FREE eBook

LEARNING asp-classic

#asp-classic

Table of Contents

About

Chapter 1: Getting started with asp-classic

Remarks

Active Server Pages (ASP), also known as Classic ASP or ASP Classic, was Microsoft's first server-side script-engine for dynamically-generated web pages. The introduction of ASP.NET led to use of the term Classic ASP for the original technology.

The default server-side scripting language for ASP is VBScript. The generated pages are meant to be viewed in a browser, so they usually use HTML markup and CSS styling.

[1] ASP is not installed by default on these versions of IIS. You need to go into the server manager features and add ASP.
See Classic ASP Not Installed by Default on IIS 7.0 and above

Versions

IIS	ASP	Released
3.0	1.0	1996-12-01
4.0	2.0	1997-09-01
5.0	3.0	2000-11-01
6.0	3.0	2003-01-01
7.0	3.0[1]	2008-01-01
7.5	3.0[1]	2009-01-01
8.0	3.0[1]	2012-01-01

Examples

Hello World

```
<!doctype html>
<html>
  <head>
    <title>Example Page</title>
  </head>
  <body>
<%
  'This is where the ASP code begins
  'ASP will generate the HTML that is passed to the browser
  'A single quote denotes a comment, so these lines are not executed
  'Since this will be HTML, we included the html and body tags
  'for Classic ASP we use Response.Write() to output our text
  'like this

  Response.Write ("Hello world")

  'Now we will end the ASP block and close our body and html tags
%>
  </body>
</html>
```

When response is sent from the Server to the Browser the output will be like this:

```
<!doctype html>
<html>
  <head>
    <title>Example Page</title>
  </head>
  <body>
 Hello world
  </body>
</html>
```

Structure of a Simple ASP Page

```
<%@ Language="VBScript" CodePage = 65001 %>
<%
Option Explicit
Response.Charset = "UTF-8"
Response.CodePage = 65001
%>
<!doctype html>
<html>
  <head>
    <title>My First Classic ASP Page</title>
  </head>

  <body>
    <%="Hello World"%>
  </body>
</html>
```

This is a very basic example of a Classic ASP page that returns the phrase "Hello World" to the browser along with the rest of the standard HTML. The HTML portions are static, i.e. the server will send them to the browser as-is. The parts delimited by `<% %>` are what the server will actually process before sending it to the client.

Note that the `<%="stuff"%>` syntax is shorthand for `<%Response.Write "stuff"%>`.

Read Getting started with asp-classic online: https://riptutorial.com/asp-classic/topic/1094/getting-started-with-asp-classic

Chapter 2: Connecting to a database

Introduction

Classic ASP utilises a technology called ActiveX Data Objects when requiring access to external data sources. The ADODB Library provides three main objects for this purpose, `ADODB.Connection`, `ADODB.Command` and the `ADODB.Recordset`.

Examples

Populating a dropdown from the database

(Caveat emptor: there are many, many programmers who go into absolute conniptions if they meet code that uses recordsets instead of commands and stored procedures.)

```
<%
dim rs, sql
dim SelectedUser
SelectedUser = request.form("user")
if IsNumeric(SelectedUser) then
    SelectedUser = CLng(SelectedUser)
else
    SelectedUser = 0
end if
%>
...
<p>Select a user: <select name="user" size="1">
<%
sql = "SELECT id, displayname FROM users WHERE active = 1 ORDER BY displayname"
set rs = server.createobject("ADODB.Recordset")
rs.open sql,"[connection string stuff goes here]",1,2
do until rs.eof
    response.write "<option value='" & rs("id") & "'"
    if rs("id") = SelectedUser then response.write " selected"
    response.write ">" & rs("displayname") & "</option>" & vbCrLf
    rs.Movenext '<- VERY VERY IMPORTANT!
loop
rs.close
set rs = nothing
%>
</select></p>
...
```

Read Connecting to a database online: https://riptutorial.com/asp-classic/topic/4991/connecting-to-a-database

Chapter 3: Looping

Examples

For Loop

In classic ASP we can specify a for loop with the *for* keyword. With the for statement we need the *next* statement which will increment the counter.

```
For i = 0 To 10
    Response.Write("Index: " & i)
Next
```

The *step* keyword can be used to changed the how the *next* statement will modify the counter.

```
For i = 10 To 1 Step -1
    'VBS Comment
Next
```

To exit a for loop, use the *Exit For* statement

```
For i = 0 To 10
    Response.Write("Index: " & i)
    If i=7 Then Exit For 'Exit loop after we write index 7
Next
```

We can also use a `For...Each` loop to perform a loop through a series of defined elements in a collection. For instance:

```
Dim farm, animal
farm = New Array("Dog", "Cat", "Horse", "Cow")
Response.Write("Old MacDonald had a Farm, ")
For Each animal In farm
    Response.Write("and on that farm he had a " & animal & ".<br />")
Next
```

Do Loop

Do while is very similar to for loop however this generally is used if our loop repetitions is unknown.

Do While:

```
'Continues until i is greater than 10
Do While i <= 10
    i = i + 1
Loop

'Or we can write it so the first loop always executes unconditionally:
```

```
'Ends after our first loop as we failed this condition on our previous loop
Do
    i = i + 1
Loop While i <= 10
```

Do Until:

```
'Ends once i equates to 10
Do Until i = 10
    i = i + 1
Loop

'Or we can write it so the first loop always executes unconditionally:
'Ends after our first loop as we failed this condition on our previous loop
Do
    i = i + 1
Loop Until i=10
```

Exiting a Do loop is similar to a for loop but just using the *Exit Do* statement.

```
'Exits after i equates to 10
Do Until i = 10
    i = i + 1
    If i = 7 Then Exit Do
Loop
```

Read Looping online: https://riptutorial.com/asp-classic/topic/5663/looping

Chapter 4: Variables

Examples

Declaring

Creating variables in VBScript can be done by using the Dim, Public, or Private statement. It is best practice to put at the top of the script "Option Explicit" which forces you to explicitly define a variable.

You can declare one variable like this:

```
Option Explicit
Dim firstName
```

Or you can several variables like this:

```
Option Explicit
Dim firstName, middleName, lastName
```

If you do not have the option explicit statement, you can create variables like so:

```
firstName="Foo"
```

This is **NOT** recommended as strange results can occur during the run time phase of your script. This happens if a typo is made later when reusing the variable.

To create an array, simply declare it with how many elements in the parameter:

```
Option Explicit
Dim nameList(2)
```

This creates an array with three elements

To set array elements, simply use the variable with the index as parameter like so:

```
nameList(0) = "John"
```

VBScript also supports multi-dimensional arrays:

```
Option Explicit
Dim gridFactors(2, 4)
```

Variable Types

VBScript is a weakly typed language; variables are all of type variant, though they usually have an implied subtype denoting the data they hold.

This means that your variable, no matter what you call it, can hold any value:

```
Dim foo
foo = "Hello, World!"
foo = 123.45
foo = #01-Jan-2016 01:00:00#
foo = True
```

Note that the above is perfectly valid code, though mixing your variables like this is amazingly poor practice.

The string subtype is always assigned by using speech marks `" "`. Unlike JavaScript and other languages, the apostrophe does not provide the same functionality.

Numbers in VBScript can include any format of number, but do have a particular subtype based on their value and whether they contain a decimal point or not.

Dates use the `# #` specifiers. Be aware that formats for a numeric date style (e.g. 01/01/2016) retains an American date format, so `#05/06/2016#` is 6th May, not 5th June. This can be circumnavigated by using a `#dd-mmm-yyyy#` format, as in the example above.

Boolean variables contain `True` or `False` values.

As explained earlier, arrays are dimensioned using a set of parentheses to define the number of elements and ranks (rows and columns), for instance:

```
Dim myArray(3, 4)
```

All elements in arrays are of type variant, allowing every single element to be of any subtype. This is very important when you need to perform tasks such as reading data from a record set or other object. In these cases, data can be directly assigned to a variable, for instance, when being returned from a record set...

```
Dim myData
....
myData = rsMyRecordset.GetRows()
....
Response.Write(myData(3,2))
```

One final type that requires some explanation is the `Object` type. Objects are basically pointers to the memory location of the object itself. Object types must be `Set`...

```
Dim myObj
Set myObj = Server.CreateObject("ADODB.ecordSet")
```

Read Variables online: https://riptutorial.com/asp-classic/topic/3195/variables

Credits

S. No	Chapters	Contributors
1	Getting started with asp-classic	Community, David Starkey, Lankymart, Martha, RamenChef, SearchAndResQ
2	Connecting to a database	Lankymart, Martha
3	Looping	Jake, Paul
4	Variables	feetwet, Jake, John Odom, Lankymart, Paul