Unit V. Regular Expression, Rollover and Frames

Introduction

- Sometime, someone enter wrong information into a form displayed on our web page? So, to prevent this by writing javascript that validates information on the form before our form is processed by the CGI application running on the web server.
- While this was useful for performing basic validation of a form, the string object lacks the power to perform sophisticated validation and formatting that is found in commercial JavaScript applications.
- In this chapter, you'll learn how to master regular expressions and use them to manipulate information in amazing ways.

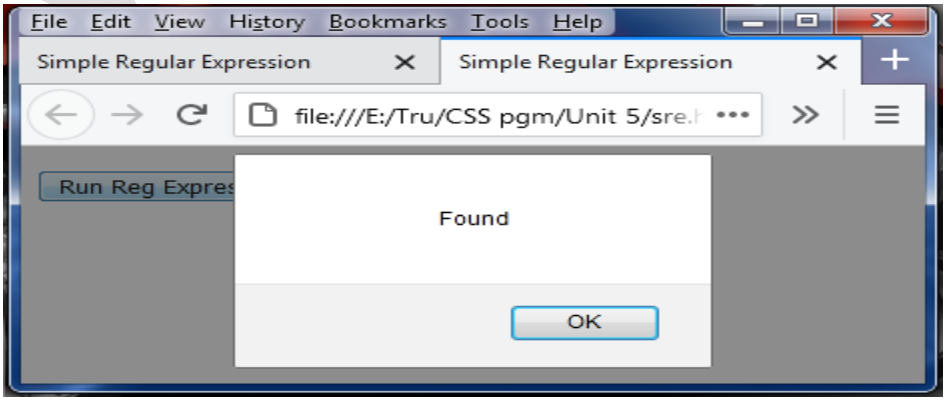
5.1 Regular Expression

- A regular expression is very similar to a mathematical expression, except a regular expression tells the browser how to manipulate text rather than numbers by using special symbols as operators.
- For example, the browser might be told to determine whether a specific character exists in one or more lines of text. Likewise, the browser might be told to replace all occurrences of a word with another word. This and more can be accomplished by writing a regular expression.
- Example, to determine whether the letter b or letter t is in the name Bob and display an appropriate message in an alert dialog box when a button is clicked on the form.

```
< !DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
 <title>Simple Regular Expression</title>
 <script language="Javascript" type="text/javascript">
 <!--
 function RegExpression()
{
```

```
var name='Bob'
re = /[bt]/
if (re.test(name))
{
alert('Found')
}
else
{
alert('Not Found')
}
}
-->
</script>
</head>
<body>
<FORM action="http://www.jimkeogh.com" method="post">
<P>
<INPUT name="Run Reg Expression" value= "Run Reg Expression" type="button" onclick= " RegExpression()"/>
</P>
</FORM>
</body>
</html>
```

Output :



- The regular expression is located in the RegExpression() function definition in the <head> tag of the web page.
- A regular expression begins and ends with a slash (/).
- You place the special symbols that make up the regular expressions between slashes.
- This tells the browser to search the text for characters that appear within the brackets.
- In this expression, two characters are within the square brackets: a b and a t, which tells the browser to determine whether the text includes a b or a t, or both. That's the regular expression.
- The regular expression is assigned to the `re` variable.
- Notice that we don't use quotation marks, which would tell the browser that the special symbols of the regular expression is part of a string, which it isn't.
- The test() method is called and passed the variable name that contains the string Bob. The test() method is one of several methods of the regular expression object.
- The browser evaluates Bob using the regular expression.
- A true is returned if either a b or a t or both are found in the name Bob; otherwise a false is returned.
- Depending on this result, the appropriate alert dialog box is displayed on the screen.

5.1.1 Language of regular expression

- A regular expression is a complex instruction that the browser has understand.
- The words of the regular expression language are called special characters and act similarly to an operator in a mathematical expression. An operator, tells the browser to perform an operation on operands, which are values.
- Special characters tell the browser to perform an operation on text.
- Table 10-1 contains special characters that are used to create a regular expression.
- We can place any number of characters, numbers, or punctuation or symbols within the brackets, and the browser will determine whether they exist in the text.

- *However, one symbol may pose a problem: suppose you want to determine whether the text contains the bracket ([) symbol? This can be troublesome since the [is a special character in a regular expression and will confuse the browser.*
- *The browser assumes the [is enclosing an operation to perform, so it won't search the text for the [character.*
- If you want to search for a symbol that is also a special character, you must precede the symbol with a backslash (\), which is known as an escape character.
- Here's how the browser reads this regular expression:

1. The / character tells the browser that this is the beginning of a regular expression.

2. The [character tells the browser to search the text for the following character(s).

3. The \ tells the browser to ignore the special meaning of the next character.

4. The [character is the character that the browser will search for in the text.

5. The] character tells the browser that there are no more characters to search for.

6. The / character tells the browser that this is the end of the regular expression.

Special Character	Description
\	Tells the browser to ignore the special meaning of the following character
^	Beginning of a string or negation operator, depending on where it appears in the regular expression
$	End of a string
*	Zero or more times
+	One or more times
?	Zero or one time; also referred to as the optional qualifier
.	Any character except a newline character (\n)
\b	Word boundary

\B	Nonword boundary
\d	Any digit, 0–9
\D	Any nondigit
\f	Form feed
g	Search the first and subsequent occurrences of the character(s)
i	Search without matching the case of the character
\n	Newline; also called a line feed
\r	Carriage return
\s	Any single whitespace character
\S	Any single non-whitespace character
\t	Tab
\v	Vertical tab
\w	Any letter, number, or underscore
\W	Any character other than a letter, number, or underscore
\xnn	The ASCII character defined by the hexadecimal number nn
\o>nn	The ASCII character defined by the octal number nn
\cx	The control character x
[abcde]	A character set that matches any one of the enclosed characters
[^abcde]	A character that does not match any of the enclosed characters
[a-e]	A character that matches any character in this range of characters; the hyphen indicates a range
[\b]	The backspace character
{n}	Exactly n occurrences of the previous subpattern or character set
{n,}	At least n occurrences of the previous subpattern or character set
{n,m}	At least n but no more than m occurrences of the previous subpattern or character set
(x)	A grouping or subpattern, which is also stored for later use
x|y	Either x or y

Table 10-1 Special Characters Used to Create a Regular Expression

5.1.2 Finding non matching characters

- Sometimes a JavaScript application prohibits certain characters from appearing within text entered into a form, such as a hyphen (-); otherwise, the character might inhibit processing of the form by the CGI program running on the web server.
- You can direct the browser to search for illegal character(s) by specifying the illegal character(s) within brackets and by placing the caret (^) as the first character in the bracket.
- Let's see how this works in the following example: /[^\-]/
- In this case, the browser is asked to determine whether the text does not contain the hyphen.
- The caret asks the browser to determine whether the following character(s) do not appear in the text.
- To find the hyphen in text, you need to escape the hyphen with the backslash, like so \-.
- Suppose you wrote the following regular expression and the browser didn't find the hyphen in the text.
- The browser responds with a false—this is because you are telling the browser to determine whether the hyphen appears in the text.
- If the hyphen appears, the browser would respond with a true.

  /[\-]/

- However, by placing a caret in the regular expression, as shown next, the browser responds with a true if the hyphen is not found in the text. This is because you are telling the browser to determine whether the hyphen does not appear in the text.

  /[^\-]/

5.1.3 Entering a range of characters

- You don't need to enter every character that you want the browser to match or not match in the text if those characters are in a series of characters, such as f through *l*. Instead of including each and every character within brackets, you can use the first and last character in the series, separated by a hyphen.
- Let's say that you need to tell the browser to match any or all of the characters f,g, h, i, j, k, or l in the text. You could write the following regular expression: `/[fghijkl]/`
- Alternatively, you could write the following regular expression, which tells the browser to match any letter(s) that appears in the series f through and including l: `/[f-l]/`
- Likewise, you can tell the browser not to match any characters in a range of characters using the same kind of regular expression, except you place the caret in front of the first character, as shown here: `/[^f-l]/`
- In this case, the browser would return true if none of the characters *f* through *l* were found.

5.1.4 Matching Digits and Nondigits

- Limiting an entry either to digits or nondigits is a common task for many JavaScript applications.
- For example, a telephone number entered by a user should be a series of digits, and a first name should be nondigits. Nondigits appearing in a phone number indicate the user entered an invalid phone number. Likewise, a first name that contains digits is likely an invalid first name.
- You can have the browser check to see whether the text has digits or nondigits by writing a regular expression.
- The regular expression must contain either \d or \D, depending on whether you want the browser to search the text for digits (\d) or nondigits (\D).
- The \d symbol, as shown in the following example, tells the browser to determine whether the text contains digits.
- The browser returns a true if at least one digit appears in the text.

- You'd use this regular expression to determine whether a first name has any digits, for example.
- If it does, the browser returns a true and your JavaScript notifies the user that an invalid first name was entered into the form.
  /\d/
- The \D symbol is used to tell the browser to search for any nondigit in the text.
- This is illustrated next. The browser returns a true if a nondigit is found.
- This is the regular expression you would use to validate a telephone number, assuming the user was asked to enter digits only.
- If the browser finds a nondigit, the telephone number is invalid and you can notify the user who entered the information into the form.
  /\D/

5.1.5 Matching punctuations and symbols

- Now, see the browser determine whether text contains or doesn't contain letters, punctuation, or symbols, such as the @ sign in an e-mail address, by using the \w and \W special symbols in a regular expression.
- The \w special symbol tells the browser to determine whether the text contains a letter, number, or an underscore, and the \W special symbol reverses this request by telling the browser to determine whether the text contains a character other than a letter, number, or underscore.
- Let's say that you were expecting a person to enter the name of a product that has a combination of letters and numbers.
- You can use the following regular expression to determine whether the product name that was entered into the form on your web page contains a symbol: /\W/
- Using \W is equivalent to using [a-zA-Z0-9_].

5.1.6 Matching words

- You might want the browser to search for a particular word within the text. A word is defined by a word boundary—that is, the space between two words.
- You define a word boundary within a regular expression by using the \b special symbol.
- Think of the \b special symbol as a space between two words.
- You need to use two \b special symbols in a regular expression if you want the browser to search for a word: the first \b represents the space at the beginning of the word and the second represents the space at the end of the word.
- Let's say you want to determine whether the name Bob appears in the text. Since you don't want the browser to match just text that contains the series of letters B-o-b, such as Bobby, you'll need to use the word boundary to define Bob as a word and not simply a series of letters.
- Here's how you'd write this regular expression: /\bBob\b/

5.1.7 Replacing a the text using regular expressions

- you can also use a regular expression to replace portions of the text by using the replace() method.
- The replace() method requires two parameters: a regular expression and the replacement text.
- Here's how the replace() method works. First, you create a regular expression that identifies the portion of the text that you want replaced.
- Then you determine the replacement text. Pass both of these to the replace() method, and the browser follows the direction given in the regular expression to locate the text.
- If the text is found, the browser replaces it with the new text that you provided.
- The next example tells the browser to replace Bob with Mary in the text. The regular expression specifies the word Bob.

- The replace() method of the string object is then called to use the regular expression to search for Bob within the text and then replace Bob with Mary.
- However, the original string isn't modified. The modified string is returned by the replace() method. You could assign the modified string to the variable containing the original string if you don't need the original string anymore.
- A common problem is to replace all occurrences of one or more characters of a string.
- You do this by creating a regular expression and calling the replace()method; however, you'll need to place the g special character at the end of the regular expression, which tells the browser to replace all occurrences of the regular expression in the string. This is shown here:

```
/\bBob\b/g

re = /\bBob\b/

text = 'Hello, Bob and welcome to our web site.'

text = text.replace(re, 'Mary')
```

5.1.8 Returning expression object properties

- In addition to methods, the regular expression object has properties that you can access from within your JavaScript by referencing the name of the regular expression object followed by the property name.
- Table 10-2 lists these properties.
- For example, let's say that you want to access the last characters that were matched by the regular expression.
- As you'll notice in Table 10-2, the lastMatch property contains the last characters that were matched by the regular expression object. You reference this by using the following expression: re.lastMatch

Regular Expression Object	Properties
$1 (through $9)	Parenthesized substring matches
$_	Same an input
$*	Same as multiline
$&	Same as lastMatch
$+	Same as lastParen
$`	Same as leftContent
$'	Same as rightContext
constructor	Specifies the function that creates an object's prototype
global	Search globally (g modifi er in use)
ignoreCase	Search case-insensitive (i modifi er in use)
input	The string to search if no string is passed
lastIndex	The index at which to start the next match
lastMatch	The last match characters
lastParen	The last parenthesized substring match
leftContext	The substring to the left of the most recent match
multiline	Whether strings are searched across multiple lines
prototype	Allows the addition of properties to all objects
rightContext	The substring to the right of the most recent match
source	The regular expression pattern itself

Table 10-2 Properties of the Regular Expression Object

5.2 Frames

- Now, We visit web sites we are able to scroll the main portion of web page while a smaller section containing navigation remained stationary on the screen.
- Although this part is also contained on a single web page, actually multiple web pages appeared on the screen at the same time, and each was displayed in a frame.
- This Frames are created using HTML, but you can interact and manipulate frames using a JavaScript.

5.2.1 Create a frame

- All frames contain at least three web pages. The first frame surrounds the other frames, and this entire collection is called the frameset. The other frames are within the frameset, and each is referred to as a child. We can give each child a unique name so we can later refer to it in our application.
- JavaScript refers to the frameset as the top or the parent. The parent frame is always at the top of the display. Child windows appear within the parent window.
- You can nest frames many layers deep—so the top level may actually still be a child frame of another frameset.
- Let's create a simple frame that contains two child windows. We'll begin by defining the frameset using the <frameset> HTML tag. The frameset can be divided into columns and rows, depending on the needs of our application. Columns divide the frameset vertically using the cols attribute of the <frameset> tag. Rows divide the frameset horizontally using the rows attribute of the <frameset> tag.
- The number of rows or columns that appear in a frameset is determined by the value assigned to these attributes.
- Each column or row is represented by a percentage that indicates the percent of the frameset that is taken up by the column or row.
- We can also specify a width and height—it doesn't have to be a percentage of the available window.
- If we want to divide the frameset evenly into two child windows.

- One child window is at the top and the other is at the bottom. Since you are dividing the frameset horizontally, they need to define the rows attribute. The top child window takes up 50 percent of the frameset, and the bottom child window takes up the other 50 percent. Here is the value that is assigned to the rows attribute to create these child windows:
  <frameset rows="50%,50%">
- After defining the frameset, we can insert a web page into each child window by using the <frame> HTML tag.
- Each child window has its own <frame> tag. You specify the web page that will be displayed in the child window by defining a value for the src attribute of the <frame> tag.

- For example, suppose that you want WebPage1.html to appear as the top child window.
- Here's what you'd need to write (although it makes sense to name the top child window topPage, you can assign any name you want to the child window):

```
<frame src="WebPage1.html" name="topPage" />
```

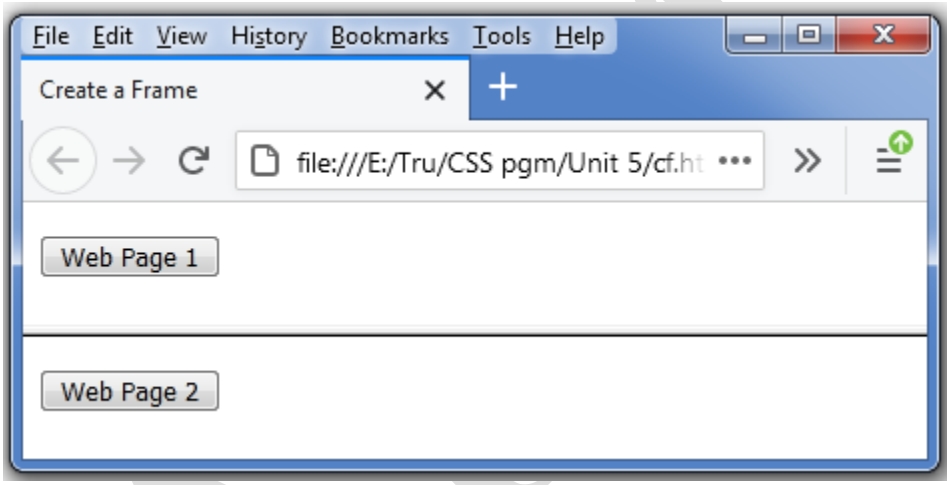
- You'll need to define a <frame> tag within the <frameset> tag for each child window contained in the <frameset> tag. The first <frame> tag within the <frameset> tag refers to the upper left–most child window. Subsequent <frame> tags refers to child windows that appear left to right, top to bottom within the <frameset> tag.

- The following example shows how to create a frameset that contains two child windows, one on the top and the other on the bottom (Figure 11-1).

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
 <title>Create a Frame</title>
</head>
```

```
<frameset rows="50%,50%">
 <frame src="WebPage1.html" name="topPage" />
 <frame src="WebPage2.html" name="bottomPage" />
</frameset>
</html>
```

Output :

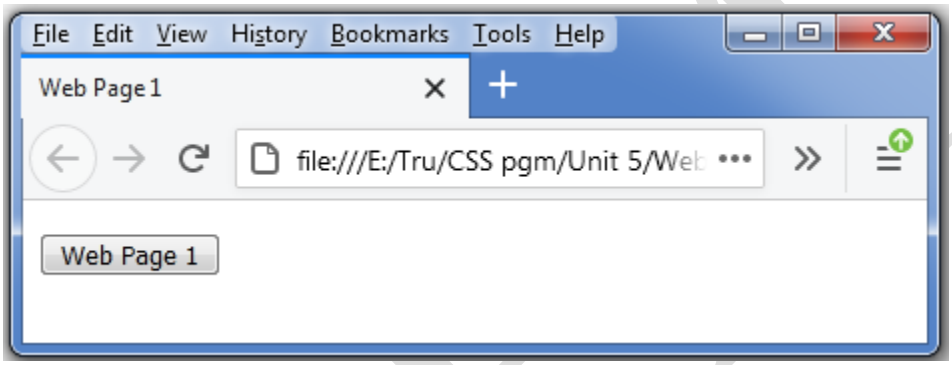


The following is WebPage1.html, which appears at the top of the frameset:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
 <title>Web Page 1</title>
</head>
 <body>
 <FORM action="http://www.jimkeogh.com" method="post">
 <P>
```

```
<INPUT name="WebPage1" value="Web Page 1" type="button" />
</P>
</FORM>
</body>
</html>
```

Output :

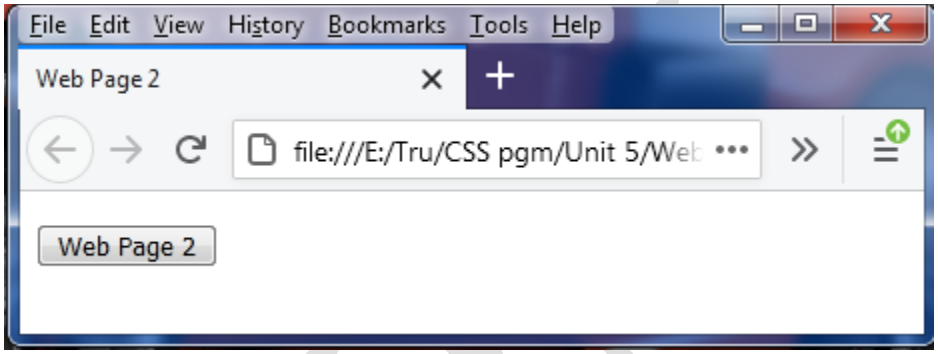


The following is WebPage2.html, which appears at the bottom of the frameset:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
 <title>Web Page 2</title>
</head>
 <body>
 <FORM action="http://www.jimkeogh.com" method="post">
 <P>
 <INPUT name="WebPage2" value="Web Page 2"  type="button" />
```

```
</P>

</FORM>

</body>

</html>
```

Output :



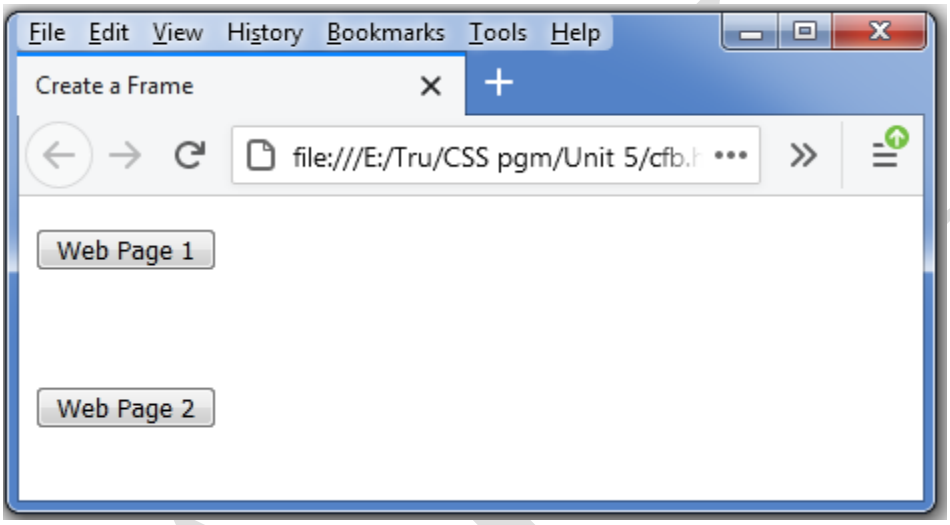
5.2.2 Invisible borders of frame

- The border can be hidden by setting the frameborder and border attributes of the <frame> tag to zero (0).
- Any value other than 0 that is assigned to the frameborder and border attributes causes the browser to display the border.
- Example,

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

 <title>Create a Frame</title>

</head>

<frameset rows="50%,50%">
```

```
<frame src="WebPage1.html" name="topPage" frameborder="0" border="0" />
<frame src="WebPage2.html" name="bottomPage" frameborder="0" border="0" />
</frameset>
</html>
```

Output :



5.2.3 Calling a child windows

- We'll begin with the simple task of calling a JavaScript function that is defined in another child window.
- You can refer to another child window by referencing the frameset, which is the parent window, and then by referencing the name of the child window, followed by whatever element within the web page of the child window that you want to access.
- Suppose that we modified WebPage1.html to include the following JavaScript function:

```
<head>
<title>Web Page 1</title>
```

```
<script language="Javascript" type="text/javascript">
<!--
function ChangeContent() {
alert("Function Called")
}
-->
</script>
</head>
```

- We'll also modify WebPage2.html to call the ChangeContent() function when the Web Page 2 button is clicked, which is shown next.
- Notice that we specified the parent (frameset) and the name of the child window (toPage) that contains the web page that defines the JavaScript ChangeContent() function.

  <INPUT name = "WebPage2" value = "Web Page 2" type = "button" onclick = "parent.topPage.ChangeContent()" />

  When the Web Page 2 button is clicked in the bottom child window, the browser calls the ChangeContent() function defined in the top child window, which displays an alert dialog box in the top child window.

- To call a JavaScript function in different frames, both pages have to be sourced from the same domain — otherwise, the browser throws a security alert and prevents it.
- If the pages are from different subdomains — for example, content1.jimkeogh.com and content2.jimkeogh.com — you can make it work as long as both pages are included in a JavaScript statement: document.domain = jimkeogh.com

  If you don't do it like this, you'll get a security alert.

5.2.4 Changing a content and focus of a child windows

- You can change the content of a child window from a JavaScript function by modifying the source web page for the child window.
- To do this, you must assign the new source to the child window's href attribute. In this example, you were able to get a reference to the parent frame's topPage element because they are both from the same domain. At that point, you have two options: if they're in the same domain,you reference it as illustrated previously, but you can also just change the frame src attribute in the frameset to point the frame to a new page.
- Let's do this in the following example. Again, we'll use the same frameset that we've been using throughout this chapter.
- However, we'll need to modify both the WebPage1.html and WebPage2.html files. In addition we'll need to define a new web page called WebPage3.html.
- Here is the new WebPage1.html file. WebPage1.html appears in the bottom child window, and when the Web Page 1 button is clicked, the content of the top child window changes from WebPage2.html to WebPage3.html.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Web Page 1</title>
<script language="Javascript" type="text/javascript">
<!--
function ChangeContent() {
parent.topPage.location.href = 'WebPage3.html'
}
-->
</script>
```

```
</head>
<body>
<FORM action="http://www.jimkeogh.com" method="post">
<P>
< INPUT name = "WebPage1" value = "Web Page 1" type = "button" onclick
    = "ChangeContent()" />
</P>
</FORM>
</body>
</html>
```

- We modified WebPage1.html in two ways: First, we defined the ChangeContent() function in the <head> tag.
- This function simply changes the value assigned to the href attribute to WebPage3.html. The original href was WebPage2.html, which is defined when we created the frameset.
- Notice that in order to change the href value, we need to reference the parent, the name of the child window, the location, and the href attribute.
- This tells the browser to go to the parent and then, within the parent, go to the topPage child window and change the source for that window.
- The following is WebPage2.html, which displays a button on the screen called Web Page 2 when the frameset is first shown on the screen. WebPage2.html is removed once the button on WebPage1.html is clicked.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Web Page 2</title>
```

```
</head>
<body>
<FORM action="http://www.jimkeogh.com" method="post">
<P>
<INPUT name="WebPage2" value="Web Page 2" type="button" />
</P>
</FORM>
</body>
</html>
```

The following is WebPage3.html, which displays a button on the screen called Web Page 3 after the button on WebPage1.html is clicked (Figure 11-4).

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Web Page 3</title>
</head>
<body>
<FORM action="http://www.jimkeogh.com" method="post">
<P>
<INPUT name= "WebPage3" value="Web Page 3" type="button" />
</P>
</FORM>
</body>
```

```
</html>
```

5.2.5 Writing to a child windows

- Typically, the content of a child window is a web page that exists on the web server.
- However, you can dynamically create the content when you defi ne the frameset by directly writing to the child window from a JavaScript.
- The JavaScript must be defi ned in the HTML fi le that defines the frameset and called when the frameset is loaded.
- This is illustrated in the next example, where the JavaScript function writes the content for the topPage child window, assuming the child is from the same domain:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
 <title>Create a Frame</title>
 <script language="Javascript" type="text/javascript">
 <!--
  function ChangeContent()
  {
  window.topPage.document.open()
  window.topPage.document.writeln(
'<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN">')
  window.topPage.document.writeln(
'<html xmlns="http://www.w3.org/1999/xhtml">')
   window.topPage.document.writeln('<head>')
```

```
 window.topPage.document.writeln( '<title>Web Page 3</title>')
 window.topPage.document.writeln('</head>')
window.topPage.document.writeln('<body>')
 window.topPage.document.writeln(
'<FORM action = "http://www.jimkeogh.com " method = "post" > ' )
 window.topPage.document.writeln('<P>')
 window.topPage.document.writeln(
'<INPUT name="WebPage3" value="Web Page 3" type = "button" />')
  window.topPage.document.writeln('</P>')
  window.topPage.document.writeln('</FORM>')
  window.topPage.document.writeln('</body>')
  window.topPage.document.writeln('</html>')
  window.topPage.document.close()
  }
 -->
 </script>
</head>
<frameset rows="50%,50%" onload="ChangeContent()">
 <frame src="WebPage1.html" name="topPage" />
 <frame src="WebPage2.html" name="bottomPage" />
</frameset>
</html>
```

- To write dynamic content to a child window, you must assign a source fi le to each frame of the frameset, even though you are dynamically creating the source for at least one of those frames.
- You'll notice in this example that WebPage1.html is assigned to the topPage frame. WebPage1.html must be a real fi le, although it won't appear in the topPage frame because the JavaScript function writes the content to that frame.
- The JavaScript function is defi ned in the <head> tag and is called when the onload event occurs. The topPage child window must be opened before the JavaScript function can write to the window. You open the child window by calling the open() method for that frame, as shown here:

```
window.topPage.document.open()
```

- Once opened, call the write() method to write HTML content to the child window to create the web page. This example displays the Web Page 3 button on a form. The final step is to call the close() method to close the window, as shown here:

```
window.topPage.document.close()
```

5.2.6 Accessing elements of another child window

- You can access and change the value of elements of another child window by directly referencing the element from within your JavaScript.
- You must explicitly specify the full path to the element in the JavaScript statement that references the element, and it must be from the same domain as the web page; otherwise, a security violation occurs.
- Let's see how this works. Suppose that a button named WebPage1 is on Form1, located on the web page that is displayed in the bottomPage frame of the frameset. The objective is to change the label of the Web Page 1 button.

- You'll need to specify the full path and then assign text to the value attribute of WebPage1, as shown here:

```
parent.topPage.Form1.WebPage1.value='New Label'
```

5.3 Rollover

- Rollovers are used to make a dreary web page come alive, by altering its appearance as the visitor scans the contents of the web page with the mouse.
- Any object on a web page can be changed with a rollover.
- Some web developers change an image that is related to the object beneath the mouse cursor. Other web developers pop up a new window that further describes the object. The only limitation is your imagination.

5.3.1 Creating rollover

- A rollover is caused by an event called onmouseover and occurs when a visitor to your web site moves the mouse over an object that appears on the page. An object can be an image, text, or any element of a form.
- You react to the onmouseover event by using the onmouseover attribute of an HTML tag that defines the object on the web page and then assign to the onmouseover attribute the action you want performed when the event occurs.
- The action can assign a new value to an attribute of an object, call a method of an object, or call a JavaScript function.
- Let's say that we want to change the image on the product page whenever the visitor moves the mouse cursor over the image.
- The <IMG> tag defines the image object. The value assigned to the src attribute of the <IMG> tag identifies the image itself. Whenever the onmouseover event occurs, we need to change the value of the src attribute to identify the new image. Here's how this is done:

<IMG height = "92" src = "7441805.gif" width = "70" border = "0" onmouseover = "src='0072253630.jpeg'">

- Here, the original image is the 7441805.gif file.
- The new image is the 0072253630.jpeg file.
- The onmouseover attribute is assigned the complete assignment statement (src='0072253630.jpeg'), which tells the browser to replace the 7441805.gif image with 0072253630.jpeg.

5.3.2 Text rollover

- You can create as many rollovers as you want on your web page; however, each one should be meaningful to the visitor.
- There is nothing more distracting to a visitor than to encounter rollovers on practically every object on a web page. Carefully placed rollovers can enhance a visitor's experience when browsing the web page.
- A clever rollover technique used by some developers is to enable a visitor to see additional information about an item described in text by placing the mouse cursor on the text.
- This eliminates the time-consuming task of using a hyperlink to display another web page that contains this additional information and reduces the information clutter found on some web pages.
- You create a rollover for text by using the onmouseover attribute of the <A> tag, which is the anchor tag. You assign the action to the onmouseover attribute the same way as you do with an <IMG> tag.
- Let's start a rollover project that displays a list of book titles. Additional information about a title can be displayed when the user rolls the mouse cursor over the book title.
- In this example, the cover of the book is displayed. However, you could replace the book cover with an advertisement or another message that you want to show about the book.

- One thing must be done; the onmouseover attribute must change the src attribute of the <IMG> tag. Therefore, the value assigned to the onmouseover attribute needs to identify explicitly the <IMG> tag that is being changed.
- To do this, we must give the <IMG> tag a unique name by assigning the name to the name attribute of the <IMG> tag. We can then reference the name in the value assigned to the onmouseover attribute of the text's <A> tag. The following segment shows how this is done.
- First, we give a name to the <IMG> tag. We'll call it cover.

```
<IMG height="92" src="7441805.gif" width="70" border="0" name="cover">
```

- Next, we reference the name cover in the src attribute to change the image that is assigned to the cover <IMG> tag.
- Notice that we use the complete document path, beginning with the document, then the object within the document (the <IMG> tag), and then the attribute of the object (src) that we're changing.

```
<A onmouseover="document.cover.src='7441805.gif'">
 <B><U>Java Demystified</U></B>
</A>
```

- The following web page displays three book titles and one book cover. The cover of the first book is shown when the page opens and is replaced with other covers as the mouse cursor is rolled over each corresponding title.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
 <title>Rollover Text</title>
</head>
<body>
```

```
<TABLE width="100%" border="0">
<TBODY>
<TR vAlign="top">
<TD width="50">
<a>
<IMG height="92" src="7441805.gif" width="70" border="0" name="cover">
</a>
</TD>
<TD>
<IMG height="1" src="" width="10">
</TD>
<TD>
<A onmouseover= "document.cover.src='7441805.gif'">
<B><U>Java Demystified</U></B>
</A>
<BR>
<A onmouseover= "document.cover.src='0072253630.jpeg'">
<B><U>OOP Demystified</U></B>
</A>
<BR>
<A onmouseover= "document.cover.src='7417436.gif'">
<B><U>Data Structures Demystified</U></B>
</A>
```

```
</TD>
</TR>
</TBODY>
</TABLE>
</body>
</html>
```

5.3.3 Multiple actions for rollover

- As you probably realize, you don't need JavaScript to use rollovers with your application, because you can react to an onmouseover event by directly assigning an action to the onmouseover attribute of an HTML tag. This direct method enables you to perform one action in response to an onmouseover event.
- However, you may find that you want more than one action to occur in response to an onmouseover event. To do this, you'll need to create a JavaScript function that is called by the onmouseover attribute when an onmouseover event happens.
- This JavaScript function is not much different from other JavaScript functions that you've created throughout this book, except this function is likely to have statements that manipulate objects on the page rather than perform calculations.
- Let's suppose a visitor rolls the cursor over a book title, as in the previous example. Instead of simply changing the image to reflect the cover of the book, you could

also display an advertisement for the book in a new window, encouraging the visitor to purchase the book (Figure 12-4).

- In this case, both the statement that changes the book cover and the statement that pops up the advertisement are contained in the JavaScript function, which is called by the onmouseover attribute of the text's anchor tag.
- The next example shows how this is done.

  First, we define the OpenNewWindow() JavaScript function in the <head> tag of the page.

  The OpenNewWindow() function has one argument, which is an integer called book that identifies the book title that the visitor selected.
- The function executes the appropriate statements depending on the book. Basically, the same three statements are executed for each book:

  • The appropriate cover is assigned the src attribute of the <IMG> tag.

  • A new window is opened by calling the window.open() method of the window object.

  • The advertisement is written to the new window using the window.write() method.

- These statements are slightly different for each book, of course, as each has a different cover, the window is positioned in a different place on the screen for each book, and the content written to the window is tailored to each book.
- We then define the rest of the page in the <BODY> tag. This is nearly identical to the preceding example, except the text reacts to two events—onmouseover and onmouseout—inside of one event.
- The onmouseover attribute responds to the onmouseover event by calling the OpenNewWindow() JavaScript function and passing it an integer that identifies the book. The onmouseout attribute reacts to the mouse cursor rolling off (onmouseout event) the text by calling the close() method of the window object,which closes the newly opened window so we don't clutter the screen with windows.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
 <title>Open Window</title>
 <script language="Javascript" type="text/javascript">
 <!--
 function OpenNewWindow(book) {
 if (book== 1)
 {
 document.cover.src='7441805.gif'
   MyWindow = window.open( '', 'myAdWin', 'titlebar=0 status=0, toolbar=0, location=0, menubar = 0, directories = 0, resizable=0, height=50, width=150,left=500, top=400')
 MyWindow.document.write( '10% Discount for Java Demystified!')
 }
 if (book== 2)
 {
 document.cover.src='0072253630.jpeg'
  MyWindow = window.open('', 'myAdWin', 'titlebar = "0" status= "0", toolbar = "0", location = "0", menubar = "0", directories = "0", resizable= "0", height = "50", width = "150", left = "500", top = "500"')
 MyWindow.document.write( '20% Discount for OOP Demystified!')
 }
 if (book== 3)
```

```
{
document.cover.src= '7417436.gif'
 MyWindow = window.open('', 'myAdWin', 'titlebar="0" status="0", toolbar="0", location="0", menubar ="0", directories="0", resizable = "0", height= "50", width = "150",
left ="500",top = "600"')
MyWindow.document.write('15% Discount for Data Structures Demystified!')
}
}
-->
</script>
</head>
<body>
<TABLE width="100%" border="0">
<TBODY>
<TR vAlign="top">
<TD width="50">
<a>
<IMG height="92" src="7441805.gif" width="70" border="0" name="cover">
</a>
</TD>
<TD>
```

```
<IMG height="1" src ="" width="10">
</TD>
<TD>
<A onmouseover = "OpenNewWindow(1)" onmouseout = "MyWindow.close()" >
<B><U>Java Demystified </U></B>
</A>
<BR>
<A onmouseover ="OpenNewWindow(2)" onmouseout = "MyWindow.close()">
<B><U>OOP Demystified</U></B>
</A>
<BR>
<A onmouseover = "OpenNewWindow(3)" onmouseout = "MyWindow.close()" >
<B><U>Data Structures Demystified</U></B>
</A>
</TD>
</TR>
</TBODY>
</TABLE>
</body>
</html>
```

5.3.4 More efficient rollover

- An efficient way of handling rollovers is to load images into an array when your web page loads. The browser loads each image once the first time the image is referenced in the web page.
- Typically, the default setting for the browser is to check the browser cache for subsequent references for the image rather than download the image again from the web server.
- However, a visitor to your web page might have changed the default setting, causing the browser to reload the image each time the image is referenced. This might cause a noticeable delay.
- Any delay in transmission is likely to be noticed by the visitor. While most visitors accept short delays when they're selecting a different web page, they tend to be unforgiving if the rollover takes longer than a second or two to display the new image.
- You can reduce this delay by creating a JavaScript that loads all the images into memory once at the beginning of the JavaScript, where they can be quickly called upon as the onmouseover event occurs.
- Downloading images when the web page is fi rst loaded is a simple three-step process:

    1. Declare an image object.
    2. Assign the image fi le to the image object.
    3. Assign the image object to the src attribute of the HTML tag that is going to react to the rollover event.

  The following example shows how this is done.

- Notice that the IMG object is declared and assigned an image in the if statement and that the IMG objects are assigned to null if the browser doesn't support rollovers.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
 <title>More Efficient Rollover</title>
 <script language="Javascript" type="text/javascript">
 <!--
 JavaDemystified = new Image
 OOPDemystified = new Image
 DataStructuresDemystified = new Image
 if (document.images) {
 JavaDemystified.src = '7441805.gif'
 OOPDemystified.src = '0072253630.jpeg'
DataStructuresDemystified.src = '7417436.gif'
 }
 else {
 JavaDemystified.src = ''
 OOPDemystified.src = ''
 DataStructuresDemystified.src = ''
document.cover = ''
}
-->
</script>
</head>
```

```
<body>
 <TABLE width="100%" border=0>
 <TBODY>
 <TR vAlign ="top">
 <TD width="50">
 <a>
 <IMG height="92" src="7441805.gif" width="70" border="0" name= "cover" >
 </a>
 </TD>
 <TD>
 <IMG height="1" src ="" width="10">
 </TD>
 <TD>
 <A onmouseover = "document.cover.src = JavaDemystified.src">
 <B><U>Java Demystified </U></B>
 </A>
 <BR>
 <A onmouseover = "document.cover.src = OOPDemystified.src">
    <B><U>OOP Demystified</U></B>
    </A>
    <BR>
    <A onmouseover = "document.cover.src= DataStructuresDemystified.src">
    <B><U>Data Structures Demystified</U></B>
```

```
</A>
</TD>
</TR>
</TBODY>
</TABLE>
</body>
</html>
```