- The response object: ASP objects, Sending text with the response object and embedded quotes, Using variables, Other response;
- The request object;
- The application and server objects: Threads, Application variables and use, The server object, Limitation of application variables;
- The session object.

ASP has 7 built-in Objects:

1. **Request** : used to get information from a visitor.
2. **Response** : used to send output to the user from the server.
3. **Application**: A group of ASP files that work together to perform some purpose is called an application. The Application object is used to tie these files together.
4. **Session**: A Session object stores information about, or change settings for a user session.
5. **Server**: used to access properties and methods on the server.
6. **ObjectContext:** used to either commit or abort a transaction managed by Component Services like database transactions that have been initiated by a script contained in an ASP page
7. **Error** : displays information about errors in scripts.

These Objects have

a) **Collection** : An Object's collections constitute different sets of keys and value pairs related to that object.
b) **Properties** : An Object's properties can be set to specify the state of the object.
c) **Methods** : An Object's methods determines the things one can do with the ASP Object.

For Example :

OBJECT *: A Book.*
METHOD : *Read a book.*
PROPERTIES : *No. of pages.*
COLLECTION : *Each page(key) has a particular text(value)*

ASP Response Object

The ASP Response object is used to send output to the user from the server.

Examples

- Write text with ASP: How to write text with ASP.
- Format text with HTML tags in ASP: How to combine text and HTML tags with ASP.
- Redirect the user to a different URL: How to redirect the user to a different URL.
- Show a random link: How to create a random link.
- Controlling the buffer: How to control the buffer.
- Clear the buffer: How to clear the buffer.
- End a script in the middle of processing and return the result: How to end a script in the middle of processing.
- Set how many minutes a page will be cached in a browser before it expires: How to specify how many minutes a page will be cached in a browser before it expires.
- Set a date/time when a page cached in a browser will expire: How to specify a date/time a page cached in a browser will expire.
- Check if the user is still connected to the server: How to check if a user is disconnected from the server.
- Set the type of content: How to specify the type of content.
- Set the name of the character set: How to specify the name of the character set.

The syntax, collections, properties and methods of the ASP Response object are as follows:

Syntax

```
Response.collection|property|method
```

Collections

Collections	Description
Cookies	The Cookies collection sets the value of a cookie. If you attempt to set the value of a cookie that does not exist, it is created. If it already exists, the new value you set overwrites the old value already written to the client machine.

Properties

Properties	Description
Buffer	The Buffer property determines whether to buffer page output or not. If set to True, then output from the page is not sent to the client until the script on that page has been processed, or until the Response object Flush or End methods are called.
CacheControl	The CacheControl property determines whether proxy servers are able to cache the output generated by ASP or not. If your page's content is large and doesn't change often, you might want to allow proxy servers to cache the page. In this case set this property to Public. Otherwise set it to Private.
Charset	The Charset property appends the name of the character set to the Content-Type header in the Response object.
ContentType	The ContentType property specifies the HTTP content type for the response. If not specified, the default is "text/html".
Expires	The Expires property specifies the length of time (in minutes) that the client machine will cache the current page. If the user returns to the page before it expires, the cached version will be displayed.
ExpiresAbsolute	The ExpiresAbsolute property specifies a date and time on which a page cached on a browser will expire. If the user returns to the same page before that date and time, the user will view the cached version of the page. If no time is specified, the page expires at midnight on the date specified. If a date is not specified, the page expires at the given time on the day that the script is run.
IsClientConnected	The IsClientConnected property is a read-only property that indicates if the client has disconnected from the web server since the last use of the Response object's Write method.
Pics	The PICS (Platform for Internet Content Selection) property adds a value to the pics-label field of the response header.
Status	The Status property specifies the value of the status line that is returned to the client machine from the web server.

Methods

Methods	Description
AddHeader	The AddHeader method allows you to add your own HTTP header with a specified value. If you add an HTTP header with the same name as a previously added header, the second header will be sent in addition to the first; adding the second header does not overwrite the value of the first header with the same name. Also, once the header has been added to the HTTP response, it cannot be removed.
AppendToLog	The AppendToLog method adds a string to the end of the Web server log entry for the current page request. You can only add up to 80 characters at a time, but you can call this method multiple times.
BinaryWrite	The BinaryWrite method writes information directly to the output without any character conversion.
Clear	The Clear method erases any buffered HTML output. It does so without sending any of the buffered response to the client. Calling Clear will cause an error if Buffer property of the Response object is not set to True.

End	The End method stops the processing of the script in the current page and sends the already created content to the client. Any code present after the call to the End method is not processed.
Flush	The Flush method sends buffered output to the client immediately. Flush will cause a run-time error if Buffer property of the Response object is not set to True.
Redirect	The Redirect method redirects the user to a different URL.
Write	The Write method writes a specified string to the output.

```
Response.Clear();//erases any buffered HTML output
Response.Write(htmlString);
Response.Flush(); // send all buffered output to client
Response.End(); // response.end would work fine now.
```

Example:

```
Response.Write("Before flush<br>")
Response.Flush()


Response.Write("After flush, before clear<br>")
Response.Clear()


Response.Write("After clear, before end<br>")
Response.End

Response.Write("After end<br>")
```

Output:

Before flush
After clear, before end

ASP Request Object

- The Request object retrieves the values that the client browser passed to the server during an HTTP request.
- It is used to get information from the user.
- Using this object, you can dynamically create web pages and perform various server-side actions based on input from the user.

QueryString Collection Examples

- Send query information when a user clicks on a link: How to send query information to a page within a link, and retrieve that information on the destination page (which is, in this example, the same page).
- A QueryString collection in its simplest use: Use the QueryString collection to retrieve the values from a form (the form uses the get method - the information sent is visible to everybody).
- How to use information from forms: How to use the values retrieved from a form.
- More information from a form: What the QueryString collection contains if several input fields have equal names. It also shows how to use the Count keyword to count the "name" property.

Form Collection Examples

- A form collection in its simplest use: How the Form collection retrieves the values from a form (the form uses the post method - the information sent is invisible to others).
- How to use information from forms: How to use the values retrieved from a form.
- More information from a form: What the Form collection contains if several input fields have equal names. It also shows how to use the Count keyword to count the "name" property.
- A form with radio buttons: How to interact with the user through radio buttons.
- A form with checkboxes: How to interact with the user through checkboxes.

Other Examples

- Get the server variables: How to get the visitor's browser type, IP address, and more.
- Create a welcome cookie: How to create a Welcome Cookie.
- Find the total number of bytes the user sent: How to find the total number of bytes the user sent in the Request object.

- The syntax, collections, properties and methods of the ASP Request object are as follows:

Syntax

```
Request[.collection|property|method](variable)
```

Collections

Collections	Description
ClientCertificate	The ClientCertificate collection provides access to the certification fields from a request issued by the web browser. Commonly, it is used when the client is requesting secure pages through SSL connection. Before using this collection, you must configure your Web server to request client certificates.
Cookies	The Cookies collection allows you to retrieve the values of the cookies sent in an HTTP request.
Form	The Form collection allows you to retrieve the data input into an HTML form posted to the HTTP request body by a form using the POST method.
QueryString	The QueryString collection allows you to retrieve the values of the variables in the HTTP query string. For example, it parses the values sent by a form using the GET Method. The QueryString collection is less capable than the Form collection, since there is a limit to the amount of data that can be sent in the header of an HTTP request.
ServerVariables	The ServerVariables collection contains the values of predefined environment variables plus all of the HTTP header values sent from the client browser to the web server.

Properties

Properties	Description
TotalBytes	The TotalBytes property is a read-only property that specifies the total number of bytes sent by the client in the HTTP request body. It is important when preparing to read data from the request body using the BinaryRead method of the Request object.

Methods

Methods	Description
BinaryRead	The BinaryRead method retrieves the data sent to the server from the client as part of a POST request and stores it in a SafeArray. A SafeArray is a special variant array that contains, in addition to its items, the number of dimensions in the array and the upper bounds of the array.

ASP Application Object

An Active Server Page application is actually a collection of ASP files in a virtual directory and associated sub-directories. The Application object is used to control and manage all items that are available to all users of an Active Server application. The Application items can be variables needed for all users in the application, or they can be instantiated objects that provide special server-side functionality.

The Application object is initialized by IIS when the first .asp page from within the given virtual directory is requested. It remains in the server's memory until either the web service is stopped or the application is explicitly unloaded from the web server (using the Microsoft Management Console).

The syntax, collections, methods and events of the ASP Application object are as follows:

Syntax

```
Application.method
```

Collections

Collections	Description
Contents	The Contents collection contains all the items added to the application through the use of scripts (not through the use of the <OBJECT> tag).

StaticObjects	The StaticObjects collection contains all session-level objects added to the application through the use of the <OBJECT> tag. The collection can be used to retrieve the value of a specific property for an object, or to retrieve all properties for all static objects.

Methods

Methods	Description
Contents.Remove	The Contents.Remove method deletes the specified item from the Application object Contents collection.
Contents.RemoveAll	The Contents.RemoveAll method deletes all the items from the Application object Contents collection.
Lock	The Lock method locks the Application Object, preventing other users from modifying the same application-level variables at the same time. The individual user retains control of the Application object until the Application.UnLock method is declared. If the Unlock method is not called explicitly, IIS will unlock the locked Application object when the script ends or times out.
Unlock	The Unlock method releases control of the locked application variables. Once Unlock has been called, other clients can again alter the values of the variables in the Application object. If the Unlock method is not called explicitly, IIS will unlock the locked Application object when the script ends or times out.

Events

Events	Description
Application_OnStart	The Application_OnStart event occurs before the first new session is created (when the first client request is received).
Application_OnEnd	The Application_OnEnd event occurs when the ASP application is explicitly unloaded from the web server or when the web service on the web server is stopped.

ASP Session Object

The Session object stores information needed for a particular user's session on the web server. It is automatically created every time when an ASP page from the web site or web application is requested by a user who does not already have a session, and it remains available until the session expires.

The Session object is user specific. It can be used to store variables specific to a particular user and IIS will maintain these variables when the client moves across pages within your web site.

The Session object is based on using cookies, so if cookies are not permitted on the client browser (because of firewall issues, browser incompatibility, or desktop/network security concerns), the Session object is rendered useless.

Examples

- Set and return the LCID: Set or return an integer that specifies a location or region. Contents like date, time, and currency will be displayed according to that location or region.
- Return the SessionID: Return a unique id for each user. The id is generated by the server.
- A session's timeout: Set and return the timeout (in minutes) of a session.

The syntax, collections, properties, methods and events of the ASP Session object are as follows:

Syntax

```
Session.collection|property|method
```

Collections

Collections	Description
Contents	The Contents collection contains all the items added to the session through the use of scripts (not through the use of the <OBJECT> tag). The collection can be used to determine the value of a specific item, or to iterate over all the items in the session.

StaticObjects	The StaticObjects collection contains all of the objects with session-level scope created through the use of the <OBJECT> tag. The collection can be used to retrieve properties of a specific object in the collection, or to use a specific method of a given object in the collection.

Properties

Properties	Description
CodePage	The CodePage property defines the code page that will be used to display the page content in the browser. A code page is a character set containing all the alphanumeric characters and punctuation used by a specific locale.
LCID	The LCID property is used to set or get the locale identifier (LCID) of the page is sent to the browser. The local identifier is used to control how dates, times and currencies will be displayed according to the specific location or region.
SessionID	The SessionID is a read-only property that uniquely identifies each current user's session. Each session has a unique identifier that is generated by the server when the session is created and stored as a cookie on the client machine.
TimeOut	The Timeout property sets or returns the timeout period (in minutes) for the Session object in this application. If the user does not refresh or request a page within the timeout period, the session ends. Can be changed in individual pages as required.

Methods

Methods	Description
Abandon	The Abandon method is used to destroy all objects stored in a Session object and releases their resources. If the Abandon method is not explicitly called, the web server will maintain all session information until the session times out.
Contents.Remove	The Contents.Remove method removes the specified item from the Session object Contents collection.
Contents.RemoveAll	The Contents.RemoveAll method removes all the items from the Session object Contents collection.

Events

Events	Description
Session_OnStart	The Application_OnStart event occurs when a new session starts, before the page that the user requests is executed.
Session_OnEnd	The Session_OnEnd event occurs when the user's session is abandoned or times out.

ASP Server Object

The Server object provides access to properties and methods on the server. Much of functionality it provides is simply functionality the web server itself uses in the normal processing of client requests and server responses.

Examples

- When was a file last modified?: Check when a file was last modified.
- Open a text file for reading: Open "Textfile.txt" for reading.
- Homemade hit counter

The syntax, properties and methods of the ASP Server object are as follows:

Syntax

```
Server.property|method
```

Properties

Properties	Description

ScriptTimeout	The ScriptTimeout property sets or returns the maximum amount of time (in seconds) that the script in the page can run before it is terminated. If you do not set a value for this property, the default value is 90 seconds.

Methods

Methods	Description
CreateObjet	The CreateObject method creates an instance of the object (a component, application or a scripting object) and returns a reference to it. Once instantiated, this object's properties and methods can be used just as you can use the properties and methods of the built-in objects that come with ASP.
Execute	The Execute method executes another .asp page without leaving the current page. It stops the execution of the current page and transfers control to the page specified in the URL. After that page has finished execution, control passes back to the calling page.
GetLastError	The GetLastError method returns an ASPError object that describes the error condition that occurred.
HTMLEncode	The HTMLEncode method applies HTML encoding to a specified string. All non-legal HTML characters are converted to their equivalent HTML entity.
MapPath	The MapPath method maps a specified virtual or relative path to a physical path on the server.
Transfer	The Transfer method stops execution of the current page and sends all current state information to another page specified in the URL. Unlike the Execute method, execution does not resume in the page calling the Transfer method.
URLEncode	The URLEncode method applies URL encoding rules to a specified string. All characters that are not valid in an URL are converted to their equivalent HTML entity.

ASP ASPError Object

The ASPError object is a new in ASP 3.0, and is available in IIS5 and later. It provides a range of detailed information about the last error that occurred in script in an ASP page. The ASPError object is only available through the GetLastError method of the Server object. The syntax and properties of the ASPError object are as follows:

Syntax

```
ASPError.property
```

Properties

Properties	Description
ASPCode	The ASPCode property returns an error code generated by IIS.
Number	The Number property returns the standard COM error code.
Line	The Line property indicates the number of the line within the .asp file that generated the error.
Category	The Category property indicates the source of the error. E.g. ASP itself, the scripting language, or an object.
Column	The Column property indicates the column position within the .asp file that generated the error.
File	The File property returns the name of the .asp file that was being processed when the error occurred.
Source	The Source property returns the actual source code, if available, of the line where the error occurred.
Desciption	The Description property returns a short description of the error.

ASPDescription	The ASPDescription property returns a detailed description of the error if it is ASP-related.

Comparison between Request and Response

S.N	Attributes	Request	Response
1	Uses	Used to read the data submitted by the client (i.e. the data comes with the client request)	Used to send instructions to the client browser.
2	Process	from client (you) to server	from Server to client (you)
3	Collections	QueryString, Form	None
4	Properties	TotalBytes	Buffer, CacheControl, ContentType, Expires, ExpiresAbsolute, IsClientConnected, Status
5	Methods	None	AddHeader, AppendToLog, Clear, End, Flush, Redirect, Write
6	Example	**Example HTML form** <form method="get" action="simpleform.asp"> First Name: <input type="text" name="fname"> Last Name: <input type="text" name="lname"> <input type="submit" value="Submit"> </form> <body> Welcome <% response.write("Last Name: " & request.querystring("lname")) %> </body>	<% Response.write(“Hello World”) %>

Comparison between Cookies and Sessions

S.N	Attributes	Cookies	Session
1	Store sides	Cookies are stored in the user's browser	Sessions are stored in server
2	Store Lifetime	Longer lifetime	Shorter lifetime
3	Storage duration	A cookie can keep information in the user's browser until deleted by user or set as per the timer	User cant disable the session
4	Accessible	Application state allows you to store global objects that can be accessed by any client.	A session is available as long as the browser is opened.
5	Destroyed	It will not be destroyed even if you close the browser	It will be destroyed if you close the browser
6	Information type	Cookies can only store string.	Can store any object
7	Future use	we can save cookie for future reference	Session can’t be.

6	Example	<%
		Response.Cookies("firstname")="Alex"
		%>
			<%
			Session("username")="hcoe"
			Session.Timeout=5 // minutes
			%>

Comparison between Application and Sessions

S.N	Attributes	Application	Session
1	Information type	Can’t store user specific information.	used to store user specific information

2	Accessible lifetime	Application variables are accessible till the application ends.	Default lifetime of the session variable is20 mins
3	Access by	Application state allows you to store global objects that can be accessed by any client.	Sessions allows information to be stored in one page and accessed in another, and it supports any type of object, including your own custom data types.
4	Id maintaining	For application object the id is maintained for whole application.	The session object is used to maintain the session of each user. If one user enter in to the application then they get session id if he leaves from the application then the session id is deleted. If they again enter in to the application they get different session id.

Example of Application Objects

Store and Retrieve Application Variables

Application variables can be accessed and changed by any page in an application.

You can create Application variables in "Global.asa" like this:

```
<script language="vbscript" runat="server">
Sub Application_OnStart
     application("vartime")=""
     application("users")=1
End Sub
</script>
```

In the example above we have created two Application variables: "vartime" and "users".

You can access the value of an Application variable like this:

There are

```
<%
        Response.Write(Application("users"))
%>
```

active connections.

The Global.asa file

The Global.asa file is an optional file that can contain declarations of objects, variables, and methods that can be accessed by every page in an ASP application.

All valid browser scripts (JavaScript, VBScript, JScript, PerlScript, etc.) can be used within Global.asa.

The Global.asa file can contain only the following:

- Application events
- Session events
- <object> declarations
- TypeLibrary declarations
- the #include directive

Note: The Global.asa file must be stored in the root directory of the ASP application, and each application can only have one Global.asa file.

Events in Global.asa

In Global.asa you can tell the application and session objects what to do when the application/session starts and what to do when the application/session ends. The code for this is placed in event handlers. The Global.asa file can contain four types of events:

a) **Application_OnStart** - Occurs when the FIRST user calls the first page in an ASP application. This event occurs after the Web server is restarted or after the Global.asa file is edited. The "Session_OnStart" event occurs immediately after this event.
b) **Session_OnStart** - This event occurs EVERY time a NEW user requests his or her first page in the ASP application.
c) **Session_OnEnd** - This event occurs EVERY time a user ends a session. A user-session ends after a page has not been requested by the user for a specified time (by default this is 20 minutes).
d) **Application_OnEnd** - This event occurs after the LAST user has ended the session. Typically, this event occurs when a Web server stops. This procedure is used to clean up settings after the Application stops, like delete records or write information to text files.

*A **Global.asa** file could look something like this:*

```
<script language="vbscript" runat="server">

sub Application_OnStart
'some code
end sub

sub Application_OnEnd
'some code
end sub

sub Session_OnStart
'some code
end sub

sub Session_OnEnd
'some code
end sub

</script>
```

Note: Because we cannot use the ASP script delimiters (<% and %>) to insert scripts in the Global.asa file, we put subroutines inside an HTML <script> element.

<object> Declarations

It is possible to create objects with session or application scope in Global.asa by using the <object> tag.
Note: The <object> tag should be outside the <script> tag!

Syntax

```
<object runat="server" scope="scope" id="id" {progid="progID"|classid="classID"}>
....
</object>
```

Parameter	Description
scope	Sets the scope of the object (either Session or Application)
id	Specifies a unique id for the object
ProgID	An id associated with a class id. The format for ProgID is [Vendor.]Component[.Version] Either ProgID or ClassID must be specified.
ClassID	Specifies a unique id for a COM class object. Either ProgID or ClassID must be specified.

Examples

The first example creates an object of session scope named "MyAd" by using the ProgID parameter:

```
<object runat="server" scope="session" id="MyAd" progid="MSWC.AdRotator">
</object>
```

The second example creates an object of application scope named "MyConnection" by using the ClassID parameter:

```
<object runat="server" scope="application" id="MyConnection"
classid="Clsid:8AD3067A-B3FC-11CF-A560-00A0C9081C21">
</object>
```

The objects declared in the Global.asa file can be used by any script in the application:

```
GLOBAL.ASA:

<object runat="server" scope="session" id="MyAd" progid="MSWC.AdRotator">
</object>
```

You could reference the object "MyAd" from any page in the ASP application:

```
SOME .ASP FILE:
<%=MyAd.GetAdvertisement("/banners/adrot.txt")%>
```

TypeLibrary Declarations

A TypeLibrary is a container for the contents of a DLL file corresponding to a COM object. By including a call to the TypeLibrary in the Global.asa file, the constants of the COM object can be accessed, and errors can be better reported by the ASP code. If your Web application relies on COM objects that have declared data types in type libraries, you can declare the type libraries in Global.asa.

Syntax

<!--METADATA TYPE="TypeLib" file="*filename*" uuid="*id*" version="*number*" lcid="*localeid*" -->

Parameter	Description
file	Specifies an absolute path to a type library. Either the file parameter or the uuid parameter is required
uuid	Specifies a unique identifier for the type library. Either the file parameter or the uuid parameter is required
version	Optional. Used for selecting version. If the requested version is not found, then the most recent version is used
lcid	Optional. The locale identifier to be used for the type library

Error Values

The server can return one of the following error messages:

Error Code	Description
ASP 0222	Invalid type library specification
ASP 0223	Type library not found

ASP 0224	Type library cannot be loaded
ASP 0225	Type library cannot be wrapped

Note: METADATA tags can appear anywhere in the Global.asa file (both inside and outside <script> tags). However, it is recommended that METADATA tags appear near the top of the Global.asa file.

Restrictions

Restrictions on what you can include in the Global.asa file:

- You cannot display text written in the Global.asa file. This file can't display information
- You can only use Server and Application objects in the Application_OnStart and Application_OnEnd subroutines. In the Session_OnEnd subroutine, you can use Server, Application, and Session objects. In the Session_OnStart subroutine you can use any built-in object

How to use the Subroutines

Global.asa is often used to initialize variables.
The example below shows how to detect the exact time a visitor first arrives on a Web site. The time is stored in a Session variable named "started", and the value of the "started" variable can be accessed from any ASP page in the application:

```
<script language="vbscript" runat="server">
     sub Session_OnStart
         Session("started")=now()
     end sub
</script>
```

Global.asa can also be used to control page access.

The example below shows how to redirect every new visitor to another page, in this case to a page called "newpage.asp":

```
<script language="vbscript" runat="server">
     sub Session_OnStart
         Response.Redirect("newpage.asp")
     end sub
</script>
```

And you can include functions in the Global.asa file.
In the example below the Application_OnStart subroutine occurs when the Web server starts. Then the Application_OnStart subroutine calls another subroutine named "getcustomers". The "getcustomers" subroutine opens a database and retrieves a record set from the "customers" table. The record set is assigned to an array, where it can be accessed from any ASP page without querying the database:

```
<script language="vbscript" runat="server">

sub Application_OnStart
          getcustomers
end sub

sub getcustomers
     set conn=Server.CreateObject("ADODB.Connection")
     conn.Provider="Microsoft.Jet.OLEDB.4.0"
     conn.Open "c:/webdata/northwind.mdb"
      set rs=conn.execute("select name from customers")
      Application("customers")=rs.GetRows
      rs.Close
      conn.Close
end sub

</script>
```

Global.asa Example

In this example we will create a Global.asa file that counts the number of current visitors.

- The **Application_OnStart** sets the Application variable "visitors" to 0 when the server starts
- The **Session_OnStart** subroutine adds one to the variable "visitors" every time a new visitor arrives
- The **Session_OnEnd** subroutine subtracts one from "visitors" each time this subroutine is triggered

The Global.asa file:

```
<script language="vbscript" runat="server">

Sub Application_OnStart
     Application("visitors")=0
End Sub

Sub Session_OnStart
     Application.Lock
     Application("visitors")=Application("visitors")+1
     Application.UnLock
End Sub

Sub Session_OnEnd
     Application.Lock
     Application("visitors")=Application("visitors")-1
     Application.UnLock
End Sub

</script>
```

To display the number of current visitors in an ASP file:

```
<!DOCTYPE html>
<html>
    <head> </head>
    <body>
          <p>There are <%response.write(Application("visitors"))%> online
now!</p>
     </body>
</html>
```

VBScript Functions

This page contains all the built-in VBScript functions. The page is divided into following sections:

- Date/Time functions
- Conversion functions
- Format functions
- Math functions
- Array functions

Date/Time Functions

Function	Description
CDate	Converts a valid date and time expression to the variant of subtype Date
Date	Returns the current system date
DateAdd	Returns a date to which a specified time interval has been added
DateDiff	Returns the number of intervals between two dates
DatePart	Returns the specified part of a given date
DateSerial	Returns the date for a specified year, month, and day
DateValue	Returns a date
Day	Returns a number that represents the day of the month (between 1 and 31, inclusive)
FormatDateTime	Returns an expression formatted as a date or time
Hour	Returns a number that represents the hour of the day (between 0 and 23, inclusive)
IsDate	Returns a Boolean value that indicates if the evaluated expression can be converted to a date
Minute	Returns a number that represents the minute of the hour (between 0 and 59, inclusive)
Month	Returns a number that represents the month of the year (between 1 and 12, inclusive)
MonthName	Returns the name of a specified month
Now	Returns the current system date and time
Second	Returns a number that represents the second of the minute (between 0 and 59, inclusive)
Time	Returns the current system time
Timer	Returns the number of seconds since 12:00 AM
TimeSerial	Returns the time for a specific hour, minute, and second
TimeValue	Returns a time
Weekday	Returns a number that represents the day of the week (between 1 and 7, inclusive)
WeekdayName	Returns the weekday name of a specified day of the week

Year	Returns a number that represents the year

Conversion Functions

Function	Description
Asc	Converts the first letter in a string to ANSI code
CBool	Converts an expression to a variant of subtype Boolean
CByte	Converts an expression to a variant of subtype Byte
CCur	Converts an expression to a variant of subtype Currency
CDate	Converts a valid date and time expression to the variant of subtype Date
CDbl	Converts an expression to a variant of subtype Double
Chr	Converts the specified ANSI code to a character
CInt	Converts an expression to a variant of subtype Integer
CLng	Converts an expression to a variant of subtype Long
CSng	Converts an expression to a variant of subtype Single
CStr	Converts an expression to a variant of subtype String
Hex	Returns the hexadecimal value of a specified number
Oct	Returns the octal value of a specified number

Format Functions

Function	Description
FormatCurrency	Returns an expression formatted as a currency value
FormatDateTime	Returns an expression formatted as a date or time
FormatNumber	Returns an expression formatted as a number
FormatPercent	Returns an expression formatted as a percentage

Math Functions

Function	Description
Abs	Returns the absolute value of a specified number

Atn	Returns the arctangent of a specified number
Cos	Returns the cosine of a specified number (angle)
Exp	Returns *e* raised to a power
Hex	Returns the hexadecimal value of a specified number
Int	Returns the integer part of a specified number
Fix	Returns the integer part of a specified number
Log	Returns the natural logarithm of a specified number
Oct	Returns the octal value of a specified number
Rnd	Returns a random number less than 1 but greater or equal to 0
Sgn	Returns an integer that indicates the sign of a specified number
Sin	Returns the sine of a specified number (angle)
Sqr	Returns the square root of a specified number
Tan	Returns the tangent of a specified number (angle)

Array Functions

Function	Description
Array	Returns a variant containing an array
Filter	Returns a zero-based array that contains a subset of a string array based on a filter criteria
IsArray	Returns a Boolean value that indicates whether a specified variable is an array
Join	Returns a string that consists of a number of substrings in an array
LBound	Returns the smallest subscript for the indicated dimension of an array
Split	Returns a zero-based, one-dimensional array that contains a specified number of substrings
UBound	Returns the largest subscript for the indicated dimension of an array

String Functions

Function	Description
InStr	Returns the position of the first occurrence of one string within another. The search begins at the first character of the string
InStrRev	Returns the position of the first occurrence of one string within another. The search begins at the last character of the string

LCase	Converts a specified string to lowercase
Left	Returns a specified number of characters from the left side of a string
Len	Returns the number of characters in a string
LTrim	Removes spaces on the left side of a string
RTrim	Removes spaces on the right side of a string
Trim	Removes spaces on both the left and the right side of a string
Mid	Returns a specified number of characters from a string
Replace	Replaces a specified part of a string with another string a specified number of times
Right	Returns a specified number of characters from the right side of a string
Space	Returns a string that consists of a specified number of spaces
StrComp	Compares two strings and returns a value that represents the result of the comparison
String	Returns a string that contains a repeating character of a specified length
StrReverse	Reverses a string
UCase	Converts a specified string to uppercase

Other Functions

Function	Description
CreateObject	Creates an object of a specified type
Eval	Evaluates an expression and returns the result
IsEmpty	Returns a Boolean value that indicates whether a specified variable has been initialized or not
IsNull	Returns a Boolean value that indicates whether a specified expression contains no valid data (Null)
IsNumeric	Returns a Boolean value that indicates whether a specified expression can be evaluated as a number
IsObject	Returns a Boolean value that indicates whether the specified expression is an automation object
RGB	Returns a number that represents an RGB color value
Round	Rounds a number
ScriptEngine	Returns the scripting language in use

ScriptEngineBuildVersion	Returns the build version number of the scripting engine in use
ScriptEngineMajorVersion	Returns the major version number of the scripting engine in use
ScriptEngineMinorVersion	Returns the minor version number of the scripting engine in use
TypeName	Returns the subtype of a specified variable
VarType	Returns a value that indicates the subtype of a specified variable

VBScript Keywords

Keyword	Description
Empty	Used to indicate an uninitialized variable value. A variable value is uninitialized when it is first created and no value is assigned to it, or when a variable value is explicitly set to empty. Example: Dim x 'the variable x is uninitialized! x="ff" 'the variable x is NOT uninitialized anymore x=Empty 'the variable x is uninitialized! **Note:** This is not the same as Null!!
IsEmpty	Used to test if a variable is uninitialized. Example: If (IsEmpty(x)) 'is x uninitialized?
Nothing	Used to indicate an uninitialized object value, or to disassociate an object variable from an object to release system resources. Example: Set myObject=Nothing
Is Nothing	Used to test if a value is an initialized object. Example: If (myObject Is Nothing) 'is it unset? **Note:** If you compare a value to Nothing, you will not get the right result! Example: If (myObject = Nothing) 'always false!
Null	Used to indicate that a variable contains no valid data. One way to think of Null is that someone has explicitly set the value to "invalid", unlike Empty where the value is "not set". **Note:** This is not the same as Empty or Nothing!! Example: x=Null 'x contains no valid data
IsNull	Used to test if a value contains invalid data. Example: if (IsNull(x)) 'is x invalid?
True	Used to indicate a Boolean condition that is correct
False	Used to indicate a Boolean condition that is not correct (False has a value of 0)