add grade to total
\[ \text{total} = \text{total} + \text{grade}; \]

add 1 to counter
\[ \text{counter} = \text{counter} + 1; \]
### JavaScript Keywords

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

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

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

*Fig. 9.1  JavaScript keywords.*
Fig. 9.2  Flowcharting the single-selection if structure.
Fig. 9.3 Flowcharting the double-selection if/else structure.
Fig. 9.4  Flowcharting the **while** repetition structure.
```html
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<!-- Fig. 9.7: average.html -->
<head>
<title>Class Average Program</title>
<script language="JavaScript">
var total, // sum of grades
    gradeCounter, // number of grades entered
    gradeValue, // grade value
    average, // average of all grades
    grade; // grade typed by user

// Initialization Phase
total = 0; // clear total
gradeCounter = 1; // prepare to loop

// Processing Phase
while (gradeCounter <= 10) { // loop 10 times
    // prompt for input and read grade from user
    grade = window.prompt("Enter integer grade:", "0");
    // convert grade from a String to an integer
    gradeValue = parseInt(grade);
    // add gradeValue to total
    total = total + gradeValue;
    // add 1 to gradeCounter
    gradeCounter = gradeCounter + 1;
}

// Termination Phase
average = total / 10; // calculate the average

// display average of exam grades
document.writeln("<H1>Class average is " + average + "</H1>");
</script>
</head>
<body>
Click Refresh (or Reload) to run the script again
</body>
</html>
```

Fig. 9.5  Class-average program with counter-controlled repetition (part 1 of 3).

© Copyright 2000 by Prentice Hall. All Rights Reserved.
For use only by instructors in classes for which Java How to Program, Third Edition is the required textbook.
Fig. 9.5  Class-average program with counter-controlled repetition (part 2 of 3).
Fig. 9.5  Class-average program with counter-controlled repetition (part 3 of 3).

© Copyright 2000 by Prentice Hall. All Rights Reserved.
For use only by instructors in classes for which Java How to Program, Third Edition is the required textbook.
Chapter 9

Initialize total to zero
Initialize gradeCounter to zero

Input the first grade (possibly the sentinel)
While the user has not as yet entered the sentinel
    Add this grade into the running total
    Add one to the grade counter
    Input the next grade (possibly the sentinel)

If the counter is not equal to zero
    Set the average to the total divided by the counter
    Print the average
else
    Print "No grades were entered"

Fig. 9.6  Pseudocode algorithm that uses sentinel-controlled repetition to solve the class-average problem.
```html
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<!-- Fig. 9.9: Average2.html -->
<head>
<title>Class Average Program: Sentinel-controlled Repetition</title>
<script language = "JavaScript" >

var gradeCounter, // number of grades entered
    gradeValue,   // grade value
    total,        // sum of grades
    average,      // average of all grades
    grade;        // grade typed by user

// Initialization phase
total = 0;        // clear total
gradeCounter = 0; // prepare to loop

// Processing phase
// prompt for input and read grade from user
grade = window.prompt("Enter Integer Grade, -1 to Quit:", "0");
// convert grade from a String to an integer
gradeValue = parseInt( grade );

while ( gradeValue != -1 ) {
    // add gradeValue to total
    total = total + gradeValue;
    // add 1 to gradeCounter
    gradeCounter = gradeCounter + 1;
    // prompt for input and read grade from user
    grade = window.prompt("Enter Integer Grade, -1 to Quit:", "0");
    // convert grade from a String to an integer
    gradeValue = parseInt( grade );
}

// Termination phase
if ( gradeCounter != 0 ) {
    average = total / gradeCounter;
    // display average of exam grades
    document.writeln("<h1>Class average is " + average + "</h1>");
} else
    document.writeln("<p>No grades were entered</p>" );
</script>
</head>
<body>
</body>
</html>
```

Fig. 9.7 Class-average program with sentinel-controlled repetition (part 1 of 2).
Fig. 9.7  Class-average program with sentinel-controlled repetition (part 2 of 2).
Initialize passes to zero
Initialize failures to zero
Initialize student to one

While student counter is less than or equal to ten
   Input the next exam result
   If the student passed
      Add one to passes
   else
      Add one to failures
   Add one to student counter

Print the number of passes
Print the number of failures
If more than eight students passed
   Print “Raise tuition”

Fig. 9.8 Pseudocode for examination-results problem.
```html
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<!-- Fig. 9.11: analysis.html -->
<head>
<title>Analysis of Examination Results</title>
<script language = "JavaScript">
// initializing variables in declarations
var passes = 0,      // number of passes
failures = 0,    // number of failures
student = 1,     // student counter
result;          // one exam result

// process 10 students; counter-controlled loop
while ( student <= 10 ) {
    result = window.prompt(
        "Enter result (1=pass,2=fail)", "0" );
    if ( result == "1" )
        passes = passes + 1;
    else
        failures = failures + 1;
    student = student + 1;
}

// termination phase
document.writeln( "<h1>Examination Results</h1>" );
document.writeln(
    "Passed: "+ passes + "<br>Failed: " + failures );
if ( passes > 8 )
document.writeln( "<br>Raise Tuition" );
</script>
</head>
<body>
<p>Click Refresh (or Reload) to run the script again</p>
</body>
</html>
```

Fig. 9.9  JavaScript program for examination-results problem (part 1 of 5).

© Copyright 2000 by Prentice Hall. All Rights Reserved.
For use only by instructors in classes for which Java How to Program, Third Edition is the required textbook.
Fig. 9.9  JavaScript program for examination-results problem (part 2 of 5).
Fig. 9.9  JavaScript program for examination-results problem (part 3 of 5).

© Copyright 2000 by Prentice Hall. All Rights Reserved.
For use only by instructors in classes for which Java How to Program, Third Edition is the required textbook.
Fig. 9.9  JavaScript program for examination-results problem (part 4 of 5).

© Copyright 2000 by Prentice Hall. All Rights Reserved.
For use only by instructors in classes for which Java How to Program, Third Edition is the required textbook.
Fig. 9.9  JavaScript program for examination-results problem (part 5 of 5).
<table>
<thead>
<tr>
<th>Assignment operator</th>
<th>Initial variable value</th>
<th>Sample expression</th>
<th>Explanation</th>
<th>Assigns</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>+=</code></td>
<td><code>c = 3</code></td>
<td><code>c += 7</code></td>
<td><code>c = c + 7</code></td>
<td>10 to <code>c</code></td>
</tr>
<tr>
<td><code>-=</code></td>
<td><code>d = 5</code></td>
<td><code>d -= 4</code></td>
<td><code>d = d - 4</code></td>
<td>1 to <code>d</code></td>
</tr>
<tr>
<td><code>*=</code></td>
<td><code>e = 4</code></td>
<td><code>e *= 5</code></td>
<td><code>e = e * 5</code></td>
<td>20 to <code>e</code></td>
</tr>
<tr>
<td><code>/=</code></td>
<td><code>f = 6</code></td>
<td><code>f /= 3</code></td>
<td><code>f = f / 3</code></td>
<td>2 to <code>f</code></td>
</tr>
<tr>
<td><code>%=</code></td>
<td><code>g = 12</code></td>
<td><code>g %= 9</code></td>
<td><code>g = g % 9</code></td>
<td>3 to <code>g</code></td>
</tr>
</tbody>
</table>

Fig. 9.10  Arithmetic assignment operators.
<table>
<thead>
<tr>
<th>Operator</th>
<th>Called</th>
<th>Sample expression</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>++</td>
<td>preincrement</td>
<td>++a</td>
<td>Increment ( a ) by 1, then use the new value of ( a ) in the expression in which ( a ) resides.</td>
</tr>
<tr>
<td>++</td>
<td>postincrement</td>
<td>a++</td>
<td>Use the current value of ( a ) in the expression in which ( a ) resides, then increment ( a ) by 1.</td>
</tr>
<tr>
<td>--</td>
<td>predecrement</td>
<td>--b</td>
<td>Decrement ( b ) by 1, then use the new value of ( b ) in the expression in which ( b ) resides.</td>
</tr>
<tr>
<td>--</td>
<td>postdecrement</td>
<td>b--</td>
<td>Use the current value of ( b ) in the expression in which ( b ) resides, then decrement ( b ) by 1.</td>
</tr>
</tbody>
</table>

Fig. 9.11 The increment and decrement operators.
Fig. 9.12 Differences between preincrementing and postincrementing.
### Table: Precedence and Associativity of Operators

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

**Fig. 9.13** Precedence and associativity of the operators discussed so far.