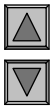
Chapter 9 - JavaScript: Control Structures II

Outline



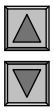
Outline

whileCounter.html

```
1   <?xml version = "1.0"?>
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. 9.1: WhileCounter.html    -->
6   <!-- Counter-Controlled Repetition -->
7
8   <html xmlns = "http://www.w3.org/1999/
9      <head>
10        <title>Counter-Controlled Repeti
11
12        <script type = "text/javascript">
13           <!--
14           var counter = 1;              // initialization
15
16           while ( counter <= 7 ) {      // repetition condition
17              document.writeln( "<p style = \"font-size: " +
18                 counter + "ex\">XHTML font size " + counter +
19                 "ex</p>" );
20              ++counter;                 // increment
21           }
22           // -->
23        </script>
24
25     </head><body></body>
26  </html>
```

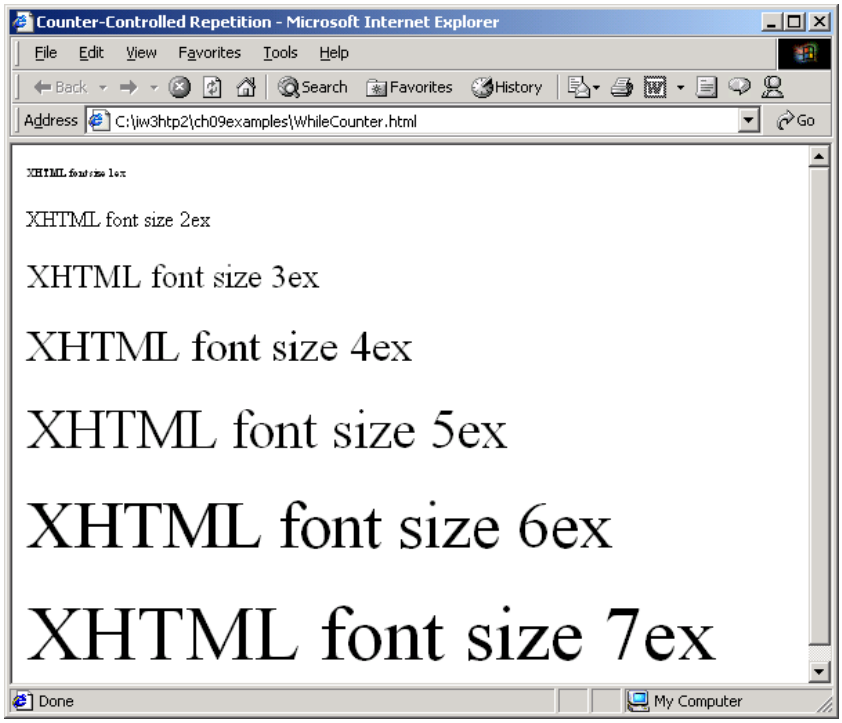
The **while** loop will continue until the value of **counter** is greater than 7.

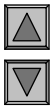
Increment the counter.



Outline

Program Output





Outline

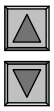
ForCounter.html

```
1   <?xml version = "1.0"?>
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. 9.2: ForCounter.html                          -->
6   <!-- Counter-Controlled Repetition with for structure -->
7
8   <html xmlns = "http://www.w3.org/1999/xhtml">
9      <head>
10        <title>Counter-Controll                    tle>
11
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>
23
24     </head><body></body>
25  </html>
```

Initialization

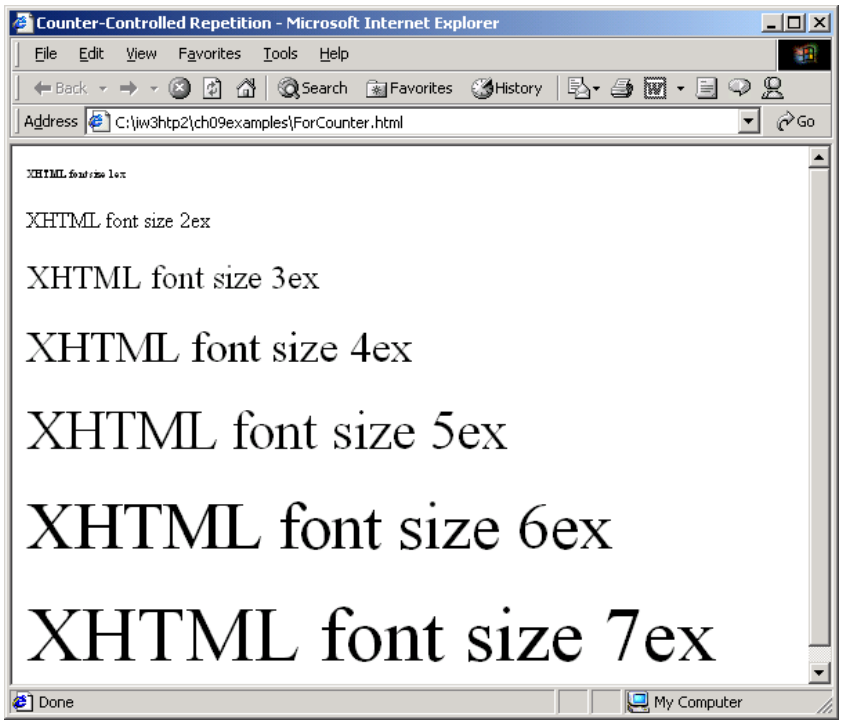
Repetition condition

Incrementing



Outline

Program Output



9.3 `For` Repetition Structure

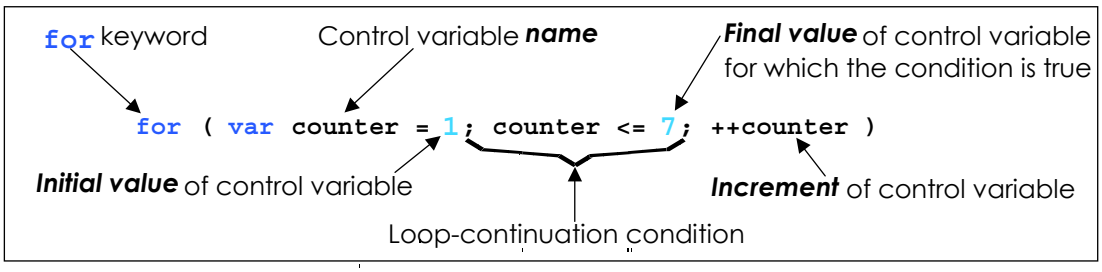


Fig. 9.3 Components of a typical for structure header.

9.4 Examples Using the `for` Structure

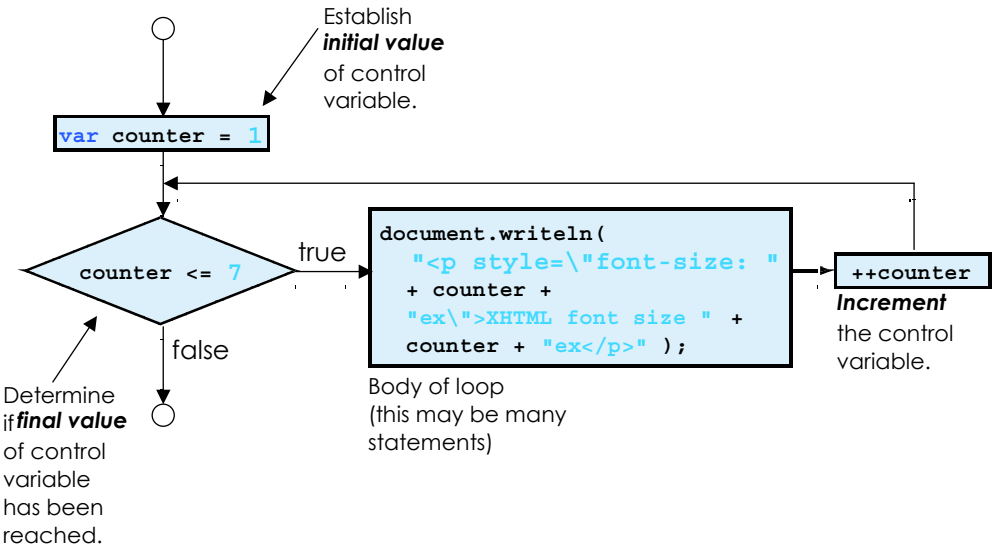
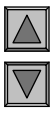


Fig. 9.4 Flowcharting a typical for repetition structure.



Outline

Sum.html

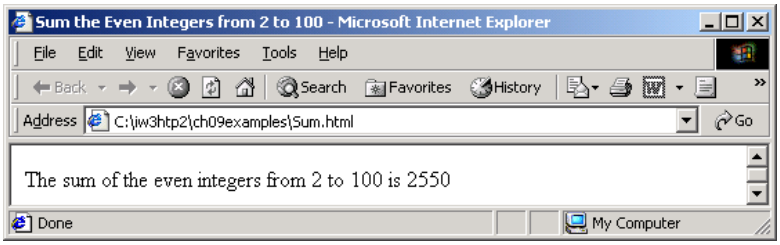
```
1   <?xml version = "1.0"?>
2   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5   <!-- Fig. 9.5: Sum.html                    -->
6   <!-- Using the for repetition structure -->
7
8   <html xmlns = "http://www.w3.org/
9      <head>
10        <title>Sum the Even Integer
11
12        <script type = "text/javas
13           <!--
14           var sum = 0;
15
16           for ( var number = 2; number <= 100; number += 2 )
17              sum += number;
18
19           document.writeln( "The sum of the even integers " +
20              "from 2 to 100 is " + sum );
21           // -->
22        </script>
23
24     </head><body></body>
25  </html>
```

The `for` loop will continue until the value of `number` is greater than 100.

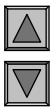
Initialization.

Repetition con

Incrementing.



Program Output



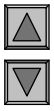
Outline

Interest.html

```
1   <?xml version = "1.0"?>
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. 9.6: interest.html            -->
6   <!-- Using the for repetition structure -->
7
8   <html xmlns = "http://www.w3.org/1999/xhtml">
9      <head>
10        <title>Calculating Compound Interest</title>
11
12        <script type = "text/javascript">
13           <!--
14           var amount, principal = 1000.0, rate = .05;
15
16           document.writeln(
17              "<table border = \"1\" width = \"100%\">" );
18           document.writeln(
19              "<caption>Calculating Compound Interest</caption>" );
20           document.writeln(
21              "<thead><tr><th ali
22           document.writeln(
23              "<th align = \"left
24           document.writeln( "</t
25
26           for ( var year = 1; year <= 10; ++year ) {
27              amount = principal * Math.pow( 1.0 + rate, year );
28              document.writeln( "<tbody><tr><td>" + year +
29                 "</td><td>" + Math.round( amount * 100 ) / 100 +
30                 "</td></tr>" );
31           }
32
33           document.writeln( "</tbody></table>" );
34           // -->
35        </script>
```

Opening **table** element.

Each iteration of the **for** loop creates a table row listing the year of the loan and the amount.

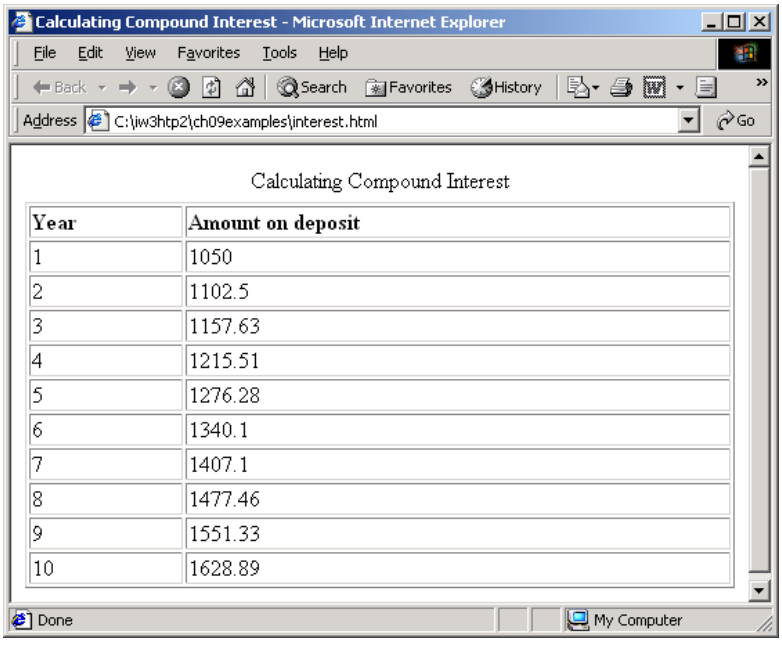


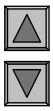
Outline

Interest.html

```
36
37     </head><body></body>
38  </html>
```

Program Output





Outline

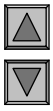
SwitchTest.html

```
1   <?xml version = "1.0"?>
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. 9.7: SwitchTest.html  -->
6   <!-- Using the switch structure -->
7
8   <html xmlns = "http://www.w3.org/1999/xhtml">
9      <head>
10        <title>Switching between XHTML List Formats</title>
11
12        <script type = "t
13           <!--
14           var choice,
15               startTag,
16               endTag,
17               validInput = true,
18               listType;
19
20           choice = window.prompt( "Select a list style:\n" +
21               "1 (bullet), 2 (numbered), 3 (lettered)", "1" );
22
23           switch ( choice ) {
24              case "1":
25                 startTag = "<u
26                 endTag = "</ul>";
27                 listType = "<h1>Bullet List</h1>";
28                 break;
29              case "2":
30                 startTag = "<ol>";
31                 endTag = "</ol>";
32                 listType = "<h1>Ordered List: Numbered</h1>";
33                 break;
```

Variable **choice** is given the value input by the user in the prompt dialog.

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.



Outline

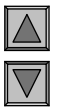
SwitchTest.html

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

If none of the **case**s match, variable **validInput** is set to **false**.

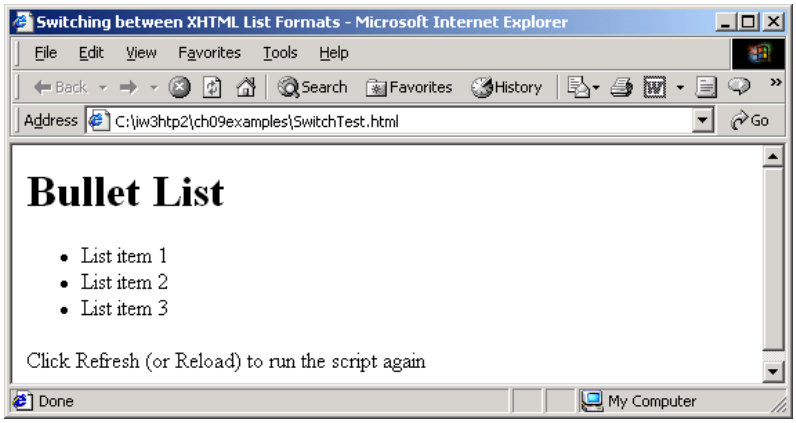
If the user input a valid value, the list is created.

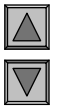
Otherwise, the message **"Invalid choice"** is displayed in the browser.



Outline

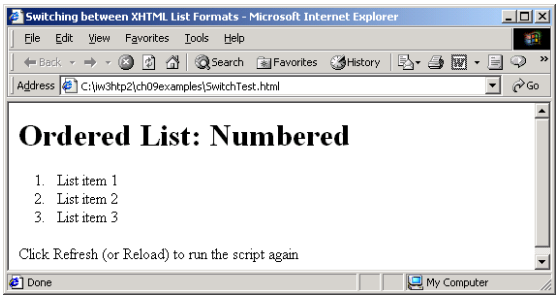
Program Output

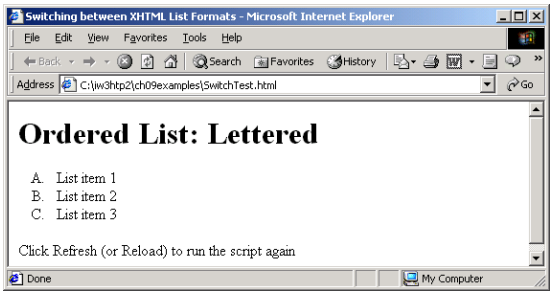




Outline

Program Output





9.5 `switch` Multiple-Selection Structure

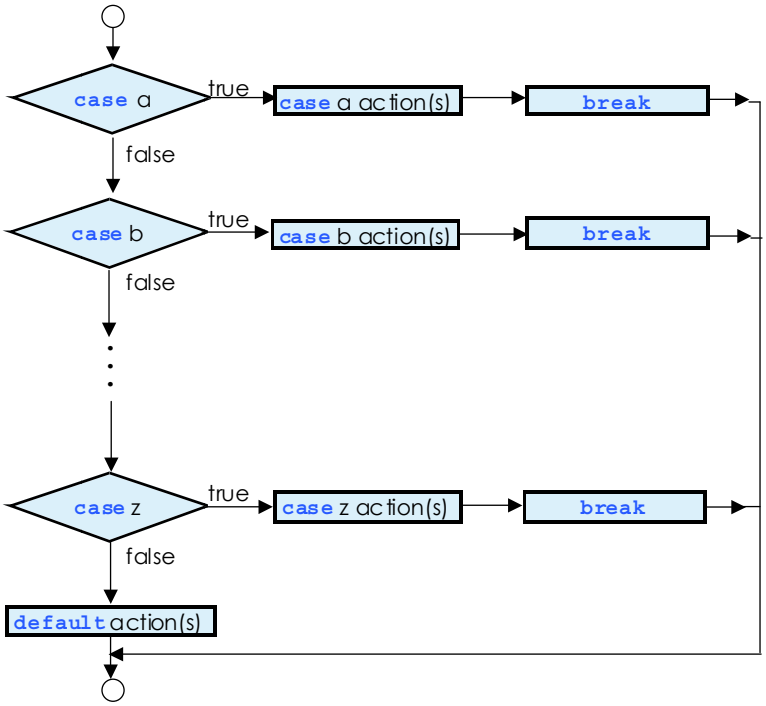


Fig. 9.8 switch multiple-selection structure.

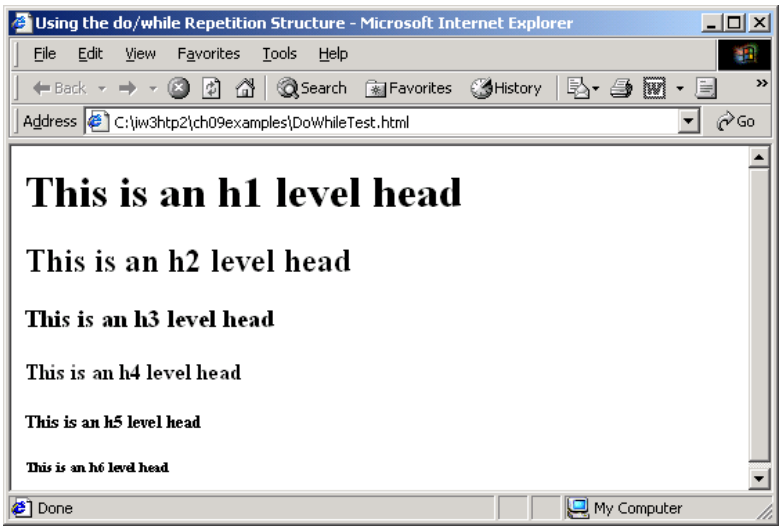
Outline

DoWhileTest.html

```
1   <?xml version = "1.0"?>
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. 9.9: DoWhileTest.html    -->
6   <!-- Using the do/while structure -->
7
8   <html xmlns = "http://www.w3.org/1999/xhtml">
9      <head>
10        <title>Using the do/while Repetition Structure</title>
11
12        <script type =
13           <!--
14           var counter
15
16           do {
17              document.writeln( "<h" +
18                 "an h" + counter + "
19                 counter + ">" );
20
21              ++counter;
22           } while ( counter <= 6 );
23           // -->
24        </script>
25
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**.



Program Output

9.6 `do/while` Repetition Structure

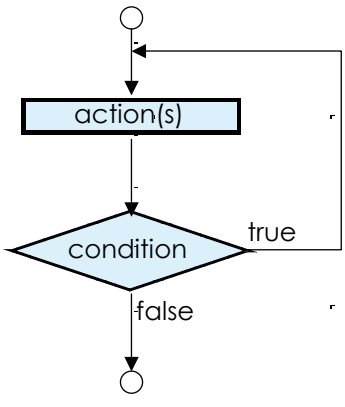
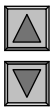


Fig. 9.10 Flowcharting the do/while repetition structure.

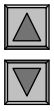


Outline

BreakTest.html

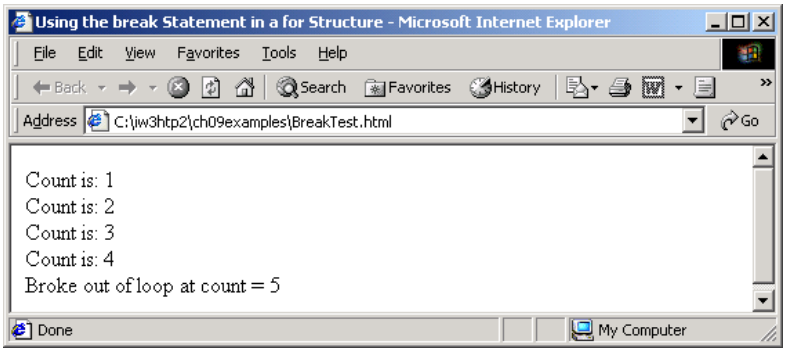
```
1   <?xml version = "1.0"?>
2   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5   <!-- Fig. 9.11: BreakTest.html    -->
6   <!-- Using the break statement    -->
7
8   <html xmlns = "http://www.w3.org/1999/xhtml">
9      <head>
10        <title>
11           Using the break St
12        </title>
13
14        <script type = "text/javascript">
15           <!--
16           for ( var count = 1; count <= 10; ++count ) {
17              if ( count == 5 )
18                 break;  // break loop only if count == 5
19
20              document.writeln( "Count is: " + count + "<br />" );
21           }
22
23           document.writeln(
24              "Broke out of loop at count = " + count );
25           // -->
26        </script>
27
28     </head><body></body>
29  </html>
```

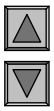
When the value of variable **count** equals **5**, the **break** statement causes program control to proceed to the first line outside the **for** loop.



Outline

Program Output



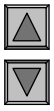


Outline

ContinueTest.html

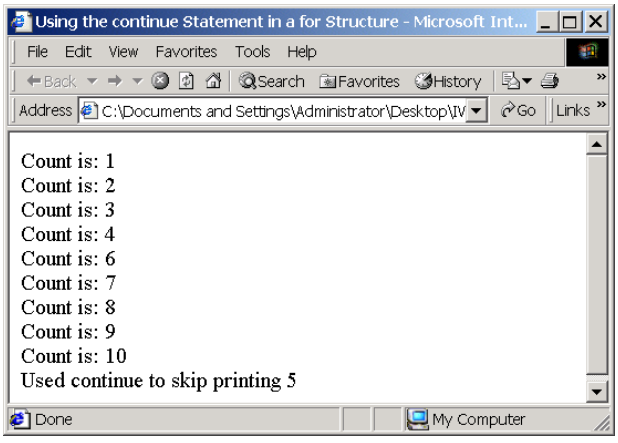
```
1   <?xml version = "1.0"?>
2   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5   <!-- Fig. 9.12: ContinueTest.html  -->
6   <!-- Using the break statement     -->
7
8   <html xmlns = "http://www.w3.org/1999/xhtml">
9      <head>
10        <title>
11           Using the continue s
12        </title>
13
14        <script type = "text/javascript">
15           <!--
16           for ( var count = 1; count <= 10; ++count ) {
17              if ( count == 5 )
18                 continue;  // skip remaining code in loop
19                            // only if count == 5
20
21              document.writeln( "Count is: " + count + "<br />" );
22           }
23
24           document.writeln( "Used continue to skip printing 5" );
25           // -->
26        </script>
27
28     </head><body></body>
29  </html>
```

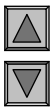
When the value of variable **count** equals **5**, the **continue** statement causes program control to proceed to the next iteration of the **for** loop.



Outline

Program Output





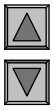
Outline

BreakLabelTest.html

```
1   <?xml version = "1.0"?>
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. 9.13: BreakLabelTest.html          -->
6   <!-- Using the break statement with a Label -->
7
8   <html xmlns = "http
9      <head>
10        <title>Using the break Statement with a Label</title>
11
12        <script type = "text/jav
13           <!--
14           stop: {   // labeled
15              for ( var row = 1;
16                 for ( var column = 1; column <= 5 ; ++column ) {
17
18                    if ( row == 5 )
19                       break stop; // jump to end of stop block
20
21                    document.write( "* " );
22                 }
23
24                 document.writeln( "<br />" );
25              }
26
27              // the following line is skipped
28              document.writeln( "This line should not print" );
29           }
30
```

stop is the label for the **break** statement.

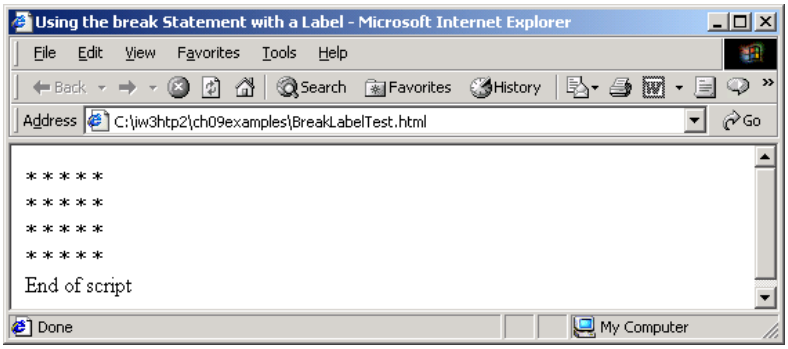
When the **break** statement is encountered, program control proceeds to the first line outside the **stop** block and not just the **for** loop where the statement is found.



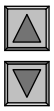
Outline

BreakLabelTest.html

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



Program Output



Outline

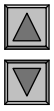
ContinueLabelTest.html

```
1   <?xml version = "1.0"?>
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. 9.14: ContinueLabelTest.html -->
6   <!-- Using the continue statement      -->
7
8   <html xmlns = "http://ww
9      <head>
10        <title>Using the continue Statement with a Label</title>
11
12        <script type = "text/javascrip
13           <!--
14           nextRow:    // target label
15              for ( var row = 1; row <
16                 document.writeln( "<br />" );
17
18                 for ( var column = 1; column <= 10; ++column ) {
19
20                    if ( column > row
21                       continue nextR
22
23
24                    document.write( "* " );
25                 }
26              }
27           // -->
28        </script>
29
30     </head><body></body>
31  </html>
```

nextRow is the label for the **continue** statement.

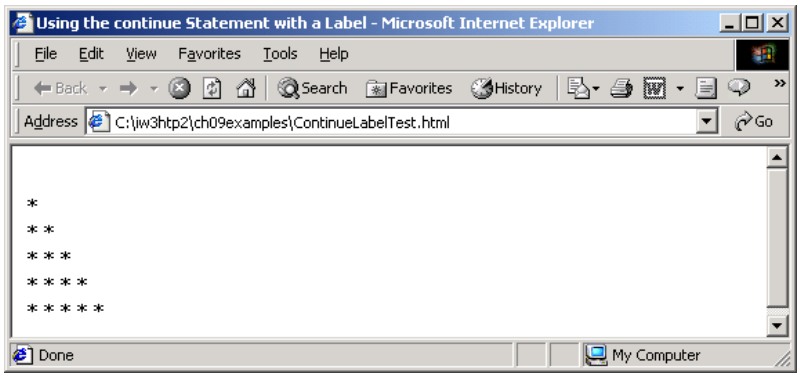
If the value of variable **column** is greater than the value of variable **row**, the **continue** statement causes the next interation of the loop.

If the **continue** statement is performed, method **write** does not print the string **"* "** in the XHTML document.



Outline

Program Output



9.9 Logical Operators

expression1	expression2	expression1 && expression2
false	false	false
false	true	false
true	false	false
true	true	true

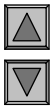
Fig. 9.15 Truth table for the && (logical AND) operator.

expression1	expression2	expression1 || expression2
false	false	false
false	true	true
true	false	true
true	true	true

Fig. 9.16 Truth table for the || (logical OR) operator.

expression	!expression
false	true
true	false

Fig. 9.17 Truth table for operator ! (logical negation).



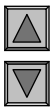
Outline

LogicalOperators.html

```
1   <?xml version = "1.0"?>
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. 9.18: LogicalOperators.html  -->
6   <!-- Demonstrating Logical Operators    -->
7
8   <html xmlns = "http://www.w3.org/1999/xhtml">
9      <head>
10        <title>Demonstrating the Logical Operators</title>
11
12        <script type = "text/javascript">
13           <!--
14           document.writeln(
15              "<table border = \"1\" width = \"100%\
16
17           document.writeln(
18              "<caption>Demonstrating Logical " +
19              "Operators</caption" );
20
21           document.writeln(
22              "<tr><td width = \"25%\">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>" );
28
29           document.writeln(
30              "<tr><td width = \"25%\">Logical OR (||)</td>" +
31              "<td>false || false: " + ( false || false ) +
32              "<br />false || true: " + ( false || true ) +
33              "<br />true || false: " + ( true || false ) +
34              "<br />true || true: " + ( true || true ) +
35              "</td>" );
```

Each expression will evaluate to **true** or **false** using the rules of logical **AND**.

Each expression will evaluate to **true** or **false** using the rules of logical **OR**.

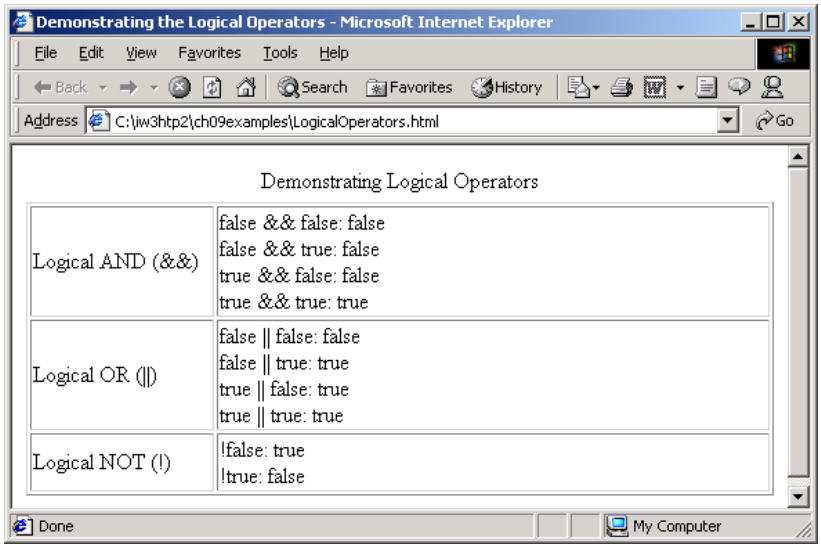


Outline

LogicalOperators.html

```
36
37          document.writeln(
38             "<tr><td width = \"25%\">Logical NOT (!)</td>" +
39             "<td>!false: " + ( !false ) +
40             "<br />!true: " + ( !true ) + "</td>" );
41
42          document.writeln( "</table>" );
43          // -->
44       </script>
45
46    </head><body></body>
47 </html>
```

These expressions demonstrate the use of logical **NOT**.



Program Output

9.10 Summary of Structured Programming

Operators	Associativity	Type
`( )`	left to right	parentheses
`++ -- !`	right to left	unary
`* / %`	left to right	multiplicative
`+ -`	left to right	additive
`< <= > >=`	left to right	relational
`== !=`	left to right	equality
`&&`	left to right	logical AND
`||`	left to right	logical OR
`?:`	right to left	conditional
`= += -= *= /= %=`	right to left	assignment

Fig. 9.19 Precedence and associativity of the operators discussed so far.

9.10 Summary of Structured Programming

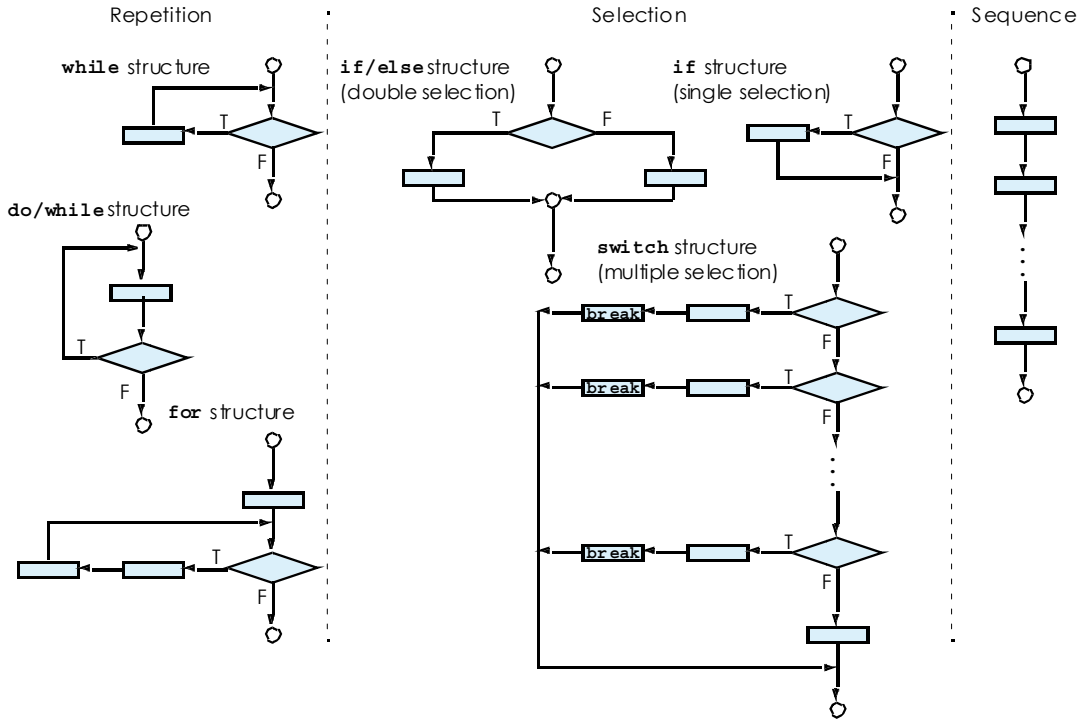


Fig. 9.20 JavaScript's single-entry/single-exit sequence, selection and repetition structures.

9.10 Summary of Structured Programming

Rules for Forming Structured Programs	
1)	Begin with the "simplest flowchart" (Fig. 9.22).
2)	Any rectangle (action) can be replaced by two rectangles (actions) in sequence.
3)	Any rectangle (action) can be replaced by any control structure (sequence, if, if/else, switch, while, do/while or for).
4)	Rules 2 and 3 may be applied as often as you like and in any order.

Fig. 9.21 Rules for forming structured programs.

9.10 Summary of Structured Programming

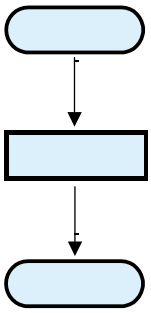


Fig. 9.22 Simplest flowchart.

9.10 Summary of Structured Programming

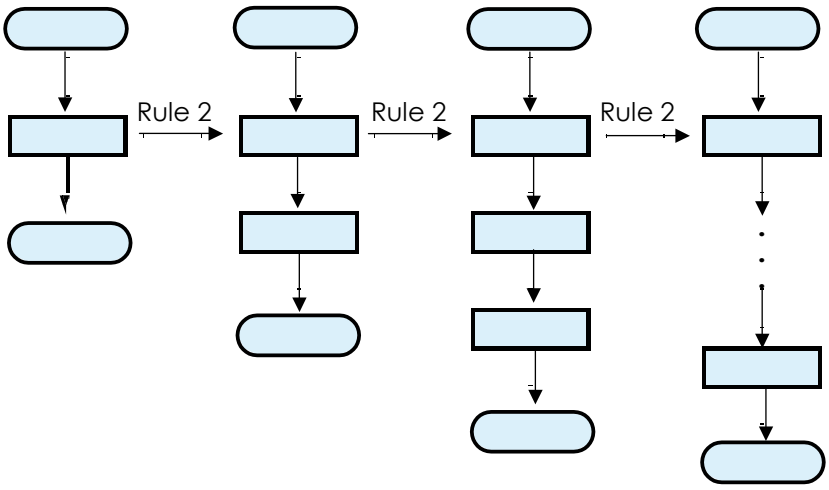


Fig. 9.23 Repeatedly applying rule 2 of Fig. 9.21 to the simplest flowchart.

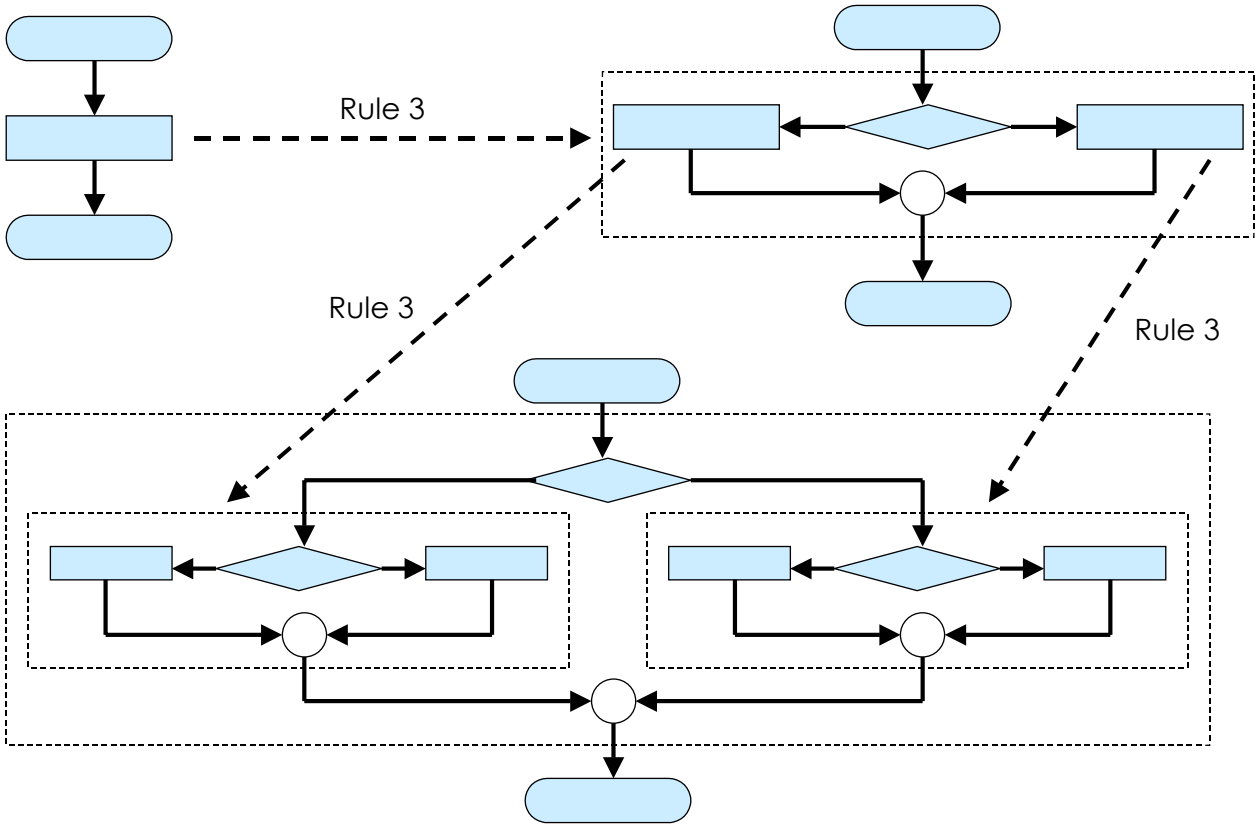


Fig. 9.24 Applying rule 3 of Fig. 9.21 to the simplest flowchart

9.10 Summary of Structured Programming

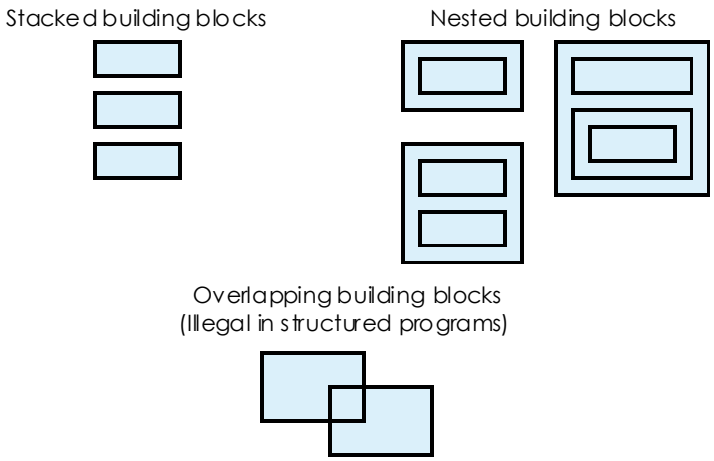


Fig. 9.25 Stacked, nested and overlapped building blocks.

9.10 Summary of Structured Programming

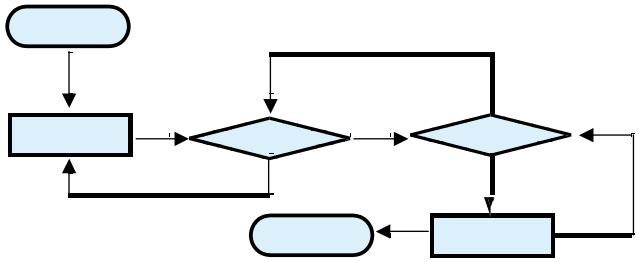


Fig. 9.26 Unstructured flowchart.