8

JavaScript: Control Statements II

Outline

8.2 Essentials of Counter-Controlled Repetition

- **Counter-controlled repetition requires**
  - **name of a control variable**
  - **initial value of the control variable**
  - **the increment (or decrement) by which the control variable is modified each time through the loop**
  - **the condition that tests for the final value of the control variable to determine whether looping should continue**

Fig. 8.1 | Counter-controlled repetition (Part 1 of 2).

```
1  <?xml version = "1.0" encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 8.1: WhileCounter.html -->
6  <!-- Counter-controlled repetition. -->
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8     <head>
9        <title>Counter-Controlled Repetition</title>
10       <script type = "text/javascript">
11          <!--
12          var counter = 1; // initialization
13
14          while ( counter <= 7 ) // repetition condition
15          {
16             document.writeln( "<p style = \"font-size: " +
17                counter + "ex\">XHTML font size " + counter +
18                "ex</p>" );
19             ++counter; // increment
20          } //end while
21          // -->
22       </script>
23    </head><body></body>
24 </html>
```

Initializes `counter`

Precedes the " with a \ to create an escape sequence so that it can be used in the string

Condition to be fulfilled with every iteration

Incrementing statement

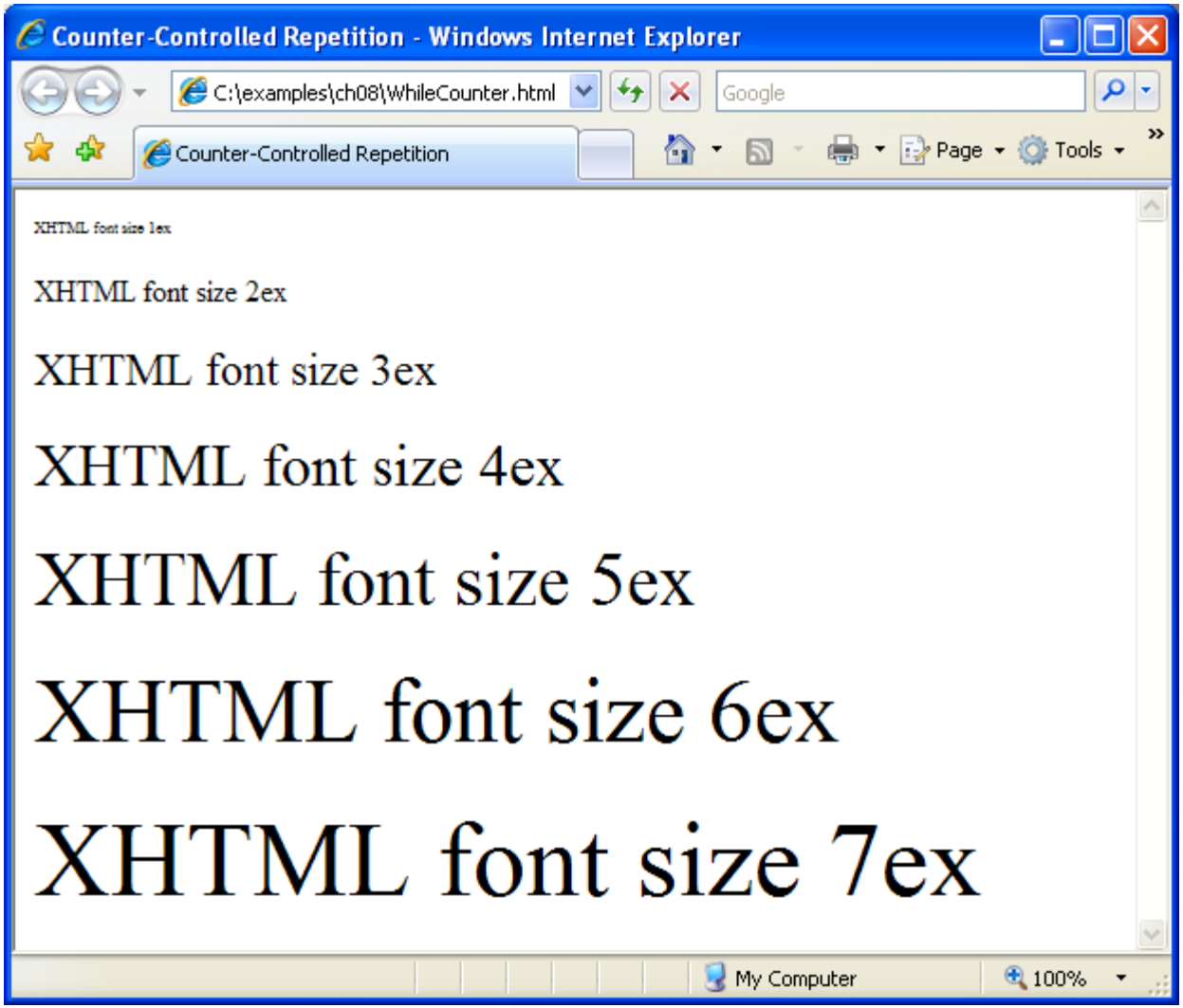


Fig. 8.1 | Counter-controlled repetition (Part 2 of 2).

8.3 for Repetition Statement

- for statement
  - Cpecifies each of the items needed for counter-controlled repetition with a control variable
  - Can use a block to put multiple statements into the body
- If the loop's condition uses a < or > instead of a <= or >=, or vice-versa, it can result in an off-by-one error
- for statement takes three expressions
  - Initialization
  - Condition
  - Increment Expression
- The increment expression in the for statement acts like a stand-alone statement at the end of the body of the for statement
- Place only expressions involving the control variable in the initialization and increment sections of a for statement

Fig. 8.2 | Counter-controlled repetition with the for statement (Part 1 of 2).

```
1  <?xml version = "1.0" encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 8.2: ForCounter.html -->
6  <!-- Counter-controlled repetition with the for statement. -->
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8     <head>
9        <title>Counter-Controlled Repetition</title>
10       <script type = "text/javascript">
11          <!--
12          // Initialization, repetition condition and
13          // incrementing are all included in the for
14          // statement header.
15          for ( var counter = 1; counter <= 7; ++counter )
16             document.writeln( "<p style = \"font-size: " +
17                counter + "ex\">XHTML font size " + counter +
18                "ex</p>" );
19          // -->
20       </script>
21    </head><body></body>
22 </html>
```

Initial value of the control variable

Condition to test whether looping should continue

Increment to occur after each iteration of the loop

Statement inside the `for` loop

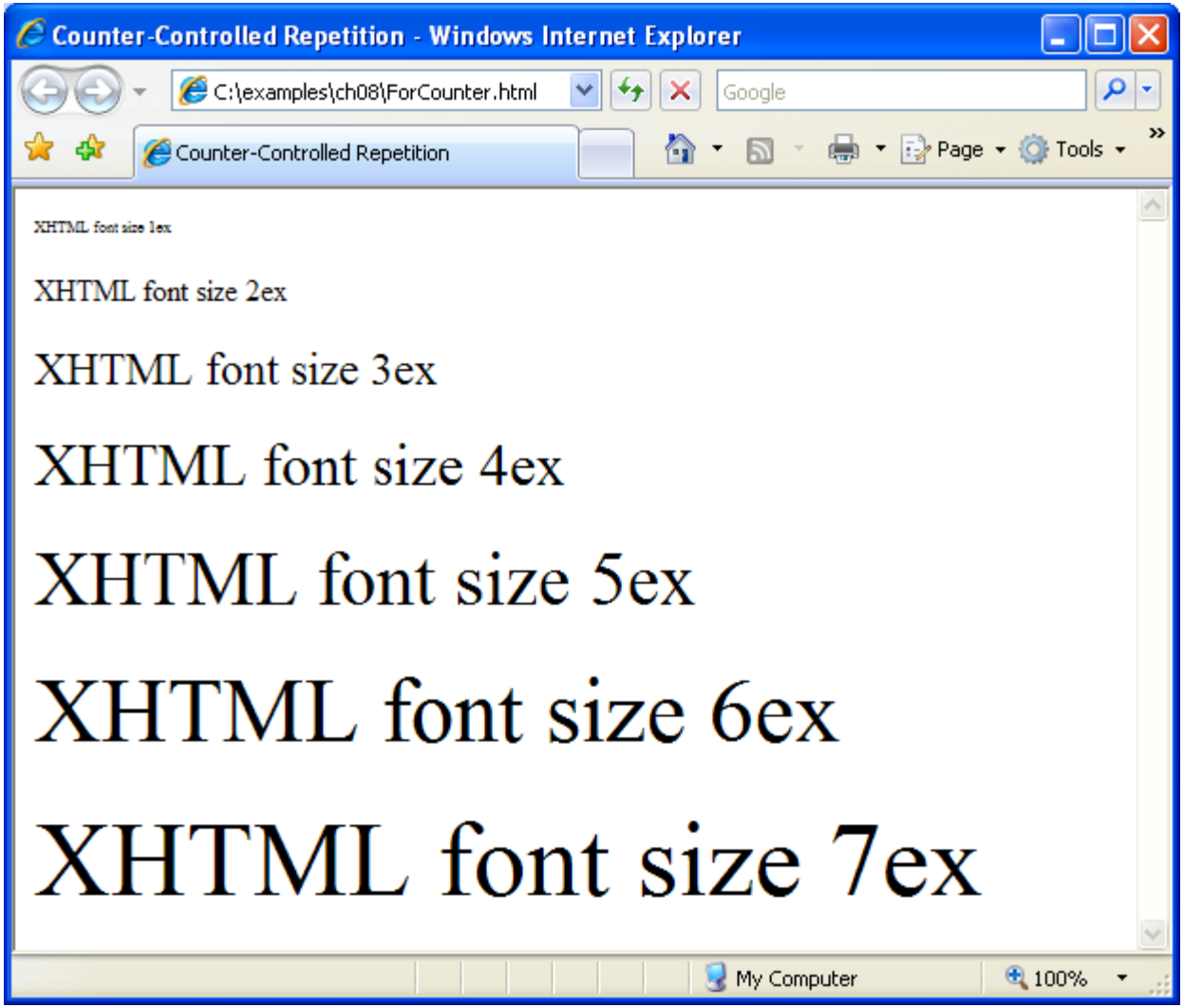


Fig. 8.2 | Counter-controlled repetition with the **`for`** statement (Part 2 of 2).

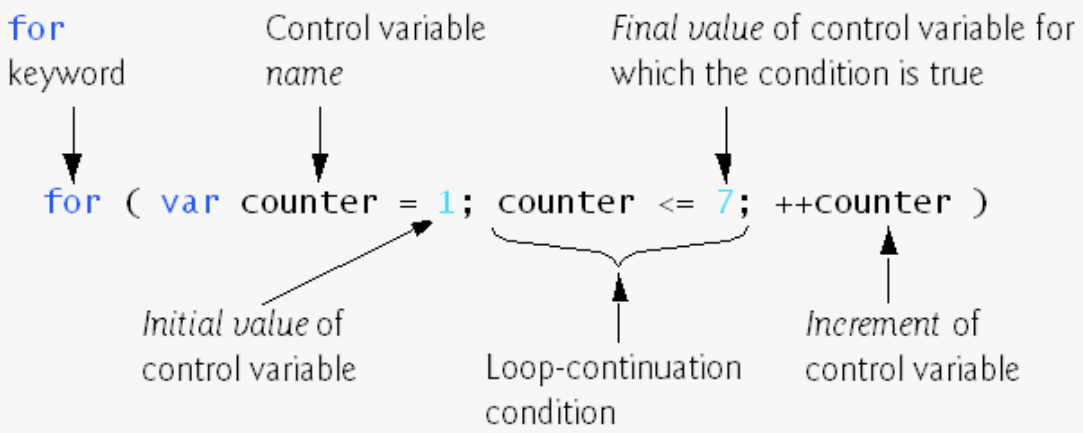


Fig. 8.3 | `for` statement header components.

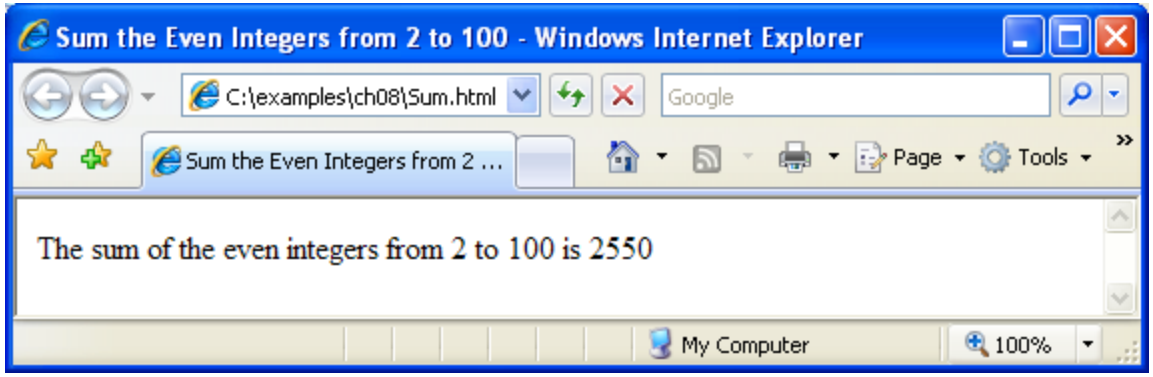
Fig. 8.5 | Summation with the for repetition structure.

```
1  <?xml version = "1.0" encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 8.5: Sum.html -->
6  <!-- Summation with the for repetition structure. -->
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8     <head>
9        <title>Sum the Even Integers from 2 to 100</title>
10       <script type = "text/javascript">
11          <!--
12          var sum = 0;
13
14          for ( var number = 2; number <= 100; number += 2 )
15             sum += number;
16
17          document.writeln( "The sum of the even integers " +
18             "from 2 to 100 is " + sum );
19          // -->
20       </script>
21    </head><body></body>
22 </html>
```

Control variable `number` begins at the value of 2

We execute the loop while `number` is less than or equal to 100

After each loop iteration is complete, increment `number` by 2



```
1  <?xml version = "1.0" encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 8.6: Interest.html -->
6  <!-- Compound interest calculation with a for loop. -->
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8     <head>
9        <title>Calculating Compound Interest</title>
10       <style type = "text/css">
11          table { width: 100% }
12          th    { text-align: left }
13       </style>
14       <script type = "text/javascript">
15          <!--
16          var amount; // current amount of money
17          var principal = 1000.0; // principal amount
18          var rate = .05; // interest rate
19
20          document.writeln(
21             "<table border = \"1\">" ); // begin the table
22          document.writeln(
23             "<caption>Calculating Compound Interest</caption>" );
24          document.writeln(
25             "<thead><tr><th>Year</th>" ); // year column heading
26          document.writeln(
27             "<th>Amount on deposit</th>" ); // amount column heading
28          document.writeln( "</tr></thead><tbody>" );
29
```

Fig. 8.6 | Compound interest calculation with a **for** loop (Part 1 of 2).

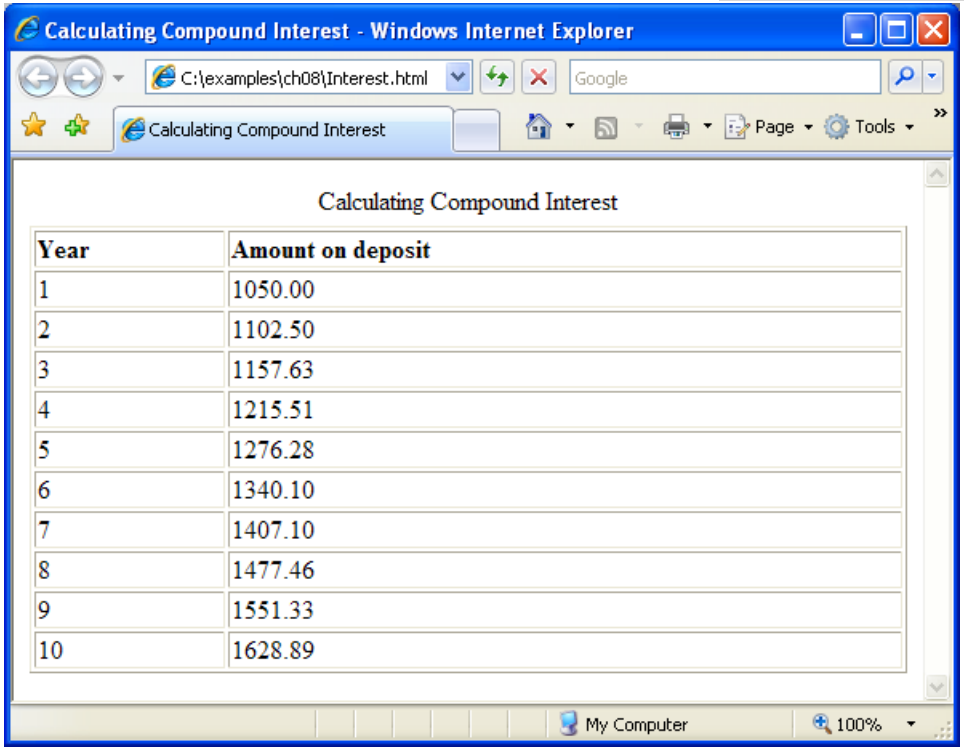
```
30          // output a table row for each year
31          for ( var year = 1; year <= 10; ++year )
32          {
33             amount = principal * Math.pow( 1.0 + rate, year );
34             document.writeln( "<tr><td>" + year +
35                "</td><td>" + amount.toFixed(2) +
36                "</td></tr>" );
37          } //end for
38
39          document.writeln( "</tbody></table>" );
40          // -->
41       </script>
42    </head><body></body>
43 </html>
```

Control variable `year` begins with a value of 1

Continue to execute the loop while `year` is less than or equal to 10

After each loop iteration, increase the value of `year` by 1

Fig. 8.6 | Compound interest calculation with a for loop (Part



8.5 switch Multiple-Selection Statement

- switch statement
  - Consists of a series of case labels and an optional default case
  - When control reaches a switch statement
    - The script evaluates the controlling expression in the parentheses
    - Compares this value with the value in each of the case labels
    - If the comparison evaluates to true, the statements after the case label are executed in order until a break statement is reached
- The break statement is used as the last statement in each case to exit the switch statement immediately
- The default case allows you to specify a set of statements to execute if no other case is satisfied
  - Usually the last case in the switch statement

8.5 `switch` Multiple-Selection Statement (Cont.)

- Each case can have multiple actions (statements)
- Braces are not required around multiple actions in a case of a `switch`
- The `break` statement is not required for the last case because program control automatically continues with the next statement after the `switch`
- Having several case labels listed together (e.g., `case 1: case 2:` with no statements between the cases) executes the same set of actions for each case

Fig. 8.7 | Using the switch multiple-selection statement (Part 1 of 4).

```
1  <?xml version = "1.0" encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 8.7: SwitchTest.html -->
6  <!-- Using the switch multiple-selection statement. -->
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8     <head>
9        <title>Switching between XHTML List Formats</title>
10       <script type = "text/javascript">
11          <!--
12          var choice; // user's choice
13          var startTag; // starting list item tag
14          var endTag; // ending list item tag
15          var validInput = true; // indicates if input is valid
16          var listType; // type of list as a string
17
18          choice = window.prompt( "Select a list style:\
19             "1 (numbered), 2 (lettered), 3 (roman)", "1" );
20
21          switch ( choice )
22          {
23             case "1":
24                startTag = "<ol>";
25                endTag = "</ol>";
26                listType = "<h1>Numbered List</h1>";
27                break;
28             case "2":
29                startTag = "<ol style = \"list-style-type: upper-
30                endTag = "</ol>";
31                listType = "<h1>Lettered List</h1>";
32                break;
```

Beginning of `switch` statement

Beginning of statements to be executed if `choice` equals "1"

Statements

Break out of `switch` statement

Beginning of statements to be executed if `choice` equals "2"

Statements

Break out of `switch` statement

8.7 | Using switch selection statement (Part 4 of 4).

```
33            case "3":
34               startTag = "<ol style = \"list-sty
35               endTag = "</ol>";
36               listType = "<h1>Roman Numbered Lis
37               break;
38            default:
39               validInput = false;
40         } //end switch
41
42         if ( validInput == true )
43         {
44            document.writeln( listType + startTag );
45
46            for ( var i = 1; i <= 3; ++i )
47               document.writeln( "<li>List item " + i + "</li>" );
48
49            document.writeln( endTag );
50         } //end if
51         else
52            document.writeln( "Invalid choice: " + choice );
53         // -->
54      </script>
55   </head>
56   <body>
57      <p>Click Refresh (or Reload) to run the script again</p>
58   </body>
59 </html>
```

Beginning of statements to be execut… quals “3”

Statements

Break out of `switch` statement

Beginning of statements to be executed if c… ng other than “1”, “2” or “3”

Statement

No break is necessary, since we’ve come to the end of the `switch` anyway

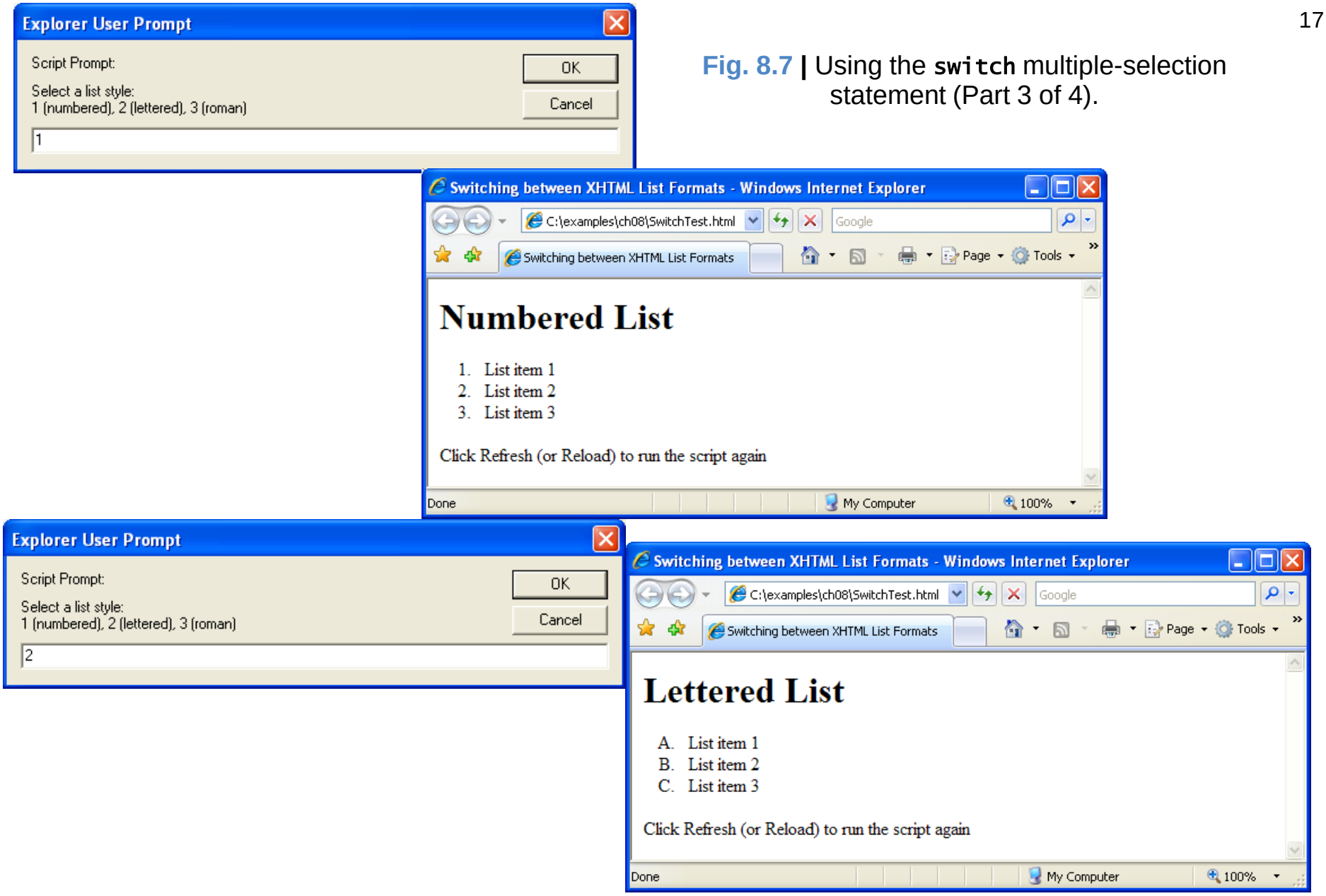


Fig. 8.7 | Using the `switch` multiple-selection statement (Part 3 of 4).

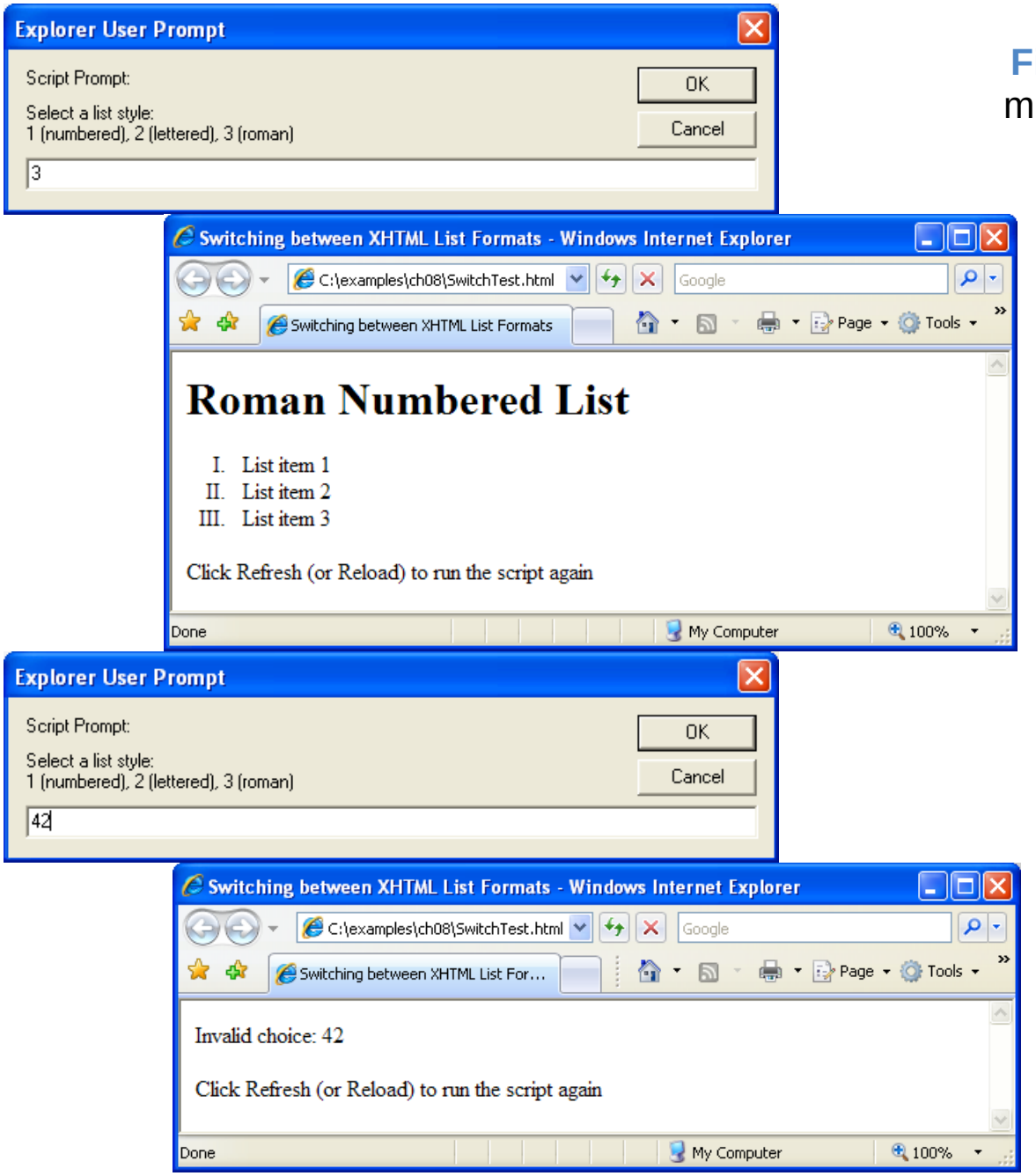


Fig. 8.7 | Using the `switch` multiple-selection statement (Part 4 of 4).

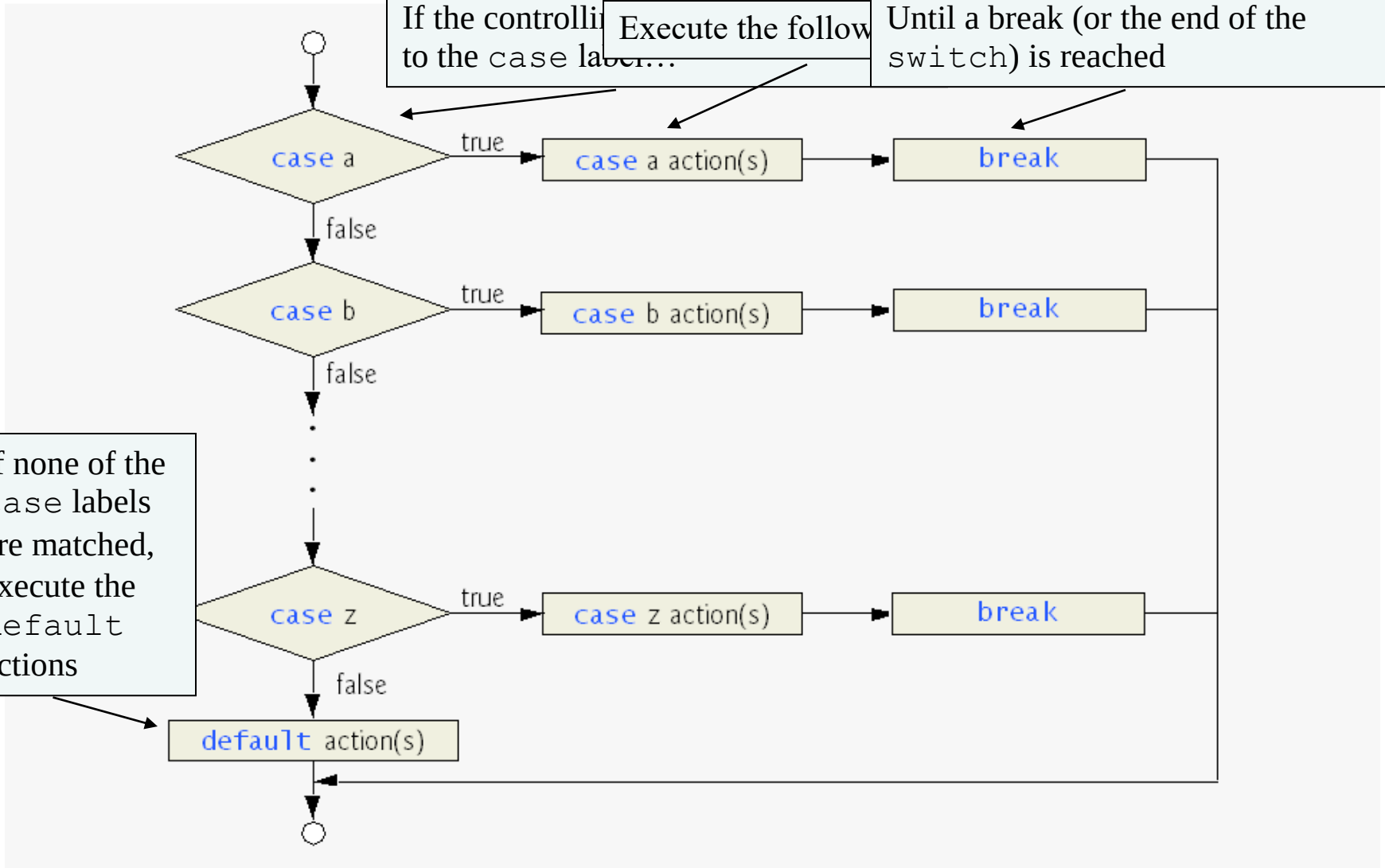


Fig. 8.8 | `switch` multiple-selection statement.

8.6 do…while Repetition Statement

- **do…while statement**
  - **tests the loop-continuation condition *after* the loop body executes**
  - ***The loop body always executes at least once***

Fig. 8.9 | Using the do while repetition statement (Part 1 of 2).

```
1  <?xml version = "1.0" encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 8.9: DoWhileTest.html -->
6  <!-- Using the do...while repetition statement. -->
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8     <head>
9        <title>Using the do...while Repetition Statement</title>
10       <script type = "text/javascript">
11          <!--
12          var counter = 1;
13
14          do {
15             document.writeln( "<h" + counter + ">This is " +
16                "an h" + counter + " level head" + "</h" +
17                counter + ">" );
18             ++counter;
19          } while ( counter <= 6 );
20          // -->
21       </script>
22
23    </head><body></body>
24 </html>
```

Perform the following actions…

Then check to see if `counter <= 6`. If it is, iterate through the loop again.

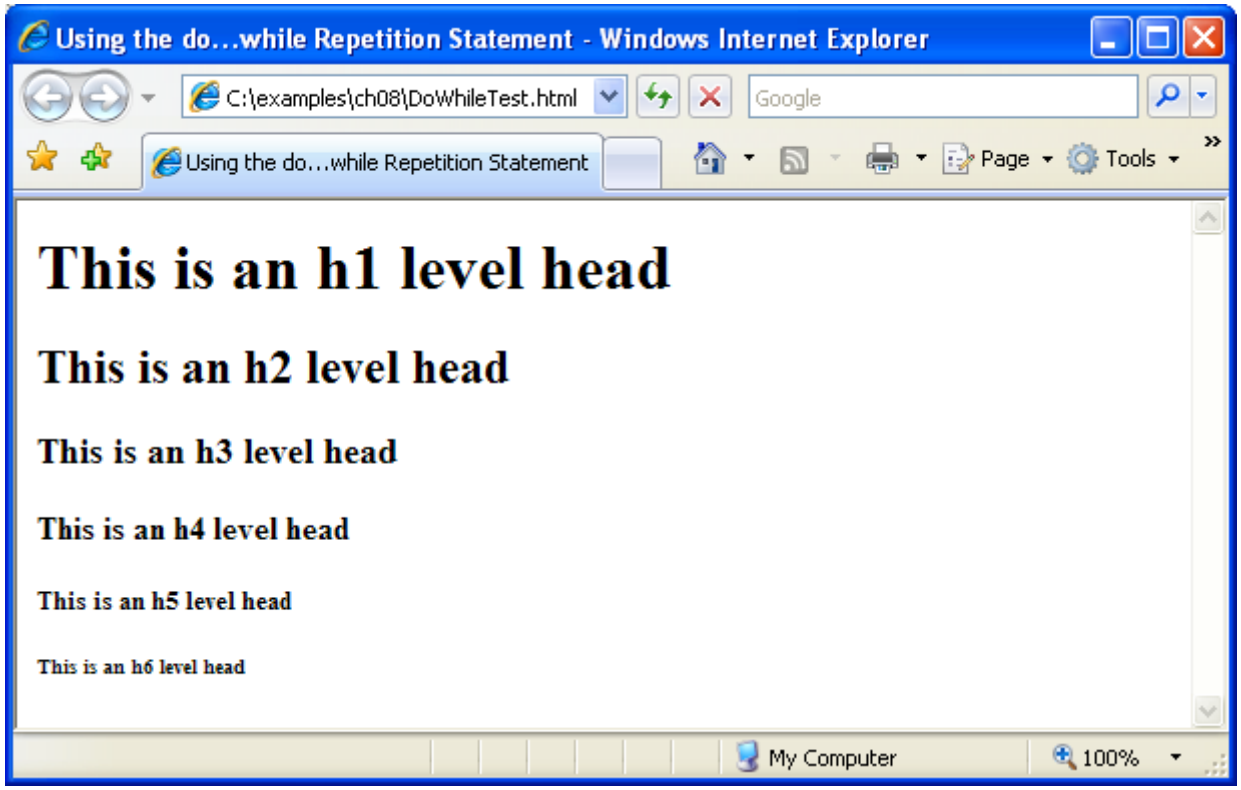


Fig. 8.9 | Using the `do while` repetition statement (Part 2 of 2).

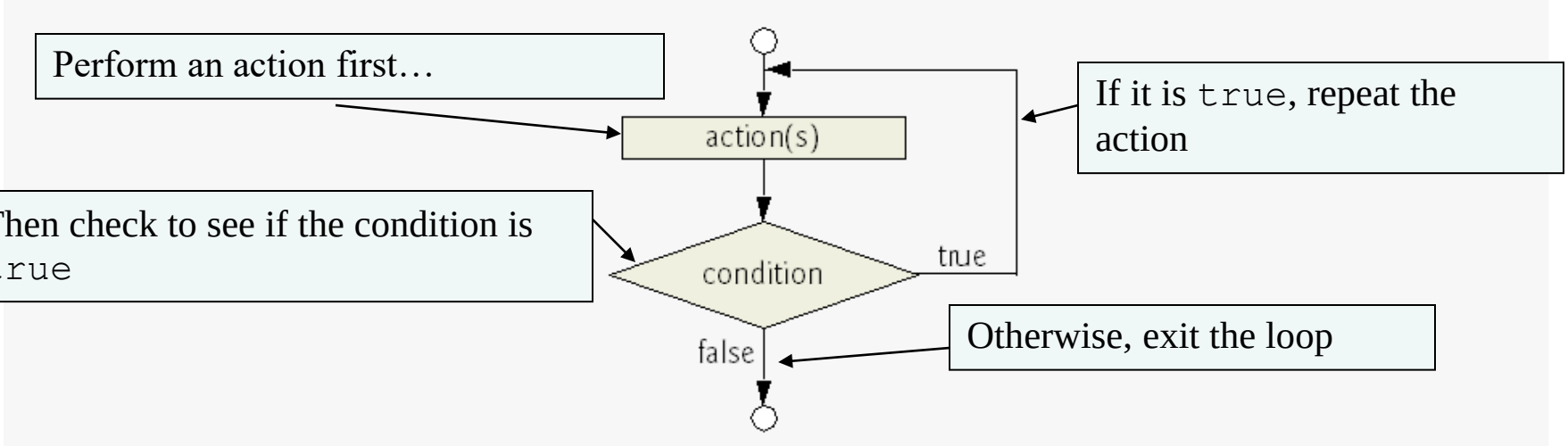


Fig. 8.10 | `do while` repetition statement flowchart.

8.7 break and continue Statements

- **break statement in a while, for, do…while or switch statement**
  - **Causes immediate exit from the statement**
  - **Execution continues with the next statement in sequence**
- **break statement common uses**
  - **Escape early from a loop**
  - **Skip the remainder of a switch statement**

8.7 break and continue Statements (Cont.)

- continue statement in a while, for or do…while
  - skips the remaining statements in the body of the statement and proceeds with the next iteration of the loop
  - In while and do…while statements, the loop-continuation test evaluates immediately after the continue statement executes
  - In for statements, the increment expression executes, then the loop-continuation test evaluates

```
1  <?xml version = "1.0" encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 8.11: BreakTest.html -->
6  <!-- Using the break statement in a for statement. -->
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8     <head>
9        <title>
10          Using the break Statement in a for Statement
11       </title>
12       <script type = "text/javascript">
13          <!--
14          for ( var count = 1; count <= 10; ++count )
15          {
16             if ( count == 5 )
17                break; // break loop only if count == 5
18
19             document.writeln( "Count is: " + count + "<br />" );
20          } //end for
21
22          document.writeln(
23             "Broke out of loop at count = " + count );
24          // -->
25       </script>
26    </head><body></body>
27 </html>
```

Exits the `for` loop immediately if `count == 5`

Fig. 8.11 | Using the **break** statement in a **for** statement (Part 1 of 2).

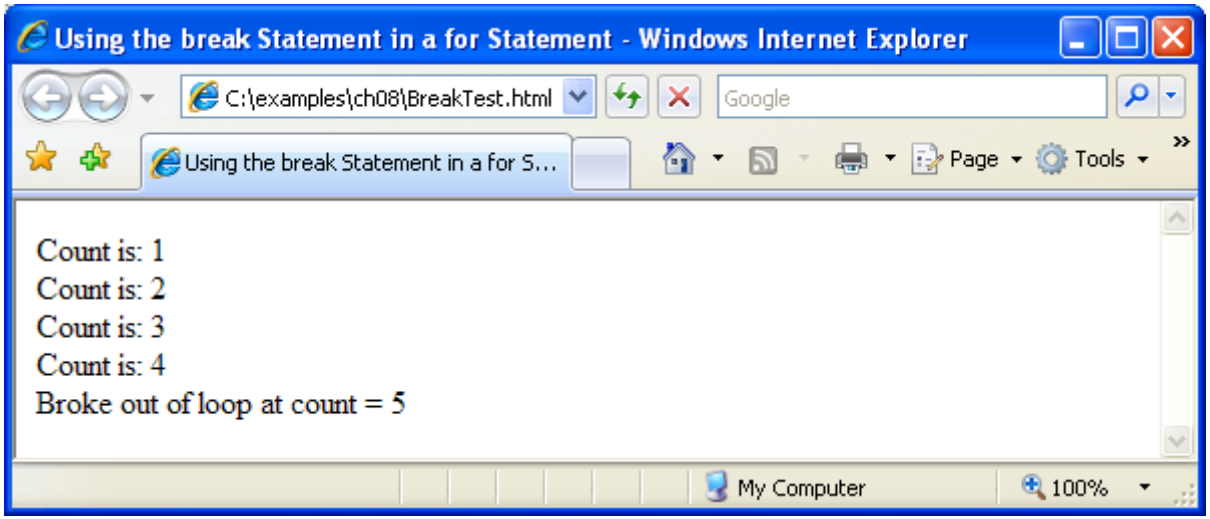


Fig. 8.11 | Using the **`break`** statement in a **`for`** statement (Part 2 of 2).

Fig. 8.12 | Using the **continue** statement in a **for** statement (Part 1 of 2).

```
1  <?xml version = "1.0" encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 8.12: ContinueTest.html -->
6  <!-- Using the continue statement in a for statement. -->
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8     <head>
9        <title>
10          Using the continue Statement in a for Statement
11       </title>
12
13       <script type = "text/javascript">
14          <!--
15          for ( var count = 1; count <= 10; ++count )
16          {
17             if ( count == 5 )
18                continue; // skip remaining loop code only if count == 5
19
20             document.writeln( "Count is: " + count + "<br />" );
21          } //end for
22
23          document.writeln( "Used continue to skip printing 5" );
24          // -->
25       </script>
26
27    </head><body></body>
28 </html>
```

If `count == 5`, skips the rest of the statements in the loop, increments `count`, and performs the loop-continuation test

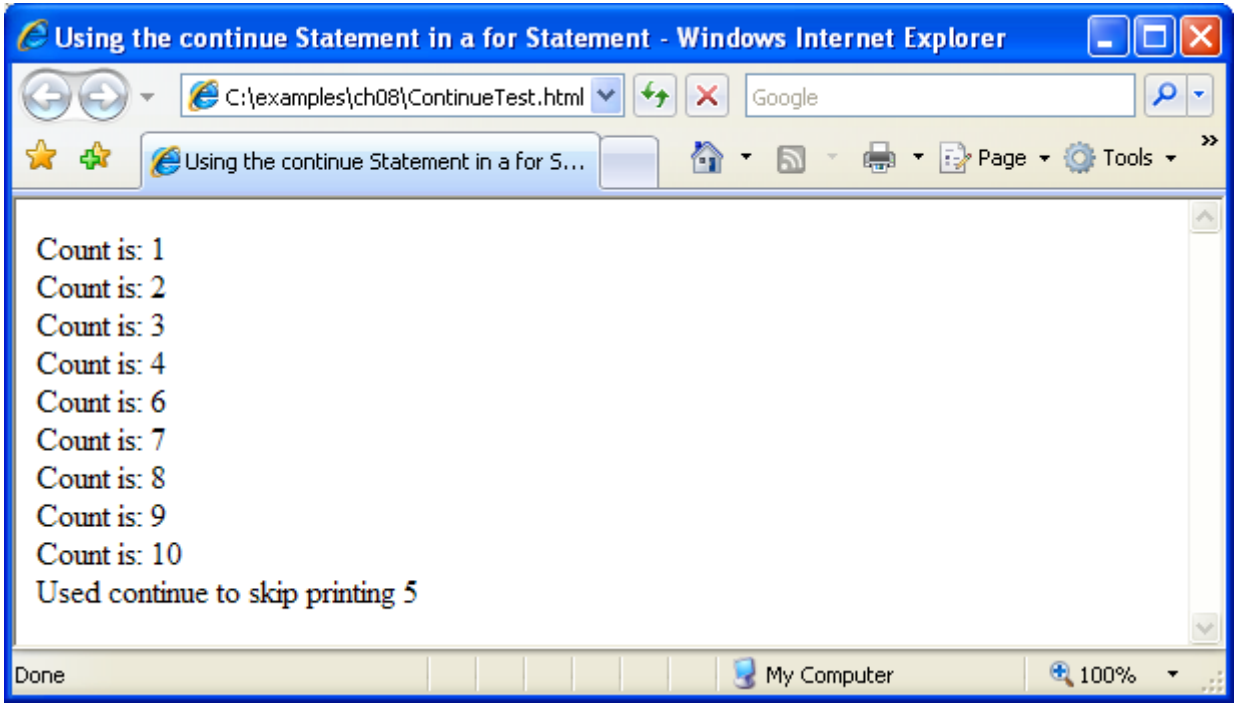


Fig. 8.12 | Using the **`continue`** statement in a **`for`** statement (Part 2 of 2).

8.8 Labeled `break` and `continue` Statements

- **To break out of a nested control statement**
  - **Use the labeled `break` statement**
  - **When executed in a `while`, `for`, `do…while` or `switch` statement, causes immediate exit from that statement and any number of enclosing repetition statements**
  - **Program execution resumes with the first statement after the specified labeled statement (a statement preceded by a label)**
- **A labeled statement can be a block (a set of statements enclosed in curly braces, `{}`)**
- **Commonly are used to terminate nested looping structures containing `while`, `for`, `do…while` or `switch` statements**

Fig. 8.13 | Labeled `break` statement in a nested `for` statement (Part 1 of 2).

```
1  <?xml version = "1.0" encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 8.13: BreakLabelTest.html -->
6  <!-- Labeled break statement in a nested for statement. -->
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8     <head>
9        <title>Using the break Statement with a Label</title>
10       <script type = "text/javascript">
11          <!--
12          stop: { // labeled block
13             for ( var row = 1; row <= 10; ++row )
14             {
15                for ( var column = 1; column <= 5 ; ++column )
16                {
17                   if ( row == 5 )
18                      break stop; // jump to end of stop block
19
20                   document.write( "* " );
21                } //end for
22
23                document.writeln( "<br />" );
24             } //end for
25
26             // the following line is skipped
27             document.writeln( "This line should not print" );
28          } // end block labeled stop
29
```

Statement label

Beginning of labeled statement

If `row == 5`, immediately go to the end of the `stop` block

End of labeled statement

```
30          document.writeln( "End of script" );
31          // -->
32       </script>
33    </head><body></body>
34 </html>
```

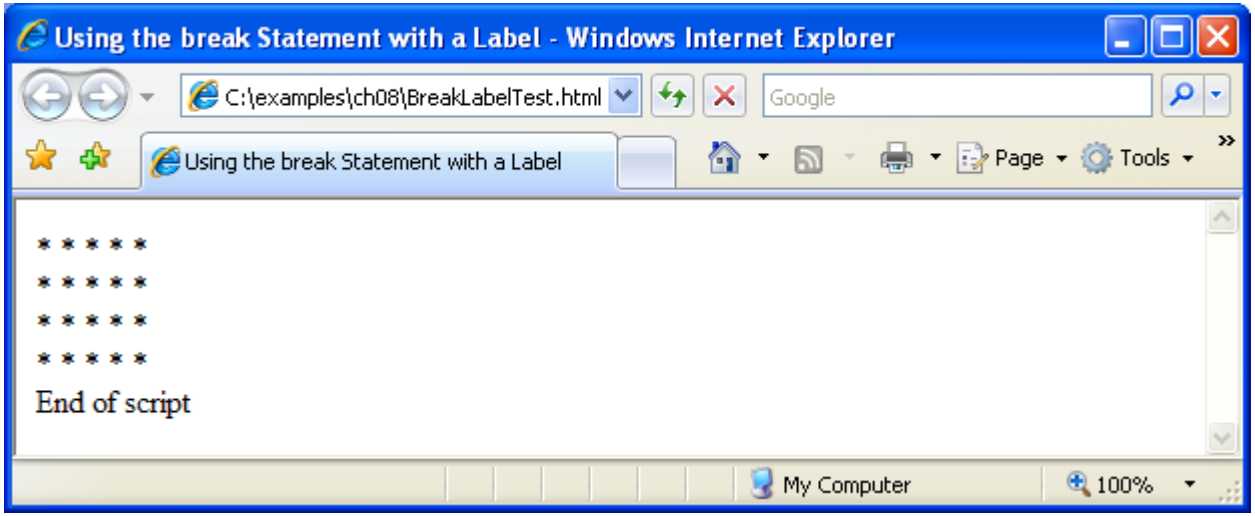


Fig. 8.13 | Labeled `break` statement in a nested `for` statement (Part 2 of 2).

8.8 Labeled `break` and `continue` Statements (Cont.)

- **Labeled `continue` statement**
  - **When executed in a repetition statement (`while`, `for` or `do…while`), skips the remaining statements in the structure's body and any number of enclosing repetition statements**
  - **Proceeds with the next iteration of the specified labeled repetition statement (a repetition statement preceded by a label)**
  - **In labeled `while` and `do…while` statements, the loop-continuation test evaluates immediately after the `continue` statement executes**
  - **In a labeled `for` statement, the increment expression executes, then the loop-continuation test evaluates**

Fig. 8.14 | Labeled `continue` statement in a nested `for` statement (Part 1 of 2).

```
1  <?xml version = "1.0" encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 8.14: ContinueLabelTest.html -->
6  <!-- Labeled continue statement in a nested for statement. -->
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8     <head>
9        <title>Using the continue Statement with a Label</title>
10       <script type = "text/javascript">
11          <!--
12          nextRow: // target label of continue statement
13             for ( var row = 1; row <= 5; ++row )
14             {
15                document.writeln( "<br />" );
16
17                for ( var column = 1; column <= 10; ++column )
18                {
19                   if ( column > row )
20                      continue nextRow; // next iteration of labeled loop
21
22                   document.write( "* " );
23                } //end for
24             } //end for
25          // -->
26       </script>
27    </head><body></body>
28 </html>
```

Statement label

Beginning of labeled statement

If `column > row`, skip all remaining statements in the `nextRow` block, perform the increment expression, then evaluate the loop-continuation test

End of labeled statement

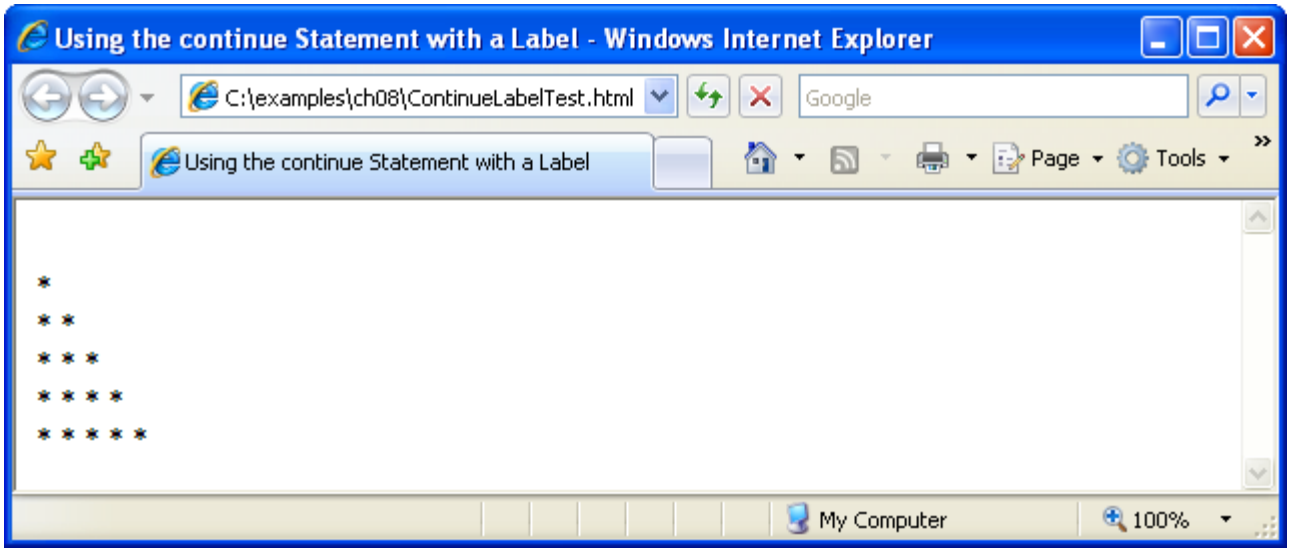


Fig. 8.14 | Labeled **`continue`** statement in a nested **`for`** statement (Part 2 of 2).

8.9 Logical Operators

- **Logical operators can be used to form complex conditions by combining simple conditions**
  - **&& (logical AND)**
  - **|| (logical OR)**
  - **! (logical NOT, also called logical negation)**
- **The && operator is used to ensure that two conditions are both true before choosing a certain path of execution**
- **JavaScript evaluates to `false` or `true` all expressions that include relational operators, equality operators and/or logical operators**

Fig. 8.18 | Demonstrating logical operators (Part 1 of 2).

```
1  <?xml version = "1.0" encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 8.18: LogicalOperators.html -->
6  <!-- Demonstrating logical operators. -->
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8     <head>
9        <title>Demonstrating the Logical Operators</title>
10       <style type = "text/css">
11          table   { width: 100% }
12          td.left { width: 25% }
13       </style>
14       <script type = "text/javascript">
15          <!--
16          document.writeln(
17             "<table border = \"1\"" );
18          document.writeln(
19             "<caption>Demonstrating Logical " +
20                "Operators</caption>" );
21          document.writeln(
22             "<tr><td class = \"left\">Logical AND (&&)</td>" +
23             "<td>false && false: " + ( false && false ) +
24             "<br />false && true: " + ( false && true ) +
25             "<br />true && false: " + ( true && false ) +
26             "<br />true && true: " + ( true && true ) +
27             "</td></tr>" );
```

Generates a truth table for the logical AND operator

```
28          document.writeln(
29             "<tr><td class = \"left\">Logical OR (||)</td>" +
30             "<td>false || false: " + ( false || false ) +
31             "<br />false || true: " + ( false || true ) +
32             "<br />true || false: " + ( true || false ) +
33             "<br />true || true: " + ( true || true ) +
34             "</td></tr>" );
35          document.writeln(
36             "<tr><td class = \"left\">Logical NOT (!)</td>" +
37             "<td>!false: " + ( !false ) +
38             "<br />!true: " + ( !true ) + "</td></tr>" );
39          document.writeln( "</table>" );
40          // -->
41       </script>
42    </head><body></body>
43 </html>
```

Fig. 8.18 | Demonstrating logical operators (Part 2 of 2).

Generates a truth table for the logical OR operator

Generates a truth table for the logical NOT operator