Server-Side Scripting
In
JavaScript/JScript And VBScript

John D. Haney
john.haney@nau.edu

and

Craig A. VanLengen
craig.vanlengen@nau.edu

College of Business Administration, Northern Arizona University
Flagstaff, AZ 86011-5066

ABSTRACT

When developing server-side scripting using Microsoft's Active Server Pages and their Internet Information Server (IIS), either VBScript or JScript are available. The language of choice for most developers is VBScript since it is closely akin to Visual Basic and Visual Basic for Applications. However, for those developers that are more familiar with Java and JavaScript, JScript is a comfortable alternative. The differences between VBScript and JScript lie primarily in the syntax and not in the functionality. The examples interact with an Oracle database: to connect to the database; create record sets; and adding, changing, and deleting records shows identical logic structure. Where the use of JScript rather than VBScript can become rather tedious is the scarcity of functions in JScript that are available in VBScript. The solution is to write comparable user-defined functions in JScript as demonstrated by the FormatCurrency function.

Keywords: IS Curriculum, programming languages, Web development

1. INTRODUCTION

When developing applications for the Web there are several scripting language choices including: JavaScript, JScript, VBScript or others that will not be discussed in this paper. This paper presents an example of maintaining a database with add, change, and delete options written in both JScript and VBScript using an Oracle database. Client-side and server-side scripting is discussed along with the scripting language features and factors to consider when developing an application on the Web.

2. DEVELOPMENT ENVIRONMENT

The differences in the scripting languages will be demonstrated by using examples that were developed in the following environment: Microsoft NT server, using Internet Information Server (IIS), and Active Server Pages (ASP). Oracle was the database used in the examples. However, Microsoft Access and SQL Server have also been used in similar examples. All of the application scripts are embedded within HTML documents that are stored on the server with the .asp extension.

3. SERVER-SIDE SCRIPTING VERSUS CLIENT-SIDE SCRIPTING

A script application is part of an HTML document. The difference in the tagging indicates whether the code executes on the server or on the client. Client-side script code is written using the <script> tag and executed on the client. Web server scripting in Microsoft Internet Information Server has the script code between the brackets and percent symbol (<%…..%>) and the code executes on the server. The HTML document has an .asp extension instead of .htm or .html.

The choice between client scripts or server scripting is based on the level of interaction needed with the database on the server. If interaction with a database is

not needed then the script could be on the client. This way the amount of network traffic is reduced. Server-side scripts must be used for database interaction, when accessing other programs that are on the server and whenever it is necessary to store data from user interaction. When security is an issue server-side scripting would be used to make the application code invisible on the client machine.

For client-side scripting the developer must be concerned about which browser the user has loaded. For client-side scripting you would want to use JavaScript or JScript to ensure that the application ran in either Internet Explorer or Netscape Communicator. A Netscape browser will ignore a client-side application developed in VBScript.

4. SERVER-SIDE SCRIPTING EXAMPLES USING VBSCRIPT AND JSCRIPT

Examples of adding, changing, and deleting of record occurrences within tables in an Oracle database are shown in both VBScript and JScript. First the embedded script that is necessary to connect and open a database is shown. Then the definition of an SQL statement and the opening of a recordset are given. And finally the code required to add, change or delete a record from a table within the database is listed.

Connecting to the Database

The example database is a catalog table that has the following fields: an id, description, price, quantity on-hand, and date purchased. The first step in interacting with the catalog table in the Oracle database is to create a connection to the Oracle database, and then to open the domain of the database that the catalog table is within.

VBScript: First an instance of the connection object is created with the Set statement. In this example **conn** is a user defined variable. Then the open method is used with the connection object to open the specific domain of the Oracle database. In this example cis440 is the name of the data source name (DSN) that points to the Oracle database, and **userid** and **pw** are the user id and password of the specific domain of the database.

```
Set conn = Server.CreateObject("ADODB.Connection")
conn.Open "cis440", "userid", "pw"
```

Jscript: In JScript the syntax varies only slightly but the effect is the same. The Set verb is not used in JScript but the rest of the statement of the creation of the connection object is the same. In JScript the syntax of the open statement uses parenthesis to surround the arguments.

```
conn = Server.CreateObject("ADODB.Connection");
conn.Open("cis440", "userid", "pw");
```

So, connecting to and opening the database in VBScript or JScript is very similar syntactically. A Set is used in VBScript on the connection, and parentheses are used in JScript on the open.

Opening a Recordset

Next a recordset must be opened. A recordset is a collection of records selected from one or more tables from the database. In this example only the Catalog table is used. First a variable is filled with the appropriate SQL statement that will select from the table. Then an instance of the recordset object is created, and then the recordset is opened. After the open the recordset will contain the selected records from the table.

VBScript: In the following example **sql** is a string variable that will contain the SQL statement, and **RS** is a variable that will contain an instance of the recordset object. In this example, once the Open method is executed the recordset will contain all of the records from the Catalog table. The 3 on the Open method defines a static open. This means that a snapshot is taken of the database table and any changes made to the table would not be reflected in the recordset. The 2 on the Open method defines a pessimistic lock where the record is locked until an update method occurs.

```
sql = "Select * From Catalog"
Set RS = Server.CreateObject("ADODB.Recordset")
RS.Open sql, conn, 3, 2
```

Jscript: The comparison between VBScript and JScript again depicts very similar syntax. In JScript a Set verb is not used on the CreateObject statement and parenthesis are used to surround the arguments on the Open statement.

```
sql = "Select * From Catalog"
RS = Server.CreateObject("ADODB.Recordset");
RS.Open (sql, conn, 3, 2);
```

Adding a record to a table within the database

VBScript: Now to the heart of the interaction with the database where records will be added, changed, or deleted. In the following example a record will be added to the Catalog table by using the recordset. First a determination will be made as to whether the record already exists. If the record exists a redirection is made to a Web page that displays a message that the record is already on file. If the record does not exist then the record is added by using the **AddNew** and **Update** methods.

First an SQL statement is created that compares the Cat_ID field, which is the primary key of the Catalog table, to a field that is supplied from a form on a Web page that is then posted to the Web page that contains

the following script. The Request.Form is an active server page object that accesses the fields posted to this Web page. In this example the value is placed into a variable that in turn is referenced in the SQL statement.

```
fCat_ID = Request.Form(“Cat_ID”)
sql = "Select * From Catalog Where Cat_ID = " & """ &
fCat_ID & """
```

Next the recordset object is created and then opened.

```
Set RS = Server.CreateObject(“ADODB.Recordset”)
RS.Open sql, conn, 3, 2
```

At this point the recordset is tested to determine whether it contains a record or not. If it contains a record then a redirection is made to a Web page, OnFile.asp that informs the user that a record cannot be added because it already exists:

```
If Not RS.Eof Then
    Response.Redirect “OnFile.asp”
EndIf
```

If the logic falls through here then a record with that Catalog ID does not exist so it is safe to add a record.

The AddNew method opens a buffer for placing values into the fields of the recordset. Again, the Request.Form object is used to access the fields posted to this Web page. The values of the form fields are placed into the recordset fields. When all of the recordset fields are filled the Update method writes from the buffer area to the table in the database.

```
RS.AddNew
fPrice = Request.Form("Price")
RS("Cat_ID") = Request.Form(“Cat_ID”)
RS("Description") =Request.Form("Description")
RS("Price") = 0
RS("Qty_OnHand") =Request.Form("Quantity")
RS("Pur_Date") = Request.Form("PDate")
RS.Update
RS.Close
```

A zero was placed into the Price field. The reason for this is when placing a value from a form using the Request.Form object into a real number in an Oracle database the decimal positions are stripped away. To work around this problem the following SQL Update statement is used. The SQL statement is built and then the Execute method updates the database.

```
sql = "Update Catalog Set Price = " & fPrice & " Where
Cat_ID = " & """ & fCat_ID & """
conn.Execute(sql)
```

Jscript: Adding a record to a table using JScript is very similar, except for the syntax of the language.

```
fCat_ID = Request.Form(“Cat_ID”);
sql = "Select * From Catalog Where Cat_ID = "  + """ +
fCat_ID + """;
RS = Server.CreateObject("ADODB.Recordset");
RS.Open (sql, conn, 3, 2);
if (!RS.eof)
      {
      NewPage = "OnFile.asp";
      Response.Redirect (NewPage);
      }
RS.AddNew();
fPrice = Request.Form("Price");
RS("Cat_ID") = Request.Form(“Cat_ID”);
RS("Description") = Request.Form("Description");
RS("Price") = 0;
RS("Qty_OnHand") = Request.Form("Quantity");
RS("Pur_Date") = Request.Form("PDate");
RS.Update();
sql = "Update Catalog Set Price = " + fPrice + " Where
Cat_ID = " + """ + fCat_ID + """;
conn.Execute(sql);
```

Changing a record in a table within the database:

VBScript: In the following example a record will be changed in the Catalog table by using the recordset. First a determination will be made as to whether the record exists or not. If the record does not exist then a redirection is made to a Web page that displays a message that the record is not on file. If the record does exist then the record is changed by using the Update method.

First an SQL statement is created that compares the Cat_ID field in the table of the database to a field that is supplied from a form on a Web page that is posted to this Web page with scripting. Note that the following script is almost identical to the Add routine. The notable difference is that on a change an AddNew method is not used, otherwise it is identical.

```
fCat_ID = Request.Form(“Stuff_ID”)
sql  = "Select * From Catalog Where Cat_ID =" & """ &
fCat_ID") & """
Set RS = Server.CreateObject("ADODB.Recordset")
RS.Open sql, conn, 3, 2
fPrice = Request.Form("Price")
RS("Description") =Request.Form("Description")
RS("Price") = 0
RS("Qty_OnHand") =Request.Form("Quantity")
RS("Pur_Date") =Request.Form("PDate")
RS.Update
RS.Close
sql = "Update Stuff Set Price = " & fPrice & " Where
Cat_ID = " & """ & fCat_ID & """
conn.Execute(sql)
```

Jscript: As with VBScript, the only difference between the Add and the Change is that on a Change the AddNew method is not used.

```
sql = "Select * From Catalog Where (Cat_ID= '" +
fCat_ID + "')";
RS = Server.CreateObject("ADODB.Recordset");
RS.Open (sql, conn, 3, 2);
FPrice = Request.Form("Price");
RS("Description") = Request.Form("Description");
RS("Price") = 0;
RS("Qty_OnHand") = Request.Form("Quantity");
RS("Pur_Date") = Request.Form("PDate");
RS.Update();
RS.Close();
sql = "Update Catalog Set Price = " + fPrice + " Where
Cat_ID = " + "'" + fCat_ID + "'";
conn.Execute(sql);
```

Deleting a record from a table within the database:

In the following example a record will be deleted from the Catalog table by using the recordset. First a determination will be made as to whether the record exists. If the record does not exist then a redirection is made to a Web page that displays a message that the record is not on file. If the record does exist then the record is deleted by using the Delete method.

VBScript:

```
fCat_ID = Request.Form("Cat_ID")
sql  = "Select * From Catalog Where Cat_ID =" & "'" &
fCat_ID") & "'"
Set RS = Server.CreateObject("ADODB.Recordset")
RS.Open sql, conn, 3, 2
```

To delete the record all that is needed is to execute the Delete method.

```
RS.Delete
RS.Close
```

Jscript:

To delete a record in JScript the logic is exactly the same as in VBScript, only the syntax is different.

```
sql = "Select * From Catalog Where (Cat_ID= '" +
fCat_ID + "')";
RS = Server.CreateObject("ADODB.Recordset");
RS.Open (sql, conn, 3, 2);
RS.Delete();
RS.Close();
```

5. COMPARISON OF SCRIPTING LANGUAGES

Server-side scripting has little concern for which scripting language is used as long as the server environment is compatible with the scripting language that is used. However, this is not true of client-side scripting. If Web pages are developed that include scripts that will be executed on the client the browser environment must be taken into account. While developing, it is recommended that the Web pages be tested in both Internet Explorer and Netscape Communicator. Do not assume that if it works in one it will work correctly in the other.

JavaScript from Netscape and JScript from Microsoft are both compliant with The European Computer Manufacturers Association (ECMA) standard on Internet scripting language (ECMA-262). This ECMA Script standard was adopted in 1998. The standard was based on JavaScript 1.1 and it is assumed that Microsoft and Netscape will abide by the standard. Of course developers must be aware of enhancements to the standard that might be offered by Microsoft and Netscape. Therefore in the discussion JavaScript and JScript will be treated as synonymous or equivalent.

VBScript is a subset of Microsoft's Visual BASIC and with client-side scripting runs in Microsoft's Internet Explorer. JavaScript which was developed by Netscape and runs in either Microsoft's Internet Explorer or Netscape's Communicator.

Comparing Variables Between the Scripts

A variable is the name of a location in memory where data is stored during application execution. VBScript allows explicit and implicit declaration of variables. Implicit is where the variable is created when it is used. The better programming approach is to explicitly declare the variables prior to use. With explicit declaration it at least appears like some planning was done prior to writing the application. By using the *Option Explicit* statement at the beginning of the script we can force all variables to be declared prior to their use in another statement. Examples of variable declarations follow:

```
Dim fName, lName
```

As far as data typing in VBScript everything is of *variant* type. This type allows the storage of different data types in the same variable. The variant data type allows several common subtypes: boolean, currency, date, double, integer, long, object, single, and string. Explicit conversion from one subtype to another is by CInt() to convert a value to an integer and CSgn() to convert a value to a single precision real number. Other conversion functions are available.

VBScript also allows arrays. An array is declared like a variable with the Dim statement. Dim students(100) declares an array with 101 elements because VBScript like Visual BASIC is zero based. Arrays can be static in size or dynamic. The dynamic array is declared without specifying the number of elements. Dim students() is an example. During program execution, when you know the size of the array you use a ReDim statement to change the size of the array, such as ReDim students(100). VBScript allows multidimensional arrays with up to 60 dimensions.

Arrays in JScript are created by defining a variable and adding square brackets to it like Student[0] = "Joe Cool". JScript allows you to mix data types in the same dimension of the array. Arrays in JScript are referred to as being *sparse*. Sparse means that only the elements of the array that are assigned a value occupy memory. So if we declare Student[1], Student[2], Student[10], we have only three elements in our array that require memory space. However, the .length method would report the array length to be 10 and not 3.

JScript like VBScript allows implicit and explicit declaration of variables. Again the better technique is to explicitly declare your variables with the var statement. A variable can be implicitly declared by initializing the value of a variable. An interesting fact is that when a variable is declared in this manner it becomes a global variable and its use and value are available to the entire script, but only after the line of code initializing it is executed. The scope of an implicitly declared variable in VBScript maintains the scope of where it is created.

JScript does not force the programmer to initialize the value of the variable. However, trying to extract a value from a variable prior to it being initialized will result in an error. VBScript does not force the programmer to initialize the value of the variable.

Like VBScript a variable in JScript is not declared to be of any data type. The type is determined when you set the value of the variable. JScript has the following data types: number, boolean, string, function, and object. The single data type of number can hold integer or real numbers. Another interesting feature is that the data type of a variable can be switched or converted from one type to another by assigning it data of a different type. The lack of strong data typing and easy conversion from one type to another can be problematic. A typeof operator is available that returns the type of data held by the variable. JScript has two explicit functions to convert a string that is a numeric character to either an integer (parseInt()) and (parseFloat()) to convert text to a real number.

Another feature of JScript that programmers should be aware of is that JScript is case-sensitive. So a variable named Student is not the same as student. This feature has confused many former BASIC and COBOL programmers.

VBScript supports user-defined constants. Instead of the DIM statement to declare a variable you use the Const statement. For example Const ConRate = .05 would create a constant with a value of .05. JScript does not support constants in the normal programming sense. Constants in most languages are assigned a value when they are created and cannot have the value changed anywhere in the program. In JScript the programmer can still declare a variable and use it as a constant, however, the system will not prevent the programmer from changing the value elsewhere in the code.

Comparing the formatting of currency fields

In VBScript, formatting as currency is a fairly simple process. The following statement would format a field named Price and write it to the Web page:

```
<%= FormatCurrency(Price) %>
```

To accomplish the same functionality in JScript, however, a user-defined function is necessary. If the name of the function defined in JScript is the same as the function in VBScript, then the code to write to the Web page remains the same.

```
<%= FormatCurrency(Price) %>
```

However, the following programmer defined function must be written:

```
function FormatCurrency(valuein) {
    var GetIt = "" + valuein;
    var FormatStr="";
    var OutDollars="";
    var tDollars="";
    var Decipos=GetIt.indexOf(".");
    if  (Decipos == -1)
    {
        Decipos = GetIt.length
    }
    var Dollars = GetIt.substring(0,Decipos);
    var DollarLen = Dollars.length;
    if  (DollarLen > 3 )
    {
        while   (DollarLen > 0)
        {
            tDollars = Dollars.substring(DollarLen -
            3, DollarLen);
            if  (tDollars.length == 3)
            {
                OutDollars = "," + tDollars +
                OutDollars;
                DollarLen = DollarLen - 3;
            }
            else
            {
                OutDollars = tDollars + OutDollars;
                DollarLen = 0;
            }
        }
    }
    else
    {
        OutDollars = Dollars;
    }
    if  (OutDollars.substring(0,1) == ",")
    {
```

```
        Dollars = OutDollars.substring (1,
        OutDollars.length);
}
else
{
        Dollars = OutDollars;
}
Cents = GetIt.substring(Decipos + 1, Decipos + 3);
if  (Cents == "")
{
        Cents = "00";
}
if     (Cents.length == 1)
{
        Cents = Cents + "0";
}
var FormatStr = "$" + Dollars + "." + Cents;
return FormatStr;
}
```

Classroom Usage: The use of Active Server Pages was applied to both a Web development class and an MIS class. In the Web development class each student developed a fully interactive Web site in a Client/Server environment. By using Active Server Pages, an Oracle database was updated by adding, changing, and deleting records. The students were primarily senior level Computer Information majors.

In the MIS class, comprised of a cross section of Business majors, the students were placed in groups of five to eight members. Each group was given a pseudo Web site of about fifty Web pages, with a storefront presence. The interaction with the database was fully functional. The pages were modified to present a Web-based company for each group.

In both of these experiences the learning process was extremely positive given the student population of each of the classes. In the case of the Web development class, a complete information system was developed with a Web interface. For the MIS class, even non-CIS majors were able to develop a fully functional Web based company.

6. SUMMARY

When developing server-side scripting using Microsoft's Active Server Pages and their Internet Information Server, either VBScript or JScript are available. The language of choice for most developers is VBScript since it is similar to Visual Basic and Visual Basic for Applications. However, for those developers that are more familiar with Java and JavaScript, JScript is a comfortable alternative.

As this paper has demonstrated, the primary differences between VBScript and JScript lie in the syntax and not in the functionality. The example used which interacts with an Oracle database for connecting to the database; creating record sets; and adding, changing, and deleting records shows identical logic structure.

Where the use of JScript rather than VBScript can become rather tedious is the scarcity of functions in JScript that are available in VBScript. The solution is to write comparable user-defined functions in JScript as demonstrated by the FormatCurrency function.

7. REFERENCES

Danesh, Arman, 1996, T*each Yourself JavaScript 1.1 in a Week, 2nd ed*. Indianapolis: Sams.Net.

Danesh, Arman and Wes Tatters, 1996, *JavaScript 1.1, Developer's Guide*. Indianapolis: Sams.Net.

Mansfield, Richard, 1997, *The Comprehensive Guide to VBScript, The Encyclopedic Reference for VBScript, HTML & Active X*. Chapel Hill: Ventana Press.

Microsoft Windows Script Technologies. VBScript. http://msdn.microsoft.com/scripting/vbscript/ (31 Aug. 2000).

Microsoft Windows Script Technologies. jscript. http://msdn.microsoft.com/scripting/jscript/ (31 Aug. 2000).

Shelly, Gary B., Thomas J. Cashman, William J. Dorin, , and Jeffery J. Quasnay, 2000, *JavaScript, Complete Concepts and Techniques*. Cambridge: Course Technology:.

Walther, Stephen, Steven Banick, Aaron Bertrand, Craig Eddy, Christian Gross, Keith McIntyre, and Jeff Spotts, 1999, *Active Server Pages 2.0*. Indianapolis: Sams.

Wyke, R. Allen, Jason D. Gilliam, and Charlton Ting, 1999, *Pure JavaScript, a Code-Intensive Premium Reference*. Indianapolis: Sams.