What you should already know

Before studying VBScript, you should already have at least a basic understanding of HTML and/or XHTML. VBScript scripts are placed on webpages with HTML/XHTML tags and without knowing these tags, you will not know where and how to place VBScript scripts on a webpage. Check out our **HTML tutorials** and **XHTML tutorials** if you are not yet familiar with these languages.

What is VBScript?

VBScript is a scripting language used to provide dynamic and interactive content on webpages.

VBScript is short for Visual Basic Scripting Edition. VBScript is a lighter version of the Visual Basic programming language, and like Visual Basic, VBScript was developed by Microsoft.

VBScript and Internet Explorer

The most important thing to know about VBScript is that it is a **proprietary** language.

VBScript was designed by Microsoft to work in Internet Explorer and browsers based on Internet Explorer's engine such as Flashpeak's **SlimBrowser**.

VBScript code will NOT work in web browsers such as Firefox. Opera, Safari, and Chrome.

Due to the proprietary nature of VBScript, it is not a very popular scripting language on the world wide web. Instead, many people prefer using languages that are not proprietary and run on a variety of web browsers.

What's the purpose of learning VBScript if it is a proprietary language and there are non-proprietary alternatives out there?

Here are some reasons:

1. **Internet Explorer is the most popular web browser** - knowing how to write code specifically for it is a plus
2. **Browser specific code** - There may come a time when you want to provide separate content on a webpage for users of a particular web browser
3. **Knowing another language** - It's always good to know more. Even if you will use VBScript alot less than other languages, you still have knowledge of another web language

What can be done with VBScript?

- **Interact with the user** - For example, you can ask the user for their name and print a custom message with it on a webpage such as "Hello Roger!"
- **Form validation** - Validate data entered by the user. For example, you can check to make sure that certain fields in a form were actually filled out, or that a piece of data contains the required amount of characters

- **Perform calculations** - You can use VBScript to perform a range of mathematical calculations ranging from simple arithmetic to complex formulas
- **Act upon events** - Specify code to run when certain events occur. For example, when the user clicks a button you can instruct VBScript to display a message in an alert box

All this and much more can be done with VBScript!

This tutorial focuses on:

- **Declaring a script**
- **Printing text on a webpage**
- **Including HTML tags in a script**
- **Including comments in a script**
- **Dealing with browsers that do not support scripts**
- **Dealing with browsers that do not support VBScript**

Declaring a script

A VBScript script is declared using HTML's <script> tag. This tag denotes that there will be a script on a webpage, and the scripting language that will be used in the script is denoted by this tags *type* attribute. When working with VBScript, it should be set to "text/vbscript"

Example:
<script type="text/vbscript"> *script content goes here* </script>

A script can be placed in the head section or the body section of a webpage, but a it will be executed differently depending on where it is placed.

A script placed in the head section of a webpage

A script placed in the head section of a webpage will be executed when it is called, or when certain events are triggered such as the clicking of a button or when a form is submitted.

Example:
<html> <head> **<script type="text/vbscript"> document.write("This is a script") </script>** <title>VBScript script in the head section</title> </head> <body> </body> </html>

A script placed in the body section of a webpage

A script placed in the body section of a webpage will be executed when the page loads. A script placed in the body section of a webpage generates the content of the page.

Example:
<html> <head> <title>VBScript script in the body section</title> </head> <body> **<script type="text/vbscript"> document.write("This is a script") </script>** </body> </html>

You can have as many scripts on a webpage as you want, this includes both scripts in the head section and the body section of a webpage.

Script in both head and body:
<html> <head> **<script type="text/vbscript"> document.write("This is a script") </script>** <title>VBScript script in the head and body sections</title> </head> <body> **<script type="text/vbscript"> document.write("This is a script") </script>** </body> </html>

Printing text on a webpage

Printing text on a webpage is accomplished with the document.write() command.

Syntax:
document.write("*textToPrint*")

Example:
<html> <head> <title>Printing text</title> </head> <body> <script type="text/vbscript"> **document.write("Here is some text")** </script> </body> </html>

Output:
Here is some text

Including HTML tags in a script

HTML tags can be included in a script using the document.write() command mentioned above. Any tags included in a script through this command will be interpreted by the web browser as regular HTML.

Example:
<html> <head> <title>HTML tags in scripts</title> </head> <body> <script type="text/vbscript"> **document.write("<b><i>Here is some bold italic text</i></b>")** </script> </body> </html>

Output:
Here is some bold italic text

Including comments in a script

Comments in VBScript are declared so that code would be easier to understand and navigate. Comments are not seen on a webpage, but only within the source code of a webpage. Comments can be placed anywhere within VBScript source code. In VBScript you can have only single line comments, no multi-line comments.

Single line comments in VBScript are declared with the apostrophe (') symbol.

Example:
<html> <head> <title>Comments</title> </head> <body> <script type="text/vbscript"> **'this is a single line comment 'this is another single line comment** document.write("Here is some text") </script> </body> </html>

Dealing with browsers that do not support scripts

There are older browsers still in use that do not recognize the <script> tag and consequently will not be able to execute scripts. In such a case, the content inside the <script> tag will be displayed on the page as regular text. To prevent this from happening, the content of a script can be placed within HTML comment tags. In such a case, older browsers that do not recognize the <script> tag will ignore the script and the content inside the <script> tag will not be displayed on the page. Browsers that can execute scripts will ignore the comments and execute the script anyway.

Example:

```
<html> <head> <title>Dealing with browsers that do not support scripts</title> </head> <body> <script type="text/vbscript"> <!-- document.write("Here is some text") --> </script> </body> </html>
```

Output:

```
Here is some text
```

Handling browsers that do not support VBScript

There may be a situation where a user is using a browser that supports scripting, but not VBScript in particular. In such a situation, you can use HTML's <noscript> tag.

Example:

```
<html> <head> <title></title> </head> <body> <script type="text/vbscript"> <!-- document.write("Your web browser supports VBScript!") --> </script> <nsocript> Your web browser does not support VBScript! </noscript> </body> </html>
```

In this example, if VBScript is supported then the script will execute and the message "Your web browser supports VBScript!" will be printed onto the page. If VBScript is not supported then the script will not be executed and the message found in the <noscript> tag which states "Your web browser does not support VBScript!" will be printed onto the page.

This tutorial focuses on:

- What is a variable?
- Declaring variables
- Option Explicit
- Naming variables
- Printing variables

What is a variable?

A variable is a container which stores information in a computer's memory. The value of a variable can change all throughout a script.

Declaring variables

In VBScript a variable is declared with the **Dim** statement.

Syntax:

```
Dim variableName
```

Example:

```
<script type="text/vbscript"> Dim title </script>
```

Once you declare a variable, you can give it a value.

Example:

```
<script type="text/vbscript"> Dim title title = "A book to read" </script>
```

NOTE: The **Dim** statement should be used only once with each variable - during declaration. After a variable has been declared, you can refer to it by its name without the **Dim** statement.

Option explicit

It is possible that within a script you will misspell a variable name and consequently get unexpected results. For example, if you misspell the variable name "title" as "titld", VBScript will automatically generate a variable with the name "titld". You can prevent this from happening by using the **option explicit** statement. This statement goes all the way at the top of a script.

Example:
<script type="text/vbscript"> **option explicit** Dim title title = "A book to read" </script>

Naming variables

When naming variables, several rules should be followed:

- **Make sure that the variable name is descriptive** - If you do not give a variable a descriptive name, it will be hard to understand what the variable refers to. For example, if you wanted to create a variable which would hold a value signifying the amount of chairs in a room, which variable name would be more appropriate - numChairs or a? The better choice for the variable name would be numChairs because it is more descriptive
- **Make sure the variable name is of appropriate length.** - Make sure the variable name is long enough to be descriptive, but not too long
- **Do not use spaces in variable names** - This is a rule that must be followed in VBScript because VBScript does not allow spaces in variable names
- **Do not use periods (.) in variable names** - Variable names cannot contain periods
- **Do not use special symbols in variable names such as !@#%^&*** - As is the rules with spaces, special symbols are not allowed in variable names. Using special symbols in variable names will generate an error. There is however one special symbol that can be used in variable names, and that symbol is the underscore (_) symbol. Variable names can only contain letters, numbers, or the underscore symbol.
- **Do not exceed 255 characters** - Variable names cannot exceed 255 characters.

Printing variables

Variables are printed by including the variable name in a document.write() command. When printing the value of a variable, the variable name should NOT be included in double quotes.

Example:
<html> <head> <title>Printing variables</title> </head> <body> <script type="text/vbscript"> Dim numChairs numChairs = 5 **document.write(numChairs)** </script> </body> </html>

Output:
```
5
```

You can also print variables together with regular text. To do this, use the **&** symbol to join the text and variable values together.

Example:
```
<html> <head> <title>Printing variables and text together</title> </head> <body> <script type="text/vbscript"> Dim teamName, teamState, numWins, numLosses teamName = "The Penguins" teamState = "Connecticut" numWins = 12 numLosses = 2 document.write("The name of the team is " & teamName) document.write("<br />The team is from " & teamState) document.write("<br />" & teamName & " have won " & numWins & " games and have lost " & numLosses & " games") </script> </body> </html>
```

Output:
```
The name of the team is The Penguins The team is from Connecticut The Penguins have won 12 games and have lost 2 games
```

This tutorial focuses on:

- **Creating procedures**
- **Using parameters**
- **Calling procedures**
- **Returning values**
- **VBScript's built-in procedures**

Creating procedures

There are two types of procedures in VBScript - Sub procedures and Function procedures.

Sub procedures

A Sub procedure can perform some actions, but CANNOT return values.

A Sub procedure is enclosed in **Sub** and **End Sub**.

Syntax:

Sub nameOfSubProcedure() *actions to perform* End Sub

Example:
```
Sub printText() document.write("Here is some text"); End Sub
```

Function procedures

A Function procedure can perform some actions, and can return values.

A Function procedure is enclosed in **Function** and **End Function**.

Syntax:

Function nameOfFunction() *actions to perform* End Function

Example:

Function printText() document.write("Here is some text"); End Function

Using parameters

Parameters are variables placed inside the parentheses of a sub procedure or a Function procedure which are used by the code inside the sub procedure or Function procedure in some way. Parameters are separated by commas.

Using parameters with a Sub procedure

Syntax:
Sub nameOfSubProcedure(*param1, param2, etc.*) actions to perform End Sub

Example:
Sub printName(name) document.write("Your name is " & name); End Sub

The above procedure takes a value supplied by its name parameter and prints a message on a webpage.

Using parameters with a Function procedure

Syntax:
Function nameOfSubProcedure(*param1, param2, etc.*) actions to perform End Function

Example:
Function printName(name) document.write("Your name is " & name); End Function

The above Function procedure takes a value supplied by its name parameter and prints a message on a webpage.

Calling procedures

Once you create a procedure, how do you actually use it? You have to call that procedure.

Calling a Sub procedure

To call a Sub procedure, refer to the name of the Sub procedure with the **Call** statement, and pass the appropriate parameter values to it (if there are any).

Syntax:
Call subProcedureToCall()

Example:
Sub printMessage() document.write("VBScript is a scripting language.") End Sub Call printMessage()

Output:
VBScript is a scripting language.

Calling a Function procedure

To call a Function procedure, refer to the name of the Function procedure with the **Call** statement, and pass the appropriate parameter values to it (if there are any).

Syntax:
Call FunctionProcedureToCall()

Example:
Function printMessage() document.write("VBScript is a scripting language.") End Function Call printMessage()

Output:
VBScript is a scripting language.

Returning values

Function procedures can be used to return values. By doing so, the Function procedure simply acts as a value after its code has been executed. This way, a variable can take the value of a Function procedure.

Syntax for returning a value from a Function procedure:

Function aFunction() *aFunction* = *value* End Function

Example:

```
Function greet() greet = "user" End Function document.write("Hello "&greet())
```

Output:

```
Hello user
```

VBScript's built-in procedures

VBScript has several built-in procedures you can use for various purposes.

- **abs()** - will return the absolute value of a number
- **Len()** - will return the length of a text string
- **IsNumeric** - will return a boolean (true/false) value indicating whether a supplied value is a number or not
- **Date()** - will return the current system date

VBScript's built-in procedures example:

```
<script type="text/vbscript"> Dim a a = "five" document.write(abs(-5)) document.write("<br />" & Len("Hello")) document.write("<br />" & IsNumeric(a)) document.write("<br />" & "Today's date is: "&Date()) </script>
```

Output:

```
5 5 False Today's date is: 1/1/2010
```

This tutorial focuses on:

- **Displaying an alert box**
- **Displaying a prompt box**

Displaying an alert box

An alert box is a box that appears with a message inside it and a title. The user will have to press the "OK" button or any other potential buttons that appear on the alert box to close it.

An alert box can be displayed using VBScript's **MsgBox** function together with the text to be displayed in the alert box, a title, and an integer indicating what kind of button or buttons should appear on the message box (0 for just an "OK" button, 1 for an "OK" and "Cancel" buttons).

Syntax:

MsgBox "*text*", *num*, "*title*"

Example:

<script type="text/vbscript"> <!-- Function aButton_OnClick() **MsgBox "Message boxes are cool!", "0", "Hi, I am a message box"** End Function --> </script> <input type="button" value="Click here for a message" name="aButton" />

Output:

Displaying a prompt box

A prompt box is used to get data from the user. A prompt box will appear with an "OK" button and a "Cancel" button. Different actions will occur depending on what button the user clicks. If the user clicks the "OK" button, the value entered into the prompt box will be set. If the user clicks the "Cancel" button, an empty string will be set.

A prompt box can be displayed using VBScript's **InputBox** function.

Syntax:
InputBox ("*MessageToShow*", "*TitleOfPromptBox*")

Example:
<script type="text/vbscript"> **'display a prompt box asking the visitor for their favorite color** function promptButton_OnClick dim favColor **favColor = InputBox("What is your favorite color?", "Favorite color")** if favColor = "" then MsgBox("You did not specify your favorite color!") else MsgBox("Your favorite color is " & favColor) end if end function </script> <input type="button" value="Click here to specify your favorite color" name="promptButton" />

Output:

This tutorial focuses on:

- The if statement
- The else statement
- The elseif statement
- Using if, else, and elseif together
- The Select statement

The if statement

The if statement tests if a certain condition is true or false and acts upon it accordingly.

Syntax:
if condition then *perform this action* end if

If the specified condition is true, then the code following the condition will be executed, otherwise it will not.

Example:
<script type="text/vbscript"> Dim aNumber aNumber= 5 **if aNumber = 5 then document.write("aNumber is equal to 5") end if** </script>

Output:
aNumber is equal to 5

The else statement

If the condition in an if statement is true, then the code following the condition will be executed. But what if you wanted something to happen if it is false? What if you wanted one thing to happen if the condition is true, and something else to happen if the condition is false? This is where the else statement comes in. The else statement works together with the if statement and executes certain code if the condition in the if statement is false.

Syntax:
if condition then *perform this action* else *perform this action* end if

If the condition in the if statement is false, then the action dictated by the else statement will be performed.

Example 1:
<script type="text/vbscript"> Dim plant plant = "cactus" **if plant = "cactus" then document.write("Great choice!") else document.write("Still a good choice, though cactuses are better.") end if** </script>

Output:
Great Choice!

Example 2:
<script type="text/vbscript"> Dim plant = "birch" **if plant = "cactus" then document.write("Great choice!") else document.write("Still a good choice, though cactuses are better.") end if** </script>

Output:
Still a good choice, though cactuses are better.

The elseif statement

The if statement tests a single condition and performs an action if that condition is true and the else statement performs an action if the condition in the if statement is false, but what if there are more than two possibilities? Surely, any condition can be only true or false, but what if you needed to test a variable for more than one value? This is where the elseif statement comes in. The elseif statement is used in conjunction with the if statement. Unlike the else statement, it does not specifically perform a certain action if the condition in the if statement is false, but rather it performs an action if the condition in the if statement is another specific value specified in the elseif statement itself.

Syntax:
if condition is one value then *perform this action* elseif condition equals another value then *perform this action* end if

Example:
<script type="text/vbscript"> Dim X X = 7 **if X = 5 then document.write("X is equal to 5") elseif X = 7 then document.write("X is equal to 7") end if** </script>

Output:
X is equal to 7

Using if, else, and elseif together

You can use the if, else, and elseif statements together when you want to check a variable for a certain value many times. If it is not any of the checked values then the code specified by the else statement will be executed.

Syntax:

if condition then *perform this action* elseif condition equals another value then *perform this action* elseif condition equals another value then *perform this action* elseif condition equals another value *perform this action* else *if the condition was not equal to any of the values tested in the if statement and all the elseif statements, then perform this action* end if

Example:

<script type="text/vbscript"> Dim X X = 5 **if X = 9 then document.write("X is equal to 9") elseif X = 7 then document.write("X is equal to 7") elseif X = 3 then document.write("X is equal to 3") elseif X = 15 then document.write("X is equal to 15") else document.write("X is not equal to 9, 7, 3, or 15. X is equal to " & X); end if** </script>

Output:

X is not equal to 9, 7, 3, or 15. X is equal to 5

The Select statement

The Select statement is specifically designed for comparing one variable to a number of possible values. It can be thought of as a substitute for the if, elseif, else structure.

Syntax:

select case variable case possible value *perform this action* case possible value *perform this action* case possible value *perform this action* case possible value *perform this action* case else *perform this action if none of the values match* end select

Example:

<script type="text/vbscript"> Dim X X = 7 **select case X case 1 document.write("X is equal to 1") case 2 document.write("X is equal to 2") case 3 document.write("X is equal to 3") case 7 document.write("X is equal to 7") case else document.write("X is not equal to any of the values specified") end select** </script>

Output:

X is equal to 7

This tutorial focuses on:

- **The for next loop**
- **The do while loop**

The for next loop

The for next loop is used to repeat a task a specific number of times.

Syntax:

for aVariable = numStart to numEnd step aVariableIncrement *code to be executed* next

Example:

<script type="text/vbscript"> for a = 1 to 10 step 2 document.write(a & " ") next </script>

Output:

1 3 5 7 9

In the above example, the **for next** loop specifies that a loop should loop 10 times and increment the looping variable by 2 each time. Within the loop body the statement **document.write(a & " ")** will print the current value of the looping variable through each iteration together with a single space.

The do while loop

The do while loop works differently then the for loop. The for loop repeats a segment of code a specific number of times, while the do while loop repeats a segment of code an unknown number of times.

Syntax:
do while condition *code to be executed* loop

Example:
<script type="text/javascript"> Dim num num = 0 **do while num < 30 num = num + 5 document.write(num & "<br />") loop** </script>

Output:
5 10 15 20 25 30

In the above example, a variable named *num* is given the value 0. The condition in the while loop is that while *num* is less than 30, 5 should be added to **num**. Once the value of *num* is greater than 30, the loop will stop executing. The loop prints the current value of *num* followed by a line break through each iteration.

This tutorial focuses on:

- **Creating an array**
- **Adding values to an array**
- **Accessing an arrays elements**
- **Modifying an arrays elements**

Creating an array

An array is created the same way a regular variable is created, with the addition of the number of the elements in the array next to the array name in parenthesis.

Syntax:
Dim arrayName(numElements)

Example:
<script type="text/vbscript"> Dim colors(4) </script>

In the above example, an array named "colors" that can store 4 elements is created.

Adding values to an array

Once you create an array, you can add values to it by specifying the array name, an index in the array and a value to be placed in that index.

Syntax:
arrayName(index) = value

Example:

<script type="text/vbscript"> Dim colors(4) **colors(0) = "green" colors(1) = "blue" colors(2) = "gray" colors(3) = "orange"** </script>

NOTE: Array indexes start at 0. So the 1st element of an array would be at index 0, the 2nd element of an array would be at index 1, and so on.

Accessing an arrays elements

To access an arrays elements, refer to the array with the appropriate index number in brackets.

Example:

<script type="text/vbscript"> Dim colors(4) colors(0) = "green" colors(1) = "blue" colors(2) = "gray" colors(3) = "orange" 'print the last element of the colors array **document.write("The last element in the colors array is " & colors(3))** </script>

Output:

The last element in the colors array is orange

Modifying an arrays elements

To modify an arrays elements, refer to the array with the appropriate index number in brackets of the value you want to change and set it to the new value.

Example:

<script type="text/vbscript"> Dim colors(4) colors(0) = "green" colors(1) = "blue" colors(2) = "gray" colors(3) = "orange" document.write("The old value of colors(1): " & colors(1)) document.write("<br />The old value of colors(3): " & colors(3)) 'change the value of colors(1) and colors(3) **colors(1) = "teal" colors(3) = "violet"** document.write("<br />The new value of colors(1): " & colors(1)) document.write("<br />The new value of colors(3): " & colors(3)) </script>

Output:

The old value of colors(1): blue The old value of colors(3): orange The new value of colors(1): teal The new value of colors(3): violet

A String is a grouping of characters sorrounded by double quotes such as "this is a string". With VBScript, you can manipulate strings in several ways such as joining two or more strings, printing the length of a string, and more.

This tutorial focuses on:

- **Creating a String**
- **Printing Strings**
- **Joining Strings**
- **String functions**

Creating a String

A String can be created by sorrounding text in double quotes and assigning it to a variable.

Example:

```
<script type="text/vbscript"> Dim aString aString= "This is a string." </script>
```

Printing Strings

A String can be printed by referencing the variable that stores the String by name.

Example:

```
<script type="text/vbscript"> Dim aString aString= "This is a string." document.write(aString) </script>
```

Output:

```
This is a string.
```

Joining Strings

You can join strings using the "&" (ampersand) operator. By joining strings, you can combine several strings into one or print several strings together at the same time.

Example:

```
<script type="text/vbscript"> Dim stringOne, stringTwo, statement, color stringOne = "Here is " stringTwo = "some text" statement = "My favorite color is " color = "green" 'join the statement and color strings into one 'and add another string to this string which is 'the period character statement = statement & color & "." 'print stringOne and stringTwo together document.write(stringOne & stringTwo) 'print a line break and the value of the statement variable document.write("<br />" & statement) </script>
```

Output:

```
Here is some text My favorite color is green.
```

String functions

VBScript provides various functions used for working with strings.

- **len()** - returns the length() of a string
- **mid()** - returns a specified number of characters from a string. This function takes three parameters - the name of the string to extract characters from, the point in the string where to start extracting characters from, and the number of characters to extract
- **ltrim()** - removes spaces from the left side of a string
- **ucase()** - converts a string into all uppercase letters

Example:

```
<html> <head> <title>String functions</title> </head> <body> <script type="text/vbscript"> Dim aString, anotherString aString = "VBScript is a scripting language." anotherString = " There are five spaces at the beginning of this string." document.write("Length of the aString string: " & len(aString)) document.write("<br />The first five characters from the aString string:" & mid(aString,1,5)) document.write("<br />The anotherString string without spaces at the beginning of it: " & ltrim(anotherString)) document.write("<br />The anotherString in all uppercase letters: " & UCase(anotherString)) </script> </body> </html>
```

Output:

```
Length of the aString string: 33 The first five characters from the aString string:VBScr The anotherString string without spaces at the beginning of it: There are five spaces at the beginning of this string. The anotherString in all uppercase letters: THERE ARE FIVE SPACES AT THE BEGINNING OF THIS STRING.
```

VBScript provides the functionality to display date and time on a webpage.

NOTE: Whenever you display the date or time on a webpage using VBScript, it will get that data from whatever the user has on their computer, whether it is correct or not. So if the current date and time is January 15th 12:30, but the users computer is set to January 10th 8:30, that is what will be displayed.

NOTE: To see the output for the examples on this page look at the page in Internet Explorer or on Firefox using the **IETab extension**.

This tutorial focuses on:

VBScript date functions

VBScript provides several functions for displaying a date on a webpage.

The Date function

The Date function is used to return the current date.

Example:

```
<script type="text/vbscript"> document.write("The current date is " & Date) </script>
```

Output:

```
The current date is 4/10/2012
```

The Day() function

Syntax:

Day(*date*)

The **date** parameter is any variable or function that represents a date. This function will return the day of the month (in numerical form from 1 to 31) from the date specified by the *date* parameter.

Example:

```
<script type="text/vbscript"> document.write("Current date: " & Date & "<br />") 'get the current day of the month from the current date document.write("<br />The current day of the month is " & Day(Date))
</script>
```

Output:

```
Current date: 4/10/2012
The current day of the month is 10
```

The Month() function

Syntax:

Month(*date*)

The *date* parameter is any variable or function that represents a date. This function will return the current month (in numerical form from 1 to 12) from the date specified by the *date* parameter.

Example:
<script type="text/vbscript"> document.write("Current date: " & Date & "<br />") **'get the current month from the current date** document.write("The current month is " & **Month(Date)**) </script>

Output:
Current date: 4/10/2012
The current month is 4

The Year() function

Syntax:
Year(*date*)

The *date* parameter is any variable or function that represents a date. This function will return the current year (in numerical form (4 digits)) from the date specified by the *date* parameter.

Example:
<script type="text/vbscript"> document.write("Current date: " & Date & "<br />") **'get the current year from the current date** document.write("The current year is " & **Year(Date)**) </script>

Output:
Current date: 4/10/2012
The current year is 2012

The IsDate() function

Syntax:
IsDate(date)

The *date* parameter is any expression or function that can be checked if it is a date or not. This function will return a boolean value (true/false) signifying if *date* is a date or not.

Example:
<script type="text/vbscript"> document.write(**IsDate(Date)** & "<br />") document.write(**IsDate("Apple")** & "<br />") document.write(**IsDate("January 1, 1900")** & "<br />") document.write(**IsDate("January first nineteen hundred")**) </script>

Output:
True
False
True
False

VBScript time functions

VBScript provides several functions for displaying time on a webpage.

The Time function

The Time function is used to return the current time.

Example:
<script type="text/vbscript"> document.write("The time is now " & **Time**) </script>

Output:
The time is now 10:27:44 AM

The Minute() function

Syntax:
Minute(*time*)

The *time* parameter is any variable or function that represents time. This function will return the current minute (from 0 to 59) from the time specified by the *time* parameter.

Example:

```
<script type="text/vbscript"> document.write("Current time: " & Time & "<br />") 'get the current minute
from the time specified document.write("The current minute is " & Minute(Time)) </script>
```

Output:

```
Current time: 10:27:44 AM
The current minute is 27
```

The Hour() function

Syntax:
Hour(time)

The *time* parameter is any variable or function that represents time. This function will return the current hour (from 0 to 23) from the time specified by the *time* parameter.

Example:

```
<script type="text/vbscript"> document.write("Current time: " & Time & "<br />") 'get the current hour
from the time specified document.write("The current hour is " & Hour(Time)) </script>
```

Output:

```
Current time: 10:27:44 AM
The current hour is 10
```

The Second() function

Syntax:
Second(*time*)

The *time* parameter is any variable or function that represents time. This function will return the current second (from o to 59) of the current minute from the time specified.

Example:

```
<script type="text/vbscript"> <!-- document.write("Current time: " & Time & "<br />") 'get the current
second from the current minute from the time specified document.write("The current second is " &
Second(Time)) --> </script>
```

Output:

```
Current time: 10:27:44 AM
The current second is 44
```

The Timer function

The Timer function is used to return the number of seconds that have elapsed since midnight (12 a.m.) for the current day.

Example:

```
<script type="text/vbscript"> document.write("Current time: " & Time & "<br />")
document.write("Number of seconds elapsed in the current day: " & Timer) </script>
```

Output:

```
Current time: 10:27:44 AM
Number of seconds elapsed in the current day: 37664.92
```

Displaying a date on a webpage

You can display a date on a webpage using the above mentioned date functions:

Example:

```
<script type="text/vbscript"> document.write("Current date: " & Date & "<br />") document.write("Current month: " & Month(Date) & "<br />") document.write("Current day: " & Day(Date) & "<br />") document.write("Current year: " & Year(Date) & "<br />") document.write("'January first eighteen hundred' a valid date format: " & IsDate("January first eighteen hundred")) </script>
```

Output:

```
Current date: 4/10/2012
Current month: 4
Current day: 10
Current year: 2012
'January first eighteen hundred' a valid date format: False
```

Displaying time on a webpage

You can display time on a webpage using the above mentioned time functions:

Example:

```
<script type="text/vbscript"> document.write("Current time: " & Time & "<br />") document.write("Current hour: " & Hour(Time) & "<br />") document.write("Current minute: " & Minute(Time) & "<br />") document.write("Current second: " & Second(Time) & "<br />") document.write("Number of seconds elapsed since current day began: " & Timer) </script>
```

Output:

```
Current time: 10:27:44 AM
Current hour: 10
Current minute: 27
Current second: 44
Number of seconds elapsed since current day began: 37664.94
```

Displaying date and time on a webpage

You can display the date and time together on a webpage using the **Now** function:

Example:

```
<script type="text/vbscript"> document.write("Date and time: " & Now) </script>
```

Output:

```
Date and time: 4/10/2012 10:27:44 AM
```

Displaying custom formatted date and time

VBScript displays date and time in one format by default but you can change this using the **FormatDateTime()** function.

Syntax:

FormatDateTime(*dateOrTime*, *format*)

The *dateOrTime* parameter is the date or time you want to format, and the *format* parameter is the type of formatting to apply.

The *format* parameter can take a value from 0 - 4, each value signifying a different type of formatting:

0 (used for dates) - display a date in the format mm/dd/yy (default)

1 (used for dates) - display a date in the format day, month and month day, year

2 (used for dates) - display a date in the format mm/dd/yy (default (same as 0))

3 (used for time) - display time in the format hh:mm:ss: PM/AM (12 hour format)

4 (used for time) - display time in the format hh:mm (24 hour format)

Example:

```
<script type="text/vbscript"> document.write("Current date: " & FormatDateTime(Date, 0) & "<br />")
document.write("Current date: " & FormatDateTime(Date, 1) & "<br />") document.write("Current date: " &
FormatDateTime(Date, 2) & "<br />") document.write("Current time: " & FormatDateTime(Time, 3) & "<br
/>") document.write("Current time: " & FormatDateTime(Time, 4)) </script>
```

Output:

```
Current date: 4/10/2012
Current date: Tuesday, April 10, 2012
Current date: 4/10/2012
Current time: 10:27:44 AM
Current time: 10:27
```

VBScript is a scripting language used in Microsoft's Internet Explorer web browser and browsers based on Internet Explorer's engine such as Flashpeak's **SlimBrowser**.

VBScript is short for Visual Basic Scripting Edition. VBScript is a lighter version of the Visual Basic programming language, and like Visual Basic, VBScript was developed by Microsoft.

NOTE: Remember that VBScript code only works in the above mentioned browsers. If you are using Firefox, download the **IETab extension** to be able to view the output of code written in VBScript inside Firefox.

The tutorials in this section included information on a variety of VBScript topics including printing text, working with variables, loop, strings, displaying date & time, and more