Outline
5.1 Introduction
5.2 Algorithms
5.3 Pseudocode
5.4 Control Structures
5.5 if Selection Statement
5.6 if…else Selection Statement
5.7 while Repetition Statement
5.8 Formulating Algorithms: Case Study 1 (Counter-Controlled Repetition)
5.9 Formulating Algorithms with Top-Down, Stepwise Refinement: Case Study 2 (Sentinel-Controlled Repetition)
5.10 Formulating Algorithms with Top-Down, Stepwise Refinement: Case Study 3 (Nested Control Structures)
5.11 Assignment Operators
5.12 Increment and Decrement Operators
5.13 Note on Data Types
5.14 Web Resources
5.1 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
5.4 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
5.5 if Selection Statement

Fig. 8.3 Flowcharting the single-selection if statement.

```
if grade >= 60
  print "Passed"
else
  ...
```
5.6 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
5.6 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
5.7 while Repetition Statement

Fig. 8.5 Flowcharting the while repetition statement.
5.8 Formulating Algorithms: Case Study 1 (Counter-Controlled Repetition)

- Counter-controlled repetition
  - Counter
    - Control the number of times a set of statements executes
  - Definite repetition
```xml
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- Fig. 8.7: average.html -->
<!-- Class Average Program -->

<html xmlns = "http://www.w3.org/1999/xhtml">
    <head>
        <title>Class Average Program</title>
        <script type = "text/javascript">
            <!--
            var total, gradeCounter, gradeValue, average, grade;
            // sum of grades
            // number of grades entered
            // grade value
            // average of all grades
            // grade typed by user
            // Initialization Phase
            total = 0; // clear total
            gradeCounter = 1; // prepare to loop
            -->
        </script>
    </head>
</html>
```

average.html
(1 of 3)
```html
// 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>" );
```

Click Refresh (or Reload) to run the script again

Class average is 79.4

Click Refresh (or Reload) to run the script again
5.9 Formulating Algorithms with Top-Down, Stepwise Refinement: Case Study 2 (Sentinel-Controlled Repetition)

- Indefinite repetition
  - Sentinel value
average2.html

-- Fig. 8.9: average2.html -->

-- Sentinel-controlled Repetition -->

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Class Average Program:
Sentinel-controlled Repetition</title>

<script type="text/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*/

</script>
</head>
</html>
// 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>
<p>Click Refresh (or Reload) to run the script again</p>
</body>
</html>
Class average is 85.66666666666667

Click Refresh (or Reload) to run the script again
5.10 Formulating Algorithms with Top-Down, Stepwise Refinement: Case Study 3 (Nested Control Structures)

- Consider problem
- Make observations
- Top-down, stepwise refinement
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<!-- Fig. 8.11: analysis.html -->
<!-- Analyzing Exam Results -->

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Analysis of Examination Results</title>

<script type="text/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");
}
</script>
</head>
</html>
```html
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>
```
**Examination Results**

Passed: 9  
Failed: 1  
Raise Tuition

Click Refresh (or Reload) to run the script again.
Examination Results

Passed: 5
Failed: 5

Click Refresh (or Reload) to run the script again
### 5.11 Assignment Operators

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

Fig. 5.12 Arithmetic assignment operators.
## 5.12 Increment and Decrement 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. 5.13 increment and decrement operators.
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Preincrementing and Postincrementing</title>

  <script type="text/javascript">
    <!--
    var c;
    c = 5;
    document.writeln("<h3>Postincrementing</h3>" );
    document.writeln( c ); // print 5
    // print 5 then increment
    document.writeln("<br />" + c++ );
    document.writeln("<br />" + c ); // print 6
  
    c = 5;
    document.writeln( "<h3>Preincrementing</h3>" );
    document.writeln( c ); // print 5
  <!-- -->
  </script>
</head>
</html>
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>

Postincrementing

5
5
6

Preincrementing

5
6
6
## 5.12 Increment and Decrement Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Associativity</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>++</td>
<td>right to left</td>
<td>unary</td>
</tr>
<tr>
<td>--</td>
<td></td>
<td></td>
</tr>
<tr>
<td>*</td>
<td>left to right</td>
<td>multiplicative</td>
</tr>
<tr>
<td>/</td>
<td></td>
<td></td>
</tr>
<tr>
<td>%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+</td>
<td>left to right</td>
<td>additive</td>
</tr>
<tr>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;</td>
<td>left to right</td>
<td>relational</td>
</tr>
<tr>
<td>&lt;=</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&gt;=</td>
<td></td>
<td></td>
</tr>
<tr>
<td>==</td>
<td>left to right</td>
<td>equality</td>
</tr>
<tr>
<td>!=</td>
<td></td>
<td></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>
<tr>
<td>+=</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-=</td>
<td></td>
<td></td>
</tr>
<tr>
<td>*=</td>
<td></td>
<td></td>
</tr>
<tr>
<td>/=</td>
<td></td>
<td></td>
</tr>
<tr>
<td>%=</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 5.15 Precedence and associativity of the operators discussed so far.
5.13 Note on Data Types

• Loosely typed
  – Automatically converts between values of different types