

Control Statements I

Outline

- 8.1 Introduction
- 8.2 Algorithms
- 8.3 Pseudocode
- 8.4 Control Structures
- 8.5 if Selection Statement
- 8.6 if...else Selection Statement
- 8.7 while Repetition Statement
- 8.8 Formulating Algorithms: Case Study 1 (Counter-Controlled Repetition)
- 8.9 Formulating Algorithms with Top-Down, Stepwise Refinement: Case Study 2 (Sentinel-Controlled Repetition)
- 8.10 Formulating Algorithms with Top-Down, Stepwise Refinement: Case Study 3 (Nested Control Structures)
- 8.11 Assignment Operators
- 8.12 Increment and Decrement Operators
- 8.13 Note on Data Types
- 8.14 Web Resources

Objectives

- In this lesson, you will learn:
 - To understand basic problem-solving techniques.
 - To be able to develop algorithms through the process of top-down, stepwise refinement.
 - To be able to use the `if` and `if...else` selection statements to choose among alternative actions.
 - To be able to use the `while` repetition statement to execute statements in a script repeatedly.
 - To understand counter-controlled repetition and sentinel-controlled repetition.
 - To be able to use the increment, decrement and assignment operators.

8.1 Introduction

- Writing a script
 - Thorough understanding of problem
 - Carefully planned approach
 - Understand the types of building blocks that are available
 - Employ proven program-construction principles

8.2 Algorithms

- Actions to be executed
- Order in which the actions are to be executed

8.3 Pseudocode

- Artificial
- Informal
- Helps programmers develop algorithms

8.4 Control Structures

- Sequential execution
 - Statements execute in the order they are written
- Transfer of control
 - Next statement to execute may not be the next one in sequence
- Three control structures
 - Sequence structure
 - Selection structure
 - if
 - if...else
 - switch
 - Repetition structure
 - while
 - do...while
 - for
 - for...in

8.4 Control Structures

- Flowchart
 - Graphical representation of algorithm or portion of algorithm
 - Flowlines
 - Indicate the order the actions of the algorithm execute
 - Rectangle symbol
 - Indicate any type of action
 - Oval symbol
 - A complete algorithm
 - Small circle symbol
 - A portion of algorithm
 - Diamond symbol
 - Indicates a decision is to be made

8.4 Control Structures

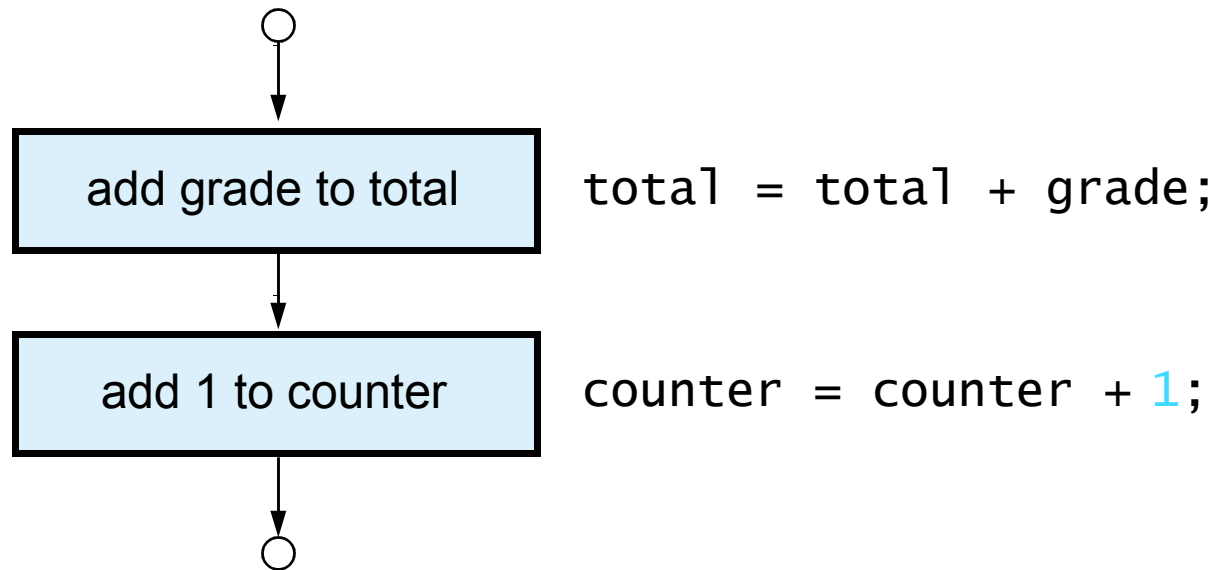


Fig. 8.1 Flowcharting JavaScript's sequence structure.

8.4 Control Structures

JavaScript Keywords				
break	case	catch	continue	default
delete	do	else	finally	for
function	if	in	instanceof	new
return	switch	this	throw	try
typeof	var	void	while	with
<i>Keywords that are reserved but not currently used by JavaScript</i>				
abstract	boolean	byte	char	class
const	debugger	double	enum	export
extends	final	float	goto	implements
import	int	interface	long	native
package	private	protected	public	short
static	super	synchronized	throws	transient
volatile				

Fig. 8.2 JavaScript keywords.

8.5 `if` Selection Statement

- Single-entry/single-exit structure
- Indicate action only when the condition evaluates to `true`

8.5 if Selection Statement

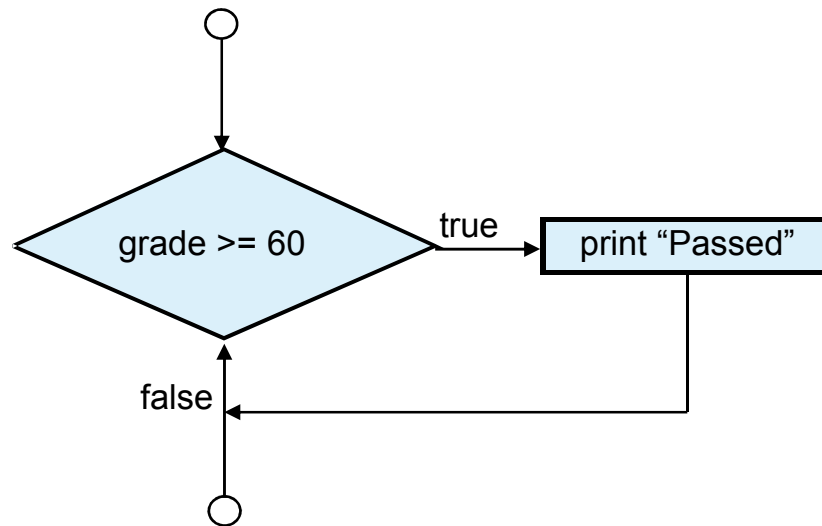


Fig. 8.3 Flowcharting the single-selection if statement.

8.6 if...else Selection Statement

- Indicate different actions to be perform when condition is true or false
- Conditional operator (?:)
 - JavaScript's only ternary operator
 - Three operands
 - Forms a conditional expression
- Dangling-else problem

8.6 if...else Selection Statement

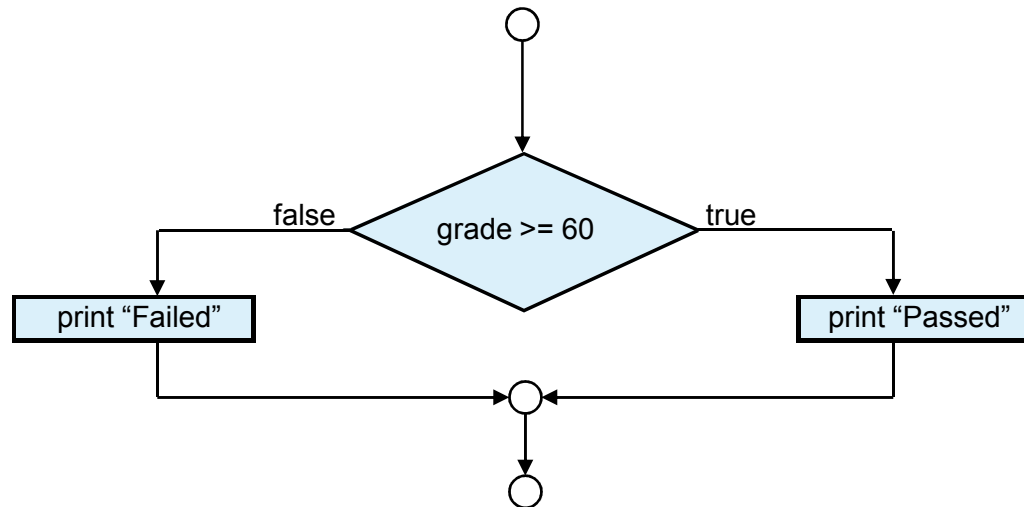


Fig. 8.4 Flowcharting the double-selection if...else statement.

8.7 while Repetition Statement

- Repetition structure (loop)
 - Repeat action while some condition remains true

8.7 while Repetition Statement

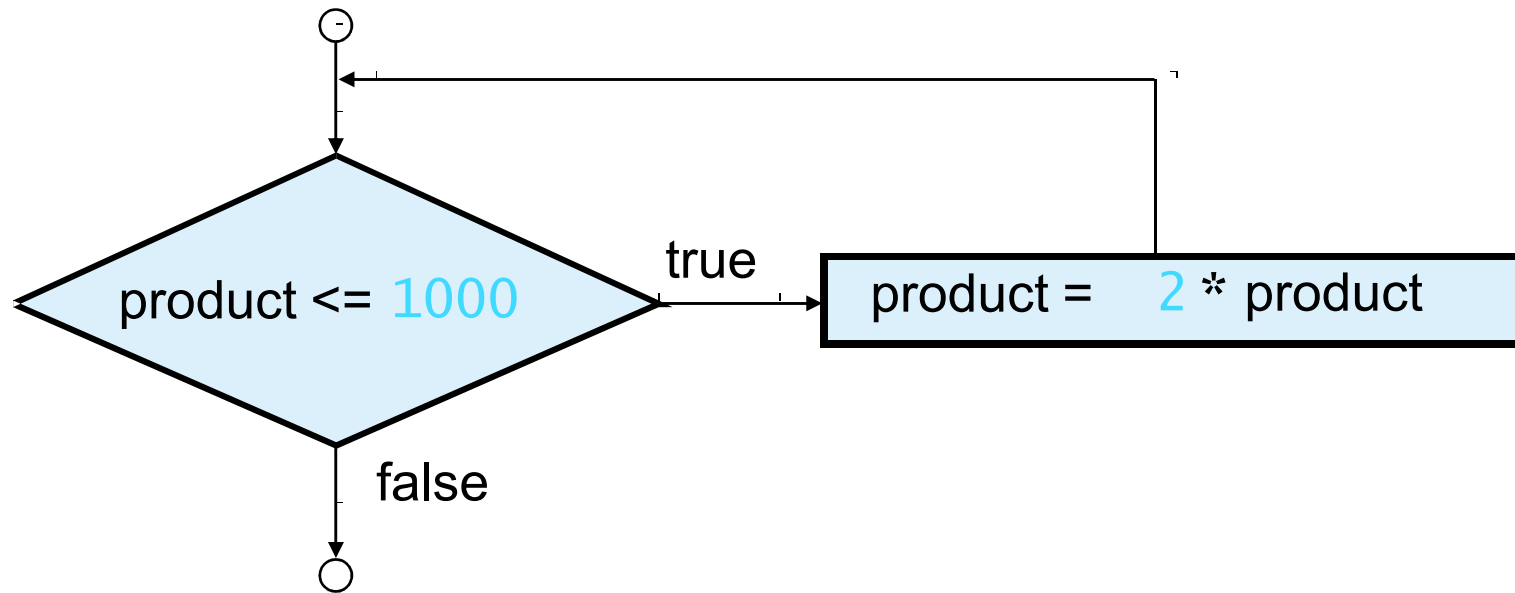


Fig. 8.5 Flowcharting the while repetition statement.

8.8 Formulating Algorithms: Case Study 1 (Counter-Controlled Repetition)

- Counter-controlled repetition
 - Counter
 - Control the number of times a set of statements executes
 - Definite repetition


```
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. 8.7: average.html -->
6 <!-- Class Average Program -->
7
8 <html xmlns = "http://www.w3.org/1999/xhtml">
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
20         // Initialization Phase
21         total = 0;         // clear total
22         gradeCounter = 1;  // prepare to loop
23
```

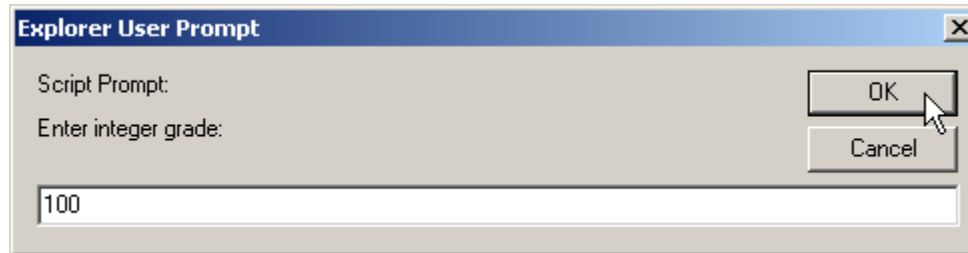
average.html
(1 of 3)

```
24 // Processing Phase
25 while ( gradeCounter <= 10 ) { // loop 10 times
26
27     // prompt for input and read grade from user
28     grade = window.prompt( "Enter integer grade:", "0" );
29
30     // convert grade from a string to an integer
31     gradeValue = parseInt( grade );
32
33     // add gradeValue to total
34     total = total + gradeValue;
35
36     // add 1 to gradeCounter
37     gradeCounter = gradeCounter + 1;
38 }
39
40 // Termination Phase
41 average = total / 10; // calculate the average
42
43 // display average of exam grades
44 document.writeln(
45     "<h1>Class average is " + average + "</h1>" );
46 // -->
47 </script>
```

average.html
(2 of 3)

```
48
49 </head>
50 <body>
51   <p>Click Refresh (or Reload) to run the script again</p>
52 </body>
53 </html>
```

average.html
(3 of 3)



8.9 Formulating Algorithms with Top-Down, Stepwise Refinement: Case Study 2 (Sentinel-Controlled Repetition)

- Indefinite repetition
 - Sentinel value

```

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. 8.9: average2.html      -->
6 <!-- Sentinel-controlled Repetition -->
7
8 <html xmlns = "http://www.w3.org/1999/xhtml">
9   <head>
10     <title>Class Average Program:
11       Sentinel-controlled Repetition</title>
12
13     <script type = "text/javascript">
14       <!--
15         var gradeCounter, // number of grades entered
16           gradeValue,    // grade value
17           total,         // sum of grades
18           average,      // average of all grades
19           grade;        // grade typed by user
20
21       // Initialization phase
22       total = 0;        // clear total
23       gradeCounter = 0; // prepare to loop
24

```

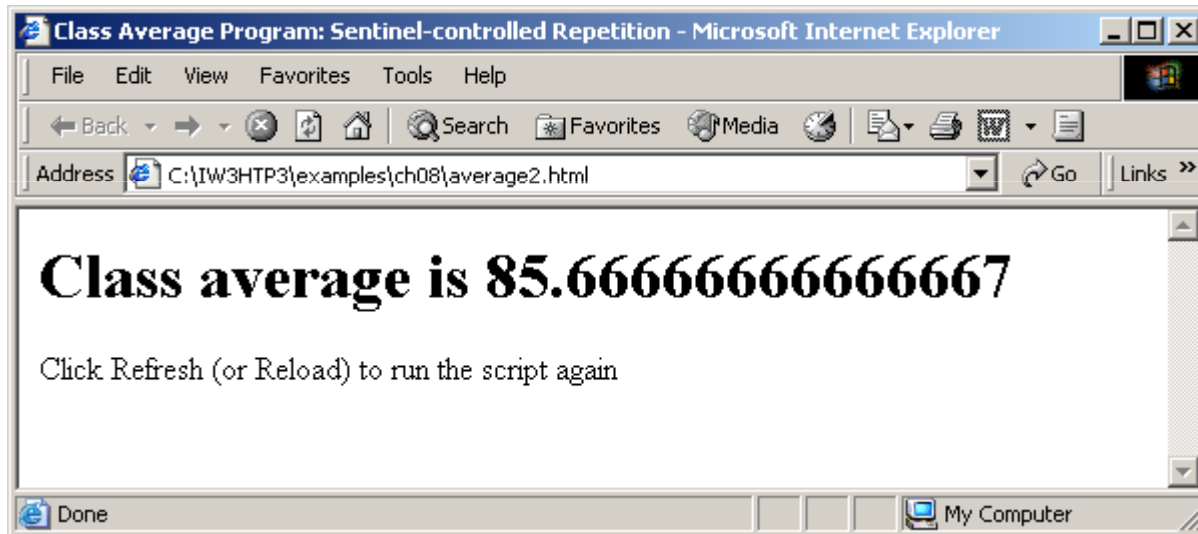
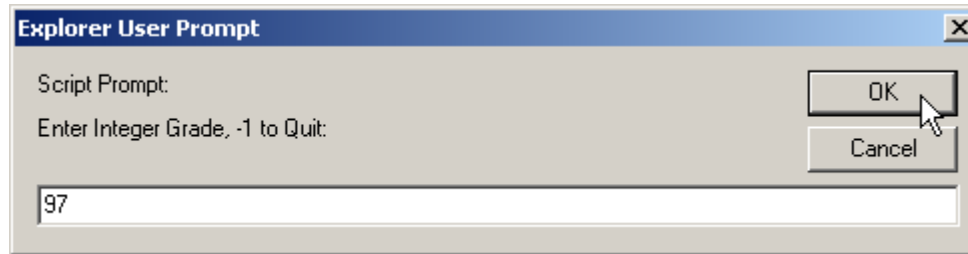
average2.html
(1 of 3)

```
25 // Processing phase
26 // prompt for input and read grade from user
27 grade = window.prompt(
28     "Enter Integer Grade, -1 to Quit:", "0" );
29
30 // convert grade from a string to an integer
31 gradeValue = parseInt( grade );
32
33 while ( gradeValue != -1 ) {
34     // add gradeValue to total
35     total = total + gradeValue;
36
37     // add 1 to gradeCounter
38     gradeCounter = gradeCounter + 1;
39
40     // prompt for input and read grade from user
41     grade = window.prompt(
42         "Enter Integer Grade, -1 to Quit:", "0" );
43
44     // convert grade from a string to an integer
45     gradeValue = parseInt( grade );
46 }
47
```

average2.html
(2 of 3)

```
48 // Termination phase
49 if ( gradeCounter != 0 ) {
50     average = total / gradeCounter;
51
52     // display average of exam grades
53     document.writeln(
54         "<h1>Class average is " + average + "</h1>" );
55     }
56     else
57     document.writeln( "<p>No grades were entered</p>" );
58     // -->
59     </script>
60 </head>
61
62 <body>
63     <p>Click Refresh (or Reload) to run the script again</p>
64 </body>
65 </html>
```

average2.html
(3 of 3)



8.10 Formulating Algorithms with Top-Down, Stepwise Refinement: Case Study 3 (Nested Control Structures)

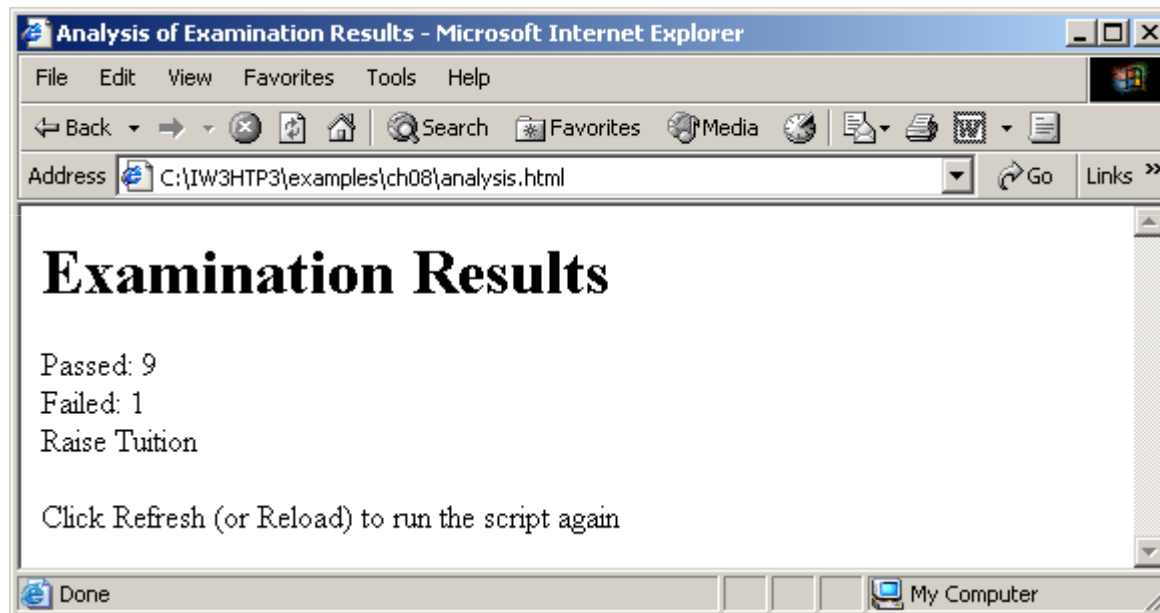
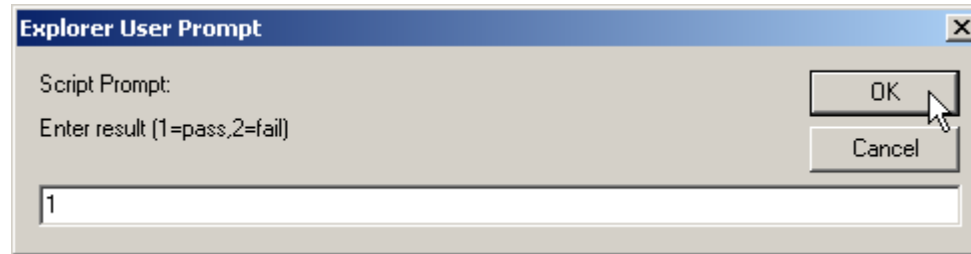
- Consider problem
- Make observations
- Top-down, stepwise refinement

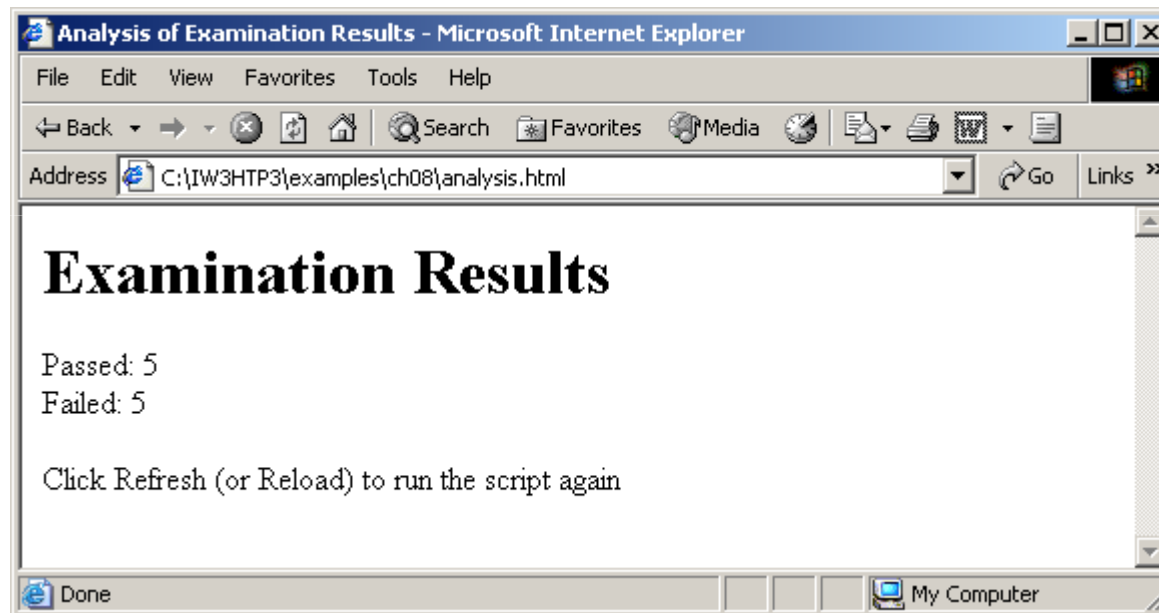
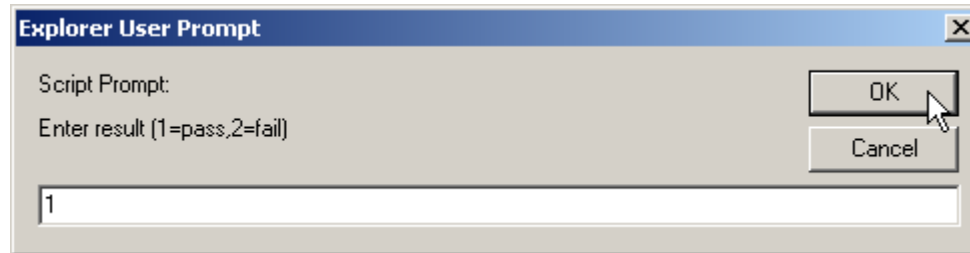
```
1 <?xml version = "1.0"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 8.11: analysis.html -->
6 <!-- Analyzing Exam Results -->
7
8 <html xmlns = "http://www.w3.org/1999/xhtml">
9   <head>
10     <title>Analysis of Examination Results</title>
11
12     <script type = "text/javascript">
13       <!--
14       // initializing variables in declarations
15       var passes = 0,      // number of passes
16           failures = 0,   // number of failures
17           student = 1,   // student counter
18           result;        // one exam result
19
20       // process 10 students; counter-controlled loop
21       while ( student <= 10 ) {
22         result = window.prompt(
23           "Enter result (1=pass,2=fail)", "0" );
24
```

analysis.html
(1 of 2)

```
25     if ( result == "1" )
26         passes = passes + 1;
27     else
28         failures = failures + 1;
29
30     student = student + 1;
31 }
32
33 // termination phase
34 document.writeln( "<h1>Examination Results</h1>" );
35 document.writeln(
36     "Passed: " + passes + "<br />Failed: " + failures );
37
38     if ( passes > 8 )
39         document.writeln( "<br />Raise Tuition" );
40     // -->
41 </script>
42
43 </head>
44 <body>
45     <p>Click Refresh (or Reload) to run the script again</p>
46 </body>
47 </html>
```

analysis.html
(2 of 2)





8.11 Assignment Operators

- Compound assignment operators
 - Abbreviate assignment expressions

8.11 Assignment Operators

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

Fig. 8.12 Arithmetic assignment operators.

8.12 Increment and Decrement Operators

- Preincrement or predecrement operator
 - Increment of decrement operator placed before a variable
- Postincrement or postdecrement operator
 - Increment of decrement operator placed after a variable

8.12 Increment and Decrement Operators

Operator	Called	Sample expression	Explanation
++	preincrement	++a	Increment a by 1, then use the new value of a in the expression in which a resides.
++	postincrement	a++	Use the current value of a in the expression in which a resides, then increment a by 1.
--	predecrement	--b	Decrement b by 1, then use the new value of b in the expression in which b resides.
--	postdecrement	b--	Use the current value of b in the expression in which b resides, then decrement b by 1.

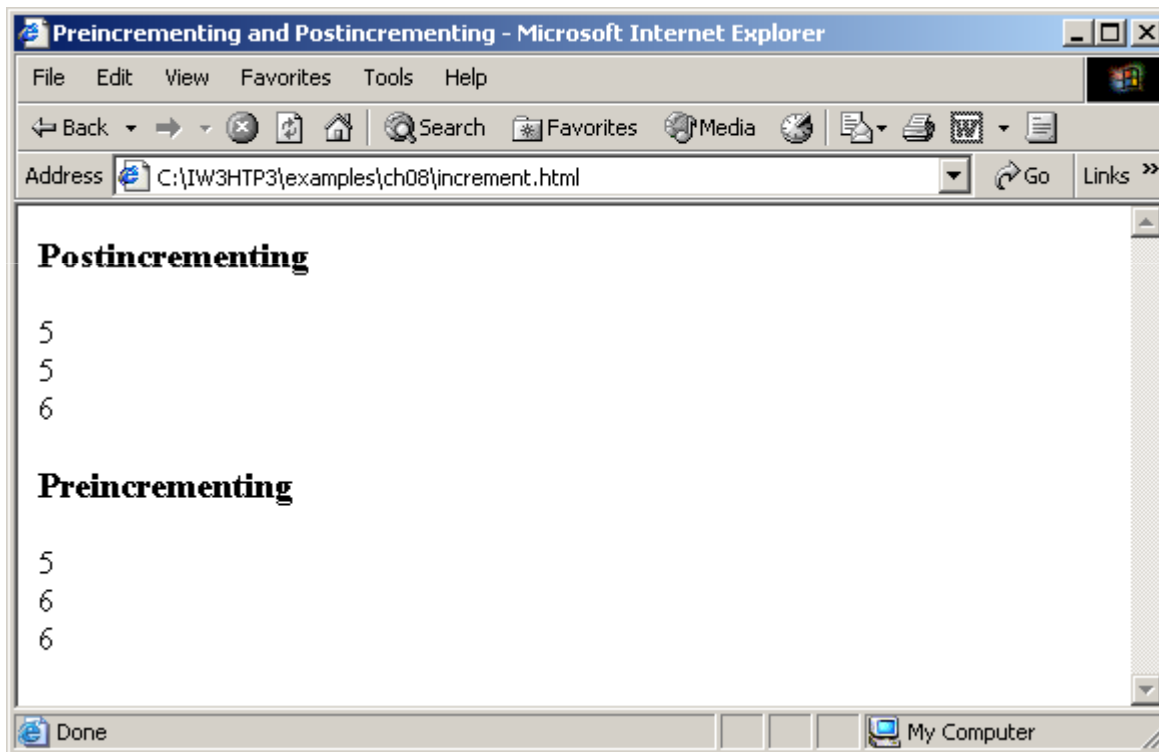
Fig. 8.13 increment and decrement operators.

```
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. 8.14: increment.html -->
6 <!-- Preincrementing and Postincrementing -->
7
8 <html xmlns = "http://www.w3.org/1999/xhtml">
9   <head>
10     <title>Preincrementing and Postincrementing</title>
11
12     <script type = "text/javascript">
13       <!--
14       var c;
15
16       c = 5;
17       document.writeln( "<h3>Postincrementing</h3>" );
18       document.writeln( c );           // print 5
19       // print 5 then increment
20       document.writeln( "<br />" + c++ );
21       document.writeln( "<br />" + c ); // print 6
22
23       c = 5;
24       document.writeln( "<h3>Preincrementing</h3>" );
25       document.writeln( c );           // print 5
```

increment.html
(1 of 2)

```
26 // increment then print 6
27 document.writeln( "<br />" + ++c );
28 document.writeln( "<br />" + c ); // print 6
29 // -->
30 </script>
31
32 </head><body></body>
33 </html>
```

increment.html
(2 of 2)



8.12 Increment and Decrement Operators

Operator	Associativity	Type
++ --	right to left	unary
* / %	left to right	multiplicative
+ -	left to right	additive
< <= > >=	left to right	relational
== !=	left to right	equality
?:	right to left	conditional
= += -= *= /= %=	right to left	assignment

Fig. 8.15 Precedence and associativity of the operators discussed so far.

8.13 Note on Data Types

- Loosely typed
 - Automatically converts between values of different types

8.14 Web Resources

- www.javascriptmall.com
- developer.netscape.com/tech/javascript
- www.mozilla.org/js/language

Control Statements II

Outline

- 9.1 Introduction
- 9.2 Essentials of Counter-Controlled Repetition
- 9.3 for Repetition Statement
- 9.4 Examples Using the for Statement
- 9.5 switch Multiple-Selection Statement
- 9.6 do...while Repetition Statement
- 9.7 break and continue Statements
- 9.8 Labeled break and continue Statements
- 9.9 Logical Operators
- 9.10 Summary of Structured Programming
- 9.11 Web Resources

Objectives

- In this lesson, you will learn:
 - To be able to use the `for` and `do...while` repetition statements to execute statements in a program repeatedly.
 - To understand multiple selection using the `switch` selection statement.
 - To be able to use the `break` and `continue` program-control statements.
 - To be able to use the logical operators.

9.1 Introduction

- Continuation of Chapter 8
 - Theory and principles of structured programming

9.2 Essentials of Counter-Controlled Repetition

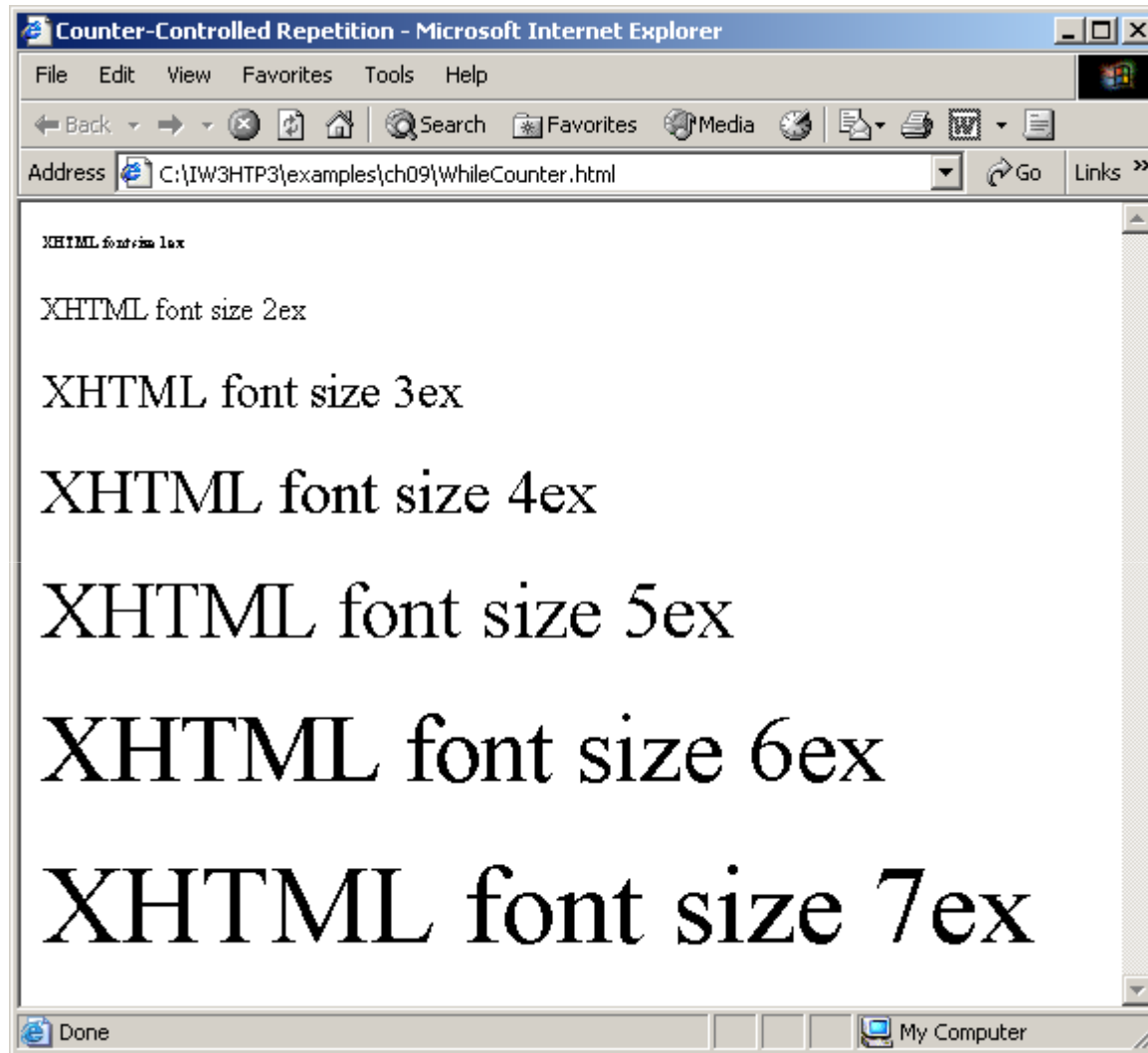
- Counter-controlled repetition
 - Name of a control
 - Initial value
 - Increment or decrement
 - Final value

```
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/xhtml">
9   <head>
10     <title>Counter-Controlled Repetition</title>
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
```

WhileCounter.html (1 of 2)

25 </head><body></body>

26 </html>



WhileCounter.html
(2 of 2)

9.3 for Repetition Statement

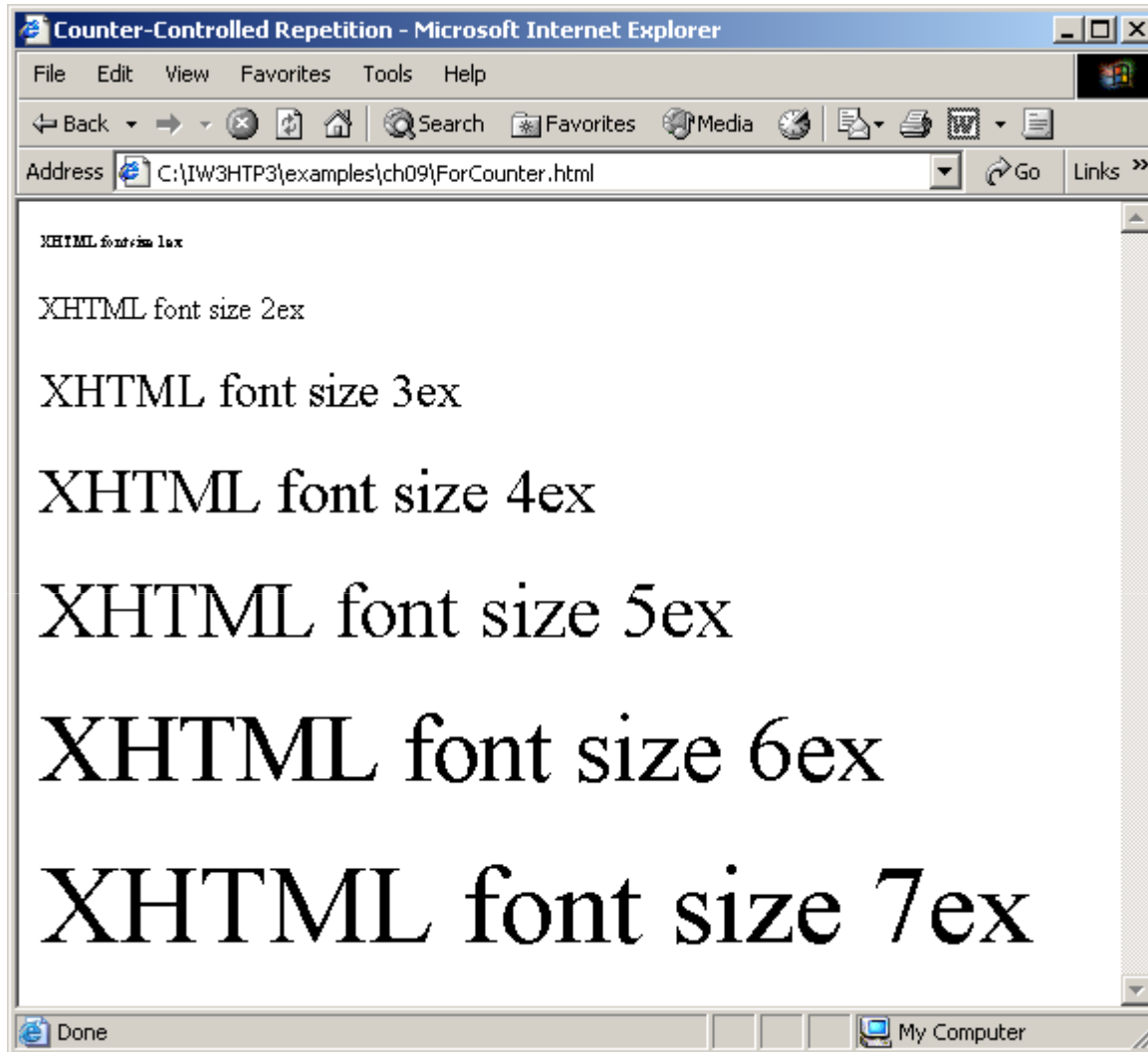
- for repetition statement
 - Handles all the details of counter-controlled repetition
 - for structure header
 - The first line

```

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 statement -->
7
8 <html xmlns = "http://www.w3.org/1999/xhtml">
9   <head>
10     <title>Counter-Controlled Repetition</title>
11
12     <script type = "text/javascript">
13       <!--
14       // Initialization, repetition condition and
15       // incrementing are all included in the for
16       // statement 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>

```

ForCounter.html (1 of 1)



9.3 for Repetition Statement

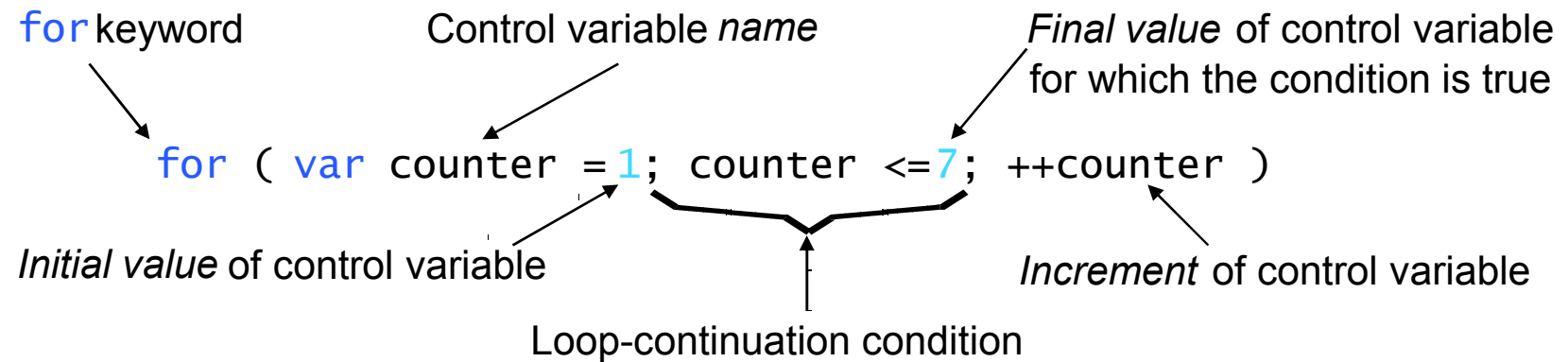


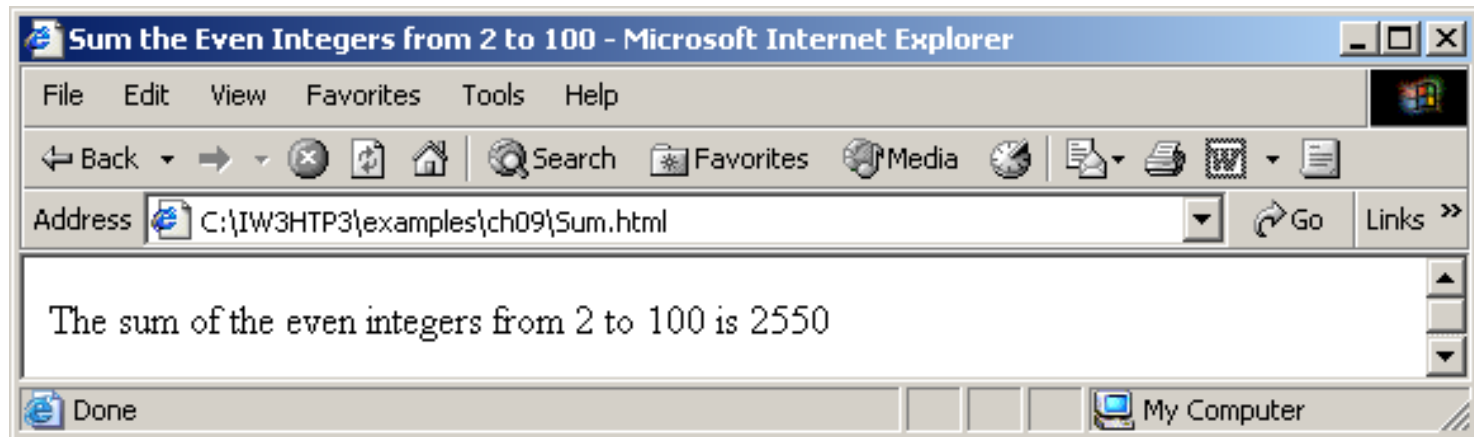
Fig. 9.3 for statement header components.

9.4 Examples Using the for Statement

- Summation with for
- Compound interest calculation with for loop
 - Math object
 - Method pow
 - Method round

```
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 statement -->
7
8 <html xmlns = "http://www.w3.org/1999/xhtml">
9   <head>
10     <title>Sum the Even Integers from 2 to 100</title>
11
12     <script type = "text/javascript">
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>
```

Sum.html
(1 of 1)



```

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 statement -->
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 align = \"left\">Year</th>" );
22      document.writeln(
23        "<th align = \"left\">Amount on deposit</th>" );
24      document.writeln( "</tr></thead>" );
25

```

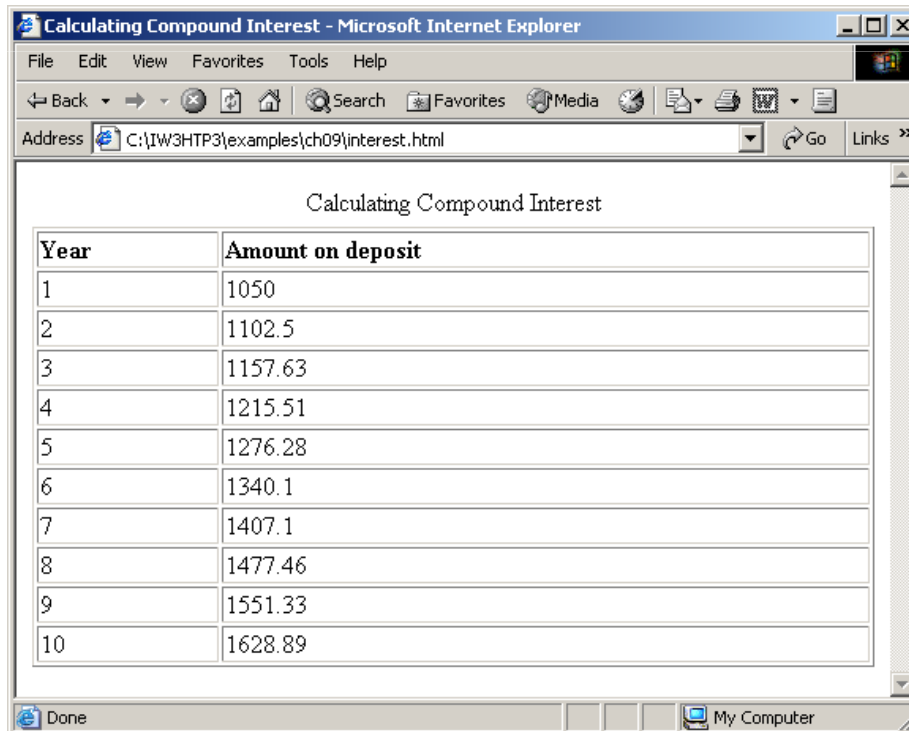
Interest.html (1 of 2)

```

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

```

Interest.html (2 of 2)



Calculating Compound Interest

Year	Amount on deposit
1	1050
2	1102.5
3	1157.63
4	1215.51
5	1276.28
6	1340.1
7	1407.1
8	1477.46
9	1551.33
10	1628.89

9.5 switch Multiple-Selection Statement

- Controlling expression
- Case labels
- Default case

```
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 statement -->
7
8 <html xmlns = "http://www.w3.org/1999/xhtml">
9   <head>
10     <title>Switching between XHTML List Formats</title>
11
12     <script type = "text/javascript">
13       <!--
14       var choice,           // user's choice
15         startTag,         // starting list item tag
16         endTag,           // ending list item tag
17         validInput = true, // indicates if input is valid
18         listType;         // list type as a string
19
20       choice = window.prompt( "Select a list style:\n" +
21         "1 (bullet), 2 (numbered), 3 (lettered)", "1" );
22
```

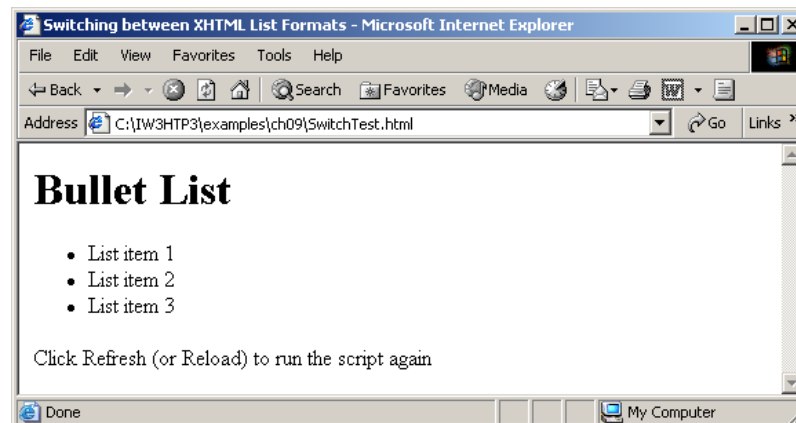
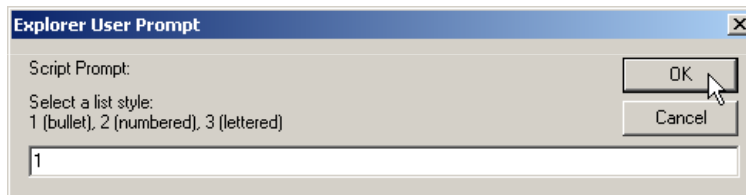
SwitchTest.html (1 of 3)

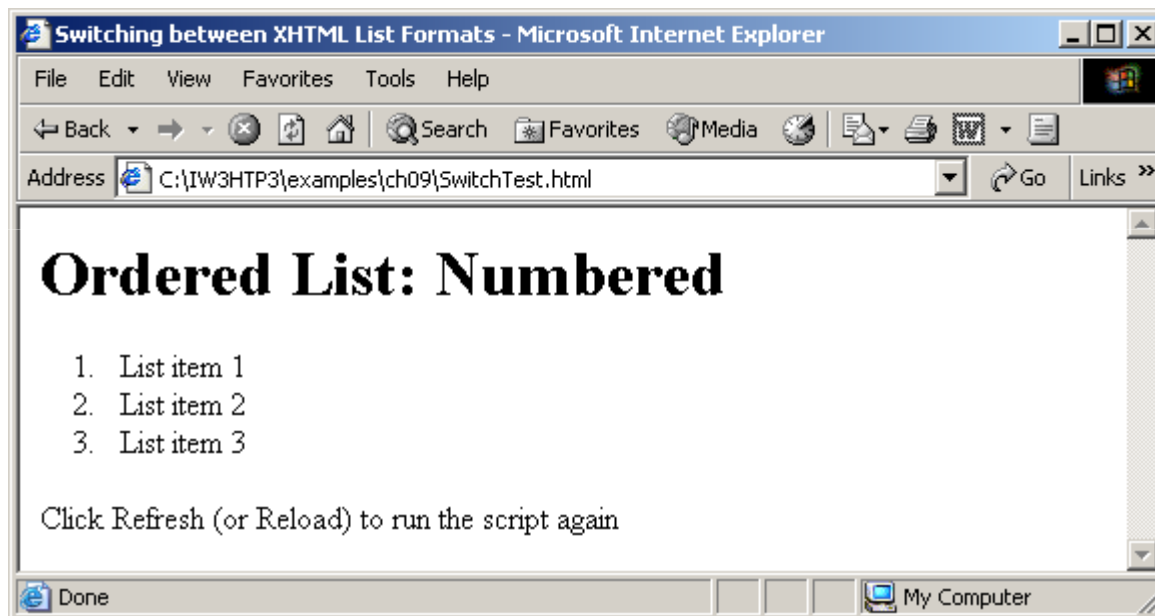
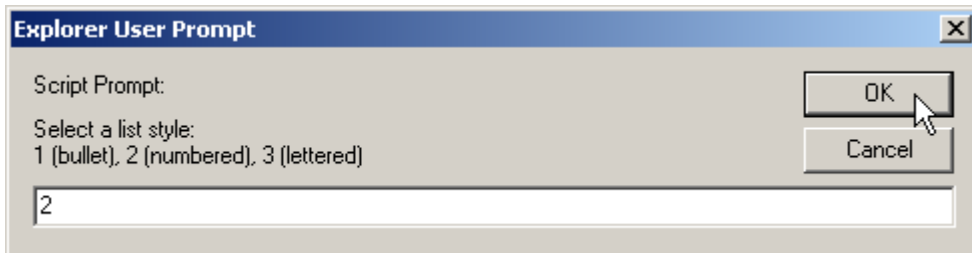

```
23     switch ( choice ) {
24         case "1":
25             startTag = "<ul>";
26             endTag = "</ul>";
27             listType = "<h1>Bullet List</h1>";
28             break;
29         case "2":
30             startTag = "<ol>";
31             endTag = "</ol>";
32             listType = "<h1>Ordered List: Numbered</h1>";
33             break;
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; i <= 3; ++i )
47             document.writeln( "<li>List item " + i + "</li>" );
```

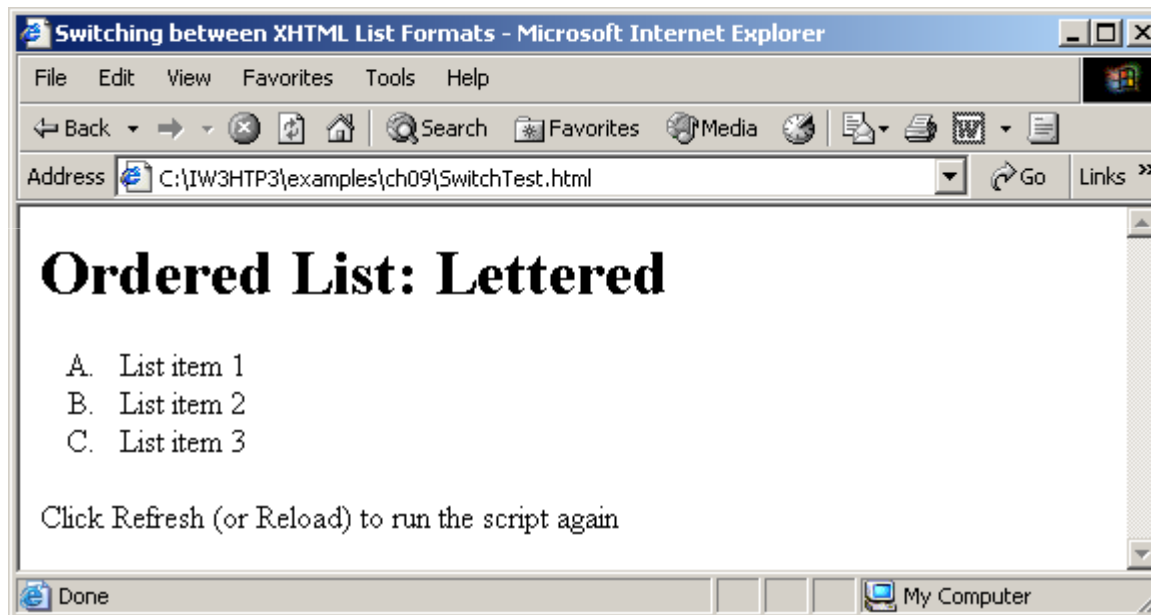
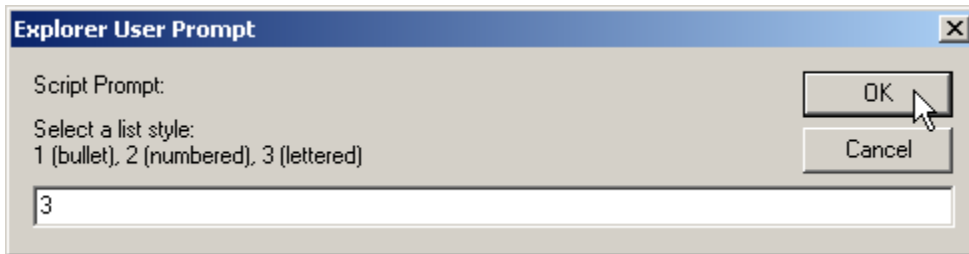
SwitchTest.html
(2 of 3)

```
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>
```

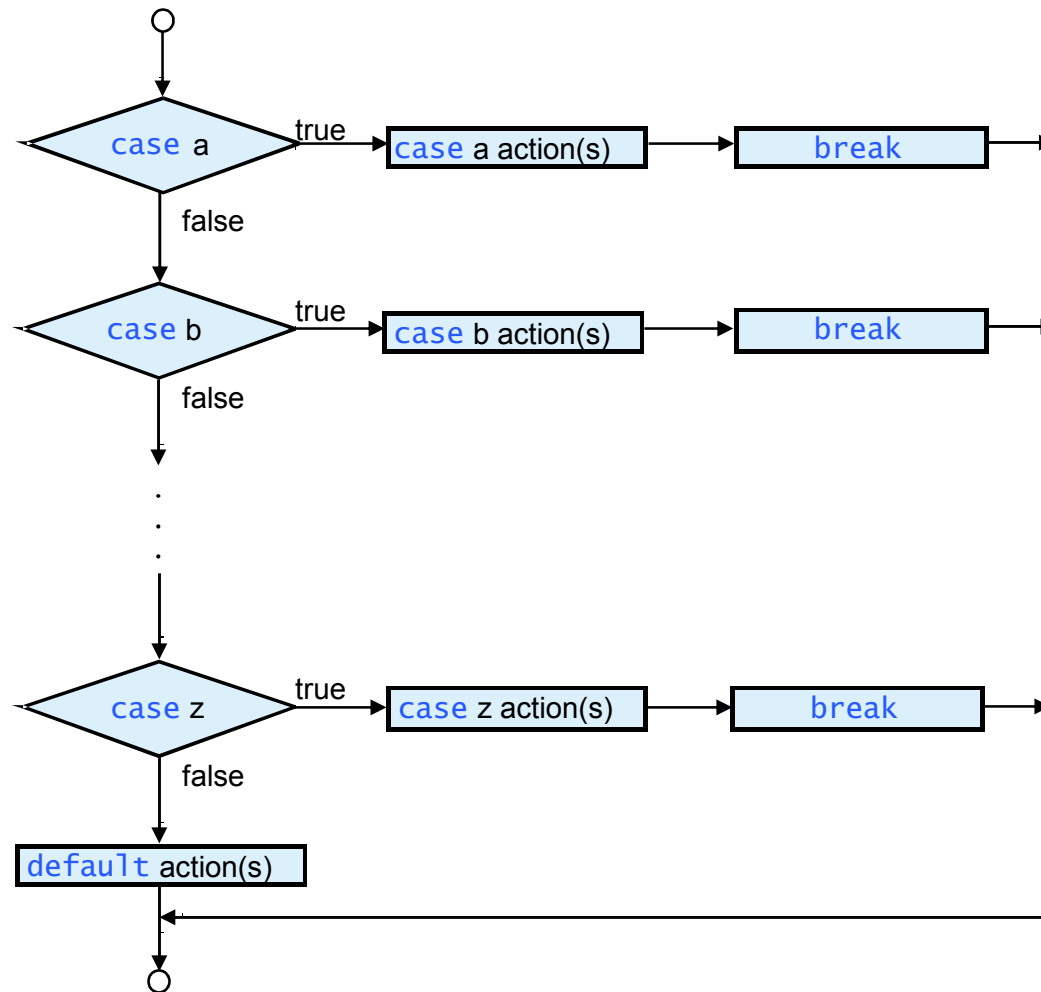
SwitchTest.html (3 of 3)







9.5 switch Multiple-Selection Statement



9.6 do...while Repetition Statement

- Similar to the while statement
- Tests the loop continuation condition after the loop body executes
- Loop body always executes at least once

```

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 statement -->
7
8 <html xmlns = "http://www.w3.org/1999/xhtml">
9   <head>
10    <title>Using the do...while Repetition Statement</title>
11
12    <script type = "text/javascript">
13      <!--
14      var counter = 1;
15
16      do {
17        document.writeln( "<h" + counter + ">This is " +
18          "an h" + counter + " level head" + "</h" +
19          counter + ">" );
20
21        ++counter;
22      } while ( counter <= 6 );
23      // -->
24    </script>

```

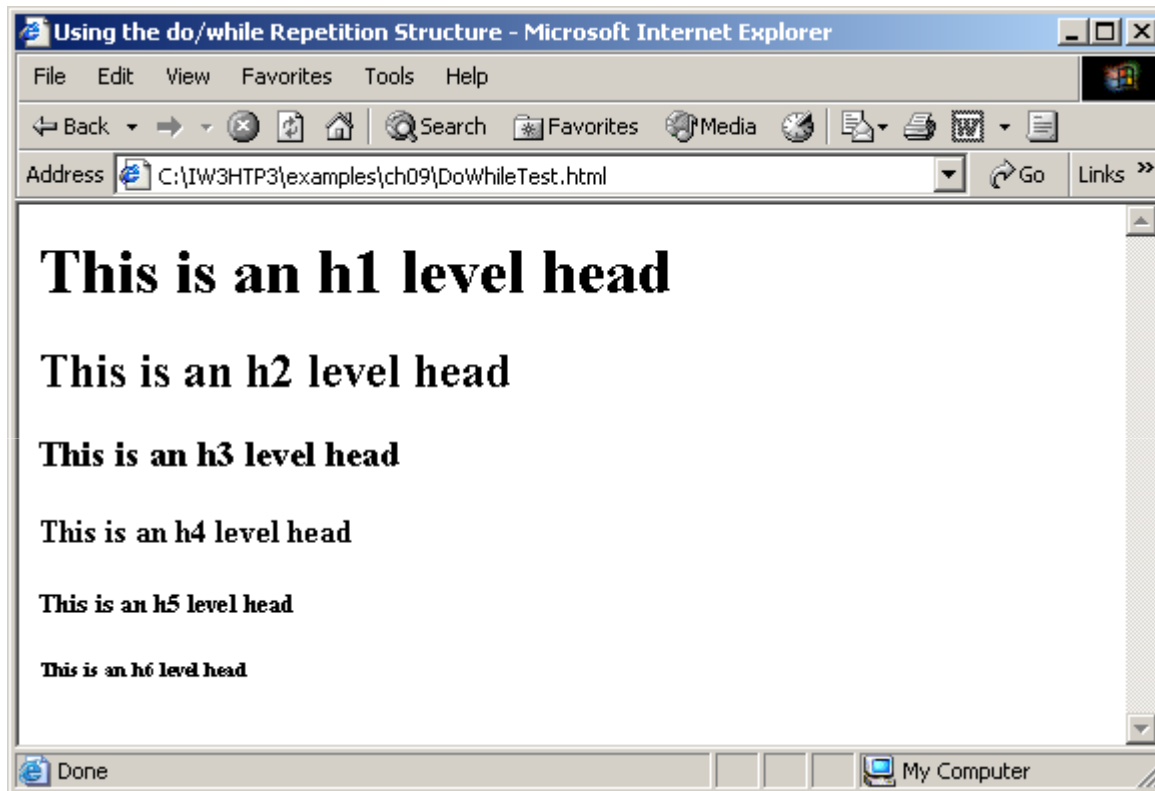
DoWhileTest.html (1 of 2)

25

26 </head><body></body>

27 </html>

DoWhileTest.html (2 of 2)



9.6 do...while Repetition Structure

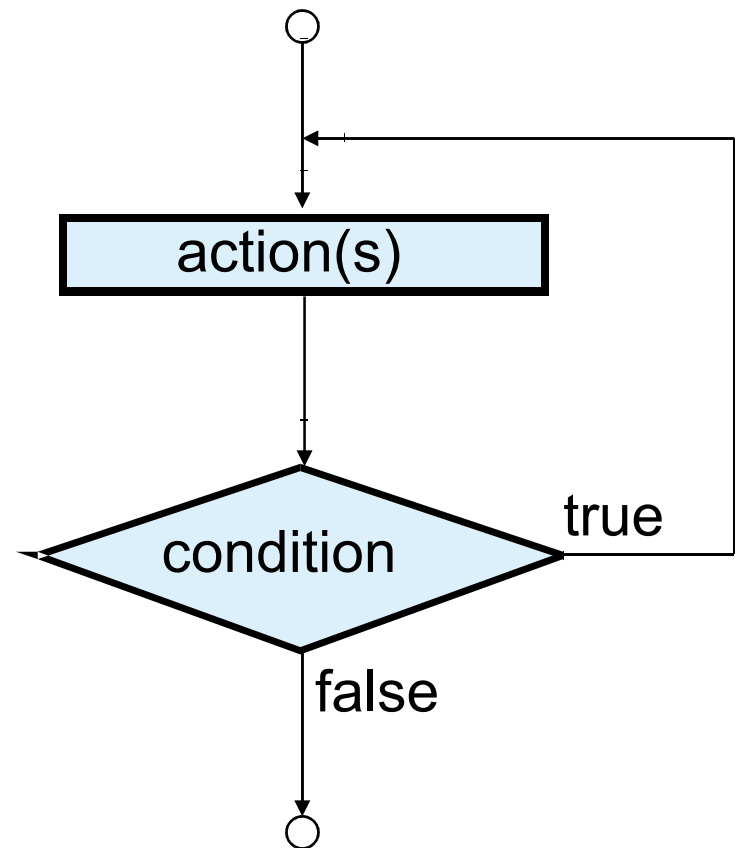


Fig. 9.10 do...while repetition statement flowchart.

9.7 break and continue Statements

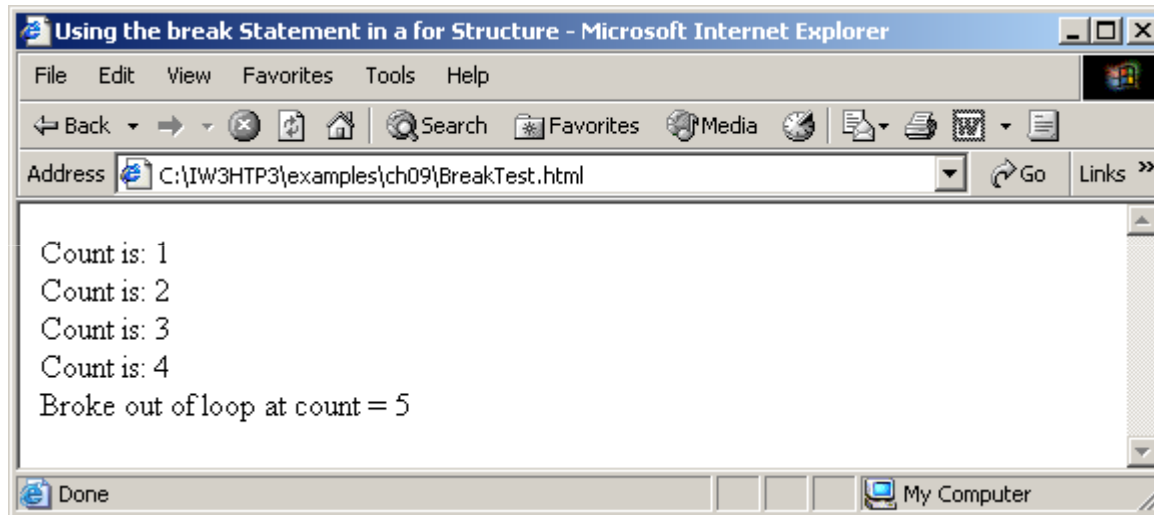
- **break**
 - Immediate exit from the structure
 - Used to escape early from a loop
 - Skip the remainder of a `switch` statement
- **continue**
 - Skips the remaining statements in the body of the structure
 - Proceeds with the next iteration of the loop

```
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 Statement in a for Structure
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
```

BreakTest.html (1 of 2)

```
23     document.writeln(  
24         "Broke out of loop at count = " + count );  
25     // -->  
26 </script>  
27  
28 </head><body></body>  
29 </html>
```

BreakTest.html
(2 of 2)

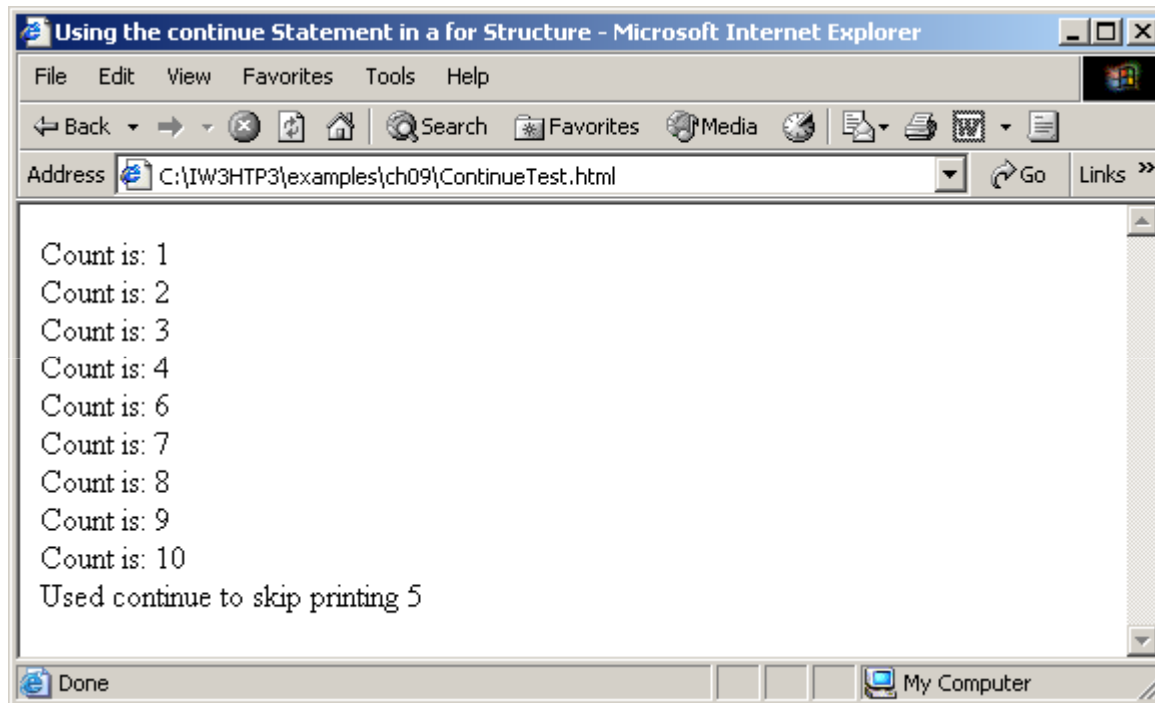


```
1 <?xml version = "1.0"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 9.12: ContinueTest.html -->
6 <!-- Using the break statement -->
7
8 <html xmlns = "http://www.w3.org/1999/xhtml">
9   <head>
10     <title>
11       Using the continue Statement in a for Structure
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
```

ContinueTest.html (1 of 2)

```
24     document.writeln( "Used continue to skip printing 5" );
25     // -->
26     </script>
27
28     </head><body></body>
29 </html>
```

ContinueTest.html (2 of 2)



9.8 Labeled break and continue Statements

- Labeled break statement
 - Break out of a nested set of structures
 - Immediate exit from that structure and enclosing repetition structures
 - Execution resumes with first statement after enclosing labeled statement
- Labeled continue statement
 - Skips the remaining statements in structure's body and enclosing repetition structures
 - Proceeds with next iteration of enclosing labeled repetition structure
 - Loop-continuation test evaluates immediately after the continue statement executes

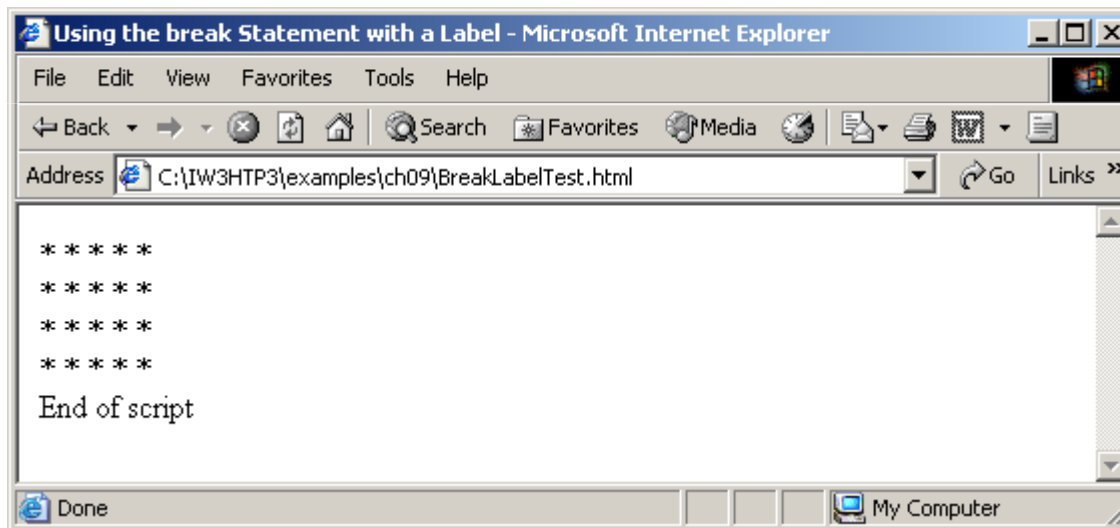
```
1 <?xml version = "1.0"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 9.13: BreakLabelTest.html      -->
6 <!-- Using the break statement with a Label -->
7
8 <html xmlns = "http://www.w3.org/1999/xhtml">
9   <head>
10    <title>Using the break Statement with a Label</title>
11
12    <script type = "text/javascript">
13      <!--
14      stop: { // labeled block
15        for ( var row = 1; row <= 10; ++row ) {
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        }

```

BreakLabelTest.html (1 of 2)


```
26
27     // the following line is skipped
28     document.writeln( "This line should not print" );
29 }
30
31 document.writeln( "End of script" );
32 // -->
33 </script>
34
35 </head><body></body>
36 </html>
```

BreakLabelTest.html (2 of 2)

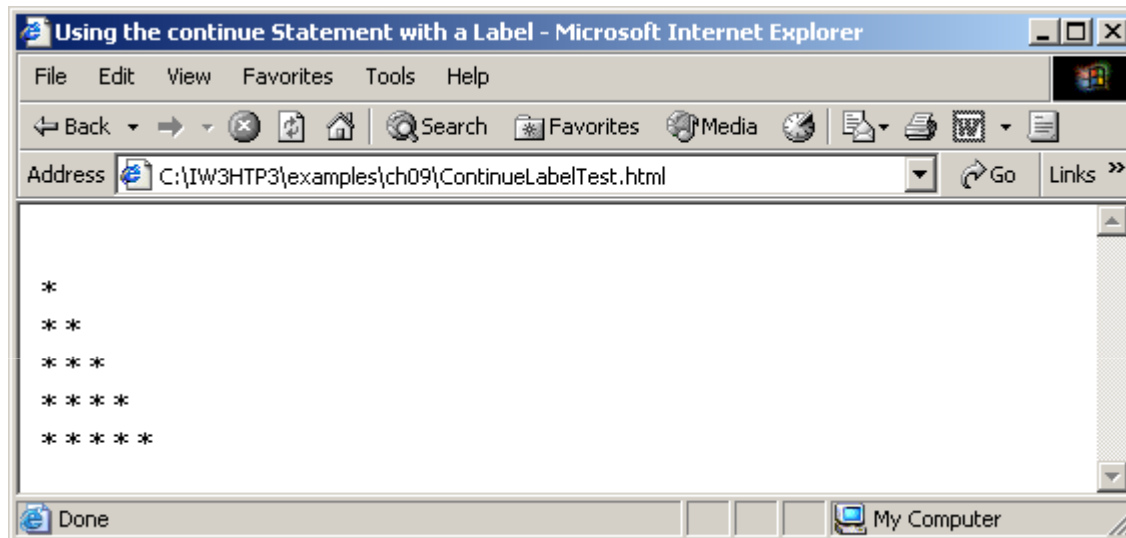


```
1 <?xml version = "1.0"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 9.14: ContinueLabelTest.html -->
6 <!-- Using the continue statement -->
7
8 <html xmlns = "http://www.w3.org/1999/xhtml">
9   <head>
10     <title>Using the continue Statement with a Label</title>
11
12     <script type = "text/javascript">
13       <!--
14       nextRow: // target label of continue statement
15         for ( var row = 1; row <= 5; ++row ) {
16           document.writeln( "<br />" );
17
18           for ( var column = 1; column <= 10; ++column ) {
19
20             if ( column > row )
21               continue nextRow; // next iteration of
22                                 // labeled loop
23
24             document.write( "* " );
25           }
```

ContinueLabelTest.html
(1 of 2)

```
26     }
27     // -->
28     </script>
29
30 </head><body></body>
31 </html>
```

ContinueLabelTest.html
(2 of 2)



9.9 Logical Operators

- More logical operators
 - Logical AND (&&)
 - Logical OR (||)
 - Logical NOT (!)

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.

9.9 Logical Operators

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).

```

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 ) +

```

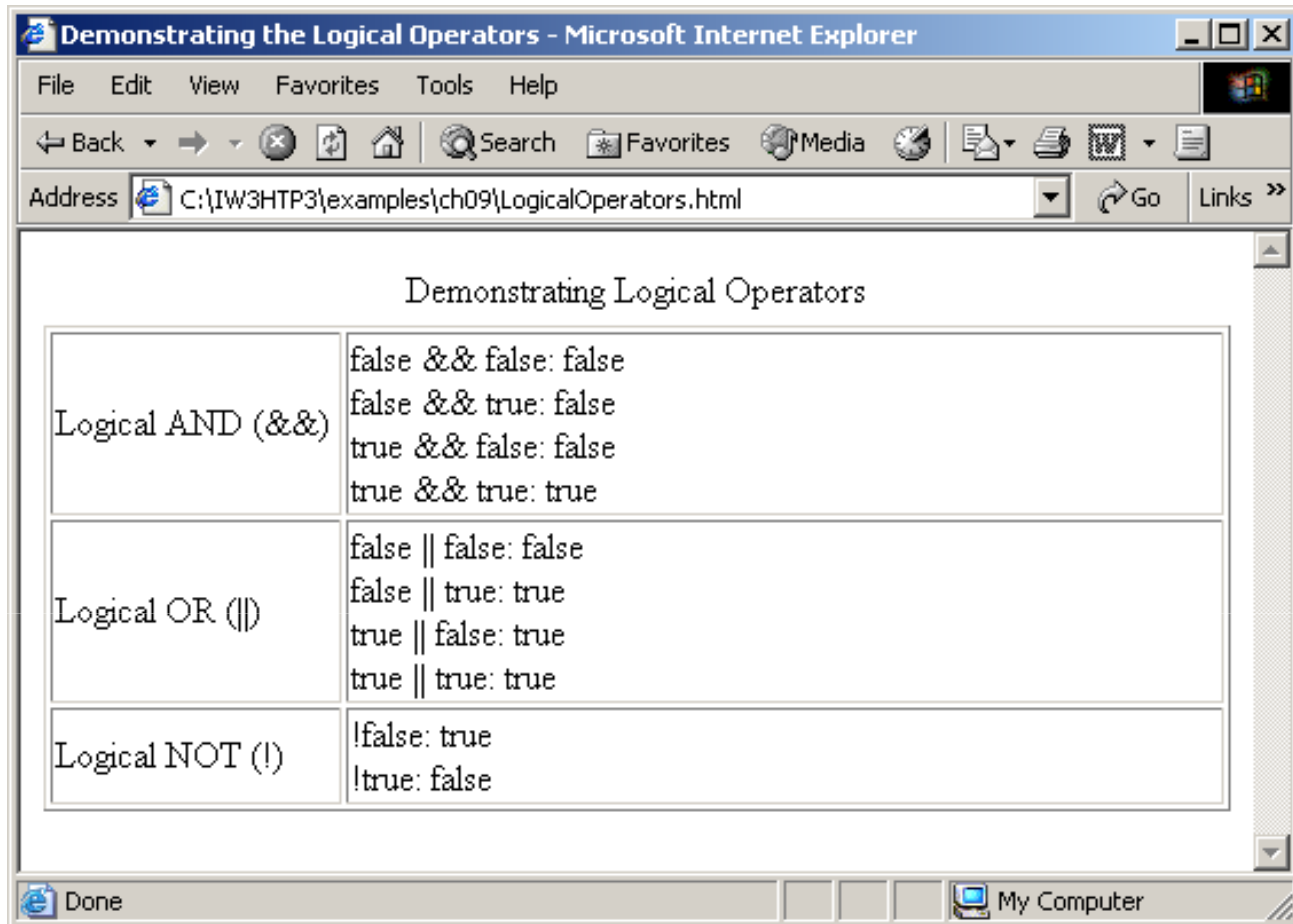
LogicalOperators.html (1 of 2)

```

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>" );
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>

```

LogicalOperators.html
(2 of 2)



9.9 Logical Operators

Operator	Associativity	Type
++ -- !	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

- Flowcharts
 - Reveal the structured nature of programs
- Single-entry/single-exit control structures
 - Only one way to enter and one way to exit each control structure
- Control structure stacking
 - The exit point of one control structure is connected to the entry point of the next control structure

9.10 Summary of Structured Programming

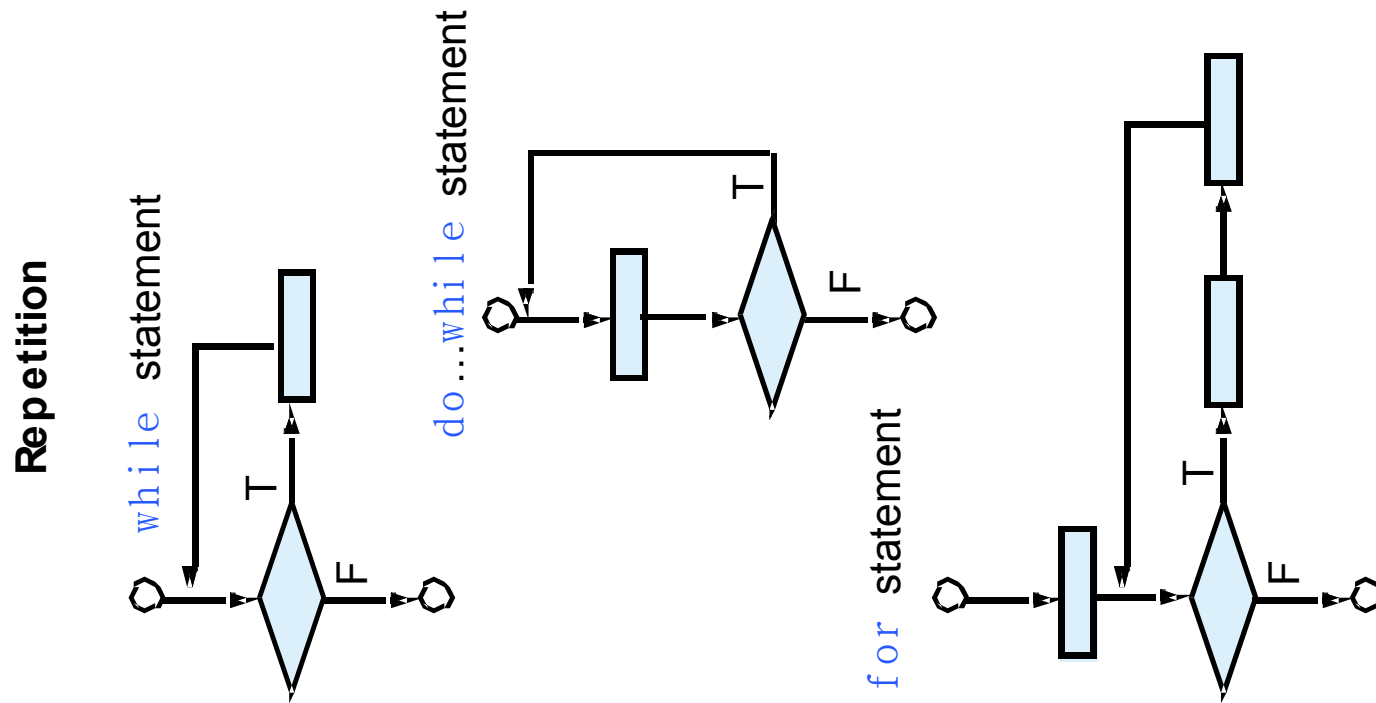


Fig. 9.20 Single-entry/single-exit sequence, selection and repetition structures. (1 of 3)

9.10 Summary of Structured Programming

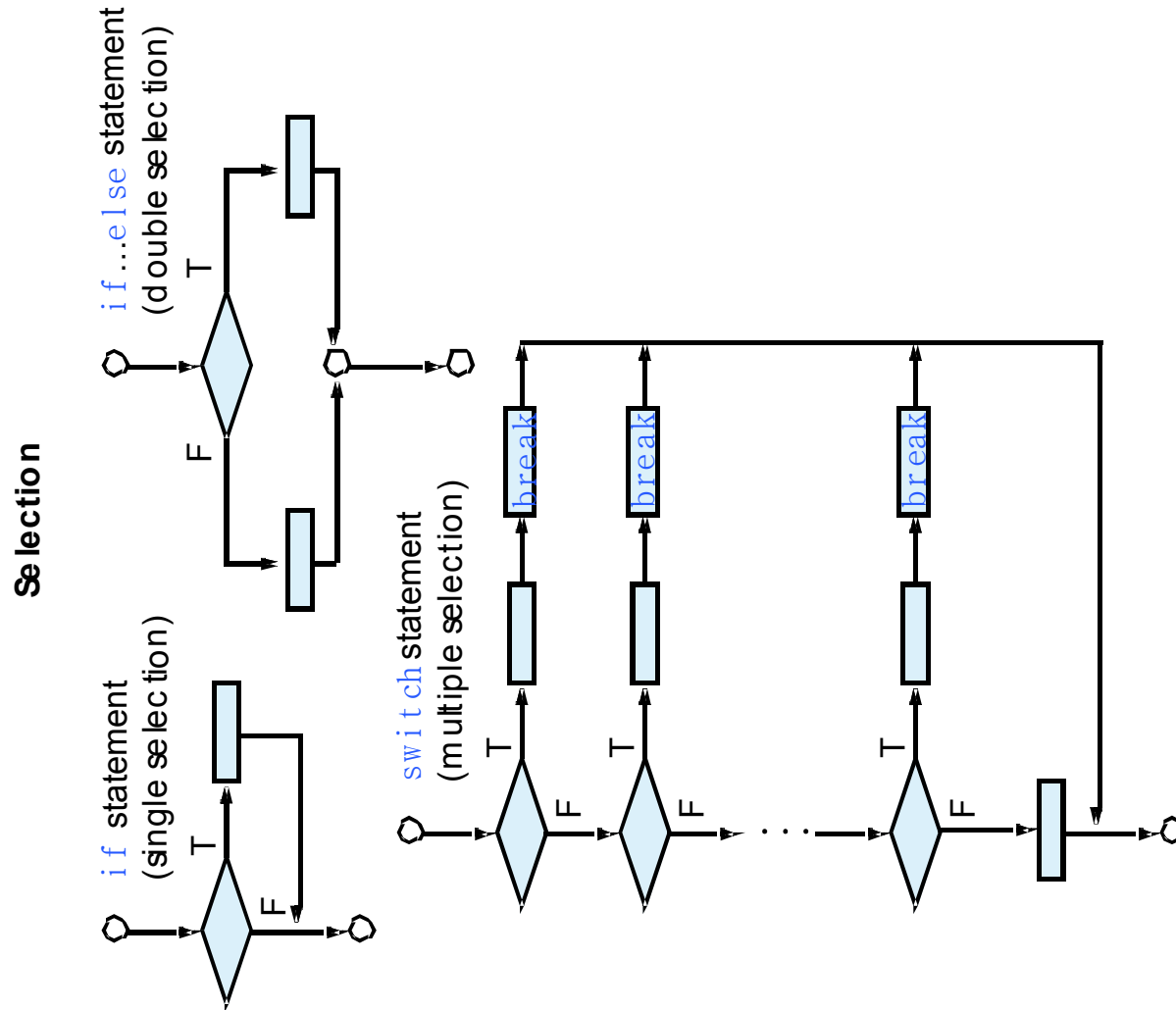


Fig. 9.20 Single-entry/single-exit sequence, selection and repetition structures. (2 of 3)

9.10 Summary of Structured Programming

Sequence

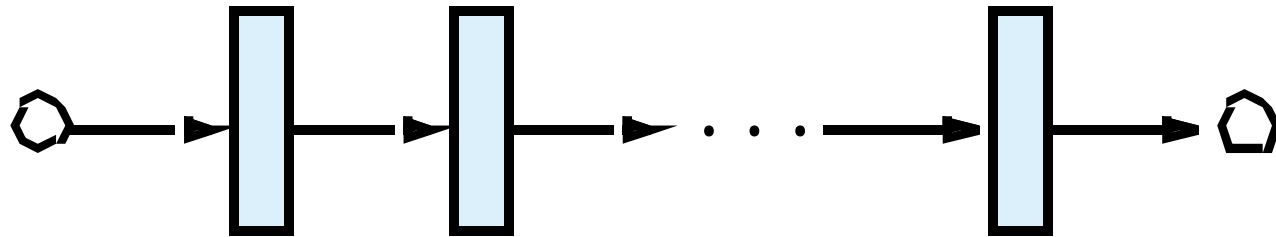


Fig. 9.20 Single-entry/single-exit sequence, selection and repetition structures. (3 of 3)

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, <i>if</i> , <i>if...else</i> , <i>switch</i> , <i>while</i> , <i>do...while</i> or <i>for</i>).
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

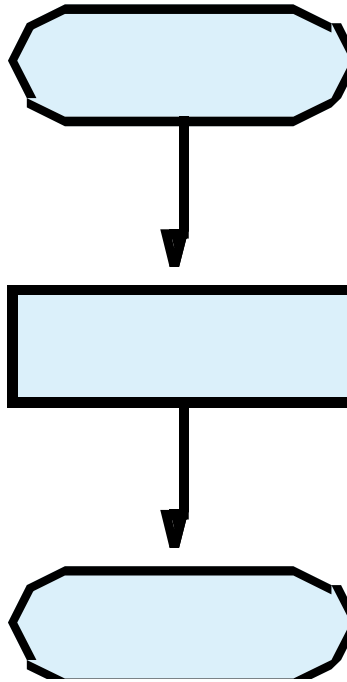


Fig. 9.22 Simplest flowchart.

9.10 Summary of Structured Programming

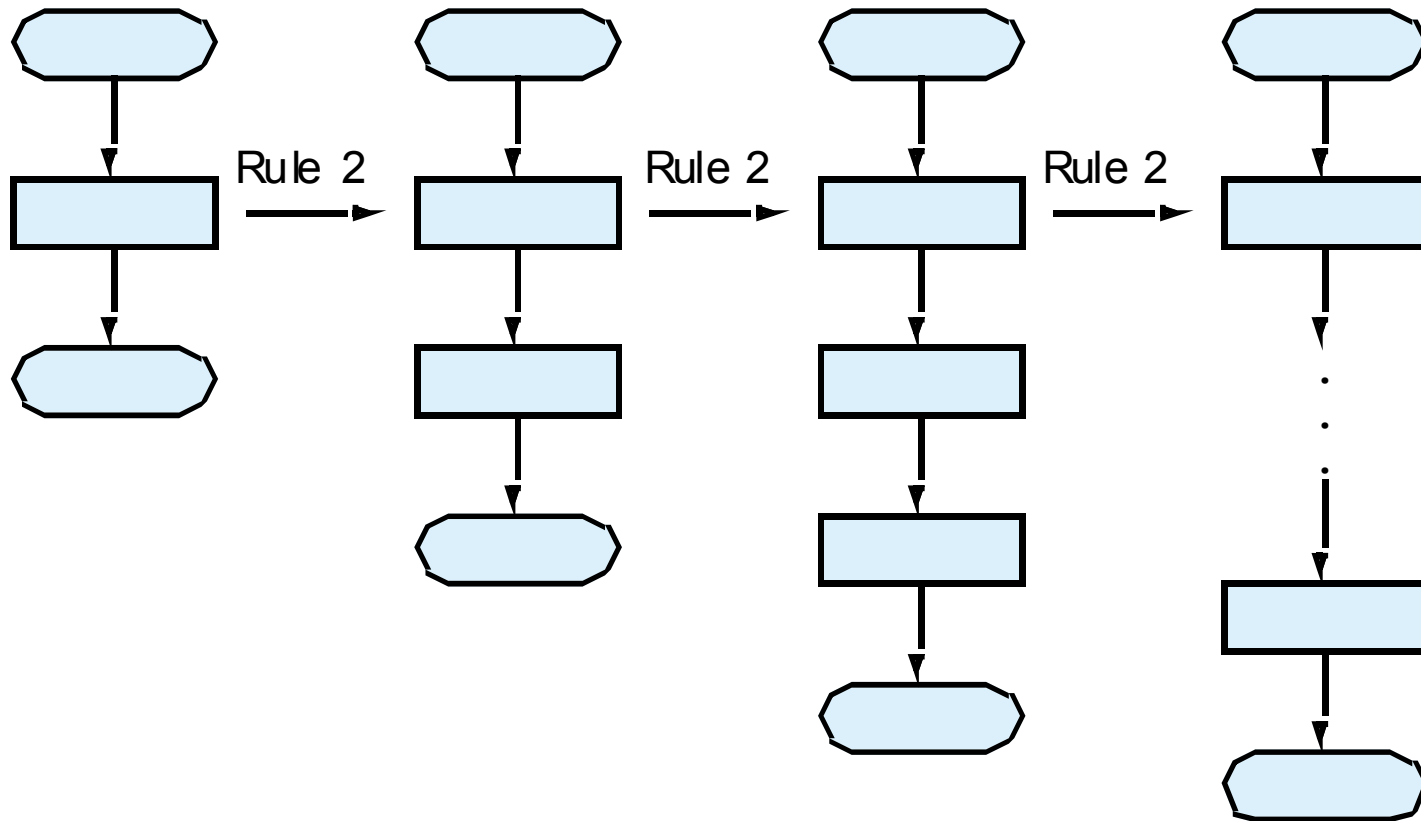


Fig. 9.23 Repeatedly applying rule 2 of Fig. 9.21 to the simplest flowchart.

9.10 Summary of Structured Programming

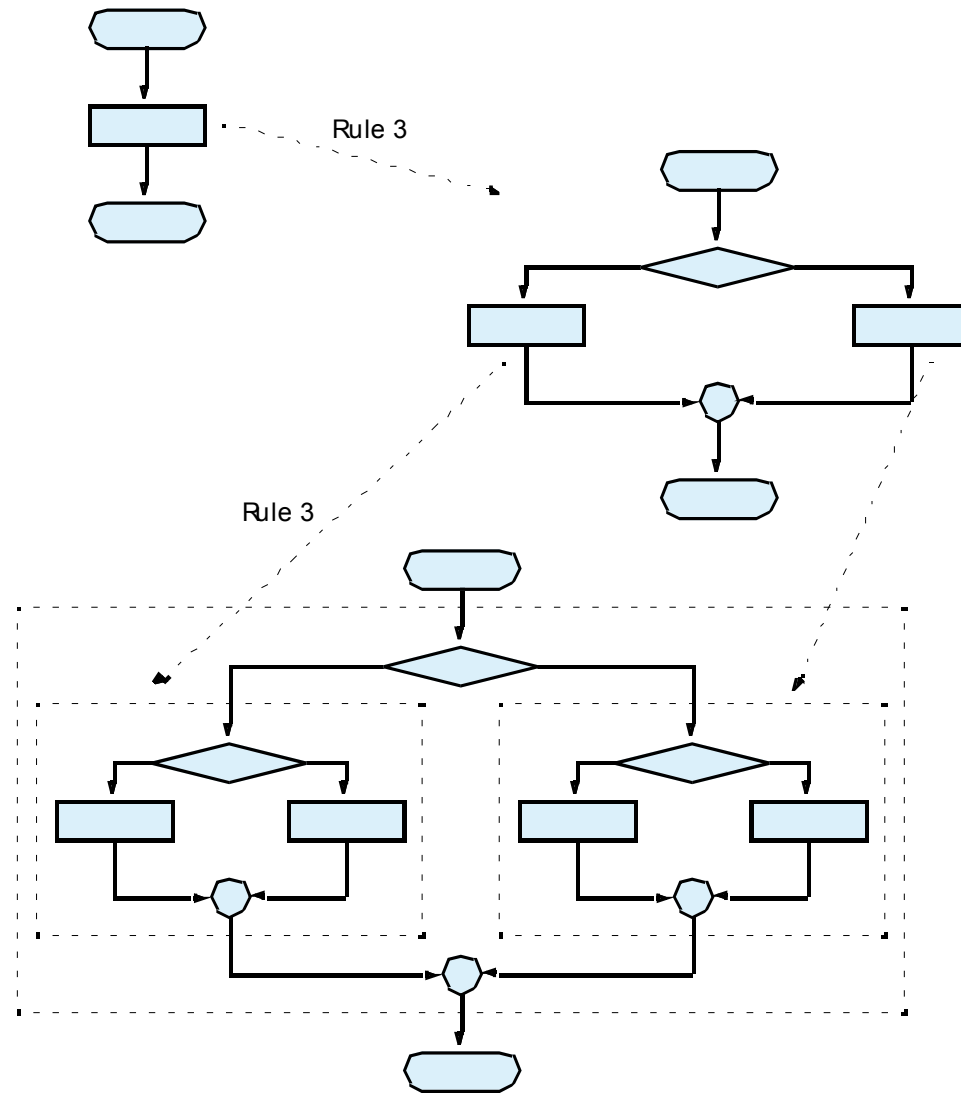
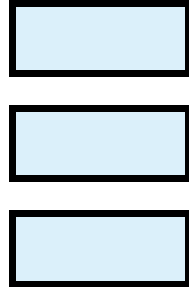


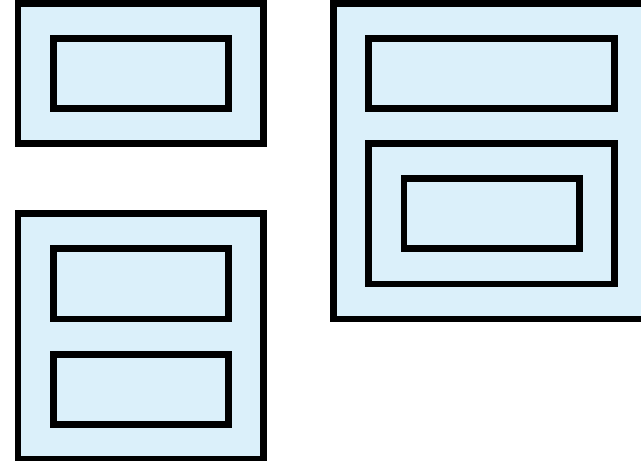
Fig. 9.24 Applying rule 3 of Fig. 9.21 to the simplest flowchart.

9.10 Summary of Structured Programming

Stacked building blocks



Nested building blocks



Overlapping building blocks
(Illegal in structured programs)

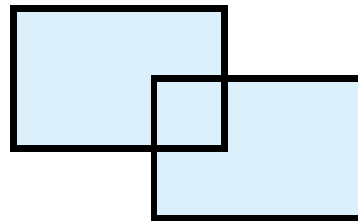


Fig. 9.25 Stacked, nested and overlapped building blocks.

9.10 Summary of Structured Programming

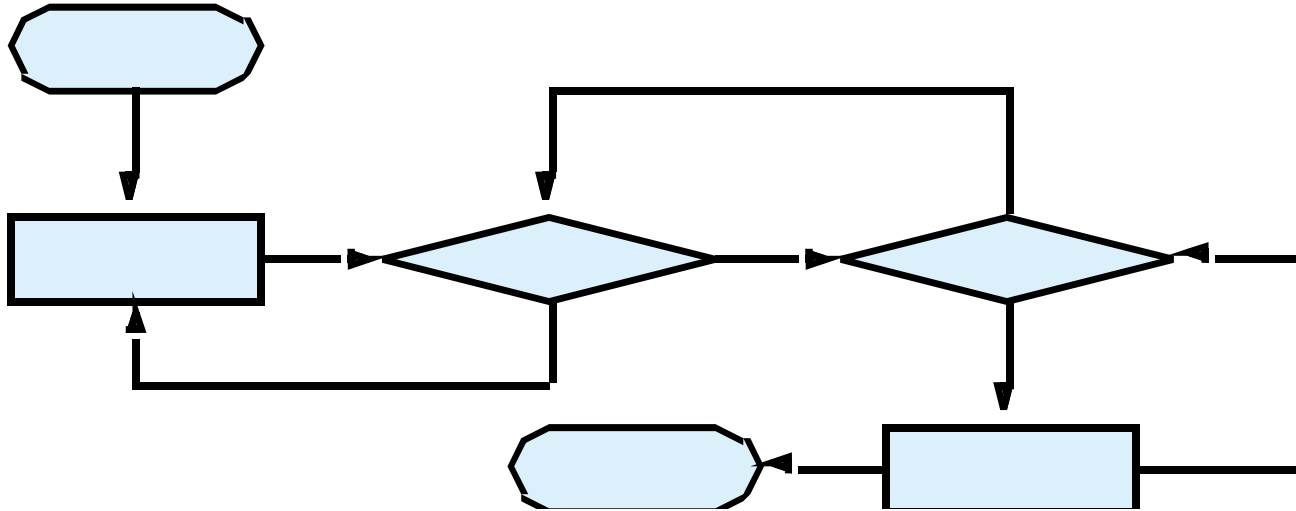


Fig. 9.26 Unstructured flowchart.