Internet Computing for Managers

Lesson 2
Vincent Ng
Department of Computing
• Simple Program: Printing a Line of Text in a Web Page
• A second JavaScript Program: Adding Integers
• Arithmetic
• Decision Making: Equality and Relational Operators
• JavaScript Internet and World Wide Web Resources
• Control structures
Welcome.html

The `script` tag indicates to the browser that the text which follows is part of a script.

The `document` object’s `writeln` method writes a line of XHTML markup in the XHTML document.
Welcome to JavaScript Programming!
<html>
  <head>
    <title>Printing a Line with Multiple Statements</title>
  </head>
  <body>
    <script type = "text/javascript">
      <!--
      document.write( "<h1 style = "color: magenta">" );
      document.write( "Welcome to JavaScript " +
      "Programming!</h1>" );
      // -->
    </script>
    <head><body></body></head>
  </body>
</html>

Using the **style** attribute, the color of the text is changed from black to magenta.

The escape sequence `\"` places a quote in the string and is not displayed in the browser.
Welcome to JavaScript Programming!
Using `break` tags, the text is displayed as three lines.
Welcome to JavaScript Programming!
<html>
<head>
<title>Printing Multiple Lines in a Dialog Box</title>
<script type = "text/javascript">
<!--
window.alert( "Welcome to\nJavaScript\nProgramming!" );
// -->
</script>
</head>
<body>
<p>Click Refresh (or Reload) to run this script again.</p>
</body>
</html>

The <code>window</code> method <code>alert</code> displays an alert dialog to the user.
The **OK** button allows the user to dismiss (or hide) the dialog.

The dialog is automatically sized to accommodate the string.

Click Refresh (or Reload) to run this script again.
Printing a Line of Text in a Web Page

<table>
<thead>
<tr>
<th>Escape sequence</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\n</td>
<td>Newline. Position the screen cursor at the beginning of the next line.</td>
</tr>
<tr>
<td>\t</td>
<td>Horizontal tab. Move the screen cursor to the next tab stop.</td>
</tr>
<tr>
<td>\r</td>
<td>Carriage return. Position the screen cursor to the beginning of the current line; do not advance to the next line. Any characters output after the carriage return overwrite the characters previously output on that line.</td>
</tr>
<tr>
<td>&quot;</td>
<td>Backslash. Used to represent a backslash character in a string.</td>
</tr>
</tbody>
</table>
| \"             | Double quote. Used to represent a double quote character in a string contained in double quotes. For example,
  window.alert( "\"in quotes\"" );
  displays "in quotes" in an alert dialog. |
| \'              | Single quote. Used to represent a single quote character in a string. For example,
  window.alert( '\\in quotes\' );
  displays 'in quotes' in an alert dialog. |

Fig. 7.5 Some common escape sequences.
```javascript
20   // read in first number from user as a string
21   firstNumber = window.prompt( "Enter first integer", "0" );
22
23   // read in second number from user as a string
24   secondNumber = window.prompt( "Enter second integer", "0" );
25
26   // convert numbers from strings to integers
27   number1 = parseInt( firstNumber );
28   number2 = parseInt( secondNumber );
29
30   // add the numbers
31   sum = number1 + number2;
32```

The `window` method `prompt` displays a prompt dialog in the browser with a message and a text field for input.

Function `parseInt` converts its string argument to an integer.
<html>
<head>
<title>An Addition Program</title>

<script type="text/javascript">
<!--
var firstNumber, // first string entered by user
    secondNumber, // second string entered by user
    number1,      // first number to add
    number2,      // second number to add
    sum;         // sum of number1 and number2
</script>
</head>
<body>
Addition.html

35     // display the results
36    document.writeln("<h1>The sum is " + sum + "</h1>" );
37     // -->
38    </script>
39
40    </head>
41    <body>
42    <p>Click Refresh (or Reload) to run the script again</p>
43    </body>
44    </html>
Adding Integers

This is the prompt to the user.

When the user clicks **OK**, the value typed by the user is returned to the program as a string. The program must convert the string to a number.

This is the text field in which the user types the value.

This is the default value if the user does not enter a number.

When the user clicks **OK**, the value typed by the user is returned to the program as a string. The program must convert the string to a number.
The sum is 117

Click Refresh (or Reload) to run the script again
Arithmetic

<table>
<thead>
<tr>
<th>JavaScript operation</th>
<th>Arithmetic operator</th>
<th>Algebraic expression</th>
<th>JavaScript expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition</td>
<td>+</td>
<td>f + 7</td>
<td>f + 7</td>
</tr>
<tr>
<td>Subtraction</td>
<td>−</td>
<td>p – c</td>
<td>p – c</td>
</tr>
<tr>
<td>Multiplication</td>
<td>×</td>
<td>bm</td>
<td>b * m</td>
</tr>
<tr>
<td>Division</td>
<td>/</td>
<td>x / y or &lt;Anchor9&gt; or x / y</td>
<td>x / y</td>
</tr>
<tr>
<td>Modulus</td>
<td>%</td>
<td>r mod s</td>
<td>r % s</td>
</tr>
</tbody>
</table>

**Fig. 7.11** Arithmetic operators.

<table>
<thead>
<tr>
<th>Operator(s)</th>
<th>Operation(s)</th>
<th>Order of evaluation (precedence)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( )</td>
<td>Parentheses</td>
<td>Evaluated first. If the parentheses are nested, the expression in the innermost pair is evaluated first. If there are several pairs of parentheses on the same level (i.e., not nested), they are evaluated from left to right.</td>
</tr>
<tr>
<td>* , / or %</td>
<td>Multiplication, Division, Modulus</td>
<td>Evaluated second. If there are several such operations, they are evaluated from left to right.</td>
</tr>
<tr>
<td>+ or −</td>
<td>Addition, Subtraction</td>
<td>Evaluated last. If there are several such operations, they are evaluated from left to right.</td>
</tr>
</tbody>
</table>

**Fig. 7.12** Precedence of arithmetic operators.
Equality and Relational Operators

Step 1. \( y = 2 \times 5 \times 5 + 3 \times 5 + 7; \)

\( 2 \times 5 \text{ is } 10 \) (Leftmost multiplication)

Step 2. \( y = 10 \times 5 + 3 \times 5 + 7; \)

\( 10 \times 5 \text{ is } 50 \) (Leftmost multiplication)

Step 3. \( y = 50 + 3 \times 5 + 7; \)

\( 3 \times 5 \text{ is } 15 \) (Multiplication before addition)

Step 4. \( y = 50 + 15 + 7; \)

\( 50 + 15 \text{ is } 65 \) (Leftmost addition)

Step 5. \( y = 65 + 7; \)

\( 65 + 7 \text{ is } 72 \) (Last addition)

Step 6. \( y = 72; \) (Last operation—place 72 into \( y \))
**Equality and Relational Operators**

<table>
<thead>
<tr>
<th>Standard algebraic equality operator or relational operator</th>
<th>JavaScript equality or relational operator</th>
<th>Sample JavaScript condition</th>
<th>Meaning of JavaScript condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equality operators</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>=</td>
<td>==</td>
<td>x == y</td>
<td>x is equal to y</td>
</tr>
<tr>
<td></td>
<td>!=</td>
<td>x != y</td>
<td>x is not equal to y</td>
</tr>
<tr>
<td><strong>Relational operators</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&gt;</td>
<td>&gt;</td>
<td>x &gt; y</td>
<td>x is greater than y</td>
</tr>
<tr>
<td>&lt;</td>
<td>&lt;</td>
<td>x &lt; y</td>
<td>x is less than y</td>
</tr>
<tr>
<td>&gt;=</td>
<td>&gt;=</td>
<td>x &gt;= y</td>
<td>x is greater than or equal to y</td>
</tr>
<tr>
<td>?</td>
<td>&lt;=</td>
<td>x &lt;= y</td>
<td>x is less than or equal to y</td>
</tr>
</tbody>
</table>

**Fig. 7.14** Equality and relational operators.
<html>
<head>
<title>Performing Comparisons</title>
<script type="text/javascript">
<!--
var first, // first string entered by user
second, // second string entered by user
number1, // first number to compare
number2; // second number to compare

// read first number from user as a string
first = window.prompt( "Enter first integer:", "0" );

// read second number from user as a string
second = window.prompt( "Enter second integer:", "0" );
</script>
</head>
</html>
// convert numbers from strings to integers
number1 = parseInt( first );
number2 = parseInt( second );
document.writeln( "<h1>Comparison Results</h1>" );
document.writeln( "<table border = \"1\" width = \"100\%">" );
Each if statement uses the equality operators to determine the relationship of the two integers input by the user.
if ( number1 >= number2 )
    document.writeln( "<tr><td>" + number1 + " >= " + number2 + "</td></tr>" );

// Display results
document.writeln( "</table>" );

// -->
</script>

</body>
<p>Click Refresh (or Reload) to run the script again</p>
</body>
</html>
Comparison Results

123 == 123
123 <= 123
123 >= 123

Click Refresh (or Reload) to run the script again
Comparison Results

100 != 200
100 < 200
100 <= 200

Click Refresh (or Reload) to run the script again
Comparison Results

200 != 100
200 > 100
200 >= 100

Click Refresh (or Reload) to run the script again
Equality and Relational Operators

<table>
<thead>
<tr>
<th>Operators</th>
<th>Associativity</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>()</td>
<td>left to right</td>
<td>parentheses</td>
</tr>
<tr>
<td>* / %</td>
<td>left to right</td>
<td>multiplicative</td>
</tr>
<tr>
<td>+ -</td>
<td>left to right</td>
<td>additive</td>
</tr>
<tr>
<td>&lt; &lt;= &gt; &gt;=</td>
<td>left to right</td>
<td>relational</td>
</tr>
<tr>
<td>== !=</td>
<td>left to right</td>
<td>equality</td>
</tr>
<tr>
<td>=</td>
<td>right to left</td>
<td>assignment</td>
</tr>
</tbody>
</table>

Fig. 7.16 Precedence and associativity of the operators discussed so far.
## Control Structures

### JavaScript Keywords

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Keyword</th>
<th>Keyword</th>
<th>Keyword</th>
<th>Keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>break</td>
<td>case</td>
<td>continue</td>
<td>delete</td>
<td>do</td>
</tr>
<tr>
<td>else</td>
<td>false</td>
<td>for</td>
<td>function</td>
<td>if</td>
</tr>
<tr>
<td>in</td>
<td>new</td>
<td>null</td>
<td>return</td>
<td>switch</td>
</tr>
<tr>
<td>this</td>
<td>true</td>
<td>typeof</td>
<td>var</td>
<td>void</td>
</tr>
<tr>
<td>while</td>
<td>with</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Keywords that are reserved, but not used by JavaScript*

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Keyword</th>
<th>Keyword</th>
<th>Keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>catch</td>
<td>class</td>
<td>const</td>
<td>debugger</td>
</tr>
<tr>
<td>enum</td>
<td>export</td>
<td>extends</td>
<td>finally</td>
</tr>
<tr>
<td>super</td>
<td></td>
<td></td>
<td>import</td>
</tr>
<tr>
<td></td>
<td>try</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Fig. 8.2* JavaScript keywords.
if/else Selection Structure

grade >= 60

true

print “Passed”

false

54   if ( number1 >= number2 )
55       document.writeln( "<tr><td>" + number1 + " >= " +
56       number2 + "</td></tr>" );
if/else Selection Structure

```
54 if ( number1 >= number2 )
55   document.writeln( number1 + " >= " + number2 );
56 else
57   document.writeln( number1 + " NOT >= " +
58       number2 );
```

Internet Computing 30
Counter-Controlled Repetition

\[ \text{while ( gradeCounter} \leq 10 ) \{ // \text{loop 10 times} \]
\[ // \text{add 1 to gradeCounter} \]
\[ \text{gradeCounter} = \text{gradeCounter} + 1; \]
\[ \} \]
Average.html

8    <html>
9    <head>
10   <title>Class Average Program</title>
11
12   <script type = "text/javascript">
13      <!--
14      var total,         // sum of grades
15      gradeCounter,    // number of grades entered
16      gradeValue,      // grade value
17      average,        // average of all grades
18      grade;          // grade typed by user
19      -->
Average.html

20   // Initialization Phase
21   total = 0;        // clear total
22   gradeCounter = 1; // prepare to loop
23

24   // Processing Phase
25   while ( gradeCounter <= 10 ) {  // loop 10 times
26       // prompt for input and read grade from user
27       grade = window.prompt( "Enter integer grade:", "0" );
28       // convert grade from a string to an integer
29       gradeValue = parseInt( grade );
30       // add gradeValue to total
31       total = total + gradeValue;
32       // add 1 to gradeCounter
33       gradeCounter = gradeCounter + 1;
34   }
// Termination Phase
average = total / 10; // calculate the average

// display average of exam grades
document.writeln("<h1>Class average is " + average + "</h1>" );

// -->
</script>

</head>
<body>
<p>Click Refresh (or Reload) to run the script again</p>
</body>
</html>
This dialog is displayed 10 times.
User input is 100, 88, 93, 55, 68, 77, 83, 95, 73 and 62.

Class average is 79.4

Click Refresh (or Reload) to run the script again.
while ( gradeValue != -1 ) {
    // add gradeValue to total
    total = total + gradeValue;
    // add 1 to gradeCounter
    gradeCounter = gradeCounter + 1;
    // prompt for input and read grade from user
    grade = window.prompt( "Enter Integer Grade, -1 to Quit:", "0" );
    // convert grade from a string to an integer
    gradeValue = parseInt( grade );
} // Termination phase
if ( gradeCounter != 0 ) {
    average = total / gradeCounter;
}

Each iteration of the loop will open a prompt dialog allowing the user to input another grade.
This dialog is displayed four times.
User input is 97, 88, 72 and −1.

Class average is 85.666666666666667

Click Refresh (or Reload) to run the script again.
<html>
<head>
<title>Analysis of Examination Results</title>

<script type = "text/javascript">
<!--[CDATA[
var passes = 0,      // number of passes
failures = 0,    // number of failures
student = 1,     // student counter
result;          // one exam result
]]></CDATA[>
</script>
</head>
</html>
// process 10 students; counter-controlled loop
while ( student <= 10 ) {
    result = window.prompt("Enter result (1=pass,2=fail)", "0");
    if ( result == "1" )
        passes = passes + 1;
    else
        failures = failures + 1;
    student = student + 1;
}

The **while** loop will continue until the value of **student** is 10 meaning 10 results were entered.

Entering a 1 into the prompt dialog means the student passed the exam. A value of 2 means the student failed.
33   // termination phase
34   document.writeln( "<h1>Examination Results</h1>" );
35   document.writeln( "Passed: " + passes + "<br />Failed: " + failures );
36   if ( passes > 8 )
37       document.writeln( "<br />Raise Tuition" );
38   // -->
39   </script>
40   </head>
41   <body>
42       <p>Click Refresh (or Reload) to run the script again</p>
43   </body>
44   </html>

If more than 8 students passed the exam, the program says to “Raise Tuition”. 
## Increment and Decrement Operators

<table>
<thead>
<tr>
<th>Assignment operator</th>
<th>Initial value of variable</th>
<th>Sample expression</th>
<th>Explanation</th>
<th>Assigns</th>
</tr>
</thead>
<tbody>
<tr>
<td>+=</td>
<td>c = 3</td>
<td>c += 7</td>
<td>c = c + 7</td>
<td>10 to c</td>
</tr>
<tr>
<td>-=</td>
<td>d = 5</td>
<td>d -= 4</td>
<td>d = d - 4</td>
<td>1 to d</td>
</tr>
<tr>
<td>*=</td>
<td>e = 4</td>
<td>e *= 5</td>
<td>e = e * 5</td>
<td>20 to e</td>
</tr>
<tr>
<td>/=</td>
<td>f = 6</td>
<td>f /= 3</td>
<td>f = f / 3</td>
<td>2 to f</td>
</tr>
<tr>
<td>%=</td>
<td>g = 12</td>
<td>g %= 9</td>
<td>g = g % 9</td>
<td>3 to g</td>
</tr>
</tbody>
</table>

Fig. 8.13 Increment and decrement operators.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Called</th>
<th>Sample expression</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>++</td>
<td>preincrement</td>
<td>++a</td>
<td>Increment a by 1, then use the new value of a in the expression in which a resides.</td>
</tr>
<tr>
<td>++</td>
<td>postincrement</td>
<td>a++</td>
<td>Use the current value of a in the expression in which a resides, then increment a by 1.</td>
</tr>
<tr>
<td>--</td>
<td>predecrement</td>
<td>--b</td>
<td>Decrement b by 1, then use the new value of b in the expression in which b resides.</td>
</tr>
<tr>
<td>--</td>
<td>postdecrement</td>
<td>b--</td>
<td>Use the current value of b in the expression in which b resides, then decrement b by 1.</td>
</tr>
</tbody>
</table>

Fig. 8.13 Increment and decrement operators.
### Increment.html

```html
<html>
<head>
<title>Preincrementing and Postincrementing</title>
</head>
<body>
<script type="text/javascript">
<!--
var c;

c = 5;
document.writeln( '<h3>Postincrementing</h3>' );
// print 5
// print 5 then increment
document.writeln( '<br />' + c++ );
// print 6
document.writeln( '<br />' + c );
// print 6

Postincrementing the variable will print the variable and then increment the value by one.
</script>
</body>
</html>
```
c = 5;
document.writeln( "<h3>Preincrementing</h3>" );
document.writeln( c ); // print 5
// increment then print 6
document.writeln( "<br />"+ ++c );
document.writeln( "<br />"+ c ); // print 6
// -->
</script>

</head><body></body>
</html>

Preincrementing the variable will increment the value by one and then print the value.
Preincrementing and Postincrementing - Microsoft Internet Explorer

Address: C:\w3htp2\ch08\examples\increment.html

Postincrementing
5
5
6

Preincrementing
5
6
6
whileCounter.html

```html
<html>
<head>
<title>Counter-Controlled Repetition</title>
<script type = "text/javascript">
<!--
var counter = 1;            // initialization
while ( counter <= 7 ) {    // repetition condition
    document.writeln( "<p style = "font-size: " +
    counter + "ex">XHTML font size " + counter +
    "ex</p>" );
    ++counter;               // increment
}
// -->
</script>
</head><body></body>
</html>
```

The `while` loop will continue until the value of `counter` is greater than 7.
forCounter.html

8    <html>
9    <head>
10   <title>Counter-Controlled Repetition</title>
12   <script type = "text/javascript">
13   <!--
14   // Initialization, repetition condition and
15   // incrementing are all included in the for
16   // structure header.
17   for ( var counter = 1; counter <= 7; ++counter )
18   document.writeln( "<p style = "font-size: " +
19        counter + "ex">XHTML font size " + counter +
20        "ex</p>" );
21   // -->
22   </script>
24   </head><body></body>
25   </html>
XHTML font size 1ex

XHTML font size 2ex

XHTML font size 3ex

XHTML font size 4ex

XHTML font size 5ex

XHTML font size 6ex

XHTML font size 7ex
<html>
<head>
<title>Calculating Compound Interest</title>
<script type = "text/javascript">
<!--
var amount, principal = 1000.0, rate = .05;
document.writeln(
    "<table border = \"1\" width = \"100\%\">" );
document.writeln(
    "<caption>Calculating Compound Interest</caption>" );
document.writeln(
    "<thead><tr><th align = \"left\">Year</th>" );
document.writeln(
    "<th align = \"left\">Amount on deposit</th>" );
document.writeln( 
    "</thead></table>"
);  
</script>
</head>
<body>

</body>
</html>
Each iteration of the `for` loop creates a table row listing the year of the loan and the amount.
### Calculating Compound Interest

<table>
<thead>
<tr>
<th>Year</th>
<th>Amount on deposit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1050</td>
</tr>
<tr>
<td>2</td>
<td>1102.5</td>
</tr>
<tr>
<td>3</td>
<td>1157.63</td>
</tr>
<tr>
<td>4</td>
<td>1215.51</td>
</tr>
<tr>
<td>5</td>
<td>1276.28</td>
</tr>
<tr>
<td>6</td>
<td>1340.1</td>
</tr>
<tr>
<td>7</td>
<td>1407.1</td>
</tr>
<tr>
<td>8</td>
<td>1477.46</td>
</tr>
<tr>
<td>9</td>
<td>1551.33</td>
</tr>
<tr>
<td>10</td>
<td>1628.89</td>
</tr>
</tbody>
</table>
switch Multiple-Selection Structure

case a
  true: case a action(s) → break
  false

  case a
    true: case a action(s) → break
    false

  ...

  case b
    true: case b action(s) → break
    false

  ...

  case z
    true: case z action(s) → break
    false

  default: default action(s)

Internet Computing
<html>
<head>
<title>Switching between XHTML List Formats</title>
<script type = "text/javascript">
<!--
var choice, // user's choice
startTag, // starting list item tag
endTag, // ending list item tag
validInput = true, // indicates if input is valid
listType; // list type as a string
choice = window.prompt( "Select a list style:\n" +
"1 (bullet), 2 (numbered), 3 (lettered)", "1" );
-->
</script>
</head>
<body>
</body>
</html>
switchTest.html

```python
23   switch ( choice ) {
24       case "1":
25           startTag = "<ul>";   endTag = "</ul>";
26           listType = "<h1>Bullet List</h1>";
27           break;
28       case "2":
29           startTag = "<ol>";    endTag = "</ol>";
30           listType = "<h1>Ordered List: Numbered</h1>";
31           break;
32       case "3":
33           startTag = "<ol type = "A">";  endTag = "</ol>";
34           listType = "<h1>Ordered List: Lettered</h1>";
35           break;
36       default:
37           validInput = false;
38   }
39
```

The value of `choice` is evaluated against each of the values of the `case` labels.

The `break` statement causes program control to proceed with the first statement after the `switch` structure.

If none of the `cases` match, variable `validInput` is set to `false`. Internet Computing
switchTest.html

```html
43   if ( validInput == true ) {
44       document.writeln( listType + startTag );
46       for ( var i = 1; i <= 3; ++i )
47           document.writeln( "<li>List item " + i + "</li>" );
49       document.writeln( endTag );
50   }
51   else
52       document.writeln( "Invalid choice: " + choice );
53       // -->
54   </script>
56   </head>
57   <body>
58       <p>Click Refresh (or Reload) to run the script again</p>
59   </body>
60   </html>
```
Extractor User Prompt

Script Prompt:
Select a list style:
1 (bullet), 2 (numbered), 3 (lettered)

OK
Cancel

Bullet List

• List item 1
• List item 2
• List item 3

Click Refresh (or Reload) to run the script again

Internet Computing
Ordered List: Numbered

1. List item 1
2. List item 2
3. List item 3

Click Refresh (or Reload) to run the script again.

Ordered List: Lettered

A. List item 1
B. List item 2
C. List item 3

Click Refresh (or Reload) to run the script again.
Ordered List: Numbered

1. List item 1
2. List item 2
3. List item 3

Click Refresh (or Reload) to run the script again.

Ordered List: Lettered

A. List item 1
B. List item 2
C. List item 3

Click Refresh (or Reload) to run the script again.
do/while Repetition Structure
DoWhileTest.html

8    <html>
9    <head>
10   <title>Using the do/while Repetition Structure</title>
12   <script type = "text/javascript">
13   <!--
14 var counter = 1;
16   do {
17   document.writeln( "<h" + counter + ">This is " +
18   "an h" + counter + " level head" + "</h" +
19   counter + "">" );
21   ++counter;
22   } while ( counter <= 6 );
23   // -->
24   </script>
26   </head><body></body>
27   </html>

Each iteration of the do/while loop writes a line of text with a header element to the XHTML document.

The loop stops when the value of counter is greater than 6.
This is an h1 level head

This is an h2 level head

This is an h3 level head

This is an h4 level head

This is an h5 level head

This is an h6 level head
Using the break Statement in a for Structure

```html
<html>
<head>
<title>Using the break Statement in a for Structure</title>
<script type="text/javascript">
<!--
for ( var count = 1; count <= 10; ++count ) {
  if ( count == 5 )
    break; // break loop only if count == 5
  document.writeln( "Count is: " + count + "\<br\>" );
}
document.writeln( "Broke out of loop at count = " + count );
// -->
</script>
</head><body>
</html>
```

When the value of variable `count` equals 5, the `break` statement causes program control to proceed to the first line outside the `for` loop.

Internet Computing 63
Count is: 1
Count is: 2
Count is: 3
Count is: 4
Broke out of loop at count = 5
When the value of variable `count` equals 5, the `continue` statement causes program control to proceed to the next iteration of the `for` loop.
Using the continue Statement in a for Structure - Microsoft Int...
stop: { // labeled compound statement
    for ( var row = 1; row <= 10; ++row ) {
        for ( var column = 1; column <= 5 ; ++column ) {

            if ( row == 5 )
                break stop; // jump to end of stop block

        }

        document.write( "* " );

    }

    document.writeln( "<br />" );

}
nextRow: // target label of continue statement
for ( var row = 1; row <= 5; ++row ) {
document.writeln( "<br />" );
}
for ( var column = 1; column <= 10; ++column ) {
    if ( column > row )
        continue nextRow; // next iteration of // labeled loop
    document.write( "* ");
}

If the value of variable column is greater than the value of variable row, the continue statement causes the next iteration of the loop.
Program Output

```
*
**
***
****
*****
```

Internet Computing