



Internet & World Wide Web

HOW TO PROGRAM

Fifth Edition

Paul Deitel • Harvey Deitel • Abbey Deitel

ALWAYS LEARNING

PEARSON

Internet & World Wide Web

HOW TO PROGRAM



FIFTH EDITION

```

1  <!DOCTYPE html>
2
3  <!-- Fig. 8.2: ForCounter.html -->
4  <!-- Counter-controlled repetition with the for statement. -->
5  <html>
6      <head>
7          <meta charset="utf-8">
8          <title>Counter-Controlled Repetition</title>
9          <script>
10
11              // Initialization, repetition condition and
12              // incrementing are all included in the for
13              // statement header.
14              for ( var counter = 1; counter <= 7; ++counter )
15                  document.writeln( "<p style = 'font-size: " +
16                      counter + "ex">HTML5 font size " + counter + "ex</p>" );
17
18          </script>
19      </head><body></body>
20 </html>

```

Fig. 8.2 | Counter-controlled repetition with the for statement.

When the for statement begins executing (line 14), the control variable `counter` is declared *and* initialized to 1. Next, the loop-continuation condition, `counter <= 7`, is checked. The condition contains the *final value* (7) of the control variable. The initial value of `counter` is 1. Therefore, the condition is satisfied (i.e., true), so the body statement (lines 15–16) writes a paragraph element in the body of the HTML5 document. Then, variable `counter` is incremented in the expression `++counter` and the loop continues execution with the loop-continuation test. The control variable is now equal to 2, so the final value is not exceeded and the program performs the body statement again (i.e., performs the next iteration of the loop). This process continues until the control variable `counter` becomes 8, at which point the loop-continuation test fails and the repetition terminates.

The program continues by performing the first statement after the for statement. (In this case, the script terminates, because the interpreter reaches the end of the script.)

Figure 8.3 takes a closer look at the for statement at line 14 of Fig. 8.2. The for statement’s first line (including the keyword `for` and everything in parentheses after it) is often called the **for statement header**. Note that the for statement “does it all”—it specifies each of the items needed for counter-controlled repetition with a control variable. Remember that a block is a group of statements enclosed in curly braces that can be placed anywhere that a single statement can be placed, so you can use a block to put multiple statements into the body of a for statement, if necessary.

A Closer Look at the for Statement’s Header

Figure 8.3 uses the loop-continuation condition `counter <= 7`. If you incorrectly write `counter < 7`, the loop will execute only *six* times. This is an example of the common logic error called an **off-by-one error**.

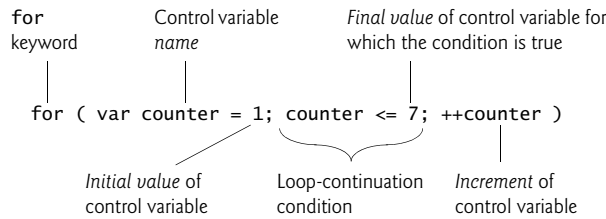


Fig. 8.3 | `for` statement header components.

General Format of a `for` Statement

The general format of the `for` statement is

```
for ( initialization; loopContinuationTest; increment )
    statements
```

where the *initialization* expression names the loop's control variable and provides its initial value, *loopContinuationTest* is the expression that tests the loop-continuation condition (containing the final value of the control variable for which the condition is true), and *increment* is an expression that increments the control variable.

Optional Expressions in a `for` Statement Header

The three expressions in the `for` statement's header are optional. If *loopContinuationTest* is omitted, the loop-continuation condition is true, thus creating an *infinite loop*. One might omit the *initialization* expression if the control variable is initialized before the loop. One might omit the *increment* expression if the increment is calculated by statements in the loop's body or if no increment is needed. The two semicolons in the header are required.

Arithmetic Expressions in the `for` Statement's Header

The initialization, loop-continuation condition and increment portions of a `for` statement can contain arithmetic expressions. For example, assume that $x = 2$ and $y = 10$. If x and y are not modified in the body of the loop, then the statement

```
for ( var j = x; j <= 4 * x * y; j += y / x )
```

is equivalent to the statement

```
for ( var j = 2; j <= 80; j += 5 )
```

Negative Increments

The "increment" of a `for` statement may be negative, in which case it's really a *decrement* and the loop actually counts *downward*.

Loop-Continuation Condition Initially **false**

If the loop-continuation condition initially is **false**, the `for` statement's body is not performed. Instead, execution proceeds with the statement following the `for` statement.



Error-Prevention Tip 8.1

Although the value of the control variable can be changed in the body of a `for` statement, avoid changing it, because doing so can lead to subtle errors.

Flowcharting a for Statement

The for statement is flowcharted much like the while statement. For example, Fig. 8.4 shows the flowchart of the for statement in lines 14–17 of Fig. 8.2. This flowchart makes it clear that the initialization occurs only once and that incrementing occurs *after* each execution of the body statement. Note that, besides *small circles* and *arrows*, the flowchart contains only *rectangle symbols* and a *diamond symbol*.

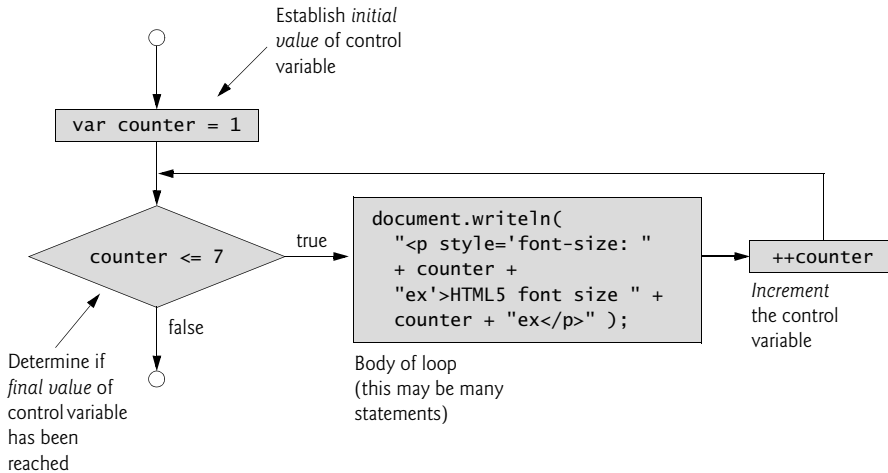


Fig. 8.4 | for repetition statement flowchart.

8.4 Examples Using the for Statement

The examples in this section show methods of varying the control variable in a for statement. In each case, we write the appropriate for header. Note the change in the relational operator for loops that *decrement* the control variable.

- a) Vary the control variable from 1 to 100 in increments of 1.

```
for ( var i = 1; i <= 100; ++i )
```

- b) Vary the control variable from 100 to 1 in increments of -1 (i.e., *decrements* of 1).

```
for ( var i = 100; i >= 1; --i )
```

- c) Vary the control variable from 7 to 77 in steps of 7.

```
for ( var i = 7; i <= 77; i += 7 )
```

- d) Vary the control variable from 20 to 2 in steps of -2.

```
for ( var i = 20; i >= 2; i -= 2 )
```



Common Programming Error 8.1

Not using the proper relational operator in the loop-continuation condition of a loop that counts downward (e.g., using `i <= 1` instead of `i >= 1` in a loop that counts down to 1) is a logic error.

Summing Integers with a for Statement

Figure 8.5 uses the for statement to sum the even integers from 2 to 100. Note that the increment expression (line 13) adds 2 to the control variable *number* *after* the body executes during each iteration of the loop. The loop terminates when *number* has the value 102 (which is *not* added to the sum), and the script continues executing at line 16.

```

1  <!DOCTYPE html>
2
3  <!-- Fig. 8.5: Sum.html -->
4  <!-- Summation with the for repetition structure. -->
5  <html>
6    <head>
7      <meta charset = "utf-8">
8      <title>Sum the Even Integers from 2 to 100</title>
9      <script>
10
11        var sum = 0;
12
13        for ( var number = 2; number <= 100; number += 2 )
14          sum += number;
15
16        document.writeln( "The sum of the even integers " +
17          "from 2 to 100 is " + sum );
18
19      </script>
20    </head><body></body>
21  </html>

```



Fig. 8.5 | Summation with the for repetition structure.

The body of the for statement in Fig. 8.5 actually could be merged into the rightmost (increment) portion of the for header by using a comma, as follows:

```

for ( var number = 2; number <= 100; sum += number, number += 2 )
;

```

In this case, the comma represents the **comma operator**, which guarantees that the expression to its left is evaluated before the expression to its right. Similarly, the initialization `sum = 0` could be merged into the initialization section of the for statement.

**Good Programming Practice 8.1**

Although statements preceding a for statement and in the body of a for statement can often be merged into the for header, avoid doing so, because it makes the program more difficult to read.

Calculating Compound Interest with the for Statement

The next example computes compound interest (compounded yearly) using the for statement. Consider the following problem statement:

A person invests \$1000.00 in a savings account yielding 5 percent interest. Assuming that all the interest is left on deposit, calculate and print the amount of money in the account at the end of each year for 10 years. Use the following formula to determine the amounts:

$$a = p (1 + r)^n$$

where

p is the original amount invested (i.e., the principal)

r is the annual interest rate

n is the number of years

a is the amount on deposit at the end of the n th year.

This problem involves a loop that performs the indicated calculation for each of the 10 years the money remains on deposit. Figure 8.6 presents the solution to this problem, displaying the results in a table. Lines 9–18 define an embedded CSS style sheet that formats various aspects of the table. The CSS property **border-collapse** (line 11) with the value **collapse** indicates that the table's borders should be merged so that there is no extra space between adjacent cells or between cells and the table's border. Lines 13–14 specify the formatting for the table, td and th elements, indicating that they should all have a 1px solid black border and padding of 4px around their contents.

```

1  <!DOCTYPE html>
2
3  <!-- Fig. 8.6: Interest.html -->
4  <!-- Compound interest calculation with a for loop. -->
5  <html>
6    <head>
7      <meta charset = "utf-8">
8      <title>Calculating Compound Interest</title>
9      <style type = "text/css">
10         table      { width: 300px;
11                      border-collapse: collapse;
12                      background-color: lightblue; }
13         table, td, th { border: 1px solid black;
14                      padding: 4px; }
15         th         { text-align: left;
16                      color: white;
17                      background-color: darkblue; }
18         tr.oddrow   { background-color: white; }
19      </style>
20      <script>
21
22         var amount; // current amount of money
23         var principal = 1000.00; // principal amount
24         var rate = 0.05; // interest rate
25

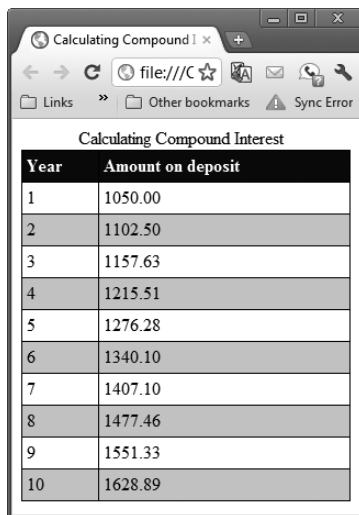
```

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

```

26     document.writeln("<table>" ); // begin the table
27     document.writeln(
28         "<caption>Calculating Compound Interest</caption>" );
29     document.writeln(
30         "<thead><tr><th>Year</th>" ); // year column heading
31     document.writeln(
32         "<th>Amount on deposit</th>" ); // amount column heading
33     document.writeln( "</tr></thead><tbody>" );
34
35     // output a table row for each year
36     for ( var year = 1; year <= 10; ++year )
37     {
38         amount = principal * Math.pow( 1.0 + rate, year );
39
40         if ( year % 2 !== 0 )
41             document.writeln( "<tr class='oddrow'><td>" + year +
42                 "</td><td>" + amount.toFixed(2) + "</td></tr>" );
43         else
44             document.writeln( "<tr><td>" + year +
45                 "</td><td>" + amount.toFixed(2) + "</td></tr>" );
46     } //end for
47
48     document.writeln( "</tbody></table>" );
49
50     </script>
51 </head><body></body>
52 </html>

```



Year	Amount on deposit
1	1050.00
2	1102.50
3	1157.63
4	1215.51
5	1276.28
6	1340.10
7	1407.10
8	1477.46
9	1551.33
10	1628.89

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

Outputting the Beginning of an HTML5 table

Lines 22–24 declare three variables and initialize `principal` to 1000.0 and `rate` to .05. Line 26 writes an HTML5 `<table>` tag, and lines 27–28 write the caption that summarizes the table's content. Lines 29–30 create the table's header section (`<thead>`), a row