”

var1=Instr(strval,”Ram”)

var2=Ucase(strval)

var3=Left(strval,4)

var4=Len(strval)

var5=trim(strval1)

var6=Mid(strval,6)

var7=Replace(strval,”Ram”,”Shyam”)

var8=Space(6)

var9=StrComp(strval,strval1)

Msgbox “Position returned by Instr function is ” & var1 & “<br />”

Msgbox “Uppercase returned by function is ” & var2 & “<br />”

Msgbox “Left characters returned by function is ” & var3 & “<br />”

Msgbox “Length returned by Len function is ” & var4 & “<br />”

Msgbox “Value returned after Trim function is ” & var5 & “<br />”

Msgbox “Value returned by Mid function is ” & var6 & “<br />”

Msgbox “New value returned after replacing is ” & var7 & “<br />”

Msgbox “String returned by Space function is ” & var8 & “<br />”

Msgbox “String Comparison returns ” & val1 & “<br />”

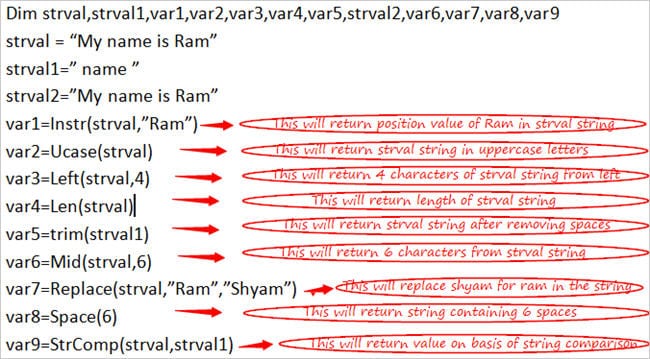
</script>

</body>

</html>

**Output is:**

Position returned by Instr function is 12  
Uppercase returned by function is MY NAME IS RAM  
Left characters returned by function is My n  
Length returned by Len function is 14  
Value returned after Trim function is name  
Value returned by Mid function is My nam  
New value returned after replacing is My name is Shyam  
String returned by Space function is “      “  
String Comparison returns 0

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/01/Example-3.jpg)

So far, we have almost covered all the topics in String.

### Working with the Cookies

Each of us must be aware of the term Cookies, let’s discuss how actually cookies work.

Whenever you visit a site or any page, the Server sends data to the browser to get it stored in the browser and it is called as a Cookie. And whenever you visit that page or site again, the browser sends that same data which is stored as Cookie to the Server.

**Cookies include some variable-length field value comprising of 5 fields.**

**They are:**

* **Expires field:** This includes all the information about the date of expiration of the Cookie. If this field is blank then it means that the cookie would get expired once any visitor quits the browser.
* **Domain field:** This includes information about the domain name of the site that you visit.
* **Path field:** This includes the information about the path of the directory or any webpage that actually set the Cookie.
* **Secure field:** This basically helps in securing the data i.e. if this field is having any value inside the Cookie then this will put a restriction to get the Cookie value retrieved only by the Secure Server.
* **Name=Value field:** This includes the information about the pair value of a ‘key’ and ‘value’ of a Cookie.

As mentioned, Cookie gets stored automatically. There is an object which stores the cookie property named as **Document.Object.**

### Reading and Writing Cookies

**document. cookie** stores the information of key-value pairs and the expiration date values of a Cookie.

document.cookie = “key1=name of the value1;key2=name of the value2,…….,expires=date”

**‘;’** is used to separate the key-value pairs.

***Let’s understand how Cookies can be written and read with the help of a Simple Example.***

<html>

<head>

<title>Let’s see method of Reading and Writing a Cookie</title>

</head>

<body>

<script language=”vbscript” type=”text/vbscript”>

Function writingcookie(name, val)

document.cookie = name & "=" & val

End Function

Function readingacookie ()

val = document.cookie

val1 = split(val,”;”)

for i = 0 to ubound(val1)

cookiename = split(val1(i),”=”)

“key is “ & cookiename (i) & “ and value is “ & cookiename (i+1)

Next

End Function

writingcookie "ram","1234"

Msgbox document.cookie & “<br />”

Msgbox readingacookie()

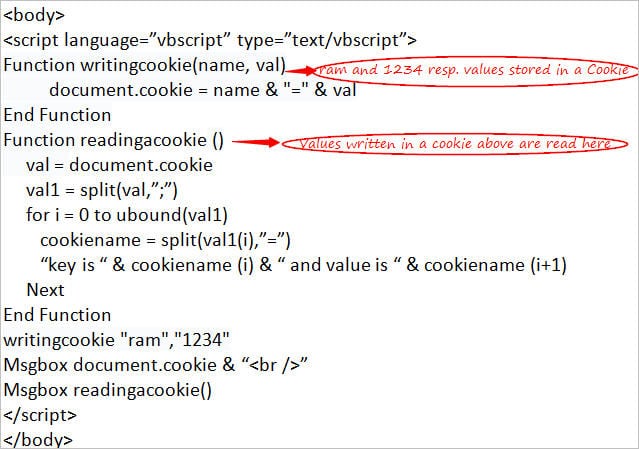
</script>

</body>

</html>

**Output is:**

ram=1234  
Key is ram and value is 1234

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/01/Example-4.jpg)

We have seen the important topics in Cookies which would help you to understand better.

### Conclusion

VBS Strings and Cookies are one among the important topics in VBScript. I’m sure that this tutorial would have briefed you about the importance and effectiveness of using Strings and Cookies.

[***Next VBScript Tutorial #10***](https://www.softwaretestinghelp.com/vbscript-events-tutorial-10/)***: My next tutorial will brief you about ‘Events’.***

# Working With Events In VBScript

**What You Will Learn:**[[hide](https://www.softwaretestinghelp.com/vbscript-events-tutorial-10/)]

* [What are VBS Events?](https://www.softwaretestinghelp.com/vbscript-events-tutorial-10/#What_are_VBS_Events)
* [Different Types of Events](https://www.softwaretestinghelp.com/vbscript-events-tutorial-10/#Different_Types_of_Events)
  + [#1) ‘On Click of Button’ Event](https://www.softwaretestinghelp.com/vbscript-events-tutorial-10/#1_On_Click_of_Button_Event)
  + [#2) ‘Mouse’ Events – Mouse Over and Mouse Out](https://www.softwaretestinghelp.com/vbscript-events-tutorial-10/#2_Mouse_Events_Mouse_Over_and_Mouse_Out)
* [List of Few other Events](https://www.softwaretestinghelp.com/vbscript-events-tutorial-10/#List_of_Few_other_Events)
* [Conclusion:](https://www.softwaretestinghelp.com/vbscript-events-tutorial-10/#Conclusion)
* [Recommended Reading](https://www.softwaretestinghelp.com/vbscript-events-tutorial-10/#Recommended_Reading)

### What are VBS Events?

**Events**are the Actions that occur when any activity is performed like any **mouse click, pressing keys, mouse hover**, etc.

With the help of writing a piece of code in programming languages like VBScript, these events can be captured and actions can be performed as per your requirements by making the best use of the Event Handling mechanism.

Different types of validations and responses can be handled in case of occurrence of an event like a window or a dialog box which appears on clicking some button, in such a case you can write a piece of code to handle such a window or dialog box.

### Different Types of Events

There are different types of events that take place in the VBScript programming language. Each event can be handled by writing some code.

In this tutorial, I will explain Events in the context of an **HTML** code (as every HTML element holds certain types of events) as well as in the context of QTP/UFT to showcase the real implementation and use of Events while working with the scripts in QTP.

**Let’s see the working of some of the important events in the context of HTML.**

#### **#1) ‘On Click of Button’ Event**

This Event occurs while **clicking any button** that is present on any HTML page.

You can handle this event as per your requirements. If you want to show some message on the click of the button then you can handle message box or if you want to move the application to the next page on the click of the button then you can write a piece of code to handle that.

**Following is an example to show the working of this Event.**

***Example:***

<html>

<head>

<title>Let’s see the working of ‘On Click of Button’ Event</title>

</head>

<body>

<script language=”vbscript” type=”text/vbscript”>

Function displaymessage(name)

Msgbox “Hello ” & name

End Function

</script>

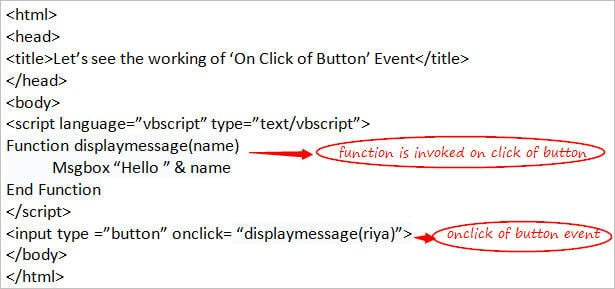
<input type =”button” onclick= “displaymessage(riya)”>

</body>

</html>

**The output is:** Hello riya

When the button is clicked then ‘on click’ the event will get triggered and ‘displaymessage’ function will get invoked and the Msgbox will display the message as mentioned.



The same Event that I have discussed above can be handled in the scripts by making a function in the QTP which can get invoked every time when any button in the application gets clicked.

Assume that there is an **Application** where there is a Window (**parent object**) inside which there are many buttons (**child objects**) and you have to click a particular button and you want the same function to get invoked on click of any of the buttons in the application.

In this scenario, you can make one function which can get invoked every time when any button is clicked.

**Let’s understand this with the help of an Example.**

***Example:***

<html>

<head>

<title>Let’s see the functionality of ‘Button Click’ Event</title>

</head>

<body>

<script language=”vbscript” type=”text/vbscript”>

Function buttonclick(Window name,Button name)

Dim flag,flag1

flag = WpfWindow(Window name).Exist

If flag = false then

Msgbox “Window name object does not exist in the application”

Exit Function

End If

flag1 = WpfWindow(Window name).WpfButton(Button name).Exist

If flag1 = false then

Msgbox “Button name object does not exist in the application”

Exit Function

Else

WpfWindow(Window name).WpfButton(Button name).Click

Msgbox “Button is clicked successfully”

End If

End Function

Msgbox buttonclick(“Login”,”Add”)

</script>

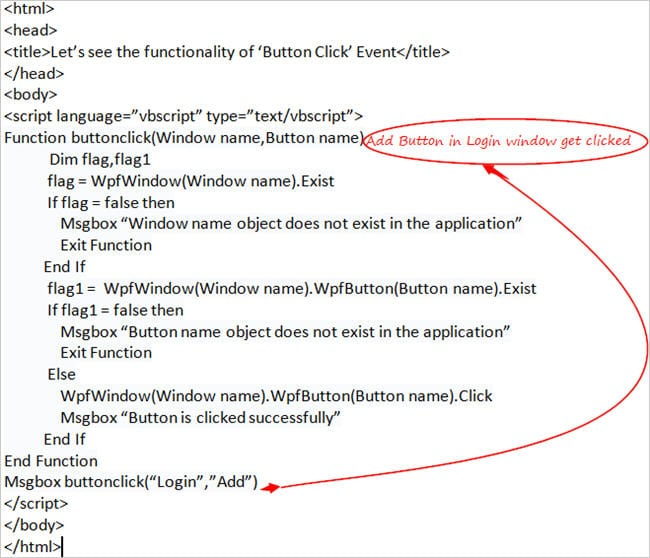
</body>

</html>

**The output is:** Button is clicked successfully

As you can see, a function named ‘buttonclick’ is created to click the button ‘Add’ inside the Window ‘Login’ in the application having WpfObjects(assumption). When a function is invoked by passing the parameters, the code written inside the function gets invoked and the button will get clicked successfully.

If the window name or button name that is passed is not correct then Msgbox will display an appropriate message based upon which is written inside the function.



#### **#2) ‘Mouse’ Events – Mouse Over and Mouse Out**

These Events occur while placing the mouse over the elements and taking out the mouse from the elements.

Like the other events, you can handle these events also as per your requirements. If you want to show some message while you mouse over an element or when the mouse focus comes out of an element then you can handle message box or if you want to move the application to the next element then you can write a piece of code to handle that.

**Following is an example to show the working of this Event.**

<html>

<head>

<title>Let’s see the working of ‘On Click of Button’ Event</title>

</head>

<body>

<script language=”vbscript” type=”text/vbscript”>

Function displaymessage(name)

Msgbox “Hello ” & name

End Function

Function displaymessage1(name)

Msgbox “Hello ” & name

End Function

</script>

<input name =”txtmsg” type =”text” OnMouseOver= “displaymessage(riya)”> <br>

<input name =”txtmsg1” type =”text” OnMouseOut= “displaymessage1(siya)”>

</body>

</html>

**Output** **is:**

Hello riya  
Hello siya

When the mouse gets over a text element then the ‘OnMouseOver’ event will get triggered and ‘displaymessage’ function will get invoked and the Msgbox will display the message as mentioned and when the mouse gets out of a text element then ‘OnMouseOut’ event will get triggered and ‘displaymessage1’ function will get invoked and Msgbox will display the message as mentioned.

These are some of the important events, hence I discussed them in detail.

Let’s see a list of some other events as well.

### List of Few other Events

Following is the list of some of the other important events that occur while working with the scripts in QTP along with their descriptions.

**They are:**

* **onchange Event:**This event occurs when any change occurs in context to the element in the Script i.e. when element changes, this event gets fired in the Script.
* **onsubmit Event:**This event occurs at the Script level when the form gets submitted in the application.
* **onreset Event:**This event occurs at the Script level when the form gets at the reset mode in the application.
* **onfocus Event:**This event occurs at the Script level when any of the elements gets a focus in the application.
* **onblur Event:**This event occurs at the Script level when the element loses its focus in the application.
* **ondblclick Event:**This event occurs at the Script level when the mouse gets double-clicked.
* **onmousedown Event:**This event occurs at the Script level when the mouse button gets pressed.
* **onmouseup Event:**This event occurs at the Script level when the mouse button gets released.
* **onkeyup Event:**This event occurs at the Script level when any of the keys that are present in the keyboard button gets released.
* **onkeydown Event:**This event occurs at the Script level when any of the keys that are present in the keyboard get pressed.
* **onkeypress Event:**This event occurs at the Script level when any of the keys that are present in the keyboard get pressed and released.
* **onmousemove Event:**This event occurs at the Script level when the pointer of the mouse moves.

### Conclusion:

I hope that this tutorial must have provided you an insight regarding the importance and effectiveness of using Events in VBScript.

[***Next Tutorial #11***](https://www.softwaretestinghelp.com/vbscript-excel-tutorial-11/)***: I will cover about ‘Excel Objects’ in my next VBScript tutorial.***

# Working With VBScript Excel Objects

**What You Will Learn:**[[hide](https://www.softwaretestinghelp.com/vbscript-excel-tutorial-11/)]

* [Overview](https://www.softwaretestinghelp.com/vbscript-excel-tutorial-11/#Overview)
* [Creating an Excel File using Excel Object](https://www.softwaretestinghelp.com/vbscript-excel-tutorial-11/#Creating_an_Excel_File_using_Excel_Object)
* [Reading/Opening an Excel File using Excel Object](https://www.softwaretestinghelp.com/vbscript-excel-tutorial-11/#ReadingOpening_an_Excel_File_using_Excel_Object)
* [Deletion from an Excel File](https://www.softwaretestinghelp.com/vbscript-excel-tutorial-11/#Deletion_from_an_Excel_File)
* [Addition & Deletion of a Sheet from an Excel File](https://www.softwaretestinghelp.com/vbscript-excel-tutorial-11/#Addition_Deletion_of_a_Sheet_from_an_Excel_File)
* [Copying & Pasting of Data from one Excel file to Another Excel File](https://www.softwaretestinghelp.com/vbscript-excel-tutorial-11/#Copying_Pasting_of_Data_from_one_Excel_file_to_Another_Excel_File)
* [Conclusion](https://www.softwaretestinghelp.com/vbscript-excel-tutorial-11/#Conclusion)
* [Recommended Reading](https://www.softwaretestinghelp.com/vbscript-excel-tutorial-11/#Recommended_Reading)

### Overview

Microsoft Excel needs to be **installed** on your computer to work with the Excel files. By creating an Excel Object, VBScript provides you the assistance to perform important operations like **Create, Open**and **Edit**Excel files.

It is very important to understand this topic as this forms the basis of the working with the Excel sheets and hence I decided to pick this as one of the topics in the series of the VBScript tutorial.

I will try to make you understand all the different codes that are required to be written to work with the excel files in an easy manner so that you can easily write a piece of code on your own.

Now, let’s move on to the practical working of Excel files by understanding the code written for different scenarios focusing mainly on the important ones.

### Creating an Excel File using Excel Object

In this section, we will see the various steps involved in creating an excel file using the Excel Object mechanism in the VBScript.

**Following is the Code for Creating an Excel File:**

Set obj = createobject(“Excel.Application”)  ‘**Creating an Excel Object**

obj.visible=True                                    ‘**Making an Excel Object visible**

Set obj1 = obj.Workbooks.Add()       ‘**Adding a Workbook to Excel Sheet**

obj1.Cells(1,1).Value=”Hello!!”         ‘**Setting a value in the first-row first column**

obj1.SaveAs “C:\newexcelfile.xls”   ‘**Saving a Workbook**

obj1.Close                                             ‘**Closing a Workbook**

obj.Quit                                                  ‘**Exit from Excel Application**

Set obj1=Nothing                                 ‘**Releasing Workbook object**

Set obj=Nothing                                   ‘**Releasing Excel object**

**Let’s understand how it works:**

* Firstly, an Excel Object with the name **‘obj’** is created using **‘createobject’** keyword and defining Excel application in the parameter as you are creating an Excel Object.
* Then an Excel Object that is created above is made **visible** to the users of the sheet.
* A **Workbook** is then added to the excel object – obj to perform actual operations inside the sheet.
* Next, the main task is performed by **adding a** **value** in the first column of the first row of the workbook that is created above.
* The workbook is then **closed** as the task has been completed.
* Excel Object is then **exited** as the task has been finished.
* Finally, both the objects – obj and obj1 are **released** by using ‘Nothing’ keyword.

**Note**: It is a good practice to release the objects using **‘Set object name = Nothing’** after the completion of the task at the end.

### Reading/Opening an Excel File using Excel Object

In this section, we will see the different steps of reading the data from an excel file using Excel Object mechanism in the VBScript. I will use the same excel file that is created above.

**Following is the Code for reading the data from an excel file:**

Set obj = createobject(“Excel.Application”)   ‘**Creating an Excel Object**

obj.visible=True                                    ‘**Making an Excel Object visible**

Set obj1 = obj.Workbooks.open(“C:\newexcelfile.xls”)    ‘**Opening an Excel file**

Set obj2=obj1.Worksheets(“Sheet1”)    **‘Referring Sheet1 of excel file**

Msgbox obj2.Cells(2,2).Value **‘Value from the specified cell will be read and shown**

obj1.Close**‘Closing a Workbook**

obj.Quit                                                  ‘**Exit from Excel Application**

Set obj1=Nothing                                 ‘**Releasing Workbook object**

Set obj2 = Nothing                               **‘Releasing Worksheet object**

Set obj=Nothing                                   ‘**Releasing Excel object**

**Let’s understand how it works:**

* Firstly, an Excel Object with the name **‘obj’** is created using**‘createobject’** keyword and defining Excel application in the parameter as you are creating an Excel Object.
* Then the Excel Object that is created above is made visible to the users of the sheet.
* Next step is to **open** an excel file by specifying the location of the file.
* Then, **worksheet** of the workbook or an excel file is specified to access the data from a particular sheet of an excel file.
* Finally, the value from the particular cell (2nd column from 2nd row) is **read** and displayed with the help of a message box.
* The workbook object is then **closed** as the task has been completed.
* Excel Object is then **exited** as the task has been finished.
* Finally, all the objects are **released** by using ‘Nothing’ keyword.

### Deletion from an Excel File

In this section, we will take a look at the steps involved in the deleting a data from an excel file using Excel Object mechanism in VBScript. I will use the same excel file that is created above.

**Following is the Code for deleting the data from an Excel file:**

Set obj = createobject(“Excel.Application”)   ‘**Creating an Excel Object**

obj.visible=True                                    ‘**Making an Excel Object visible**

Set obj1 = obj.Workbooks.open(“C:\newexcelfile.xls”)    ‘**Opening an Excel file**

Set obj2=obj1.Worksheets(“Sheet1”)    **‘Referring Sheet1 of excel file**

obj2.Rows(“4:4”).Delete           **‘Deleting 4th row from Sheet1**

obj1.Save()                                   **‘Saving the file with the changes**

obj1.Close**‘Closing a Workbook**

obj.Quit                                                  ‘**Exit from Excel Application**

Set obj1=Nothing                                 ‘**Releasing Workbook object**

Set obj2 = Nothing                               **‘Releasing Worksheet object**

**Let’s understand how it works:**

* Firstly, an Excel Object with the name **‘obj’** is created using ‘createobject’ keyword and defining Excel application in the parameter as you are creating an Excel Object.
* Then an Excel Object that is created above is made visible to the users of the sheet.
* Next step is to **open** an excel file by specifying the location of the file.
* Then, **worksheet** of the workbook or an excel file is specified to access the data from the particular sheet of an excel file.
* Finally, the 4th row is **deleted** and the changes are **saved** on the sheet.
* The workbook object is then **closed** as the task has been completed.
* Excel Object is then **exited** as the task has been finished.
* Finally, all the objects are **released** by using ‘Nothing’ keyword.

### Addition & Deletion of a Sheet from an Excel File

In this section, let’s see the different steps of adding and deleting an excel sheet from an excel file using Excel Object mechanism in VBScript. Here also I will use the same excel file that is created above.

**Following is the Code for this scenario:**

Set obj = createobject(“Excel.Application”)   ‘**Creating an Excel Object**

obj.visible=True                                    ‘**Making an Excel Object visible**

Set obj1 = obj.Workbooks.open(“C:\newexcelfile.xls”)    ‘**Opening an Excel file**

Set obj2=obj1.sheets.Add  **‘Adding a new sheet in the excel file**

obj2.name=”Sheet1”     **‘Assigning a name to the sheet created above**

Set obj3= obj1.Sheets(“Sheet1”)  **‘Accessing Sheet1**

obj3.Delete       **‘Deleting a sheet from an excel file**

obj1.Close**‘Closing a Workbook**

obj.Quit                                                  ‘**Exit from Excel Application**

Set obj1=Nothing                                 ‘**Releasing Workbook object**

Set obj2 = Nothing                               **‘Releasing Worksheet object**

Set obj3 = Nothing**‘Releasing Worksheet object**

Set obj=Nothing                                   ‘**Releasing Excel object**

**Let’s understand how it works:**

* Firstly, an Excel Object with the name **‘obj’** is created using ‘createobject’ keyword and defining Excel application in the parameter as you are creating an Excel Object.
* Then an Excel Object that is created above is made visible to the users of the sheet.
* Next step is to **open** an excel file by specifying the location of the file.
* The worksheet is then **added** to an excel file and a **name** is assigned to it.
* Then, worksheet of the workbook or an excel file is accessed (created in the earlier step) and it is **deleted**.
* The workbook object is then **closed** as the task has been completed.
* Excel Object is then **exited** as the task has been finished.
* Finally, all the objects are **released** by using ‘Nothing’ keyword.

### Copying & Pasting of Data from one Excel file to Another Excel File

In this section, we will see the different steps involved in copying/pasting data from one excel file to another excel file using Excel Object mechanism in the VBScript. I have used the same excel file that was used in the above scenarios.

**Following is the Code for this scenario:**

Set obj = createobject(“Excel.Application”)   ‘**Creating an Excel Object**

obj.visible=True                                    ‘**Making an Excel Object visible**

Set obj1 = obj.Workbooks.open(“C:\newexcelfile.xls”)    ‘**Opening an Excel file1**

Set obj2 = obj.Workbooks.open(“C:\newexcelfile1.xls”)    ‘**Opening an Excel file2**

obj1.Worksheets(“Sheet1”).usedrange.copy  **‘Copying from an Excel File1**

obj2.Worksheets(“Sheet1”).usedrange.pastespecial  **‘Pasting in Excel File2**

obj1.Save**‘ Saving Workbook1**

obj2.Save**‘Saving Workbook2**

obj1.Close**‘Closing a Workbook**

obj.Quit                                                 ‘**Exit from Excel Application**

Set obj1=Nothing                                ‘**Releasing Workbook1 object**

Set obj2 = Nothing                              **‘Releasing Workbook2 object**

Set obj=Nothing                                  ‘**Releasing Excel object**

**Let’s understand how it works:**

* Firstly, an Excel Object with the name **‘obj’** is created using ‘createobject’ keyword and defining Excel application in the parameter as you are creating an Excel Object.
* Then the Excel Object that is created above is made visible to the users of the sheet.
* Next step is to **open** 2 excel files by specifying the location of the files.
* Data is **copied** from Excel file1 and **pasted** to Excel file2.
* Both the Excel Files has been **saved**.
* The workbook object is then **closed** as the task has been completed.
* Excel Object is then **exited** as the task has been finished.
* Finally, all the objects are **released** by using a ‘Nothing’ keyword.

These are some of the important scenarios which are required in the proper understanding of the concept. And they form the foundation to work and deal with the codes for handling different types of scenarios while dealing with the Excel Objects in the script.

### Conclusion

Excel plays a prime major role everywhere. I’m sure that this tutorial must have given you a great insight regarding the importance and effectiveness of using VBS Excel Objects.

[***Next Tutorial #12***](https://www.softwaretestinghelp.com/vbscript-connection-objects-tutorial-12/)***: Our next tutorial will cover ‘Connection Objects’ in the VBScript.***

# Working With VBScript ADODB Connection Objects

**What You Will Learn:**[[hide](https://www.softwaretestinghelp.com/vbscript-connection-objects-tutorial-12/)]

* [Overview](https://www.softwaretestinghelp.com/vbscript-connection-objects-tutorial-12/#Overview)
* [Properties and Methods of Connection Objects](https://www.softwaretestinghelp.com/vbscript-connection-objects-tutorial-12/#Properties_and_Methods_of_Connection_Objects)
* [Establishing a Database Connection using ADODB Connection Object](https://www.softwaretestinghelp.com/vbscript-connection-objects-tutorial-12/#Establishing_a_Database_Connection_using_ADODB_Connection_Object)
* [Exporting Data to Excel File Using ADODB Connection Object](https://www.softwaretestinghelp.com/vbscript-connection-objects-tutorial-12/#Exporting_Data_to_Excel_File_Using_ADODB_Connection_Object)
* [Exporting Data to Text File Using ADODB Connection Object](https://www.softwaretestinghelp.com/vbscript-connection-objects-tutorial-12/#Exporting_Data_to_Text_File_Using_ADODB_Connection_Object)
* [Conclusion](https://www.softwaretestinghelp.com/vbscript-connection-objects-tutorial-12/#Conclusion)
* [Recommended Reading](https://www.softwaretestinghelp.com/vbscript-connection-objects-tutorial-12/#Recommended_Reading)

### Overview

Any Databases like SQL Server, PL/SQL, etc. need to be **installed** on your computer to work with the Database Connections.

As such, there is not any straight-forward mechanism to get connected to the database in the QTP. But with the help of **ADODB Objects**, you can interact with the database and work with the SQL Queries to fetch the data from the database.

**ADO** stands for **ActiveX Data Objects** and this provides the mechanism to act as an intermediary between the QTP and the Database.

This topic formed the basis of the working with the Database and it would be very helpful for you in the long run if you have a better understanding of it.

I will try to make you understand all the different codes, properties, and methods that are required to be written to work with the database in an easy manner so that you can easily write a piece of code on your own.

Now, let’s start with the Properties and Methods that are helpful in establishing a connection with the database.

### Properties and Methods of Connection Objects

There are different properties and methods that support to extract data from the database.

**List of ADODB Connection Object properties are as follows:**

**#1) Connection string**:

This is a very useful property which is used for creating a database connection and includes connection details like the details of the **Driver, Database Server name, Username, and Password**.

**#2) Connection Timeout**:

This is used for defining the required time for **waiting** for a connection to get established.

**#3) Provider**:

This property provides all provider related details i.e. the name of the **Connection Provider**.

**#4) State**:

This provides the information about the state of the connection i.e. if the Connection is **ON or OFF**.

The above-mentioned ones are the properties of a Connection Object. However, there is a **RecordSet Object**also (will we discuss in a while).

**List of its properties are as follows:**

**#1) BOF:**

This property is used to know the position of the current record. If the position of the current record is present **before the first record** of the recordset, then this property will return true.

**#2) EOF:**

This is just the reverse of the above one. If the position of the current record is present **after the last record**of the recordset, then this property will return true.

**Note:** Values of EOF and BOF will be false when there are no records in the recordset. This is useful in case of validating empty records i.e. when there are no records in the recordset.

**#3) MaxCount:**

This is useful in setting the **maximum** values of rows/records that can be returned from the database i.e. if you want to fetch maximum 20 rows at a time from the data then you can set this property as 20.

**Let’s now take a look at Methods:**

**List of ADODB Connection Object and RecordSet Object methods are as follows:**

* **Open:**This method is used to open a database connection object/recordset object.
* **Execute:**This is used to execute a SQL Query that is provided.
* **Close:**This is used to close a database connection/recordset which is opened.
* **Cancel:**This is a recordset method which is used for canceling the existing execution of the Database Connection.
* **clone:** This is a recordset method and is used for getting the clone of the existing recordset object.
* **move:** This is a recordset method which is used to move the pointer at a specified location inside a recordset on the basis of the count that is mentioned.
* **count:**This is a recordset method and is used to get the total number of fields present in the recordset i.e. total number of columns in a table.
* **item(i):** This is a recordset method. This is used to get the item whose index is specified by the total number of fields present in a recordset.

These are the different properties and methods that you will use while dealing with the Connection related objects.

Let’s move on to the practical implementation to know the working of these objects.

### Establishing a Database Connection using ADODB Connection Object

In this section, we will see the different steps involved in creating a database connection using the Connection Object mechanism in VBScript.

**Following is the Code for creating a connection:**

Set obj = createobject(“ADODB.Connection”) ‘**Creating an ADODB Connection Object**  
Set obj1 = createobject(“ADODB.RecordSet”) ‘**Creating an ADODB Recordset Object**  
Dim dbquery       **‘Declaring a database query variable bquery**  
Dbquery=”Select acctno from dbo.acct where name = ‘Harsh’” **‘Creating a query**  
obj.Open“Provider=SQLQLEDB;Server=.\SQLEXPRESS;UserId=test;Password=P@123;Database =AUTODB”    **‘Opening a Connection**  
obj1.Open dbquery,obj   **‘Executing the query using recordset**  
val1 = obj1.fields.item(0)  **‘Will return field value**  
msgbox val1**‘Displaying value of the field item 0 i.e. column 1**  
obj.close                             **‘Closing the connection object**  
obj1.close                           **‘Closing the connection object**  
Set obj1=Nothing              ‘**Releasing Recordset object**  
Set obj=Nothing                ‘**Releasing Connection object**

**Let’s see how it works:**

* Firstly, a **Connection Object**with the name ‘obj’ is created using ‘createobject’ keyword and ADODB Connection in the parameter as you are creating a Database Connection Object is defined.
* Then a **Recordset Object** is created to access the records of a particular table from the database using ‘obj1’. Recordset Object provides a way to access the records from the database.
* After that, a variable is declared for a **query** ‘dbquery’ to store the query to fetch the records from the database.
* Then a Connection Object that is created above is **opened** to start working with the object by providing all the details about the provider like SQL SERVER, server name, database name, user id, and password.
* A Query is then **executed** after opening recordset object by mentioning query and the connection object.
* Next, the main task is performed to fetch the value of acct no. of a particular person as per the query using **fields** of the recordset that is mentioning the index of the item where acct no. is located like item(0) in this case. Value of a field is then displayed with the help of a message box.

**Note:** Fields represent ‘columns’ and recordset represents ‘rows’ of a database table.

* The objects are then **closed** as the task has been completed.
* Finally, both the objects – obj and obj1 respectively are **released** by using ‘Nothing’ keyword.

**Note**: It is a good practice to release the objects using **‘Set object name = Nothing’** after the completion of the task at the end.

We have seen all about establishing a connection with the database and reading & displaying of data using a combination of the database and QTP.

Let’s see a few other scenarios using Connection Object.

### Exporting Data to Excel File Using ADODB Connection Object

In this section, we will see the different steps involved in exporting the data to an excel file from the database using the Connection Object mechanism in VBScript.

**Following is the Code for this scenario:**

Set obj = CreateObject("ADODB.Connection")

Set obj1 = CreateObject("ADODB.RecordSet ")

Set obj2 = CreateObject("Excel.Application")

Set obj3 = obj2.Workbooks.Open ("C:\Users\Riya.xlsx")

Set obj4 = obj3.Worksheets(1)

obj.Provider =("Microsoft.ACE.OLEDB.12.0")

obj.Open "C:\Users\newdb.autodb"

obj1.Open "Select name, age from person", obj

obj4.cells(1, 1) = "Name"

obj4.cells(1, 2) = "Age"

row = 2

While obj1.EOF = False

obj4.cells(row, 1) = obj1.Fields("Name")

obj4.cells(row, 2) = obj1.Fields("Age")

obj1.MoveNext

row = row+1

Wend

obj3.Save

obj2.Quit

obj1.Close

obj.Close

Set obj4 = Nothing

Set obj3 = Nothing

Set obj2 = Nothing

Set obj1 = Nothing

Set obj = Nothing

**Let’s see how it works:**

* **Firstly, objects –** **Connection Object, RecordSet Object and Excel Object**with the name ‘obj, ‘obj1’ and ‘obj2’ respectively are created using ‘createobject’ keyword.
* **Then, a workbook object –** ‘obj3’ is created mentioning the location of the excel file and after that **worksheet** object (‘obj4’) is created to specify the sheet in the excel file where the data needs to be exported.
* Then a Connection Object that is created above is **opened** to start working with the object and the Provider details are mentioned.
* A Query is then **executed** after opening the recordset object by mentioning query and connection object.
* Next, the main task is performed to store the values of name and age of a particular person in the excel file as per the query using **fields** of the recordset in the cells of the sheet from the database. The loop will go on until the EOF is not false (we have already discussed EOF).
* Excel workbook is then **saved**.
* Excel application will **quit** as the task has been completed.
* The objects are then **closed** as the task has been completed.
* Finally, all the objects are **released** by using ‘Nothing’ keyword.

### Exporting Data to Text File Using ADODB Connection Object

In this section, we will see the different steps involved in exporting data to a text file from the database using the Connection Object mechanism in the VBScript.

**Following is the Code for this scenario:**

Set obj = CreateObject("ADODB.Connection")

Set obj1 = CreateObject("ADODB.RecordSet ")

Set obj2 = CreateObject("Scripting.FileSystemObject")

Set obj3 = obj2.OpenTextFile("C:\Users\Riya.xlsx")

obj.Provider =("Microsoft.ACE.OLEDB.12.0")

obj.Open "C:\Users\newdb.autodb"

obj1.Open "Select name, age from person", obj

obj3.WriteLine "Name Age"

obj3.WriteLine "------"

While obj1.EOF = False

obj3.WriteLine obj1.Fields("Name") & “ “ & obj1.Fields("Age")

obj1.MoveNext

Wend

obj3.Close

Set obj3 = Nothing

Set obj2 = Nothing

obj1.Close

obj.Close

Set obj1 = Nothing

Set obj = Nothing

**Let’s see how it works:**

* **Firstly, objects –** **Connection Object, RecordSet Object and File Object**with the name ‘obj, ‘obj1’ and ‘obj2’ respectively are created using ‘createobject’ keyword.
* Then, a file object is opened mentioning the **location** of the text file where the data needs to be exported.
* Then a Connection Object that is created above is **opened** to start working with the object and the Provider details are mentioned.
* A Query is then **executed** after opening the recordset object by mentioning query and connection object.
* Then, name and age are written using **writeline** method of a file system object.
* Next, the main task is performed to write the values of name and age in the text file of a particular person using the **fields** of the recordset in the cells of the sheet from the database. The loop will go on until the EOF is not false (we have already discussed EOF).
* A File object is then **closed**and the file related objects are released.
* The objects are then **closed** as the task has been completed.
* Finally, the connection object and recordset object are **released** by using a ‘Nothing’ keyword.

These are some of the prime scenarios which are important in the proper understanding of the concept. They form the foundation to work and deal with the codes for handling different types of scenarios while dealing with Connection Objects in the script.

**Now, let’s understand the implementation of these scenarios through a simple example.**

**Example:**

<html>

<head>

<title>Let’s see implementation of Exporting data in files</title>

</head>

<body>

<script language=”vbscript” type=”text/vbscript”>

Function Exporttoexcelfile() ‘Function for exporting data to excel file

Set obj = CreateObject("ADODB.Connection")

Set obj1 = CreateObject("ADODB.RecordSet ")

Set obj2 = CreateObject("Excel.Application")

Set obj3 = obj2.Workbooks.Open ("C:\Users\Riya.xlsx")

Set obj4 = obj3.Worksheets(1)

obj.Provider =("Microsoft.ACE.OLEDB.12.0")

obj.Open "C:\Users\newdb.autodb"

obj1.Open "Select name, age from person", obj

obj4.cells(1, 1) = "Name"

obj4.cells(1, 2) = "Age"

row = 2

If obj1.EOF = True Then

Msgbox “No records found on the table!!”

End If

While obj1.EOF = False

obj4.cells(row, 1) = obj1.Fields("Name")

obj4.cells(row, 2) = obj1.Fields("Age")

obj1.MoveNext

row = row+1

Wend

obj3.Save

obj2.Quit

obj1.Close

obj.Close

Set obj4 = Nothing

Set obj3 = Nothing

Set obj2 = Nothing

Set obj1 = Nothing

Set obj = Nothing

End Function

Function Exporttotextfile() ‘Function for exporting data to text file

Set obj = CreateObject("ADODB.Connection")

Set obj1 = CreateObject("ADODB.RecordSet ")

Set obj2 = CreateObject("Scripting.FileSystemObject")

Set obj3 = obj2.OpenTextFile("C:\Users\Riya.xlsx")

obj.Provider =("Microsoft.ACE.OLEDB.12.0")

obj.Open "C:\Users\newdb.autodb"

obj1.Open "Select name, age from person", obj

obj3.WriteLine "Name Age"

obj3.WriteLine "------"

If obj1.EOF = True Then

Msgbox “No records found on the table!!”

End If

While obj1.EOF = False

obj3.WriteLine obj1.Fields("Name") & “ “ & obj1.Fields("Age")

obj1.MoveNext

Wend

obj3.Close

Set obj3 = Nothing

Set obj2 = Nothing

obj1.Close

obj.Close

Set obj1 = Nothing

Set obj = Nothing

End Function

Call Exporttoexcelfile() ‘Calling Function for exporting data to excel file

Call Exporttotextfile() ‘Calling Function for exporting data to text file

</script>

</body>

</html>

**Note:** For Output of an Example, you can go the location of the files and check if data is exported or not in the respective files.

### Conclusion

I am sure that by now you must have gained knowledge about the importance and effectiveness of using **VBS ADODB Connection Objects**.

[***Next tutorial #13***](https://www.softwaretestinghelp.com/vbscript-files-tutorial-13/)***: I will cover ‘File System Objects’ in my next tutorial.***

# VBScript File Objects: CopyFile, DeleteFile, OpenTextFile, Read And Write Text File

I decided to pick this as one of the topics in the series of the VBScript tutorial just because of its importance. File Object forms the basis of working with files

I will try to make you understand all the different codes, properties, and methods that are required to be written to work with the files in an easy manner so that you can easily write a piece of code by yourself.

Let’s start with the Properties and Methods which will be used while working with the files by providing access to handle operations like **create, delete, move, etc., on a file**. Hence, it is too important to gain an understanding of these before moving into the coding part.

**What You Will Learn:**[[hide](https://www.softwaretestinghelp.com/vbscript-files-tutorial-13/)]

* [Properties and Methods of a File Object](https://www.softwaretestinghelp.com/vbscript-files-tutorial-13/#Properties_and_Methods_of_a_File_Object)
* [Copying a File Using File Object](https://www.softwaretestinghelp.com/vbscript-files-tutorial-13/#Copying_a_File_Using_File_Object)
* [Deleting a File Using File Object](https://www.softwaretestinghelp.com/vbscript-files-tutorial-13/#Deleting_a_File_Using_File_Object)
* [Moving a File Using File Object](https://www.softwaretestinghelp.com/vbscript-files-tutorial-13/#Moving_a_File_Using_File_Object)
* [Creating a File Using File Object](https://www.softwaretestinghelp.com/vbscript-files-tutorial-13/#Creating_a_File_Using_File_Object)
* [Opening a Text File and Writing the Text Using File Object](https://www.softwaretestinghelp.com/vbscript-files-tutorial-13/#Opening_a_Text_File_and_Writing_the_Text_Using_File_Object)
* [Reading from a Text File Using File Object](https://www.softwaretestinghelp.com/vbscript-files-tutorial-13/#Reading_from_a_Text_File_Using_File_Object)
* [Conclusion](https://www.softwaretestinghelp.com/vbscript-files-tutorial-13/#Conclusion)
* [Recommended Reading](https://www.softwaretestinghelp.com/vbscript-files-tutorial-13/#Recommended_Reading)

### Properties and Methods of a File Object

There are different properties and methods that support to perform operations on a file.

**List of properties are as follows:**

* **Attributes**: This is used to know the number of attributes supported by a particular file.
* **DateCreated**: This returns the date along with the time of the creation of a particular file.
* **DateLastAccessed**: This returns the date along with the time when a particular file was last accessed.
* **DateLastModified**: This returns the date along with the time when a particular file was last modified.
* **Drive**: This returns the drive in which the particular file is located.
* **Name**: This returns the name of a particular file.
* **ParentFolder**: This returns the parent folder of a particular file as if the file is stored in C drive so it will return C:\.
* **Path**: This returns the path of a particular file as if the file is stored in C drive and the name of the file is a test so it will return C:\test.txt.
* **Size**: This returns the size of a particular file in bytes.
* **Type**: This returns the type of a particular file i.e. file type description like a file which ends with .vbs, for that “VBScript” will be returned.

These are the properties of a File Object. There is a **Files**Object also (this is a collection of file object) and let’s see its properties as follows:

* **Item:**This property is used to know the value of an item which is passed as a parameter.When a particular filename is passed as an item then this will return the full name of the file including the location of the file.
* **Count:**This is used to know the count of the File objects that are present in the collection.

**Now, let’s move to Methods.**

**List of some of the important File related methods are as follows:**

* **CopyFile/CopyFolder:**This is used to copy the mentioned file/folder to a specific destination.
* **DeleteFile/DeleteFolder:**This is used to delete a particular specified file/folder.
* **MoveFile/MoveFolder:**This is used to move the particular file/folder to the new destination as specified.
* **OpenTextFile:** This is used to open the file that is specified as a parameter and it returns as an instance of a text stream so that it can behave like a text file and operations of reading, writing & appending can be performed on that. If you want to open a text file for reading only then you can pass the constant value **1**, **2** in case of writing and **8** for appending purpose.
* **CreateTextFile:**This is used to create a text file that is specified as a parameter and it returns as an instance of text stream so that it can behave like a text file and the operations of reading, writing etc., can be performed.
* **FileExists/FolderExists/DriveExists:**This is used to check if the mentioned file/folder/drive exists or not. This returns True if it exists else False.
* **GetFile/GetFolder/GetDrive:**This is used to get the file/folder/drive object of the mentioned file/folder/drive which is specified as a parameter.

The above mentioned are the different properties and methods that you will use while dealing with the File Objects.

Now, let’s move on to the practical implementation and see the working of these objects.

### Copying a File Using File Object

**Following is the Code for copying a file:**

Set obj = createobject(“Scripting.FileSystemObject”) ‘**Creating a File Object**  
Dim loc,loc1                                                        **‘Declaring variables**  
src=”C:\app\pictures\img1.jpg”       ‘**Mentioning source location of the file to be copied**  
dest=”C:\app1”                                                   **‘Mentioning the destination**  
obj.CopyFile src,dest                           **‘CopyFile Method is used for copying the file**  
Set obj=Nothing                                                     ‘**Releasing File object**

**Let’s see how it works:**

* Firstly, a **File Object**with the name ‘obj’ is created using ‘createobject’ keyword and File System Object in the parameter is defined.
* Then, **variables** are declared for mentioning the destination and source location of the file to be copied.
* A **CopyFile method** is then used to copy the source file to the destination mentioned above.
* Finally, the object – obj is **released** by using a ‘Nothing’ keyword.

### Deleting a File Using File Object

**Following is the Code for deleting a file:**

Set obj = createobject(“Scripting.FileSystemObject”) ‘**Creating a File Object**  
Dim filename1                                                       **‘Declaring variables**  
filename1=”C:\app\pictures\img1.jpg” ‘**Mentioning name and location of the file to be deleted**  
obj.DeleteFile filename1               **‘DeleteFile Method is used for deleting the file**  
Set obj=Nothing                                                       ‘**Releasing File object**

**Let’s see how it works:**

* Firstly, a **File Object**with the name ‘obj’ is created using ‘createobject’ keyword and File System Object in the parameter is defined.
* Then, the **variable** is declared for mentioning the location of the file which has to be deleted.
* A **DeleteFile method** is then used to delete the file.
* Finally, the object – obj is **released** by using a ‘Nothing’ keyword.

### Moving a File Using File Object

**Following is the Code for moving a file:**

Set obj = createobject(“Scripting.FileSystemObject”) ‘**Creating a File Object**  
Dim filename1,filename2                                   **‘Declaring variables**  
filename1=”C:\app\pictures\img1.jpg” ‘**Mentioning the name and source location of the file to be moved**  
filename2=”C:\Users\img1.jpg” ‘**Mentioning the name and destination location of the file to be moved**  
obj.MoveFile filename1,filename1 **‘MoveFile Method is used for moving the file to the destination**  
Set obj=Nothing                                           ‘**Releasing File object**

**Let’s see how it works:**

* Firstly, a **File System Object**with the name ‘obj’ is created using ‘createobject’ keyword and File System Object in the parameter is defined.
* Then, the **variables** are declared for mentioning both the source location and the destination location of the file to be moved.
* A **MoveFileFile method** is then used to move the file.
* Finally, the object – obj is **released** by using a ‘Nothing’ keyword.

### Creating a File Using File Object

**Following is the Code for creating a text file:**

Set obj = createobject(“Scripting.FileSystemObject”) ‘**Creating a File Object**  
src=”C:\Users\Riya\file1.txt”          ‘**Mentioning name and location of the file to be created**  
obj CreateTextFile src                    **‘CreateTextFile Method is used for creating the file**  
Set obj=Nothing                                                ‘**Releasing File object**

**Let’s see how it works:**

* Firstly, a **File Object**with the name ‘obj’ is created using ‘createobject’ keyword and File System Object in the parameter is defined.
* Then, the **variable** is declared for mentioning the name and location of the file which has to be created.
* A **CreateTextFile method** is then used to create the file that is mentioned above.
* Finally, the object – obj is **released** by using a ‘Nothing’ keyword.

**Note**: In the same way, a **Folder** can be created, deleted and copied using CreateFolder, DeleteFolder and CopyFolder methods respectively.

### Opening a Text File and Writing the Text Using File Object

**Following is the Code for writing text inside a file:**

Set obj = CreateObject(“Scripting.FileSystemObject”)   ‘**Creating a File Object**  
Const ForWriting = 2               **‘Defining Constant Value to write in a file**  
Set obj1 = obj.OpenTextFile(“C:\app.txt”, ForWriting) **‘Opening a text file and writing text inside it**  
obj1.WriteLine(“This text is written in a file”)      **‘Text is written using WriteLine method**  
obj1.Close                                                                  **‘Closing a File**  
Set obj=Nothing                                                     ‘**Releasing File object**

**Let’s see how it works:**

* Firstly, a **File Object**with the name ‘obj’ is created using ‘createobject’ keyword and File System Object in the parameter is defined.
* A **constant** value is then defined for writing purpose as VBScript cannot automatically access the COM objects and so to it is required to define a constant value to pass a writing parameter value in the OpenTextFile method.
* Then, a text file is opened using **‘OpenTextFile’ method** and the writing operation is performed.
* A text is then **written** inside a file.
* Then, File is **closed**.
* Finally, the objects – obj and obj1 are **released** by using a ‘Nothing’ keyword.

**Note**: In the same way, Appending operation can also be performed on a file by defining the constant value as 8.

### Reading from a Text File Using File Object

**Following is the Code for reading text from a file:**

Set obj = CreateObject(“Scripting.FileSystemObject”)    ‘**Creating a File Object**  
Const ForReading = 1                      **‘Defining Constant Value to read from a file**  
Set obj1 = obj.OpenTextFile(“C:\app.txt”, ForReading) **‘Opening a text file and reading text from it**  
Dim str,str1  
str=obj1.ReadAll **‘All text from the file is read using ReadAll**  
Msgbox str                                **‘Contents of a file will be displayed through the message box**  
Do while obj1.AtEndofStream                  **‘Reading text line wise using Do Loop and ReadLine**  
str1=obj1.ReadLine  
Msgbox str1  
Loop  
obj1.Close                                         **‘Closing a File**  
Set obj=Nothing                              ‘**Releasing File object**

**Let’s see how it works:**

* Firstly, a **File Object**with the name ‘obj’ is created using ‘createobject’ keyword and File System Object in the parameter is defined.
* A **constant** value is then defined for the reading purpose as VBScript cannot automatically access the COM objects. Hence it is required to define a constant value to pass a reading parameter value in the OpenTextFile method.
* Then, a text file is opened using **‘OpenTextFile’ method** and the reading operations are performed.
* A whole text is then read from a file using ‘**ReadAll’**.
* Another way of reading from a file is line-wise. Do loop is used to read a text from a file line-by-line using **‘ReadLine’**.
* Then, the File is **closed**.
* Finally, the objects – obj and obj1 are **released** by using a ‘Nothing’ keyword.

These are some of the prime scenarios which should be understood properly. They form the foundation to work and deal with the codes for handling different types of scenarios while dealing with File Objects in the script.

Given below are the different types of **Examples**by taking a reference to the above scenarios and topics.

**Example1:**

**Making use of ‘Count’ and ‘Item’ properties of ‘Files’ object along with ‘GetFolder’ method**

<html>

<head>

<title>Let’s see implementation of Files Object with properties</title>

</head>

<body>

<script language=”vbscript” type=”text/vbscript”>

Dim obj, obj1, obj2, itm, cnt

Set obj= CreateObject(“Scripting.FileSystemObject”)

Set obj1=obj.GetFolder(“C:\Users\Riya”)

Set obj2=obj1.Files

itm=obj2.Item(“riya.vbs”)

cnt=obj2.Count

Msgbox(itm)

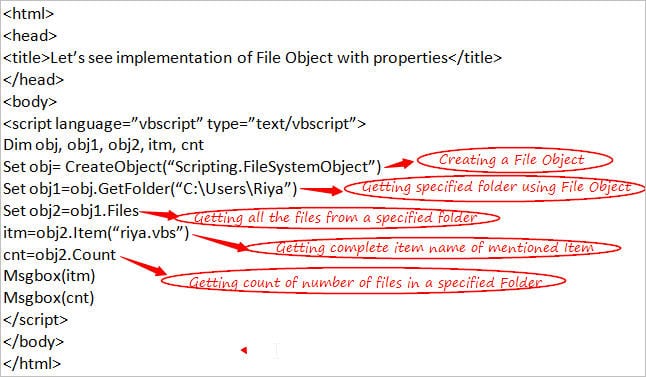
Msgbox(cnt)

</script>

</body>

</html>

**Output is:**C:\Users\Riya\riya.vbs  
6 (assuming total of 6 files available in a folder)

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/01/Example1.jpg)

**Example2:**

**Making use of different properties and methods of a ‘File’ object**

<html>

<head>

<title>Let’s see implementation of a File Object</title>

</head>

<body>

<script language=”vbscript” type=”text/vbscript”>

Dim obj, obj1

Set obj= CreateObject(“Scripting.FileSystemObject”)

Set obj1=obj.GetFile(“C:\Users\Riya.vbs”)

Msgbox “DateCreated of File is “& obj1.DateCreated & “<br>”

Msgbox “Attributes of File is “& obj1.Attributes & “<br>”

Msgbox “DateLastAccessed of File is “& obj1.DateLastAccessed & “<br>”

Msgbox “DateLastModified of File is “& obj1.DateLastModified & “<br>”

Msgbox “Name of File is “& obj1.Name & “<br>”

Msgbox “Drive of File is “& obj1.Drive & “<br>”

Msgbox “ParentFolder of File is “& obj1. ParentFolder & “<br>”

Msgbox “Path of File is “& obj1. Path & “<br>”

Msgbox “Size of File is “& obj1. Size & “<br>”

Msgbox “Type of File is “& obj1. Type & “<br>”

obj1.Copy “C:\Users\Riya.vbs”,”C:\Users”

obj1.Move “C:\Users\Riya.vbs”,”D:\”

obj1.Delete

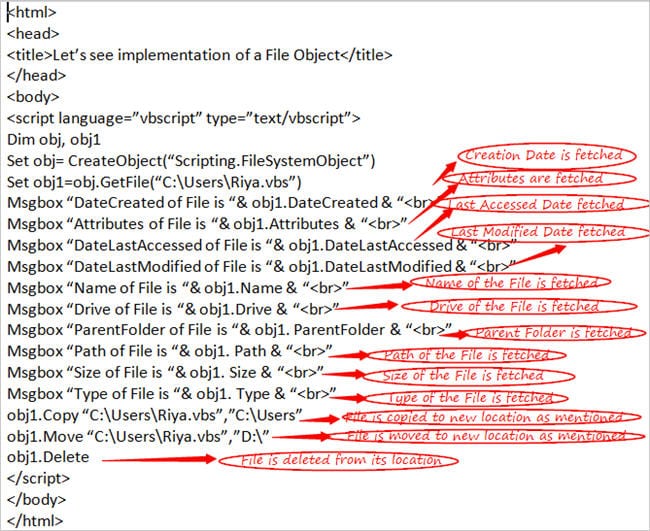
</script>

</body>

</html>

**Note**: I am using **Copy, Move and Delete** methods instead of CopyFile, MoveFile and DeleteFile because directly reference of a particular file is used and is saved in ‘obj1’.

**Output is:**DateCreated of File is 30/12/2017 04:04:28  
Attributes of File is 20  
DateLastAccessed of File is 30/11/2017 02:04:38  
DateLastModified of File is 11/10/2017 06:06:48  
Name of File is Riya.vbs  
The drive of File is C:  
ParentFolder of File is C:\  
Path of File is C:\Users\Riya.vbs  
Size of File is 600  
Type of File is VBScript Script File

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/01/Example2.jpg)

### Conclusion

We explained the importance and effectiveness of using **VBScript File Objects** which in turn would help you in working with the file related scenarios in an easy manner.

[***Next Tutorial #14***](https://www.softwaretestinghelp.com/vbscript-error-handling-tutorial-14/)**: I will cover ‘VBScript Error Handling’ concept in my next tutorial.**

# VBScript Error Handling: VBScript On Error, On Error GoTo 0, On Error Resume Next

**What You Will Learn:**[[hide](https://www.softwaretestinghelp.com/vbscript-error-handling-tutorial-14/)]

* [Overview](https://www.softwaretestinghelp.com/vbscript-error-handling-tutorial-14/#Overview)
* [Purpose of Error Handling](https://www.softwaretestinghelp.com/vbscript-error-handling-tutorial-14/#Purpose_of_Error_Handling)
* [Methods of Error Handling in the VBScript](https://www.softwaretestinghelp.com/vbscript-error-handling-tutorial-14/#Methods_of_Error_Handling_in_the_VBScript)
  + [#1) On Error Resume Next](https://www.softwaretestinghelp.com/vbscript-error-handling-tutorial-14/#1_On_Error_Resume_Next)
  + [#2) Err Object:](https://www.softwaretestinghelp.com/vbscript-error-handling-tutorial-14/#2_Err_Object)
* [Conclusion](https://www.softwaretestinghelp.com/vbscript-error-handling-tutorial-14/#Conclusion)
* [Recommended Reading](https://www.softwaretestinghelp.com/vbscript-error-handling-tutorial-14/#Recommended_Reading)

### Overview

**Error Prevention** is an aspect of Error handling which means taking effective measures inside a script in order to avoid the occurrence of errors.

**Errors may include any of the following:**

**#1)**Making use of **Exist**property to check the existence of an object before making any operations on the same.

**#2) Synchronization** techniques to deal with the delay and wait for operations inside a script.

**#3)** Making use of an **Option Explicit** statement to avoid misspelled words or spelling problems.

**Now, let’s dive deep about Error Handling as it’s important to understand how to deal with the errors while working with the scripts.**

### Purpose of Error Handling

The main purpose of performing a testing activity is to **find and resolve the errors. T**hough it is not possible to have a 100% error Free S/W, still you can take measures to bring down the error count as much as possible by making use of Error Handling Mechanism in your scripts.

Situations like issues in mathematical computations or any type of errors can be handled with the help of Error Handling.

**Now, Let’s see some of the methods of Error Handling in the VBScript.**

### Methods of Error Handling in the VBScript

VBScript basically supports 2 main methods to handle errors in the scripts.

**They are as follows:**

#### **#1) On Error Resume Next**

Most of us must have come across this method in some of the other programming languages. This method, as the name itself suggests, **moves the control of the cursor to the next line of the error statement.**

Which means, if any runtime error occurs at a particular line in the script then the control will move into the next line of the statement where the error has occurred.

**A Simple Example**:

In this case, the division is by 0 and if you do not want your script to get stuck due to this error then you put ***‘On Error Resume Next’*** at the top of your script as shown below.

On Error Resume Next (***Putting error handling statement)***  
Dim result  
result = 20/0 (***Performing division by 0 Scenario)***  
If result = 0 Then (***Checking the value of result variable)***  
Msgbox “Result is 0.”  
Else  
Msgbox “Result is non-zero.”  
End If

#### **#2) Err Object:**

This method is basically used to capture the **details of the Error**. If you want to know more about the Error like Number, description, etc., then you can do so by accessing the properties of this Object.

As this is an intrinsic object, there is no need to create an instance of this object to access its properties i.e. you can use this directly in your scripts.

**Following is the list of properties of Err Object with their details:**

**Number**: This will tell you the error number i.e. the integer value of the type of the error occurred.

**Description**: This will tell you about the error i.e. the description of the error.

**Raise**: This will let you raise the specific error by mentioning its number.

**Clear**: This will clear the error i.e. will set to error handler to nothing.

**Let’s use the same Example in this case also:**

Dim result  
on error resume next  
result = 20/0 ‘(***Performing division by 0 Scenario***)  
If Err.Number <> 0 Then ‘(***Making use of Err Object’s Number property***)  
Msgbox “Number of the Error and Description is “& Err.Number & ” ” & Err.Description ‘(***Give details about the Error***)  
Err.Clear ‘(***Will Clear the Error***)  
End If  
on error goto 0

**One more to the list:**

**#3) On Error GoTo 0:**

This method is however not an Error Handler mechanism directly because this is used to **disable any error handler** that is used in the script. This will set the handler to nothing i.e. no more error handler will be supported in the script.

**Also read =>>**[**How to handle errors in VBA**](https://www.softwaretestinghelp.com/error-handling-in-vba/)

### Conclusion

I hope this tutorial must have provided insight regarding the importance and effectiveness of using Error Handling. This tutorial will, in turn, help you in dealing with the VBscript errors in a more effective manner.

**Finally, this is the last topic of**[**VBScript tutorial series**](https://www.softwaretestinghelp.com/vbscript-tutorial-1/)**.**

[**Next Tutorial #15**](https://www.softwaretestinghelp.com/popular-vbscript-interview-questions-answers/)**:** I will cover some **interview questions related to VBScript** in my next tutorial which will include questions from all the topics that I covered till now in this series.

# 60+ Top VBScript Interview Questions And Answers [2022 LIST]

In my personal experience, there is no specific way or formula to get through an interview process and it totally depends on you and the type of Interviewer but still, it’s good to be prepared on your profile for your own confidence and knowledge, so that you don’t feel hesitant to answer the questions asked during an Interview.

### VBScript Interview Questions And Answers

**Following is the list of basic but most important interview questions in VBScript that can be asked in an Interview:**

**Q #1) What is** **VBScript language used for and which earlier language is it modeled upon?**

**Answer:** VBScript is a lightweight primary scripting language which is used for automation of the Scripts in QTP (Quick Test Professional) tool. This is modeled upon Visual Basic language.

**Q #2) What are the environments supported by VBScript language?**

**Answer: VBScript can run in the following 3 environments:**

* **IIS (Internet Information Server)** **–** This is Microsoft’s Web Server.
* **WSH (Windows Script Host) –**This is the hosting environment of the Windows Operating System.
* **IE (Internet Explorer) –** This is the most frequently used environment to run scripts and this is the simple hosting environment.

**Q #3) Which data type/types are supported by VBScript language and what are their specialties?**

**Answer:** There is only one data type that is supported by VBScript language and it is called as the ‘Variant’. If we use this data type in the String context then this will behave like a String and if we use this in Numeric context then this will behave like a Number. This is the specialty of the Variant data type**.**

**Q #4) What is the extension of the VBScript file?**

**Answer:** VBScript file is saved with an extension of **.vbs**.

**Q #5) How are Comments handled in the VBScript language?**

**Answer:** Any Statement that starts with a single quote **(‘)** or with the keyword **‘REM’** is treated as a Comment.

**Q #6) Which respective symbols are used to separate a line and to break the lengthy statement into multiple statements in the VBScript language?**

**Answer:** **Colons (:)** act as a line separator and **Underscore (\_)** is used to break the lengthy statement into multiple statements in the VBScript language.

**Q #7) What are the keywords in the VBScript language?**

**Answer:** There are some words that work as Reserved Words and they cannot be used as Variables names, Constant names or any other Identifier names, these are known as keywords. Some of the keywords in the VBScript language are Not, Nothing, Preserve, Optional, etc.

**Q #8) Is VBScript language a Case-Sensitive language and what does it mean?**

**Answer:** No. This actually means that variable names, keywords, constants, and other identifiers are not required to be typed with a consistent capitalization of letters i.e. if you type ‘Optional’ keyword as OPTIONAL, optional or Optional then these all mean the same in the VBScript language.

**Q #9) What are the naming conventions while declaring a variable in the VBScript language?**

**Answer: Following are the rules for declaring a Variable name:**

* It must always start with a letter. **For Example,** output, name, etc. Variable Name should not start with a number or any special character like \_va123, 12non, etc.
* It cannot exceed the limit of 255 characters.
* It should never contain a period (.).

**Q #10) Which keyword is used to declare a variable in the VBScript language?**

**Answer:** The **Dim** keyword is used to declare a variable in the VBScript language. However, depending upon the scope of the variable, public or private keywords can also be used.

**Q #11) What are the 2 ways in which a variable can be declared in the VBScript language?**

**Answer: Two ways to declare a variable are:**

* **Implicit Declaration**: When variables are used directly without declaration, it is termed as Implicit Declaration. However, it’s not a good practice because at any point if the variable name is not spelled correctly in the script then it can produce wrong results while running and at times, it will not even be easy to be detected by the user.
* **Explicit Declaration:**Declaring the variables before using them is known as the Explicit Declaration of variables.

**Q #12) What is the use of Option Explicit Statement?**

**Answer:** This provides a mechanism where the user has to declare all the variables using Dim, Public or Private Statements before using them in the Script.

If the user tries to use the variables which are not declared in case of Option Explicit then an error occurs. It is always recommended to use ‘Option Explicit’ at the top of the code so that even if you use the wrong name of the variable unintentionally, you can then correct it immediately without any confusion.

**Q #13) How are values assigned to the variables in the VBScript language?**

**Answer:** Values are assigned with the help of **Equal (=)** Operator. The name of the variable comes on the left and the value which is assigned to the variable is on the right-hand side of the ‘=’ Operator.

**Q #14) How are values assigned to String type and Numeric type variables?**

**Answer:** If the variable to which value is to be assigned is of String type then it can be assigned using double quotes**(“ ”)** and if the variable to which value is to be assigned is of the Numeric type then it can be assigned without using double-quotes.

**Q #15) Explain the scope of the variables using Dim, Public, and Private keywords respectively.**

**Answer:** If the variable is declared using **Dim** keyword inside the function then its scope will be limited to the function level only i.e. this variable cannot be accessed once the function ends.

If the variable is declared using a **Private** keyword inside the function then its scope will not be limited till function level only but it can be accessed everywhere in that particular script.

If the variable is declared using **Public** keyword inside the function then its scope will not be limited till function level alone, but it can be accessed everywhere in that particular script and also in the other scripts.

**Q #16) How can constants be declared in the VBScript language?**

**Answer:** Constants are named memory locations within a program that never change their values during the execution of the script. **‘Const’** keyword is used to declare Constants in the VBScript language.

**Q #17) Which constant is used for print and display functions and works as same as pressing Enter key?**

**Answer: vbCrLf**is used for print and display functions representing a carriage return with line feed character having values as Chr(13) & Chr(10). This works in the same manner as in the case of pressing an Enter key. This is a pre-defined constant of the VBScript language.

**Q #18) How many types of Operators are available in the VBScript language?**

**Answer:**There are 4 types of Operators that are supported by the VBScript language.

**They are:**

* Arithmetic Operators
* Comparison Operators
* Logical Operators
* Concatenation Operators

**Q #19) Which Operator is used for fetching the modulus of the 2 numbers in the VBScript language?**

**Answer:** **MOD** Operator is used for fetching the modulus of the 2 numbers in the VBScript language.

**Q #20) Which Operator is used to perform the comparison among 2 operands in the VBScript language?**

**Answer: ‘==’** Equal Operator is used for performing the comparison among 2 operands in the VBScript language i.e. if we check 1 == 2 then it will give False.

**Q #21) Which Operator is used to concatenate the 2 values in the VBScript language?**

**Answer:** **‘&’** operator is used to concatenate the 2 values in the VBScript language.

**Q #22) If we take 2 strings as “Good” and “Bad” then what will ‘+’ and ‘&’ operators return?**

**Answer:** ‘&’ and ‘+’ both work as a Concatenation Operator in case of the String values. Hence these both will return the same result as GoodBad.

**Q #23) Which Operator can be used to change the value of the operand or change the state of the condition?**

**Answer:** **‘NOT’** Operator is used as a logical operator and can be used to change the value of the operand or change the state of the condition i.e. if the condition is True then this will change it to False and vice versa.

**Q #24) Out of the different types of Operators, which are evaluated first and last in the VBScript language?**

**Answer:** ‘Arithmetic’ Operators are evaluated first and ‘Logical’ Operators are evaluated at last in the VBScript language.

**Q #25) Which conditional statement is the most convenient one to use in the case of multiple conditions in the VBScript language?**

**Answer:** **‘Select Case’** is the most convenient one to use in case of multiple conditions in the VBScript language as in case of SELECT Statement, you can directly move to the exact case without wasting time to go into each condition one by one.

**Q #26) What are the different types of loops available in the VBScript language?**

**Answer:** The loops which are available in the VBScript language are broadly categorized into 3 types and they are – **For Loops, Do Loops**and**While Loops.**

**Q #27) Which loop is used in the case of arrays in the VBScript language?**

**Answer:** **For Each Loop** is used in the case of Arrays. This is an extension of For Loop only. In case of repeating the code for each index value of an array then you can use ‘For Each Loop’.

**Q #28) What is the difference between For Loop and While Loop?**

**Answer:** This is a very important Interview Question which is asked multiple times.

For Loop is used when we exactly know the number of times loop (i.e. for i = start to end) has to be executed unlike in case of While Loop.

In ‘For Loop’ in the VBScript, the counter is incremented automatically if not mention step keyword by 1 when loops go to the next keyword whereas in ‘While Loop’, the counter condition has to be mentioned explicitly inside the brackets.

**Q #29) What is the difference between Do Until Loop and Do While Loop?**

**Answer:** Do While Loop first checks the condition and if it is true only after that the statements are executed and in case of Do Until, the loop will be executed until the condition becomes false.

**Q #30) How many types of Procedures are available in the VBScript language?**

**Answer:** There are 2 types of Procedures in the VBScript language – Sub Procedures and Function Procedures.

**Sub** is a type of procedure that includes a set of statements inside the block of the code and after execution, it does not return any value.

The **Function** is a type of procedure that includes a set of statements inside the block of the code and after execution, it may return value also. This can take input if required, depending on the situation.

**Q #31) What are the differences between Sub Procedures and Function Procedures?**

**Answer: Differences are as follows:**

* Sub Procedure never takes an input while the Function Procedure may take an input if required.
* Sub Procedure starts and ends with using Sub and End Sub respectively while Function Procedure starts and ends with Function and End Function respectively.
* The most important difference is Sub Procedure never returns a value while Function Procedure may return a value.

**Q #32) What are the 2 ways to pass a value to the Function?**

**Answer: The 2 ways to pass a value to the function are:**

* **Pass by Value:**When arguments are passed and any changes that take place in the Called procedure in the value of a variable does not persist then it means it is passing by value. The keyword used in this case is **ByVal**.
* **Pass by Reference:**When arguments are passed and any changes that take place in the Called procedure in the value of a variable persist then it means it is passing by reference. The keyword used in this case is **ByRef**.

**Q #33) Which In-Built function is used to format the number in the VBScript language?**

**Answer:** **FormatNumber** Conversion function is used to convert the specified expression in the form of a Number.

**Q #34) Which In-Built functions are used to convert the specified expression in the form of Date and String in the VBScript language?**

**Answer: cDate**is one of the frequently used conversion functions for converting the expression which includes Date or Time parameter into Date subtype.

**cStr** is the conversion function which is used for converting the expression into the String subtype.

**Q #35) How are Arrays declared in the VBScript language?**

**Answer:** Declaration of Array can be done in the same manner in which variables are declared but with a difference that array variable is declared by using **parenthesis ‘()’**.

The **Dim** keyword is used to declare an Array.

**Ways to declare an Array:**There are 3 ways in which an Array can be declared.

**They are as follows:**

**Way 1:** Dim array1()

Here, array1 is the name of an array and since parenthesis is empty it means that the size of an array is not defined here.

If you want to declare an array by mentioning its size then it can be done in the following way.

**Way 2**: Dim array1(5)

Here, array1 is declared with the size as 5 which states it holds 6 values considering the index of an array always starts from 0. These 5 values can be of integer type, string or character types.

**Way 3**: array1 = Array(1,2,3,4,5,6)

Here, Array Function is used to declare an array with a list of the arguments inside the parenthesis and all integer values are passed directly inside the parenthesis without any need of mentioning the size of an array.

**Note**: **Index value of an Array can never be a negative value.**

**Q #36) What are lbound and ubound in the VBScript language?**

**Answer: lbound**indicates the smallest subscript or index of an Array in the  VBScript language and this always returns 0, as the index value of an Array always starts from 0.

**ubound**returns the largest subscript of a defined array or can say it indicates the size of an Array. If the size of an array is 5 then the value of the ubound is 5.

**Q #37) Which In-Built function related to an Array joins substrings into one string in the VBScript language?**

**Answer:** Join function combines multiple substrings into a String. Here, the string returned value includes various substrings in an array and thus joins all the substrings into one string.

**Syntax:** Join(array,[delimiter]. Using delimiter is an optional condition.

**Q #38) How many types of Arrays are available in the VBScript language?**

**Answer: There are basically 2 types of Arrays that are used in the VBScript:**

* **Single Dimensional Array**: This is a simple type of array which is used more often in the scripts.
* **Multi-Dimensional Array**: When an Array is having more than 1 dimension then it is known as a multi-dimensional array. Normally, a 2-dimensional array is the one which is used most of the times i.e. there will be rows and columns in an array. The maximum dimension of an array can reach to 60

**Q #39) When are REDIM statement and PRESERVE keyword used in the VBScript language?**

**Answer:** **This is a very important Interview Question that has been asked many times**.

**REDIM**statement is used to re-define the size of an array. When the array is declared without any size, an array can be declared again using REDIM with the feasibility of specifying the size of an array.

**PRESERVE** keyword is used to preserve the contents of a current array when the size of an array gets changed.

***Let’s understand the usage of these keywords with the help of a simple example:***

<html>

<head>

<title>Let’s see implementation of Redim and Preserve</title>

</head>

<body>

<script language=”vbscript” type=”text/vbscript”>

Dim array1()

REDIM array1(3)

array1(0) = “hello”

array1(1) = 12

array1(2) = 13

array1(3) = “how are you”

REDIM PRESERVE array1(5)

array1(4) = 15

array1(5) = 16

For i = 0 to ubound(array1)

Msgbox “Value present at index ” & i & ” is “ & array1(i) & “<br />”

Next

</script>

</body>

</html>

**Q #40) What is the use of the Date function in the VBScript language?**

**Answer:** Date function displays the current system Date and Time in the VBScript language.

**Q #41) Which Date function is used in the VBScript language to find the difference between the 2 dates?**

**Answer:** **DateDiff** function is used to fetch the difference between the 2 dates that are specified as parameters on the basis of the interval specified.

**Q #42) What is the use of the FormatDateTime function in the VBScript language?**

**Answer:** This is a format function which is used to **convert the Date to some specific format** based on the parameters that are supplied to the function. The syntax of this is FormatDateTime(Date, Format). This is a widely used format function.

**Q #43) Which function is used in the VBScript language to convert the specified expression into a Date type value?**

**Answer: cDate** is used to convert a valid expression into a Date type value. The syntax of this is cDate(date) i.e. any valid Date/Time expression will be converted into a particular Date.

**Q #44) What is the use of the Instr function?**

**Answer:** This is used to find the position value of the substring at its first occurrence inside the main string.

This function requires 2 strings to be specified to perform this search operation and the search operation starts from the first character.

**Syntax:** is InStr(name of string1, the name of string2).

If the name of string1 or string2 is null or “ ” then this function will return null and 0 respectively. This return >=1 values when the string is found and 0 in the case when the string is not found.

**Q #45) How to get the length of the string by making use of the String function?**

**Answer:** **Len** function is used to get the length of a specified string i.e. the total number of characters of a specified String.

**Syntax:** Len(name of the string).

**Q #46) Which function is used to perform string comparison?**

**Answer:** **StrComp** is used to compare the 2 strings and return values on the basis of the comparison. This returns 0 if string1 = string2,-1 if string1<string2,1 if string1>string2 and null if any of the strings is null.

**Syntax:** StrComp(name of the string1, name of the string2).

**Q #47) How can the spaces from the string be removed?**

**Answer:** **Trim** function is used to trim/remove the spaces from both the sides of the specified String.

**Syntax:** Trim(name of the string).

**Q #48) How can you fetch the value of a Cookie?**

**Answer:** document.cookie stores the information of key-value pairs and expiration date values of a Cookie.

document.cookie = “key1=name of the value1;key2=name of the value2,…….,expires=date”.

**‘;’** is used to separate the key-value pairs.

**Q #49) What are Events in the VBScript language?**

**Answer:** Events are the **Actions** that occur when any activity is performed like any mouse click, pressing the keys, mouse hover, etc. With the help of writing a piece of code in the programming languages like VBScript, these events can be captured and actions can be performed as per your requirements by making the best use of the Event Handling mechanism.

**Q #50) Which Event is triggered when mouse focus comes out of an element in the VBScript language?**

**Answer:** **MouseOut** Event is triggered when mouse focus comes out of an element in the VBScript language.

**Q #51) When does ‘On Click of Button’ event gets triggered in the VBScript language?**

**Answer:** This Event occurs in case of the clicking of any button that is present on any HTML page.

**Q #52) Which object is used to work with the excel sheets in the VBScript language and what statement is used to create this object?**

**Answer:** **Excel Objects** provide supports to the coders to work and deal with the Excel Sheets.

**Set obj = createobject(“Excel.Application”)** is the way to create an Excel Object.

**Q #53) Which object is used to work with the database in the VBScript language and what statement is used to create this object?**

**Answer:** **Connection Objects** provide supports to the Coders to work and deal with the database. As such, there is not any straight-forward mechanism to connect to the database in QTP but by making use of **ADODB Objects**, you can interact with the database and work with the SQL Queries to fetch the data from the database.

**ADO stands for ActiveX Data Objects** and this provides a mechanism to act as an intermediary between the QTP and the Database.

**Set obj = createobject(“ADODB.Connection”)**  is the way to create a Connection Object.

**Q #54) What is the use of the ‘Open’ method to work with the database in the VBScript language and what connection string is passed in the same and what is its usage?**

**Answer:** This is used to open a database connection object/recordset object.

***obj.Open“Provider=SQLQLEDB;Server=.\SQLEXPRESS;UserId=test;Password=P@123;Database =AUTODB”*** is the connection string for opening a database connection.

The connection string is a very useful property and this is used for creating a database connection and includes connection information like details of the Driver, Database Server Name, Username, and Password.

**Q #55) Why is it recommended to close the database connection every time after the work is completed?**

**Answer:** **This is a very important Interview Question which has been asked many times.**

It is recommended to close the resource after its usage is completed although it’s not mandatory as library or driver ultimately will close the connection but this is required to avoid any negative effects due to improper closing of the connections which can even lead to a restriction in the access of the database by some of the users.

**Q #56) What is the use of the RecordSet object and which statement is used to create such an object?**

**Answer:** The **RecordSet** object is used for holding the records of the query which are extracted from the database.

**Set obj = createobject(“ADODB.RecordSet”)** is the statement for creating a RecordSet object.

**Q #57) How can you create a file object to work with the files in the VBScript language?**

**Answer:** **Set obj = createobject(“Scripting.FileSystemObject”)** is the statement for creating a File object.

**Q #58) What methods are used to create text files and open text files in the VBScript language?**

**Answer:** **CreateTextFile** and **OpenTextFile** methods are used to create open text files and open text files respectively in the VBScript language.

**Q #59) What is the purpose of the Err object in the VBScript language?**

**Answer:** This is basically used to capture the**details about the Error** i.e. if you want to know about the Error Number, description and other details then you can do so by accessing the properties of this Object.

**Q #60) Why is Error handling required?**

**Answer:** You can take measures to get minimum number of errors as possible by making use of Error Handling Mechanism in your scripts. Situations like issues in mathematical computations or any type of error can be handled with the help of Error Handling.

**Q #61) What purpose does ‘On Error Resume Next’ serves?**

**Answer:** **On Error Resume Next** moves the control of the cursor to the next line of the error statement i.e. if any runtime error occurs at any particular line in the script then the control will move into the next line of the statement where the error has occurred.

### Conclusion

This is all about VBScript interview questions. I hope this tutorial must have provided you a great insight regarding the questions that can be asked during an Interview and you can now confidently handle your interview process.

This has brought me to the end of the [**VBScript learning tutorial series**](https://www.softwaretestinghelp.com/vbscript-tutorial-1/) and I hope you must have now gained enough knowledge of the VBScript language. Do practice all the topics for better understanding and knowledge.

***Happy Reading!! Happy Testing!!***