Lab 5: Adding VBScript in HTML

Lab Overview

Objectives

After completing this lab, you will be able to:

- Add VBScript code to an HTML page.
- Write Sub and Function procedures, and call them from VBScript code.
- Write event procedures for standard HTML and ActiveX controls on an HTML page.

Scenario

Controls on a Web page have limited functionality by themselves. Adding VBScript code to a Web page provides a way of linking controls together so they can interact with one another. Using VBScript, you can also add new functionality that is unique to the current situation.

Lab Setup

In this lab, you will add functionality to the Product List Web page, Products.htm. You can continue to edit the page you were working on in Lab 4, or use the page provided in the \Labs\Lab05 directory.

If you need help, the solutions for all of the exercises in this lab can be found in the \Labs\Lab05\Solution directory.

This demonstration shows the solution to this lab.

Estimated time to complete this lab: **120 minutes**.

Exercises

The following exercises provide practice working with the concepts and techniques covered in this chapter.

Exercise 1: Writing Event Procedures

In this exercise, you will write VBScript code to calculate the total cost of items selected in the Product table.

Exercise 2: Linking Controls Together

In this exercise, you will write event procedures for the ActiveX spin controls in the Product table to coordinate them with the corresponding Quantity text boxes.

Exercise 3: Validating Control Values

In this exercise, you will write VBScript code to validate user entries in controls.

Optional Exercise: Adding Code to the Customer Data Page

In this exercise, you will write event procedures for the Country list box on the Customer Data Web page to change the Mask properties of the masked edit controls to reflect the format used by the selected country.

Exercise 1: Writing Event Procedures

Now that the Products Web page contains Quantity controls, you can write script code to total the cost of the items selected by the user.

➤ Add SCRIPT tags to the Products Web page

1. Using Notepad, open the Products Web page, Products.htm, from the root of your Web site.
2. Add the <SCRIPT> tags to the <HEAD> section of the document. Set the LANGUAGE parameter to VBScript for VBScript code.

➤ Write a function to calculate the total cost of the selected products

1. Write a Function procedure, CalcSubTotal, in the <SCRIPT> section of the Products page that calculates the sum of *quantity* * *price* for each item in the Products table.

Note You must hard-code the prices of the different items, because you can't read values out of the HTML table from VBScript (for example, Quantity1.Value * 34.00 + Quantity2.Value * 13.25). To make the page work in a truly dynamic fashion, you would need to place the prices in Hidden controls.

2. Return the result to the calling procedure.

   If you are unsure of the syntax of the function, look at this solution code:

```
Function CalcSubTotal()
    Dim price
    price = Quantity1.Value * 34
    price = price + Quantity2.Value * 13.25
    price = price + Quantity3.Value * 21.05
    price = price + Quantity4.Value * 15
    price = price + Quantity5.Value * 24
    price = price + Quantity6.Value * 10
    CalcSubTotal = price
End Function
```

➤ Calculate Subtotal, Tax, and Total costs

1. To the <SCRIPT> section of the Products Web page, add an event procedure for the OnClick event of the Calculate button.
2. In the event procedure:

   a. Call CalcSubTotal.

   b. Set the **Value** of the Subtotal text box to the calculated cost.

   c. Multiply the subtotal by your state's sales tax, and put the result in the Tax text box.

   d. Add the Subtotal and the Tax, and fill in the Total text box with this new value.

   If you are unsure of the syntax of the event procedure, look at this solution code:

```
Sub CalcTotal_OnClick()
    SubTotal.Value = CalcSubtotal()
    Tax.Value = SubTotal.Value * .0825
    Total.Value = CDbl(SubTotal.Value) + CDbl(Tax.Value)
End Sub
```

3. Save your changes to the Products Web page.

➢ **Test the Web page**

- Open the Products Web page from Internet Explorer, and type numeric values into the Quantity text boxes. Then, select the Calculate button.

Note If you want the Total price to display with only two decimal places, you need to add VBScript code to the CalcTotal_OnClick event procedure to round off the value.

Exercise 2: Linking Controls Together

The Products Web page has spin buttons associated with each Quantity text box. This provides the user with an alternative way of entering a quantity for a product in the list.

In this exercise, you will write event procedures for the spin buttons to coordinate them with the Quantity text boxes.

➢ **Write SpinUp event procedures**

1. If it isn't already open, open the Products Web page, Products.htm, with Notepad.
2. Write an event procedure for the SpinUp event of each spin button in the <SCRIPT> section of the Products Web page.

   Note You can start by writing an event procedure for just the first spin control, and when you have debugged it, duplicate the procedure for the other controls.

   ---

3. In the procedure, increment the value by 1 in the corresponding Quantity text box, and recalculate the subtotal by calling CalcSubTotal.

   If you are unsure of the syntax of the event procedure, look at this solution code:

```
Sub Spin1_SpinUp()
    Quantity1.Value = Quantity1.Value + 1
    SubTotal.Value = CalcSubTotal()
End Sub
```

➢ **Write SpinDown event procedures**

1. Write an event procedure for the SpinDown event of each spin button in the <SCRIPT> section of the Product List Web page.
2. In the procedure:
   a. If the value of the corresponding Quantity text box is 0, do nothing.
   b. Decrement the value of the corresponding Quantity text box by 1.
   c. Recalculate the subtotal.

   If you are unsure of the syntax of the event procedure, look at this solution code:

```
Sub Spin1_SpinDown()
    If (Quantity1.Value > 0) Then
        Quantity1.Value = Quantity1.Value - 1
        SubTotal.Value = CalcSubTotal()
    End If
End Sub
```

3. Save your changes to the Products page.

➢ **Test the Spin button event procedures**

- Open the Products Web page from Internet Explorer, and test the spin button event procedures by clicking the up and down arrows.

Exercise 3: Validating Control Values

Next, you need to validate what is entered in the Quantity text boxes. Instead of duplicating the validation code in the event procedure for each control, you will write one validation function and call it from each OnChange event procedure.

➢ **Write a function to validate entries in the Quantity controls**

1. If it isn't already open, open the Products Web page, Products.htm, with Notepad.
2. In the <SCRIPT> section of the Products Web page, write a function procedure called Validate that takes one parameter (a value) from a Quantity text box.
3. In the Validate procedure:
   a. Test the parameter to make sure it is numeric. If it is non-numeric, notify the user with the **Alert** method, and return False.
   b. Test the parameter to make sure it is positive. If it is negative, notify the user with the **Alert** method, and return False.
   c. Return True if the parameter is numeric and positive.

   If you are unsure of the syntax of the function, look at this solution code:

```
Function Validate(newNumber)
    If IsNumeric(newNumber) Then
        If newNumber > 0 Then
            Validate = True
        Else
            Alert "You can't order a negative quantity."
            Validate = False
        End If
    Else
        Alert newNumber & " is not a valid quantity."
        Validate = False
    End If
End Function
```

➢ **Write an event procedure for each Quantity control**

1. Write an event procedure for the OnChange event of each Quantity control in the <SCRIPT> section of the Products Web page.

Note You can start by writing an event procedure for just the first Quantity control, and when you have debugged it, duplicate the procedure for the other controls.

---

2. In the OnChange event procedure:
   a. Call the Validate function, and pass the value of the text box as the parameter.
   b. If Validate returns False, set the value of the text box to 0.
   c. Call CalcSubTotal, and set the **Value** property of the SubTotal text box to the new subtotal.

   If you are unsure of the syntax of the event procedure, look at this solution code:

```
Sub Quantity1_OnChange()
    If Validate(Quantity1.Value) = False Then
        Quantity1.Value = 0
    End If
    SubTotal.Value = CalcSubTotal()
End Sub
```

3. Save your changes.

➢ **Test the OnChange event procedures**

- Open the Products Web page from Internet Explorer, and type valid and invalid values into the Quantity text boxes.

Note The OnChange event procedure of a text box runs, even when you leave the text box to go to another control.

Optional Exercise: Adding Code to the Customer Data Page

In this optional exercise, you will add code to the Customer Data form created in Lab 4. You will add a button to the form, and an event procedure for its OnClick event that verifies that all fields are filled. You will also write an event procedure for the Country list box that sets the **Mask** property of the Phone and Fax masked edit controls.

➢ **Validate that all text boxes are filled**

1. Open the Customer Data page, CustData.htm, with Notepad.
2. Write a function procedure named Verify:
   a. Check the values of all text boxes to ensure that they are not NULL.

Note When a control is on a form, you must use a fully qualified object name to retrieve and set its properties. For example, the fully qualified name of the Address text control is `Document.Forms(0).Address.`

---

   b. Check the **ClipText** property of the masked edit controls to ensure that they are not NULL.
   c. Do not validate the selection in the Country list box, because it is impossible not to have a selection.
   d. If any control is not filled in, return False.
   e. If all controls are filled in, return True.

   If you are unsure of the implementation of this function, look at this solution code:

```
Function Verify()
    Set x = Document.Forms(0)
    If x.Name.Value <> "" and _
       x.Title.Value <> "" and _
       x.Company.Value <> "" and _
       x.Email.Value <> "" and _
       x.Street.Value <> "" and _
       x.City.Value <> "" and _
       x.State.Value <> "" and _
       x.Zip.Value <> "" and _
       x.Phone.ClipText <> "" and _
       x.Fax.ClipText <> "" Then
        Verify = True
    Else
        Verify = False
    End If
End Function
```

3. Add a generic button named Test (TYPE="Button" NAME="Test") to the form.
4. Write an OnClick event procedure for the Test button, and call the Verify function.
   a. If all controls are valid, display a message box to that effect.
   b. If any control is invalid, display a message box to that effect.
5. Save the changes to the Customer Data Web page.

➤ Test the Web page

- Open the Customer Data Web page from Internet Explorer, and test the Verify function by clicking the Test button.

➤ Change the Mask for the Phone and Fax controls when a new country is selected

The format of phone numbers is different in different countries. Therefore, change the **Mask** property of the Phone and Fax masked edit controls when a new country is selected.

1. Write a Sub procedure named ChangeMask:
   a. Retrieve the item selected in the Country list box.
   b. Change the **Mask** properties of the Phone and Fax masked edit controls to the appropriate value for the selected country. Use the information in the following table:

Country	Phone Number Format
Brazil, Spain	(12)123-1234
USA	(123) 123-1234
France	12.12.12.12
Germany, Switzerland	1234-123456

   If you are unsure of the implementation of this procedure, view this solution:

```
Sub ChangeMask()
    Set x = Document.Forms(0)
    If x.Country.Value = "Brazil" or _
       x.Country.Value = "Spain" Then
        x.Phone.Mask = "(##)###-####"
        x.Fax.Mask = "(##)###-####"
    Elseif x.Country.Value = "USA" Then
        x.Phone.Mask = "(###)###-####"
        x.Fax.Mask = "(###)###-####"
    Elseif x.Country.Value = "France" Then
        x.Phone.Mask = "##.##.##.##"
        x.Fax.Mask = "##.##.##.##"
    Elseif x.Country.Value = "Germany" or _
           x.Country.Value = "Switzerland" Then
        x.Phone.Mask = "####-######"
        x.Fax.Mask = "####-######"
    End If
End Sub
```

2. Write an OnClick event procedure for the Country list box, and call the ChangeMask function.
3. Save the changes to the Customer Data Web page.

➤ Test the Web page

- Open the Customer Data Web page from Internet Explorer, and test the new list box event procedure.