

ASP & JavaScript: Enhancing Applications

Dennis Salguero, Beridney Computer Services

Introduction

Within Internet application development, there is a distinct division of application processing, which is shared between the client (web browser) and the Internet server. With technologies like Active Server Pages (ASP), a programmer focuses on server-side development for the majority of the application. Most client-side programming is reserved for simple things like data validation or for creating visual effects like special buttons on menu bars. This presentation will demonstrate some advanced JavaScript techniques that ASP programmers of all levels can use to expand their applications.

This presentation is aimed at ASP programmers since it will help a developer to produce more sophisticated applications. It is beneficial if the programmers have worked with JavaScript before, but not essential. In addition, Internet application developers that work in other languages, such as Cold Fusion or PHP, may still benefit because this presentation focuses on client-side programming, and this is somewhat independent of the language that is used for the server portion of the application.



Dennis Salguero currently runs his own computer consulting firm, Beridney Computer Services, which specializes in Microsoft Office, Visual Basic and Internet application development. A native of Los Angeles and a graduate of the George Washington University, he currently resides in the suburbs of Washington, D.C. He was a contributing author to "Outlook 2000 Programming" by Wrox Press and he is a co-author for "Professional Access 2000 Programming", also by Wrox. In his free time he enjoys reading, golf and chess. He can be reached at dms@beridney.com.

Internet Growth & Development

The Internet has risen in recent years to become a major part of our lives, both in home and business environments. Hardware companies have quickly raised the processing capabilities of Internet servers, and the price point has dropped, leading to the appearance of web farms, offering almost limitless web application growth possibilities for companies. As a consequence, the strain on developers to produce ever more sophisticated, automated and efficient applications has also increased.

Higher demands from Provider and User

These days, the average user has come to expect to receive a higher quality browsing experience, without HTTP 404 (or other) errors. These errors aren't tolerated as readily as they used to be, causing loss of web traffic to your sites if you're not careful. Businesses are also changing their demands. Corporations now want more than "just a web page", with easy access to online credit card processing, credit checks, automated transactions, and many others. Internet application consultancy is very competitive, with business clients expecting relatively quick development times, a large knowledge base with a variety of programming languages, legacy systems, and advanced database systems.

Common Challenges

Keeping in mind that since most business clients expect a sophisticated application, a developer will have to find solutions to many challenges. For example, many companies have information in a variety of databases executing on various platforms and will need to consolidate these into one application.

Client: "Can you interface this application with our LDAP server for user password verification?"

Developer: "Um, what's LDAP?"

Can you guess if this developer won the contract or not?

Language Considerations

A major factor to consider is your choice of language. Which server platform is being used for example - Windows NT/2000 or UNIX? While there are some ASP ports to UNIX, Cold Fusion is still one of the best solutions for a Solaris box. With a Linux version on the way, Cold Fusion will remain the main option for a UNIX box. Below is an overview of the different languages available:

- **ASP** - Allows dynamic content on web pages - a part of IIS (thus must run on Windows NT/2000). Although there are UNIX ports for ASP available, Windows is still the most popular platform for ASP.
- **ColdFusion** - Allaire's answer to ASP, also allowing dynamic web page content. Whether ASP or CF is better remains a hotly contested subject
- **PHP** - The language of choice for dynamic content on Linux servers.

Server-Side vs. Client-Side Programming

This section will examine the two-tier programming components available, client-side and server-side. Naturally, it is up to a developer to decide which level(s) to employ, depending on the needs of the application. The following table examines the advantages and disadvantages associates with each option.

Server-side	Client-side
<p>Advantage - Processing Capabilities</p> <p>Keeping most of the application development on the server allows for the fastest processing of code and the most efficient level of error trapping.</p>	<p>Advantage - Interactivity</p> <p>Client-side still provides the highest level of interactivity. Developers can use both JavaScript and VBScript to interact with the users and provide other functions such as data validation.</p>
<p>Disadvantage - Data Transfer</p> <p>Once the information is processed on the server, it must be transferred back to the client machine for display. While there have been advances in broadband Internet connections and they are slowly proliferating through the home market, the data transfer rates remain a concern.</p>	<p>Disadvantage - Compatibility</p> <p>VBScript will work with IE, but it will not work for users of Netscape Navigator. However, JavaScript works on both browsers, though there is still a dependency between the version of the browser and the code used.</p>

In addition, it is becoming more common for an application to provide content based on the web browser that is used to access the site. Server-side processing does not allow you to do this while client-side does.

Introduction to JavaScript

For the benefit of the ASP community who aren't familiar with JavaScript, or who would appreciate a refresher, here is a small lesson in the basic syntax and use of JavaScript.

Roots of the Language

```
<script language="JavaScript">
if(you_know_c++)
{
you_know_JavaScript;
}
else
{
you_are_in_trouble;
}
</script>
```

It is no secret that JavaScript (and not MS's rival, JScript - an important distinction!) has its roots based in C++. If you have some experience working with C++, then many of the syntax rules of JavaScript will be familiar to you. However, C++ has also been called one of the hardest languages to learn, so if you don't have any experience with C++, you might find that JavaScript presents a formidable challenge to your programming skills.

Combining ASP and JavaScript

This section will show how programmers can begin to implement JavaScript into their ASP code.

Basic Syntax

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
  <title>Page Title</title>

  <script language="JavaScript">
  function showdate()
  {
    var currentdate = new Date();
    alert("The current date and time is: "+ currentdate);
  }

</script>
</head>

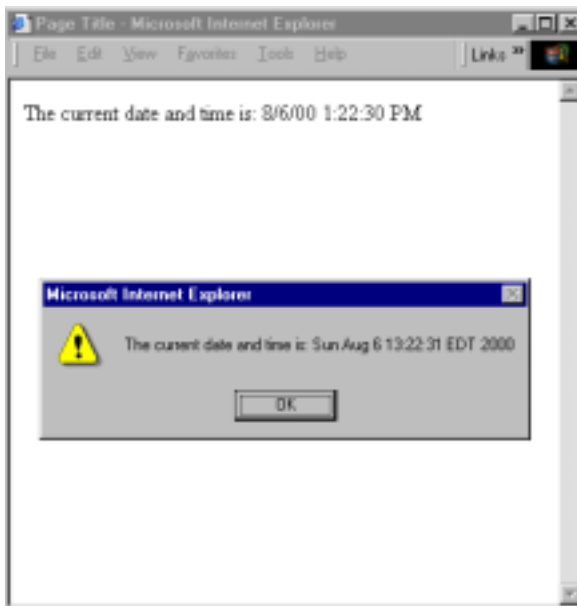
<body onload="showdate();">

  <%
    Dim currentdate
    currentdate = Now()

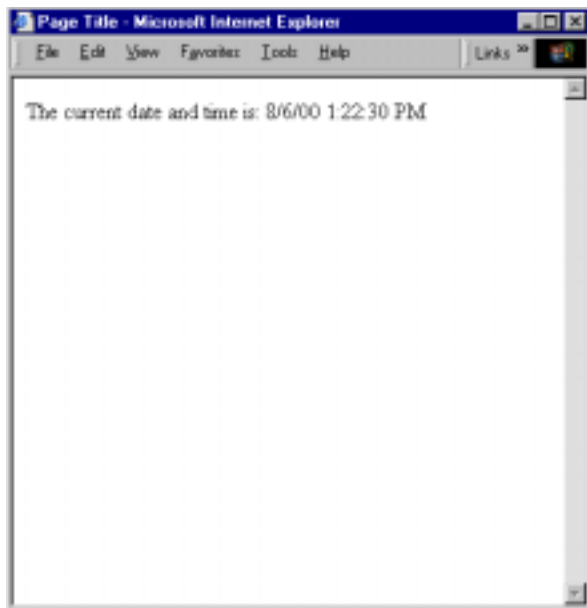
    Response.Write ("The current date and time is: " & currentdate)
  %>

</body>
</html>
```

This code produces an output in two stages. When the page first loads, the date and time are displayed in an alert box, using JavaScript.



Once the alert box is removed, the time and date is displayed using VBScript code.



Let's take a close look at the code that created this page to get a feel for the way we can integrate ASP with JavaScript.

First, look at the declaration of our JavaScript code and the function.

```
<script language="JavaScript">
  function showdate()
  {
    var currentdate = new Date();
    alert("The current date and time is: "+ currentdate);
  }
</script>
```

All JavaScript code should, when possible, be placed in the head portion of the HTML code. In the first line of the function, we are declaring the variable and setting to the date format. We then use the alert function to format the message that will appear.

```
<body onload="showdate();">
```

In order to trigger our JavaScript function, we use the `onLoad` event of the HTML document body to start our function.

Note the following points, which will save you many headaches when debugging your JavaScript code:

- All variables must be declared.
- Unlike ASP, JavaScript does not enforce upper and lower case variables. For example, in ASP, the variable `MyName` and `myname` are the same, but in JavaScript, these are two completely different variables. The same goes for the name of the function and the commands in the code. For example starting our code with `function showdate()` will not cause an error, but starting with `FUNCTION showdate()` will cause an error.
- All JavaScript lines must end with a semi-colon ("`;`"), and all functions need to be encased with brackets ("`{`" and "`}`"), or the code will not compile properly.

For a quick review of ASP, let's look at the rest of the code in our page.

```

<%
  Dim currentdate
  CurrentDate = Now()

  Response.Write ("The current date and time is: " & currentdate)
%>

```

As you can see, we also declare our variable in ASP, even though it's not required - this is merely good programming practice. It is all too easy to develop bad habits such as not declaring your ASP variables - be careful, as JavaScript enforces language rules far more strictly than, say, VBScript - not giving in to these bad habits will make you a much better developer at the end of the day.

We do not even have to give the variable a data type, nor do we have to keep the same text case throughout the code (but heed the words above!). We set this variable to the current date and time and then display it on our page.

Common Uses for JavaScript

JavaScript's most common use is for document validation – with a piece of code similar to the one shown below:

```

<script language="JavaScript" src="form_validation.js">
</script>

<script language="JavaScript">

function validate()
{
  var cansubmit = false;
  cansubmit = ForceEntry(document.frmMain.txtFName, _
    "You must enter a first name");
  if (cansubmit) cansubmit = ForceEntry( _
    document.frmMain.txtLName, "You must enter a last name");
  if (cansubmit) cansubmit = ForceEntry( _
    document.frmMain.txtCompany, "You must enter a company");
  if (cansubmit) cansubmit = ForceEntry( _
    document.frmMain.txtEMail, "You must enter an e-mail address");
  if (cansubmit) cansubmit = ForceEntry(document.frmMain.txtURL, _
    "You must enter a company URL");

  return cansubmit;
}

</script>

```

As you can see, we are using the document object and then declaring a form/text field property combination to validate. We then pass an error statement that the function will display if the value in the text field does not pass the validation rule.

Also, at the top of the code, you can see how we can use a JavaScript file that has functions that we want to re-use on other pages. You declare them with a line like the one shown here.

```

<script language="JavaScript" src="form_validation.js">
</script>

```

This code can result in message boxes like these if the validation rules are not met:



Possible Uses for JavaScript:

Unfortunately, most ASP developers stop using JavaScript once they are able to use it for data validation. There are many other uses available for JavaScript, some of which are described below.

Application Expansion

There are times when an application will require client-side processing for certain operations. A developer that doesn't have knowledge of JavaScript will eventually hit a stopping point where they can no longer develop a portion of their application.

User Interactivity

JavaScript introduces a whole new level of user interactivity with alert boxes and other formats.

Data Modification

Through the use of data arrays, JavaScript can be used to both display and modify data, and to present a much more comprehensive user interface.

Client-Side Processing

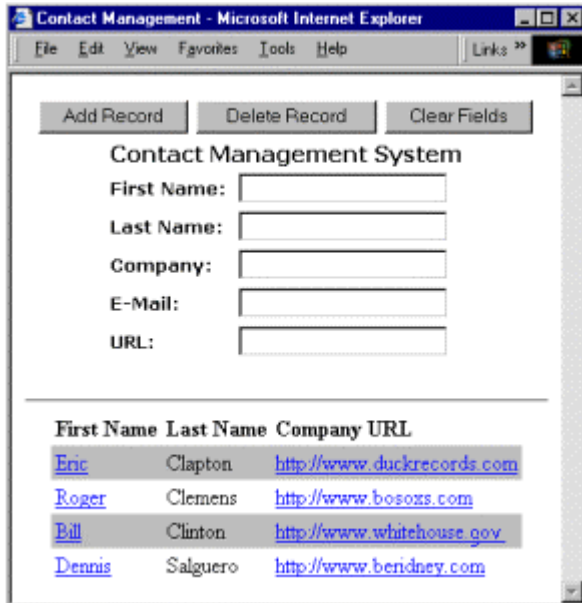
With JavaScript, we are able to take some of the processing requirements off the server and put them on the client, with only a small reduction in speed and efficiency.

Sample Application

This section explores a sample application that combines ASP along with some advanced JavaScript techniques. We will look at a Contact Management Application. This has a variety of components:

- Database development
- Contact information
- Information display
- User interaction
- Data validation
- Links

The final application is shown below:



As you can see, the application consists of an entry form that can be used to enter contact information. There is also a series of buttons that a user utilizes to manipulate the data. At the bottom of the page, the current data in the application is displayed.

Database Development

Selection Criteria

This is perhaps one of the most important parts of an Internet application. In many ways, the database that you choose will influence the speed of your application as well as the user load level that it can handle. When it comes to databases, it is best to err on the side of having a database that can handle more users than you anticipate. After all, some applications that were created decades ago, that no one expected to last more than just a few years, are still in use in many companies.

In this case, since this is a small application, we will be using an Access database. However, with ODBC connections, it is possible to prototype an application with Access and then point the ODBC connection to the final database, such as SQL Server.

Table Design

You should have a working knowledge of database tables and their structures, and normalization rules, used to create tables that can be used most effectively, e.g. splitting the first and last name into different fields and using a unique primary key to track the entries, as is shown below (it's the table we eventually used):



ODBC Connection

A Windows server lets you create an ODBC connection to a variety of databases. In this case we will create one for the Access database and can then alter this connection to use an alternate, more sophisticated database package.

Contact Information

In this section, we will study the different components of the sample application and how each one is created.

HTML Formatting

The first part of our application contains very little functionality, but does set a solid foundation for the rest of our application. The full code of the first iteration is shown below.

```
<html>
<head>
  <title>Contact Management</title>
</head>

<body bgcolor="White">

<center>

<form name="frmMain" method="POST" action="something.asp">
<!-- Start hidden form fields --->
<input type="hidden" name="txtMode" value="1">
<input type="hidden" name="txtID" value="">
<!-- End hidden form fields --->

<table border="0" cellpadding="2" cellspacing="2">
<tr>

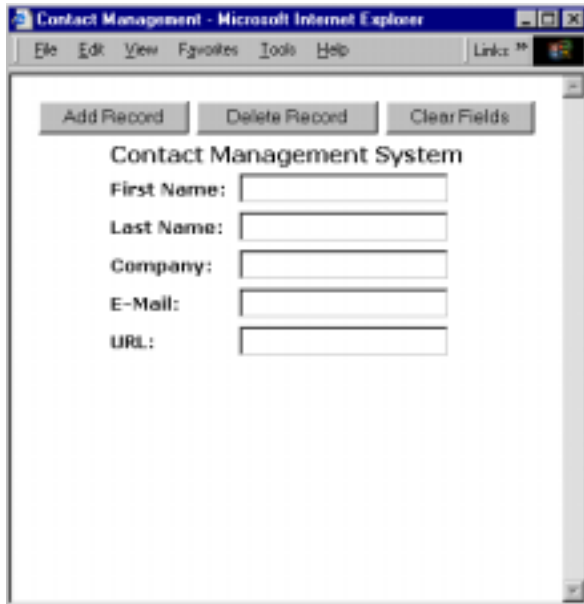
<td>
<input type="button" name="btnAdd" value="Add Record" onclick="javascript:
addrecord();"></td>
<td><input type="button" name="btnDelete" value="Delete Record"
onclick="javascript: deleterecord();"></td>
<td>
```

```



```

This code produces the output shown below.



HTML Buttons

The HTML buttons for our application are created with the code shown here:

```
<table border="0" cellpadding="2" cellspacing="2">
<tr>
<td>
<input type="button" name="btnAdd" value="Add Record" onclick="javascript:
addrecord();"></td>
<td><input type="button" name="btnDelete" value="Delete Record"
onclick="javascript: deleterecord();"></td>
<td>
<input type="button" name="btnNew" value="Clear Fields" onclick="javascript:
clearfields();"></td>
</tr>
</table>
```

As you can see, we are using the `onClick` event of each button to trigger the corresponding JavaScript function. While we haven't seen the functions just yet, understanding the use of buttons and their events is critical to creating highly interactive pages.

HTML Form

Next, we need to declare the form on our page that will contain our text fields. An important note here is that you should name your form since it will make the JavaScript references easier to write.

```
<form name="frmMain" method="POST" action="something.asp">
<!-- Start hidden form fields -->
<input type="hidden" name="txtMode" value="1">
<input type="hidden" name="txtID" value="">
<!-- End hidden form fields -->

<table border="0" cellpadding="2" cellspacing="2">
<tr>
```

```

<td>
<input type="button" name="btnAdd" value="Add Record" onclick="javascript:
addrecord();"></td>
<td><input type="button" name="btnDelete" value="Delete Record"
onclick="javascript: deleterecord();"></td>
<td>
<input type="button" name="btnNew" value="Clear Fields" onclick="javascript:
clearfields();"></td>
</tr>
</table>

<table border="0" cellpadding="2" cellspacing="0">

<tr>
<td colspan="2">
  <font face="Verdana" size="3">
    <b>Contact Management System</b>
  </font>
</td>
</tr>

<tr>
<td>
  <font face="Verdana" size="2">
    <b>First Name:</b>
  </td><td><input type="text" name="txtFName"></td>
</tr>

<tr>
<td>
  <font face="Verdana" size="2">
    <b>Last Name:</b>
  </td><td><input type="text" name="txtLName"></td>
</tr>

<tr>
<td>
  <font face="Verdana" size="2">
    <b>Company:</b>
  </td><td><input type="text" name="txtCompany"></td>
</tr>

<tr>
<td>
  <font face="Verdana" size="2">
    <b>E-Mail:</b>
  </td><td><input type="text" name="txtEMail"></td>
</tr>

<tr>
<td>
  <font face="Verdana" size="2">
    <b>URL:</b>
  </td><td><input type="text" name="txtURL"></td>
</tr>

</table>

```

```
</form>
```

Hidden Form Fields

If you look closely at the code above, you can see that we included 2 hidden form fields at the start of the form (also shown below). As we delve deeper into the code, you will see why we use these fields. Overall, it is important to understand how to declare hidden form fields and find creative ways to use them to your advantage.

```
<!-- Start hidden form fields --->  
<input type="hidden" name="txtMode" value="1">  
<input type="hidden" name="txtID" value="">  
<!-- End hidden form fields --->
```

Information Display

This section will focus on the code necessary to display the data in our application.

Data Query

This first thing we need to do is create a recordset that queries our database and finds all of the current contacts in our table. Realize that our database is stored on the server, which makes this server-side code. The full code is shown below:

```
<%  
Dim objConn  
Dim objRS  
  
Dim BGColor  
  
BGColor = "White"  
  
Set objConn = Server.CreateObject("ADODB.Connection")  
Set objRS = Server.CreateObject("ADODB.Recordset")  
  
objConn.Open "Contacts"  
  
If Request.Form("txtMode")<>" " Then  
Process  
End If  
  
strSQL = "SELECT * FROM tblContacts ORDER BY LName"  
  
objRS.Open strSQL, objConn  
%>
```

You can see that this is fairly straightforward code - we are simply declaring and using our connection object (with the ODBC connections named "Contacts") to execute our SQL statement. This results in a recordset object, `objRS`, which contains all of our contacts.

Table Formatting

Next, we need to create a table that will format the data and then display it in a neat and organized table. This is shown in the code below:

```
<% If (objRS.EOF) AND (objRS.BOF) Then  
  
Response.Write("<b>There are currently no contacts in the database</b>")  
  
End If  
%>
```

```

Else %>
<hr>
<table border="0" cellpadding="3" cellspacing="0">
<tr>
<td>
  <b>First Name</b>
</td>

<td>
  <b>Last Name</b>
</td>

<td>
  <b>Company URL</b>
</td>
</tr>

<%
  Do While NOT objRS.EOF

  If BGColor="Silver" Then
    BGColor = "White"
  Else
    BGColor="Silver"
  End If
%>
<tr bgcolor="<%= BGColor %>">
<td>

<a href="javascript: details('<%= objRS.Fields("ContactID") %>');"><%=
objRS.Fields("FName") %></a>
</td>

<td>
  <%= objRS.Fields("LName") %>
</td>

<td>
<a href="javascript: navigate('<%= objRS.Fields("URL") %>');"><%=
objRS.Fields("URL") %></a>
</td>
</tr>

<%
  objRS.MoveNext
  Loop
%>

</table>

<% End If %>

```

Let's take a closer look at each part of this code.

```

<% If (objRS.EOF) AND (objRS.BOF) Then

  Response.Write("<b>There are currently no contacts in the database</b>")

Else %>

```

In this section, we are simply checking for an empty recordset, which means that we don't need to display a table and the process should end. The rest of the code is a very straightforward combination of ASP and HTML, until we get to the following section.

JavaScript links

```
<tr bgcolor="<%= BgColor %>">
<td>

<a href="javascript: details('<%= objRS.Fields("ContactID") %>');"><%=
objRS.Fields("FName") %></a>
</td>
.
.
.
<td>
<a href="javascript: navigate('<%= objRS.Fields("URL") %>');"><%=
objRS.Fields("URL") %></a>
</td>
</tr>
```

In this section we are creating the hyperlinks for the JavaScript functions that we will be using in our application. Notice that we start off by declaring the link, just like you would for any other page. However, instead of including a link to document name, we are including the name of the JavaScript function that we will be using and including any variables that they require.

This produces links like the ones shown here:



All of this code results in the client output seen earlier, on page 8):

User Interaction

In this section we will focus on creating the functions that will run our form.

JavaScript Programming – Button Functions

The full text of the button functions is shown below:

```
<script language="JavaScript">
function clearfields()
{
document.frmMain.txtMode.value=1;
document.frmMain.btnAdd.value="Add Record";
document.frmMain.txtFName.value=" ";
document.frmMain.txtLName.value=" ";
document.frmMain.txtCompany.value=" ";
document.frmMain.txtEMail.value=" ";
document.frmMain.txtURL.value=" ";
}

function addrecord()
{
```

```

    var valid = validate();
    if (valid)
    {
        document.frmMain.action='default.asp';
        document.frmMain.submit();
    }
}

function deleterecord()
{
    var choice = window.confirm("Are you sure that you want to delete this
record?");
    if(choice)
    {
        document.frmMain.txtMode.value=3;
        document.frmMain.action='default.asp';
        document.frmMain.submit();
    }
}
</script>

```

By taking a look at the above first function (the `clearfields` function), we can start to see some of the JavaScript commands and properties that we will be using. The purpose of said function is to clear out the current information and prepare our form for our new data. You can see that we are using the `document` object, and then declare the form name and then the form component and its property value. Again, remember that text case is vital and your code will generate an error if one of these is incorrect.

You should also notice that we are setting the value for one of the hidden form fields, `txtMode`. This variable will be used when we submit our form to determine whether we need to add, update or delete the record that we are working with. The values are as follows:

txtMode Value	Database Action
1	Add the new record
2	Update the current record
3	Delete the current record

In the case of the second function (`addrecord`), we are clearing out the values so that we can add a new record, so the value of `txtMode` is set to a "1". The function serves to add new records, therefore, we need to validate the entry and then submit our form. At this time, we can declare the page that we want to submit our form to and then issue that function command.

The `deleterecord` function is triggered when a user wants to delete a record. While we could just submit the page right away, it is much more prudent to present the user with a confirmation window and a chance to change their mind. In this case, we are using the `window.confirm` command, which returns a Boolean value. We then check this value and submit the page if the user confirmed the selection. Notice that we also set the `txtMode` value to the appropriate value if the user confirms the selection.

JavaScript Programming – Data Array

The next part of our application requires creating the JavaScript array that will store our database information on the client-side. We start by creating the contact function, which will allow us to create an array that we can reference with conventional names instead of

JavaScript array reference numbers. For this function, we need to declare the length of the array, which is also the number of variables that we are passing to the function.

```
<script language="JavaScript" src="jsobjects.js">
</script>
.
function contact(ID, FName, LName, Company, EMail, URL)
{
    this.length = 6;
    this.id = ID;
    this.fname = FName;
    this.lname = LName;
    this.company = Company;
    this.email = EMail;
    this.url = URL;
}
```

The next part of this task is actually a combination of ASP and JavaScript. Recall that ASP is executed on the server-side while JavaScript is executed on the client-side. By using the ASP `Response.Write` method, we can write JavaScript code to the client-side. Look at the code shown below:

```
<%
Sub CreateJSArray

If IsObject(objRS) Then
    If (objRS.EOF) Then
        objRS.MoveFirst
    End If
End If

Response.write("<script language=""JavaScript"">" & chr(10))

\'(above line) here we are using ASP to create the required \'JavaScript code.
We are also using the contact function to create \'our array.

Response.write("<!--" & chr(10))
Response.Write("var ary_Contact = new Array();" & chr(10))

Do

sArrayString = "ary_Contact[" & objRS.Fields("ContactID") & "] = new
contact(" & _
objRS.Fields("ContactID") & "," & _
chr(34) & objRS.Fields("FName") & chr(34) & "," & _
chr(34) & objRS.Fields("LName") & chr(34) & "," & _
chr(34) & objRS.Fields("Company") & chr(34) & "," & _
chr(34) & objRS.Fields("EMail") & chr(34) & "," & _
chr(34) & objRS.Fields("URL") & chr(34) & ")" & _
";"
Response.Write(sArrayString & chr(10))

objRS.MoveNext

Loop Until objRS.EOF

Response.write("//-->" & chr(10))
Response.write("</script>")
```

```
End Sub
%>
```

When this page is displayed on the browser, it results in the in HTML output shown here:

```
<!-- HTML Output -->

<script language="JavaScript">
<!--
var ary_Contact = new Array();

ary_Contact[12] = new
contact(12,"Eric","Clapton","None","eclapton@duckrecords.com","http://www.du
ckrecords.com");

ary_Contact[10] = new contact(10,"Roger","Clemens","Boston Red
Socks","rceleemens@bosoxs.com","http://www.bosoxs.com");

ary_Contact[14] = new contact(14,"Bill ","Clinton","US Goverment - Head of
Interns","bclinton@whitehouse.gov","http://www.whitehouse.gov");

ary_Contact[1] = new contact(1,"Dennis","Salguero","Beridney Computer
Services","dennis@beridney.com","http://www.beridney.com");
//-->
```

As you can see, then end result is that we have a JavaScript array, `ary_Contact`, organized by the `ContactID`, shown on our HTML page. Naturally, this array contains the information from our table and it is now ready to be dynamically displayed on our form.

Data Validation

JavaScript Include File

Even though it is not vital to this particular application, here is the validation file that is being used by this application. Notice that if the validation fails, this function uses a JavaScript alert box, with a custom message, to alert the user.

```
function ForceEntry(objField, Message)
{
    var strField = new String(objField.value);
    if (isWhitespace(strField)) {
        alert(Message);
        objField.focus();
        objField.select();
        return false;
    }

    return true;
}

function isWhitespace (s)
{
    var i;

    // Is s empty?
    if (isEmpty(s)) return true;

    // Search through string's characters one by one
    // until we find a non-whitespace character.
```

```

    // When we do, return false; if we don't, return true.

    for (i = 0; i < s.length; i++)
    {
    // Check that current character isn't whitespace.
    var c = s.charAt(i);

    if (whitespace.indexOf(c) == -1) return false;
    }

    // All characters are whitespace.
    return true;
}

```

Javascript Implementation

We can implement this function by using a validation function like the one shown here. Notice that we set a particular value to FALSE, and then let the function change this value to TRUE if and only if all of the validation is correct.

```

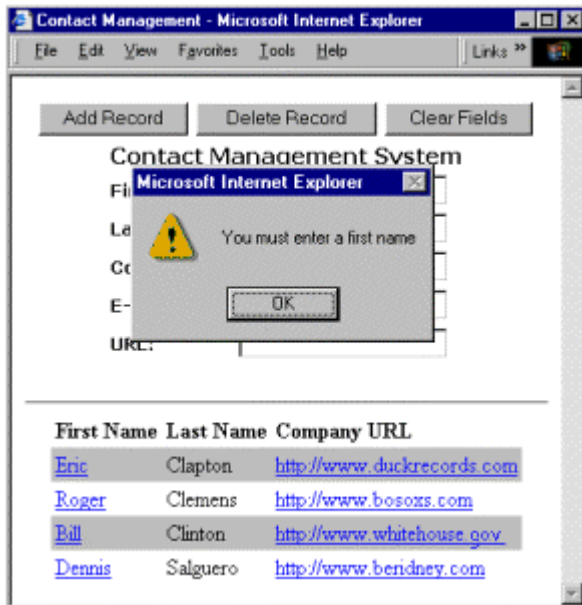
function validate()
{
    var cansubmit = false;
    cansubmit = ForceEntry(document.frmMain.txtFName,"You must enter a first
name");
    if (cansubmit) cansubmit = ForceEntry(document.frmMain.txtLName, "You must
enter a last name");
    if (cansubmit) cansubmit = ForceEntry(document.frmMain.txtCompany,"You must
enter a company");
    if (cansubmit) cansubmit = ForceEntry(document.frmMain.txtEMail,"You must
enter an e-mail address");
    if (cansubmit) cansubmit = ForceEntry(document.frmMain.txtURL,"You must
enter a company URL");

    return cansubmit;
}

function addrecord()
{
    var valid = validate();
    if (valid)
    {
        document.frmMain.action='default.asp';
        document.frmMain.submit();
    }
}

```

Incorrect validation will result in a screen like the one shown here:



Links

JavaScript Functions

One of the last steps in our JavaScript coding is to utilize the JavaScript array that we have created. Recall that our data array is organized according to the unique ContactID of each record. Each link looks like the one shown here:



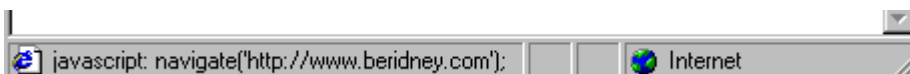
Therefore, we pass this value to the function shown here:

```
function details(ID)
{
  document.frmMain.txtMode.value=2;
  document.frmMain.btnAdd.value="Update Record";
  document.frmMain.txtID.value=ary_Contact[ID].id;
  document.frmMain.txtFName.value=ary_Contact[ID].fname;
  document.frmMain.txtLName.value=ary_Contact[ID].lname;
  document.frmMain.txtCompany.value=ary_Contact[ID].company;
  document.frmMain.txtEMail.value=ary_Contact[ID].email;
  document.frmMain.txtURL.value=ary_Contact[ID].url;
}
```

Here you can see why using the naming function was important to our code. We can call each element of our data array with a conventional name, instead of having to use the index value, which would get confusing very quickly! Also notice that we have set the txtMode value to "2" since this is an update of our current record.

JavaScript Navigation

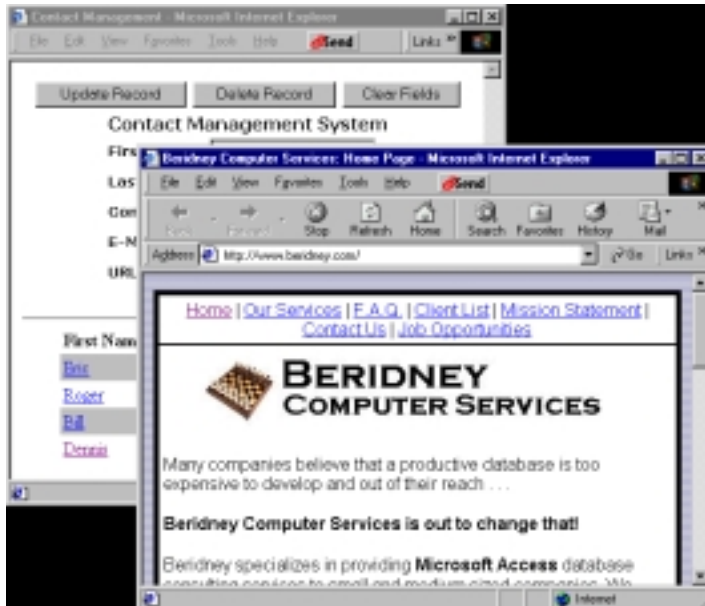
Our second JavaScript link contains a URL and looks like the one shown below:



This link triggers the function shown here:

```
function navigate(strURL)
{
  window.open(strURL);
}
```

We are using the `window.open` property to open a brand new window with the URL that was passed as one of the function parameters. When you click on the link, you will get an output like the one shown below:



While this function navigates to another web page, we can use the same function to open up sub-windows with additional information or new menu bars.

Future of Internet Application Development

This section will wrap up the presentation with a view of the future requirements of Internet applications.

Importance of Scalability

Since the Internet is still fairly new, it is not always easy to have the vision necessary to see how long your applications are going to be used. Therefore, an application should be created with future scalability in mind. For example, creating a database that follows normalization rules will make it much easier to move your data into a new database or create new queries. Also, something as simple as using relative pathing in your application will make it easier to move your application to a new server.

Client Messaging Services

More and more clients are demanding advanced messaging services. In many cases, e-mail is not a sufficient interface. Some clients will demand new messaging mediums like fax or voice to interface with e-mail. Fortunately, there are companies developing VB components to help with these kinds of requirements.

Modular Development

In many cases, you will work with a team from either your client company or other companies. Development is being aimed toward modular development where individual tasks are composed

of creating specific include files or functions. At the head of this, there is one person putting all of the components together.

Security

No matter your target platform or development language, there are always going to be some security concerns to deal with. In addition, this can't be a one-time concern; many times a company's security is breached because an administrator didn't install the latest update or patch.

Reports

Online reports are perhaps one of the most exciting developments for Internet application developers. Although there are some companies attempting to provide turnkey solutions already, there is still room for new solutions to develop.

Conclusion

As a developer, it is your duty to produce the most advanced applications that you can while combining the most efficient and reliable technology available. With ASP and JavaScript, the possibilities are literally endless. This examination of data arrays and other JavaScript methods can be used in your applications, but it should also spur you to learn more of the topics that you aren't necessarily proficient with yet and which can help your development.

Resources

1. The JavaScript library for data validation was provided by:
<http://www.4guysfromrolla.com>
2. A good resource for JavaScript including games:
<http://www.javascript.com>