INFORMATION RESOURCE CENTER

USCG Training Center Petaluma

Introduction to VBScript

INFORMATION RESOURCE CENTER

Introduction to VBScript

Information Resource Center
USCG Training Center Petaluma
599 Tomales Road
Petaluma, CA 94952
Telephone (707) 765-7523

Table of Contents

Section 1

Introduction to VBScript

What is VBScript?

VBScript is a scripting language that addsinteractivity to web pages. Scripting languages are programming languages that provide control in a host environment, in this case a web browser. Scripts are interpreted by the browser and cannot be run alone. In short, scripting languages function because the browsers know how to interpret the code.

VBScript was developed by Microsoft and is fully supported in Internet Explorer. VBScript is not fully supported in other browsers, so its applications on the Internet can be somewhat limited. However, VBScript can be a very useful tool in developing pages for the CGWEB since all Coast Guard workstations use Internet Explorer.

VBScript allows users to interact with web pages instead of just viewing content. Additionally VBScript can be used to create Active Server Pages to provide even more interactivity. VBScript can perform many functions such as validating form data, performing calculations, changing content, providing feedback, and many others.

What You Will Learn

In this course you will learn some of the basics of VBScript. VBScript is easy to learn, but does have some complexity and thus takes time and practice to learn. This course will provide the foundation for further learning while allowing you to implement VBScript right away.

Some of concepts include:

- Creating and using variables
- Writing basic functions
- Interpreting form data
- Using conditional statements

Understanding VBScript

Before we get into scripting, there are a few concepts that need to be discussed. Having an understanding of these concepts will help you understand how to write scripts. These are:

- Objects
- Methods
- Properties
- Variables
- Operators
- Statements
- Procedures
- Events

Objects

Objects are elements of a web page that can be manipulated using a specified set of behaviors. VBScript does not inherently define objects, rather it relies on the browser to define them.

Properties

Properties are like adjectives that describe objects or elements on a web page. VBScript can be used to set or retrieve properties of various elements and objects.

Methods

Methods are procedures or functions that acts on an object. If the object is a noun and properties are the adjectives, methods would be the verbs. Many objects and elements share common methods while others have specific methods.

Variables

Variables are containers that store something. Variables can be used to store just about anything. We won't get into the specifics of what can be stored at this point, but just remember that variables are containers for 'stuff'. The 'stuff' is usually text or numbers. The type of 'stuff' will determine what properties and methods can be used with the variable

Operators

Operators are symbols that when applied to data (or variables) cause the computer to perform an operation on the data. In other words, they tell the computer what to do with data and variables. There are four classes of operators in VBScript: arithmetic, comparison, logical, and string concatenation.

Statements

Statements are instructions that tell VBScript how to execute methods or how to evaluate data and variables. They also provide controls within scripts

Procedures

Procedures are groups of statements that are called on to perform a specific task. There are two types of procedures – subroutines and functions.

Events

Events are a type of procedure that are called automatically by user interaction. An example of an event would be when a user clicks a button.

Getting Help

A Scripter's Reference for VBScript is available online at http://cgweb.tcpet.uscg.mil/iishelp. To locate the reference follow these steps.

1. Go to the web site
2. Once at the web site, on the left side of the page click the plus (+) sign next to **Internet Information Server**. The list will expand to show more options.
3. Click the plus (+) sign next to **Scripter's Reference**. Another list of options will open. You will see the **VBScript Language Reference**.

From there you can click **VBScript Language Reference** or click the plus (+) sign for more options.

The language reference does not provide any tutorials or instructional content, it simply provides the syntax for various operators, objects, properties and methods.

Learning Resources

A simple tutorial is also available on the IISHelp site mentioned above. Instead of expanding the Scripter's Reference options click the plus (+) sign next to **Web Applications** and you will see the VBScript Tutorial.

Another online tutorial is available at http://idm.internet.com/corner/wrox/progref/vbt/index.html. This tutorial covers many of the same topics covered in this workbook.

Section 2

Scripting Basics

Now that the general concepts have been covered, it's time to get started writing some scripts. This first lesson will cover:

- Where scripts go
- Creating variables
- Writing basic functions
- Using basic forms

Before we start with the scripts, let's have a look at what you will be creating. To see a sample of the web page including the completed script go to http://cgweb.tcpet.uscg.mil/webstuff/training/scripts/vbexm1.htm.

Inserting Scripts in Web Pages

Scripts can be placed anywhere in a web page where they are needed. In general, the <head> section is a good place for scripts because they are not included in the body of the page and FrontPage likes them there, but there may be times when you need to put scripts in other places. As a web author, the choice is yours. Put the scripts where they work best for your application.

Script Tags

Scripts in web pages are enclosed in the <script> tags. The script tag has a start and an end. The end tag is required or your page will not display properly. Additionally, it is a good idea to include the script language in the tag. An example is shown below.

```
<script language="VBScript">
   text of script
</script>
```

Writing Basic Procedures

If you recall earlier we discussed procedures and events. It's now time to put that information to use. You may recall that a procedure is a group of statements designed to perform a specific task. There are actually two types of procedures- functions and subroutines. The two are actually quite

similar, but serve different purposes. This lesson will focus on subroutines and leave functions for a later discussion.

When you write a subroutine you need to tell VBScript that you are creating a subroutine by using the *sub* statement followed by the name of the subroutine. Additionally you need to specify where the subroutine ends by using the *end sub* statement. An example is shown below.

```
sub myroutine
   some statements
end sub
```

Calling Subroutines

Subroutines can be called from within a script or by an event. When you want to call a subroutine from a script you use the *call* statement, as shown here.

```
Call myroutine
```

When naming subroutines you can use just about any name you want. VBScript is very open when it comes to names, just be sure to use the exact name when you call the subroutine

Event Procedures

Event procedures are special subroutines that are called by events. Events are usually something the user does on the web page, such as click a button. Unlike subroutines, event procedures have strict naming rules. The name of the event procedure must be the object that triggers the event followed by the actual event name. An example is shown below.

```
sub mybutton_OnClick()
   some statements
end sub
```

In this example the object which triggers the event is called “mybutton” and the event is “OnClick”. So when the button is clicked, the subroutine will be called.

In the coming exercises you will be creating event procedures.

Statements

When writing any VBScript code it is important that each statement be placed on its own line. VBScript will return errors if you try to combine several statements in one line. So no matter how short or long, only put one statement per line. Statements should be separated by a **Return**. When the script executes each line is seen as a separate statement.

Getting Started

Before we begin, you need to open FrontPage and create a new web page. Follow these steps to do this:

1. Open FrontPage. The **Getting Started** window will appear. If it does not, go to the **File** menu, select **New** and trace to **FrontPage Web**.
2. Click **Create a New FrontPage Web**.
3. Click **OK**. The **New FrontPage Web** window will open.
4. In the list of templates click **Empty Web**.
5. Click the **Change** button.
6. Enter the path to your web. Your web will be saved on your **U:\ drive**. Use the path given below.
   U:\vbscript
7. Click **OK**.
8. Enter **VBScitpt Class** as the title for the web.
9. Click **OK**. FrontPage will create a web and take you to Navigation View.
10. Click the **New Page** button. A new page will be created.
11. Double click the new page. The page will open in FrontPage Editor.

Working In FrontPage

FrontPage does support scripts by allowing you to insert them into your page at the location of the cursor. However, there are no indicators on the page to tell you a script exists. Because of this it is best to work in HTML view. You should switch to HTML view now.

Using Notepad

If you want, you can create your web pages in Notepad. Since we are using a client side scripting language you pages will work just be viewing a local file in Internet Explorer. If you want to use Notepad, you should create a folder to store your pages for the class.

Message Box

In this first exercise you will create a simple subroutine and handler that displays a message box. A sample of the web page is available at:

http://cgweb.tcpet.uscg.mil/webstuff/training/scripts/exercise1.htm

You've seen the example of what our script will do, now it is time to start creating it yourself. The script actually has two parts- the subroutine and the event caller. Remember the subroutine is the statements that will be executed when the event is triggered. In our case the subroutine will be called by a button click.

Exercise – Creating the Subroutine

First we'll put the subroutine into the web page, then we'll explain how it works.

1. In FrontPage Editor, click after the </title> tag and press return to insert a blank line.

2. Type in the start script tag:
   ```
   <script language="VBScript">
   ```
3. Press enter to start a new line.
4. Type the following text as you see it here:
   ```
   sub mysubmit_OnClick()
         MsgBox "This is an Alert Box"
   End Sub
   ```
5. Type the end script tag:
   ```
   </script>
   ```
6. Save the page.

Examining the Code

The first line of the script starts the subroutine with the word `sub` which tells VBScript that a new subroutine is starting. Next, the routine is given a name and told what event will cause the procedure to execute. In our script the event is `mysubmit_OnClick()`. This means that when the button named "mysubmit" is clicked, the procedure will execute.

The next line will generate a message box displaying the text in quotation marks.

The last line tells the browser where the procedure ends by using the `End Sub` statement.

Exercise – Creating the event handler

To create the event caller we need to create a simple button.

1. After the `<body>` tag enter the follow tag:
   ```
   <input type="button" name="mysubmit" value="Click Me">
   ```
2. Save the page.
3. View the page in Internet Explorer to make sure it functions correctly.

You should get a web page with a single button that when clicked displays a message box..

Review

You have successfully created a procedure that opened a message box when a button was clicked. That's the basics of VBScript. You create a procedure that will be executed when the user does something. Obviously this is an oversimplified example, but the concept is generally the same. The trick is learning the syntax to make it work the way you want it to.

Creating and Using Variables

Variables are probably the most important aspect of scripting and programming. Essentially it is impossible to create a custom script without using variables. Variables are the containers that hold information that is used in your script. As an example of how variables are used we will add a text input box to our web page that will used as a variable in our script. A sample of the page is available at:

http://cgweb.tcpet.uscg.mil/webstuff/training/scripts/exercise2.htm

Creating Variables

To create a variable in VBScript you simply state the variable name and what it holds. For example if we want a variable to hold the person's name we would use the following statement:

```
Name ="George Washington"
```

In this example the variable is `name` and the value is "George Washington". We include quotes because the value is textual information.

You can also create variables without specifying their value. To do this you need to use the `dim` statement, as shown here.

```
dim name
```

This creates a variable called name that has no value. You can assign a value at a later time with the following statement:

```
Name = somevalue
```

You can create variables using either method, but it is good practice to use the dim statement to create the variable and then assign it a value at a later time. When you do this you can declare all the variables at the beginning of the script, which is a handy way of keeping track of your variables.

Types of Variables

Variables can hold many types of data or information. The type of data that is used is called a **variant**. The two most common used types of variables are text and numbers. When text is used in a variable it is referred to as a **string**. (String is the actual variant type). When text is used as a variant the value must be in quotation marks, as shown above.

Numbers can be one of four variant types – **integer**, **long**, **single**, and **double**. Integer and long are numbers which do not contain decimal points, while single and double can have decimals. We don't need to get into the specifics of each type of numerical variable, VBScript generally determines which one to use. All you really need to do is declare the variable and assign it a value. When numbers are used you simple state the number, as in the example below:

```
count = 3
```

Variable Scope

Variables, once created, can have a limited life span depending on when the variable is created. In general, variables that are created within a procedure can only be used by the procedure. Variables created at the script level can be used anywhere in the script. Take a look at the following script:

```
<script language="VBSCRIPT">
dim firstname
dim lastname
```

```
sub mysubmit_onClick()
   dim fullname
   fullname = "Thomas Jefferson"
   some statements
End sub
</script>
```

In this script the variables **firstname** and **lastname** can be used anywhere in the script, but **fullname** can only be used inside the subroutine. This concept becomes important when you start creating multiple subroutines or functions and need to pass variables between them.

Using Variables

Now it's time to add variables to our script. We'll do this by creating a text box that allows the users to enter their name. When the button is clicked the name will display in the message box. In the previous exercise we created the following script:

```
<script language="VBScript">
sub mysubmit_OnClick()
   MsgBox "This is an Alert Box"
End Sub
</script>
```

In the next exercise we will add variables to the script that will be displayed when the button is clicked. The first thing we need to do is create the variables and assign them a value.

Exercise – Creating Variables

Follow these steps to create variable. As before, we'll go over the code when you're done.

1. In FrontPage Editor, insert a blank line after `sub mysubmit_onclick()`.
2. Using the **dim** statement create a variable called **name**.
3. Assign the value of **name** as `namebox.value` (no quotation marks).
4. Change the MsgBox line to read:
   `MsgBox "Your name is " &  name`
5. Move the cursor down so that it is just before the <input> tag for the button.
6. Enter the following text to create a text input box.:
   `Please enter your name:<input type="text" name="namebox" size ="20">`
7. Save the page.

When your done your script code should look like this:

```
<script language="VBScript">
sub mysubmit_OnClick()
   dim name
   name = namebox.value
   MsgBox "Your name is " & name
End Sub
</script>
```

When the button on your page is clicked it should display the name that was typed in the text box.

Examining the Code

We added a few lines to our code. First we created a variable and assigned it a value. The value we assigned is the value of a text box that we created. So when the button is clicked VBScript will use the text that the user entered in the text box as the value of `name`.

The other change we made was to the message box. Now we are using the concatenation operator to combine a text string and a variable. VBScript will display the text followed by the value of `name`.

Naming Objects

It is important to note the VBScript does not inherently know what that the value of the `name` variable should be the text that was entered into the text box. When we created the text box we gave it a name using the `name` attribute. The name attribute is highlighted in the tag below.

```
Please enter your name:<input type="text" name="namebox" size ="20">
```

In our script we used the name to tell VBScript what the value of the `name` variable should be. The line from the script is shown here:

```
name = namebox.value
```

The bottom line is that if you want your script to use values from text boxes**, you must name the text boxes and assign them to a variable**. If you don't, you will get errors. This becomes increasingly important when you have multiple text boxes (or other form elements) on a page.

Concatenation Operator

Perhaps the most commonly used operator is the concatenation operator - &. The operator is commonly used to combine strings and/or variables. Let's suppose we have two variables, **firstname** and **lastname**. If you want to combine these variables to create a **fullname** variable you would use the concatenation operator as shown below:

```
Fullname = fistname & lastname
```

Review

In this section we covered some the principles of writing basic VBScript scripts. This included:

- Where scripts are placed in a web page
- Writing basic procedures
- Calling procedures using a button
- Creating and using variables.

You also learned how to use a message box and the concatenation operator.

In the next section you will expand on what you have learned by creating more complex procedures.

Section

3

Variables, Statements, and Conditions

Using Numeric Variables and Statements

In the previous exercises we worked with only string variables. However, if you recall the example of the page that you will create it used numerical information to return a result. Using numbers in variables is pretty much the same as using text, except you have many more options for manipulating the data.

To declare a variable as a number variant you simply assign the value as a number as shown below.

```
studentnum = 3
```

This statement assigns the numerical value of 3 to the variable `studentnum`. Notice that there are no quotation marks around the number. With numbers you cannot use quotation marks, or else VBScript will think the variable is a text string.

Now that the variable is a number you can perform a variety of mathematical functions using the variable. For example you can create a new variable based on the number. This is shown below.

```
myvar = 7
newvar = myvar * 2
```

This will return 14 as the value of `newvar`.

Project Overview

The page you will be creating will take the input from two text boxes, multiply them by a constant and return the value. A sample of the page is available at:

http://cgweb.tcpet.uscg.mil/webstuff/training/scripts/vbexm1.htm

In order to create this page our script must include:

- Variables to hold the value of each text box
- A statement to multiply the numbers
- A statement to return a value.

Our web page will need to include the following:

- Two text boxes for entering data
- A button to submit the data
- A place to display the result.

Everything needed to create this page and the script was basically covered in the first section. However, in the next exercise you will be writing statements that use numbers instead of text strings.

Exercise- Return a Value

In this exercise you will write a procedure that returns a numerical value. We'll start with the web page.

1. Create a new web page.
2. In FrontPage Editor, add the following line to you web page:
   `Pier Diem Cost Calculator`
3. Make the line a Heading 1
4. Switch to **HTML** view.
5. On the page create two text boxes separated by a line break.
6. Label the first as "Number of students" and use the **`name`** attribute to name the text box "numstudents".
7. Label the second as "Length of class" and use the **`name`** attribute to name the text box "numlength".
8. Add another line break after the second text box.
9. Using the following tag, add a button to the page:
   `<input type="button" name="mysubmit" value="Submit">`
10. Save the page.

When you are done your page should like the example below

```
<html>
<head>
<title>New Page 1</title>
</head>
<body>
<h1>Per Diem Cost Caluculator</h1>
<p>Number of Students: <input type="text" name="numstudents"
size="20"><br>
Length of class: <input type="text" name="numlength" size="20">
</p>
<p><input type="button" name="mysubmit" value="submit"> </p>
</body>
</html>
```

The button you create was called **mysubmit**, so we need a procedure that will be called by the button. Follow these steps to create the procedure.

1. In the head section type the script tag and press return.
2. Enter the follow text to start the procedure:
   `Sub mysubmit_OnClick()`
3. Press return to start a new line.
4. Using the **dim** statement create variables called **student, length** and **diem**.
5. Assign the value of the **numstudents** text box to the **student** variable using the following statement:
   `student = numstudents.value`
6. Assign the value of the **numlength** text box to the **length** variable.
7. Assign the **diem** variable the following value:
   `student*length*10.13`
8. Add a statement to show a message box displaying the value of **diem**.

When you are done your page should function like the example at http://cgweb.tcpet.uscg.mil/webstuff/training/scripts/vbl3ex1.htm.

For a complete listing of the web page code see Appendix A.

Changing Text Box Values

You script should accurately display the per diem cost in a message box. But you can also use VBScript to display the information in a text box on the page. Actually, VBScript will change the value of the text box to the value of a variable.

To do this we simply need to add another text box to our page and have VBScript put the value of the **diem** variable in the text box.

Exercise – Changing Text Boxes

You need to be working in HTML view in FrontPage editor to complete this exercise.

1. Just before the </body> tag add the following text:
   `The total per diem cost :`
2. On the same line add a text box with the name **totaldiem**.
3. In your script, delete the line with the message box.
4. Add the following line:
   `totaldiem.value = diem`
5. Save the page
6. Preview the page in the Internet Explorer.

When you are done your page should function like the sample page at http://cgweb.tcpet.uscg.mil/webstuff/training/scripts/vbl3ex1b.htm.

For a complete listing of the code see Appendix A.

Conditional Statements

Right now the Per Diem Cost Calculator works. We can put in numbers and are told what the total cost is. However, what happens if you enter some text into one of the boxes? Since the script thinks you should be using numbers, you get an error. To prevent this from happening we need a way to make sure that only numbers are entered into the two text boxes. In other words, we need to validate the data entered into the form.

To do this we need to tell VBScript to make sure the value of the text boxes are numbers, and if they are not numbers to stop the script. Also, we should show a message stating that only numbers can be used. In order to accomplish this we must use conditional statements.

Conditional statements present a statement that can only be true or false. VBScript evaluates the statement to determine if it is true or false and then acts based on the value of the statement. Probably the most common statement is the *If…Then* statement. Let's take a look at a simple example.

```
If the time is before 12 noon, then display "Good Morning" or else
display "Hello"
```

In this statement the condition is whether or not the time is before noon. That statement can either be true or false. If it is true, "Good Morning" is display. If it is false, "Hello" is displayed. The same principle is used in VBScript.

In VBScript the syntax of the *If…Then* statement is as follows:

```
If condition then
   Do something
Else
   Do something different
End if
```

In our script we want to check to see if the value of the text box is a number. Fortunately, VBScript has a function that will do this. The *IsNumberic* function determines if data is a number or not. We can combine the *IsNumeric* function with an *If…Then* statement to check the text box values.

```
If IsNumeric(numstudents.value) <> true then
Msgbox "Please enter a number"
End if
```

In this case we are using the inequality operator <>to evaluate the *If..Then* statement. In other words the first line says "if the isnumeric function does not return true then". This is a backwards way of saying "if it's not a number". You could put use the following code to accomplish the same thing.

```
 If isnumeric(numstudents.value) = False Then
```

Also notice there is no *Else* statement. In this case it is not needed, so it is left out. VBScript will continue with the script if **numstudents** is a number.

Exercise- Conditional Statements

Follow these steps to add the condition to your script.

1. Before the **dim** statements add the following text:
```
If isnumeric(numstudents.value) = False Then
      MsgBox "You must enter a number for the number of
students"
      numstudents.value = ""
      numstudents.focus
End If
```
2. Add a similar **If…then** statement that will check the value of **numlength**.
3. Save the page.
4. Preview it in Internet Explorer and try entering text into the fields.

When you are done your page should function like the example at
http://cgweb.tcpet.uscg.mil/webstuff/training/scripts/vbl3ex1c.htm

For a complete list of the code see appendix A.

Exiting Subroutines

Your script should have worked, well almost. You should have been alerted to put numbers in the text boxes, but the script still ran and returned an error. This is because we did not tell VBScript to halt the script if the field values were not numbers. To do this we need to add the *End Sub* statement to the *If...Then* statement.

Exercise- Exiting Subroutines

To add the *exit sub* statement to your script, follow these steps.

1. Just before each **end if** statement add the following text:
```
Exit Sub
```
2. Save the page and view it in Internet Explorer

When you are done the page should function like the example at
http://cgweb.tcpet.uscg.mil/webstuff/training/scripts/vbl3ex1d.htm

Review

In this section you created a web page that calculated the per diem cost for a class based on input from the user. The web page also validated the data that was entered to ensure that it was a number.

Some of the scripting concepts covered included:

- Using numbers in variables
- Using If…Then statements
- Using the IsNumeric statement
- Exiting subroutines

In the next section you will learn how to create and use arrays.

Section

4

Working with Arrays

By now you should be pretty comfortable using variables and assigning value to variables. In this section you will work with a special type of variable called an **Array**. Arrays are variables that store a series of data elements in a single variable. For example, you could create an array called **months** and use it to store the names of each month of the year. You can then refer to each month separately as if it were stored in its own variable.

The elements of an array are numbered in a zero-based list. The first data element in the list is numbered 0, the next data element is 1, and so on. To refer to a specific element in an array use the following format

```
varname(x)
```

Where **varname** is the variable name and **x** is the number of the element.

Creating Arrays

There are two types of arrays – fixed length and dynamic. In a fixed length array you specify how many data elements you will have when you create the array. In a dynamic array you do not specify the number of data elements.

Fixed Length Arrays

You create arrays the same way you create any variable except that you need to declare the variable as an array. To do this you follow the variable name with the number of elements enclosed in a set of parenthesis.

```
dim myarray(3)
```

This will create an array that can hold four data elements. Remember arrays are zero-based meaning that the first element in the array is numbered 0.

Setting the value of any element of an array is the like setting the value of any other variable, except you need to specify the array element as shown below:

```
myvar(2)="Cats"
```

Dynamic Arrays

Dynamic arrays are used when you don't know how many data elements will populate an array. When you first start writing scripts, it may be hard to think of a situation where you would need a dynamic array, but be assured you will need them.

Creating a dynamic array is the same as creating a fixed length array, except you do not specify the number of data elements.

```
Dim myarray()
```

Setting the data elements in a dynamic array is slightly different than in a fixed length array. Before you can set any data elements you need to dimension the array, or specify the number of data elements. To do this you must use the *ReDim* statement as shown below.

```
ReDim myarray(count-1)
```

In this example the array can a number of date elements equal to the value of **count-1**. In this case **count** is a variable storing a number.

If later you want to re-dimension the array, but want to preserve the data elements that may be in the array you must use the *redim preserve* statement. This will ensure that any existing data elements will not be erased.

```
ReDim Preserve myarray(count+1)
```

Example of Dynamic Arrays

One place where dynamic arrays can be used is with online shopping carts. Since you do not know how many items a person might have in the cart you would use a dynamic array to store the items. By using the *redim preserve* statement you can increase the array as the shopper adds items to the cart.

Exercise- Creating Arrays

One place where you need to use arrays is with the date function of VBScript. VBScript can tell you the month, but only by number. For an example of this go to http://cgweb.tcpet.uscg.mil/webstuff/training/scripts/vbl3ex2.htm. This page will show you the current month number. In order to show the name of the month, we need an array to hold the names of the months that will correspond to the number VBScript returns.

Follow these steps to complete the exercise.

1. Create a new page.
2. Switch to **HTML** view.
3. Enter the following script:

```
sub mysubmit_OnClick()
      currentdate = now
      curmon = Month(currentdate)
```

```
        msgbox "The current Month is " & curmon
end sub
```

4. In the body of the page enter the following text
```
<p>Show the current month:</p>
<p><input type="button" value="Click for current month"
name="mysubmit"></p>
```
5. Save the page.
6. View it in Internet Explorer.

When you are done you should have a page that will display the number of the month when you click the button.

Now we will add the array.

1. At the beginning of script create an array called **months** with **13** data elements.
2. Starting with **months(1)** assign each data element the name of the month that corresponds to the data element number. For example:
   `months(1) = “January”`
   Be sure each data element is on its own line.
3. Change the message box line to the following text:
   `msgbox "The current Month is " & months(curmon)`
4. Save the page.
5. View the page in Internet Explorer.

When you are done the current month name should be displayed when you click the button.

Appendix A

Return A Value Code

```
   <html>

   <head>
   <title>New Page 1</title>
   </head>
   <script language="VBScript">
   sub mysubmit_OnClick()
      dim students
      dim length
      dim diem

      students = numstudents.value
      length = numlength.value
      diem = students * length * 10.13
      msgbox diem
   end sub
   </script>


   <body>

   <h1>Per Diem Cost Calculator</h1>

   <p>Number of Students: <input type="text" name="numstudents"
   size="20"><br>
   Length of class: <input type="text" name="numlength" size="20"> </p>

   <p><input type="button" name="mysubmit" value="submit"> </p>
   </body>
</html>
```

Changing Text Boxes Code

```
<html>

<head>
<title>New Page 1</title>
</head>
<script language="VBScript">
sub mysubmit_OnClick()
   dim students
   dim length
   dim diem

   students = numstudents.value
   length = numlength.value
   diem = students * length * 10.13
   totaldiem.value = diem
end sub
</script>


<body>

<h1>Per Diem Cost Calculator</h1>

<p>Number of Students: <input type="text" name="numstudents"
size="20"><br>
Length of class: <input type="text" name="numlength" size="20"> </p>

<p><input type="button" name="mysubmit" value="submit"> </p>

<p>Total per diem cost:<input type="text" name="totaldiem"
size="20"></p>
</body>
</html>
```

Conditional Statements Code

```
<html>

<head>
<title>New Page 1</title>
</head>
<script language="VBScript">
sub mysubmit_OnClick()
   If isnumeric(numstudents.value) = False Then
         MsgBox "You must enter a number for the number of students"
         numstudents.value = ""
         numstudents.focus
   End If

   If isnumeric(numlength.value) = False Then
         MsgBox "You must enter a number for the class length"
         numlength.value = ""
         numlength.focus
   End If
   dim students
   dim length
   dim diem

   students = numstudents.value
   length = numlength.value
   diem = students * length * 10.13
   totaldiem.value = diem
end sub
</script>


<body>

<h1>Per Diem Cost Calculator</h1>

<p>Number of Students: <input type="text" name="numstudents"
size="20"><br>
Length of class: <input type="text" name="numlength" size="20"> </p>

<p><input type="button" name="mysubmit" value="submit"> </p>

<p>Total per diem cost:<input type="text" name="totaldiem"
size="20"></p>
</body>
</html>
```

Exiting Subroutines Code

```
<html>

<head>
<title>New Page 1</title>
</head>
<script language="VBScript">
sub mysubmit_OnClick()

   If isnumeric(numstudents.value) = False Then
         MsgBox "You must enter a number for the number of students"
         numstudents.value = ""
         numstudents.focus
         exit sub
   End If

   If isnumeric(numlength.value) = False Then
         MsgBox "You must enter a number for the class length"
         numlength.value = ""
         numlength.focus
         exit sub
   End If
   dim students
   dim length
   dim diem

   students = numstudents.value
   length = numlength.value
   diem = students * length * 10.13
   totaldiem.value = diem
end sub
</script>


<body>

<h1>Per Diem Cost Calculator</h1>

<p> </p>

<p>Number of Students: <input type="text" name="numstudents"
size="20"></p>

<p>Length of class: <input type="text" name="numlength" size="20">
</p>

<p><input type="button" name="mysubmit" value="submit"></p>

<p>Total per diem cost:<input type="text" name="totaldiem"
size="20"></p>
</body>
</html>
```