7

JavaScript: Control Statements I

Outline

7.4 Control Structures (Cont.)

- **JavaScript provides three selection structures.**
  - **The `if` statement either performs (selects) an action if a condition is true or skips the action if the condition is false.**
    - **Called a single-selection structure because it selects or ignores a single action or group of actions.**
  - **The `if…else` statement performs an action if a condition is true and performs a different action if the condition is false.**
    - **Double-selection structure because it selects between two different actions or group of actions.**
  - **The `switch` statement performs one of many different actions, depending on the value of an expression.**
    - **Multiple-selection structure because it selects among many different actions or groups of actions.**

7.4 Control Structures (Cont.)

- **JavaScript provides four repetition statements, namely, `while`, `do...while`, `for` and `for...in`.**
- **Keywords cannot be used as identifiers (e.g., for variable names).**

Fig. 7.2 | JavaScript keywords.

JavaScript keywords				
break	case	catch	continue	default
delete	do	else	false	finally
for	function	if	in	instanceof
new	null	return	switch	this
throw	true	try	typeof	var
void	while	with		
Keywords that are reserved but not used by JavaScript				
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				

7.6 `if...else` Selection Statement (Cont.)

- **Conditional operator (`?:`)**
  - **Closely related to the `if…else` statement**
  - **JavaScript's only ternary operator—it takes three operands**
  - **The operands together with the `?:` operator form a conditional expression**
  - **The first operand is a boolean expression**
  - **The second is the value for the conditional expression if the boolean expression evaluates to `true`**
  - **Third is the value for the conditional expression if the boolean expression evaluates to `false`**

7.6 `if...else` Selection Statement (Cont.)

- **Nested `if…else` statements**
  - **Test for multiple cases by placing `if…else` statements inside other `if…else` structures**
- **The JavaScript interpreter always associates an `else` with the previous `if`, unless told to do otherwise by the placement of braces (`{}`)**
- **The `if` selection statement expects only one statement in its body**
  - **To include several statements, enclose the statements in braces (`{` and `}`)**
  - **A set of statements contained within a pair of braces is called a block**

7.6 `if...else` Selection Statement (Cont.)

- **A logic error has its effect at execution time.**
- **A fatal logic error causes a program to fail and terminate prematurely.**
- **A nonfatal logic error allows a program to continue executing, but the program produces incorrect results.**

7.7 while Repetition Statement

- while
  - Allows the programmer to specify that an action is to be repeated while some condition remains true
  - The body of a loop may be a single statement or a block
  - Eventually, the condition becomes false and repetition terminates

7.8 Formulating Algorithms: Counter-Controlled Repetition

- **Counter-controlled repetition**
  - **Often called definite repetition, because the number of repetitions is known before the loop begins executing**
- **A *total* is a variable in which a script accumulates the sum of a series of values**
  - **Variables that store totals should normally be initialized to zero before they are used in a program**
- **A *counter* is a variable a script uses to count—typically in a repetition statement**

Fig. 7.7 | Counter-controlled repetition to calculate a class average (Part 1 of 3).

```
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. 7.7: average.html -->
6  <!-- Counter-controlled repetition to calculate a class average. -->
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8     <head>
9        <title>Class Average Program</title>
10       <script type = "text/javascript">
11          <!--
12          var total; // sum of grades
13          var gradeCounter; // number of grades entered
14          var grade; // grade typed by user (as a string)
15          var gradeValue; // grade value (converted to integer)
16          var average; // average of all grades
17
18          // Initialization Phase
19          total = 0; // clear total
20          gradeCounter = 1; // prepare to loop
21
22          // Processing Phase
23          while ( gradeCounter <= 10 ) // loop 10 times
24          {
25
26             // prompt for input and read grade from user
27             grade = window.prompt( "Enter integer grade:", "0" );
28
```

Stores the sum of grades

Sets `total` to 0

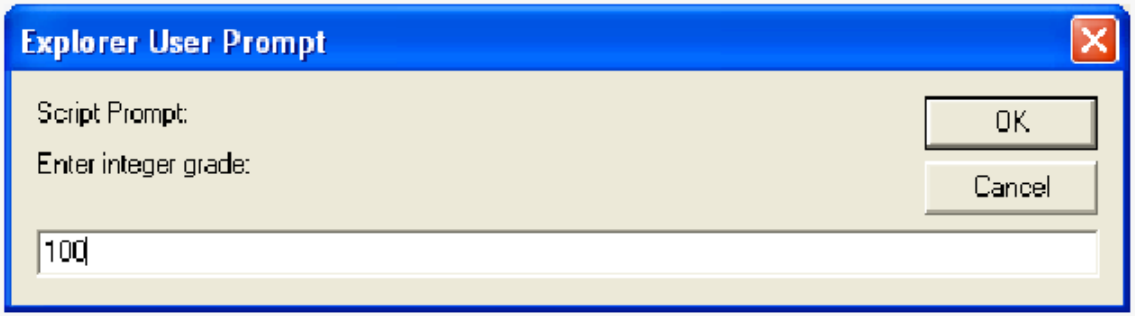
Sets `gradeCounter` to 1 in preparation for the loop

Continues the cycle until `gradeCounter` is greater than 10

```
29             // convert grade from a string to an integer
30             gradeValue = parseInt( grade );
31
32             // add gradeValue to total
33             total = total + gradeValue;
34
35             // add 1 to gradeCounter
36             gradeCounter = gradeCounter + 1;
37          } // end while
38
39          // Termination Phase
40          average = total / 10; // calculate the avera
41
42          // display average of exam grades
43          document.writeln(
44             "<h1>Class average is " + average + "</h1>" );
45          // -->
46       </script>
47    </head>
48    <body>
49       <p>Click Refresh (or Reload) to run the script again<p>
50    </body>
51 </html>
```

Increments `gradeCounter` by 1 after each iteration of the loop

Fig. 7.7 | Counter-controlled repetition to calculate a class average (Part 2 of 3).



This dialog is displayed 10 times. User input is 100, 88, 93, 55, 68, 77, 83, 95, 73 and 62.

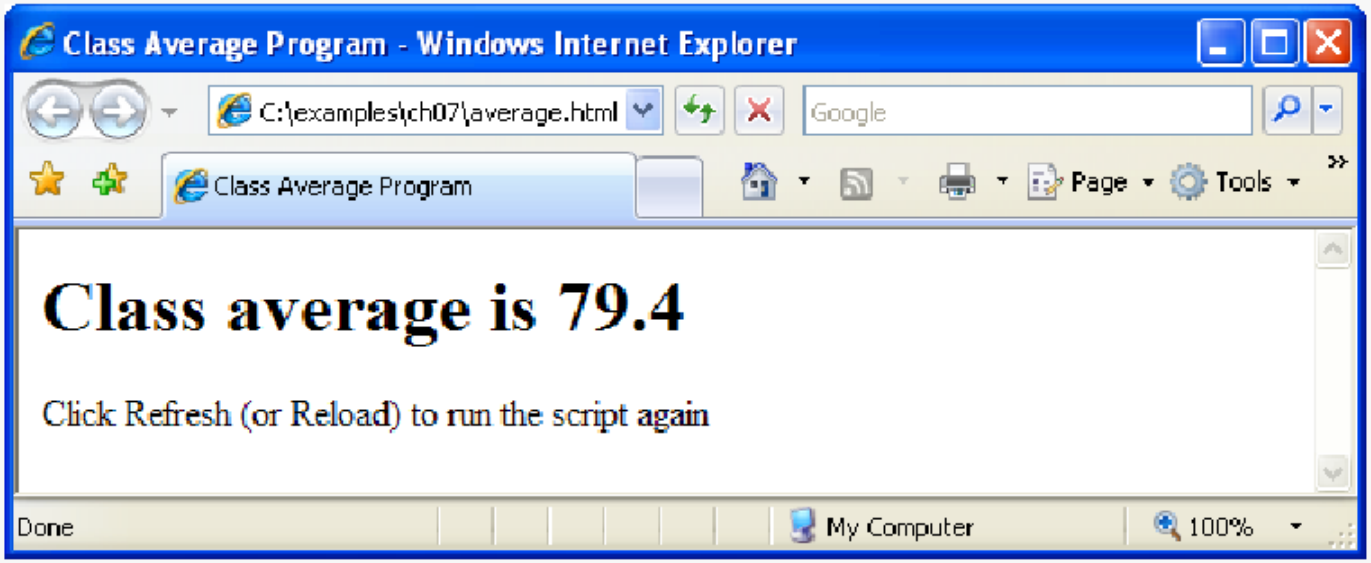


Fig. 7.7 | Counter-controlled repetition to calculate a class average (Part 3 of 3).

7.10 Formulating Algorithms: Nested Control Statements

- **Control structures may be nested inside of one another**

```
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. 7.11: analysis.html -->
6  <!-- Examination-results calculation. -->
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8     <head>
9        <title>Analysis of Examination Results</title>
10       <script type = "text/javascript">
11          <!--
12          // initializing variables in declarations
13          var passes = 0; // number of passes
14          var failures = 0; // number of failures
15          var student = 1; // student counter
16          var result; // one exam result
17
18          // process 10 students; counter-controlled loop
19          while ( student <= 10 )
20          {
21             result = window.prompt( "Enter result (1=pass,2=fail)", "0" );
22
23             if ( result == "1" )
24                passes = passes + 1;
25             else
26                failures = failures + 1;
27
28             student = student + 1;
29          } // end while
30
```

Outer control structure

Start nested control structure

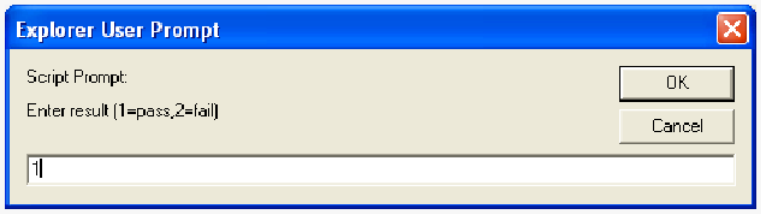
End nested control structure

Increment for `while` loop

Fig. 7.11 | Examination-results calculation (Part 1 of 3).

```
31          // termination phase
32          document.writeln( "<h1>Examination Results</h1>" );
33          document.writeln(
34             "Passed: " + passes + "<br />Failed: " + failures );
35
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>
```

Additional control structure



This dialog is displayed 10 times. User input is 1, 2, 1, 1, 1, 1, 1, 1, 1 and 1.

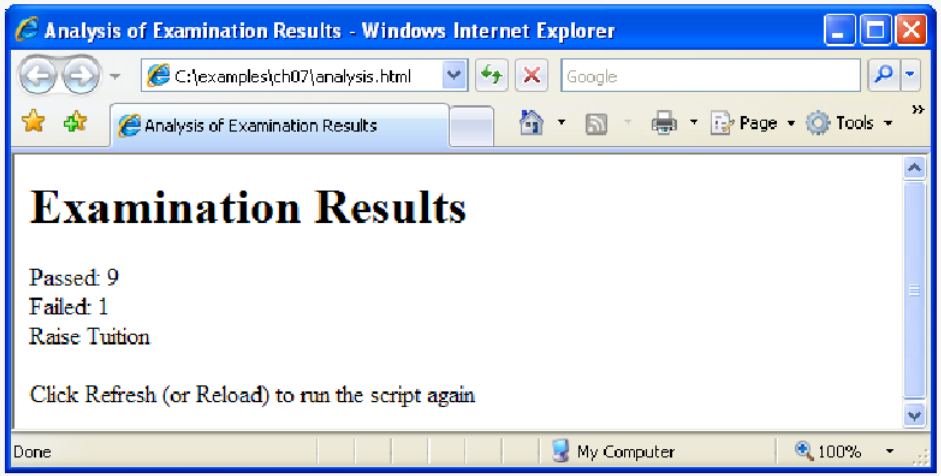


Fig. 7.11 | Examination-results calculation (Part 2 of 3).

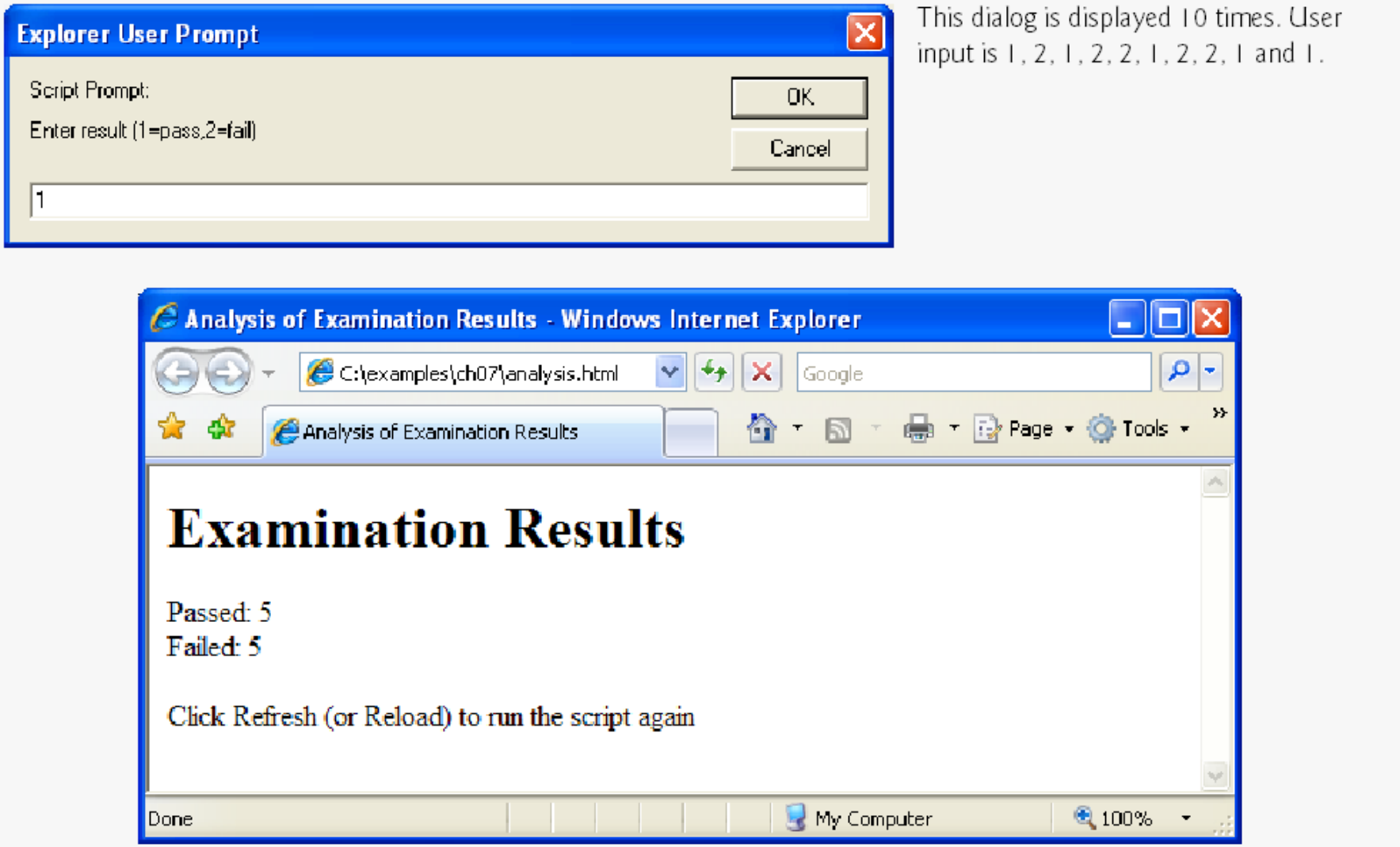


Fig. 7.11 | Examination-results calculation (Part 3 of 3).

7.11 Assignment Operators

- **JavaScript provides the arithmetic assignment operators +=, -=, *=, /= and %=, which abbreviate certain common types of expressions.**

Fig. 7.12 | Arithmetic assignment operators.

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

7.12 Increment and Decrement Operators

- **The increment operator, ++, and the decrement operator, --, increment or decrement a variable by 1, respectively.**
- **If the operator is prefixed to the variable, the variable is incremented or decremented by 1, then used in its expression.**
- **If the operator is postfixed to the variable, the variable is used in its expression, then incremented or decremented by 1.**

Fig. 7.13 | Increment and decrement operators.

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

Fig. 7.14 | Preincrementing and postincrementing (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. 7.14: increment.html -->
6  <!-- Preincrementing and Postincrementing. -->
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8     <head>
9        <title>Preincrementing and Postincrementing</title>
10       <script type = "text/javascript">
11          <!--
12          var c;
13
14          c = 5;
15          document.writeln( "<h3>Postincrementing</h3>" );
16          document.writeln( c ); // prints 5
17          // prints 5 then increments
18          document.writeln( "<br />" + c++ );
19          document.writeln( "<br />" + c ); // prints 6
20
21          c = 5;
22          document.writeln( "<h3>Preincrementing</h3>" );
23          document.writeln( c ); // prints 5
24          // increments then prints 6
25          document.writeln( "<br />" + ++c );
26          document.writeln( "<br />" + c ); // prints 6
27          // -->
28       </script>
29    </head><body></body>
30 </html>
```

Prints value of `c`, then increments it

Increments `c`, then prints its value

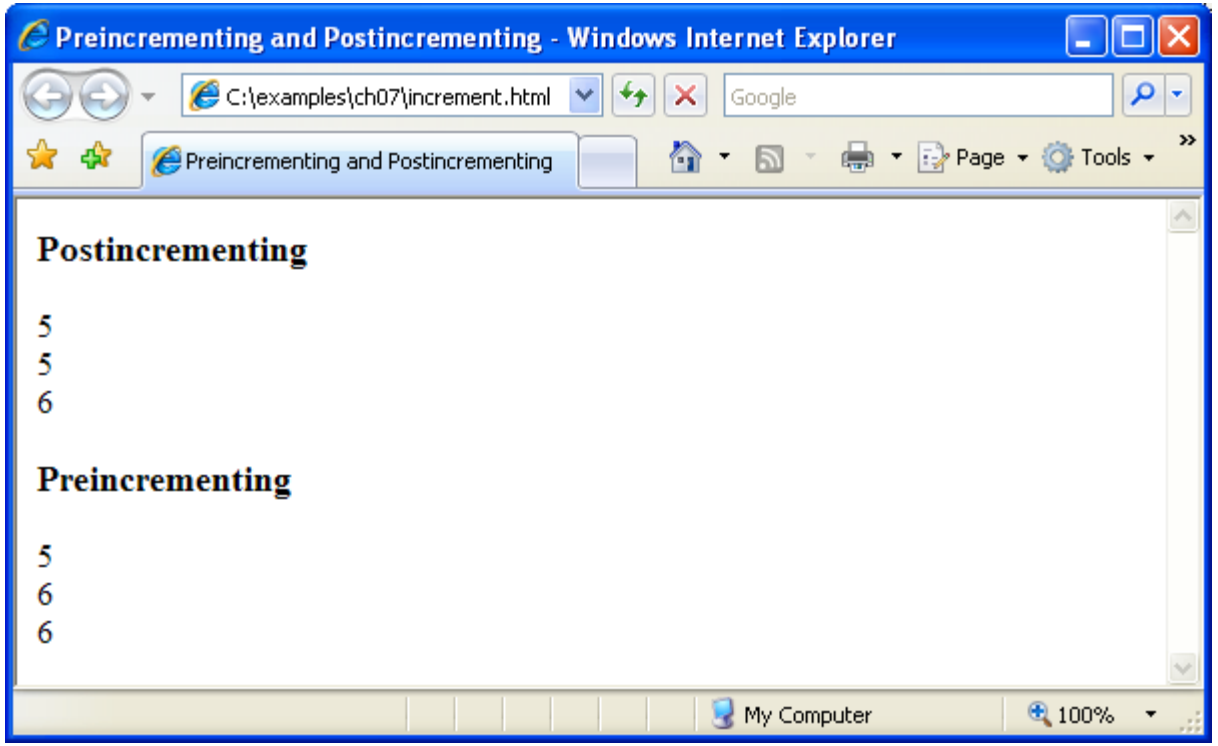


Fig. 7.14 | Preincrementing and postincrementing (Part 2 of 2).

Fig. 7.15 | Precedence and associativity of the operators discussed so far.

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