

JavaScript: Control Statements I

Erick Kurniawan, S.Kom



JavaScript: Control Statements I

Garis Besar

- 1 Control Structures
- 2 if Selection Statement
- 3 if...else Selection Statement
- 4 while Repetition Statement
- 5 Formulating Algorithms: Case Study 1 (Counter-Controlled Repetition)
- 6 Formulating Algorithms with Top-Down, Stepwise Refinement: Case Study 2 (Sentinel-Controlled Repetition)
- 7 Formulating Algorithms with Top-Down, Stepwise Refinement: Case Study 3 (Nested Control Structures)
- 8 Assignment Operators
- 9 Increment and Decrement Operators



Garis Besar

- Pada kuliah ini anda akan belajar :
 - Mengerti dasar problem-solving techniques.
 - Dapat menggunakan algorithms dengan process of top-down, stepwise refinement.
 - Dapat menggunakan `if` dan `if...else` selection statements
 - Dapat menggunakan `while` repetition statement untuk mengeksekusi statements dari script secara berulang
 - Untuk mengerti counter-controlled repetition dan sentinel-controlled repetition.
 - Dapat menggunakan increment, decrement dan assignment operators.



Control Structures

- Tiga struktur control
 - Sequence structure
 - Selection structure
 - if
 - if...else
 - switch
 - Repetition structure
 - while
 - do...while
 - for
 - for...in

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.

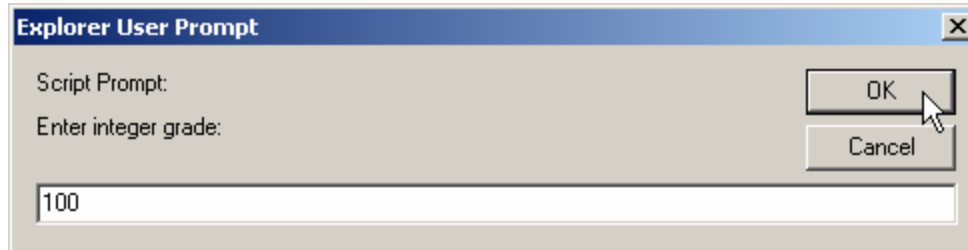


Formulating Algorithms: Case Study 1 (Counter-Controlled Repetition)

```
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
```

```
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>
```

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



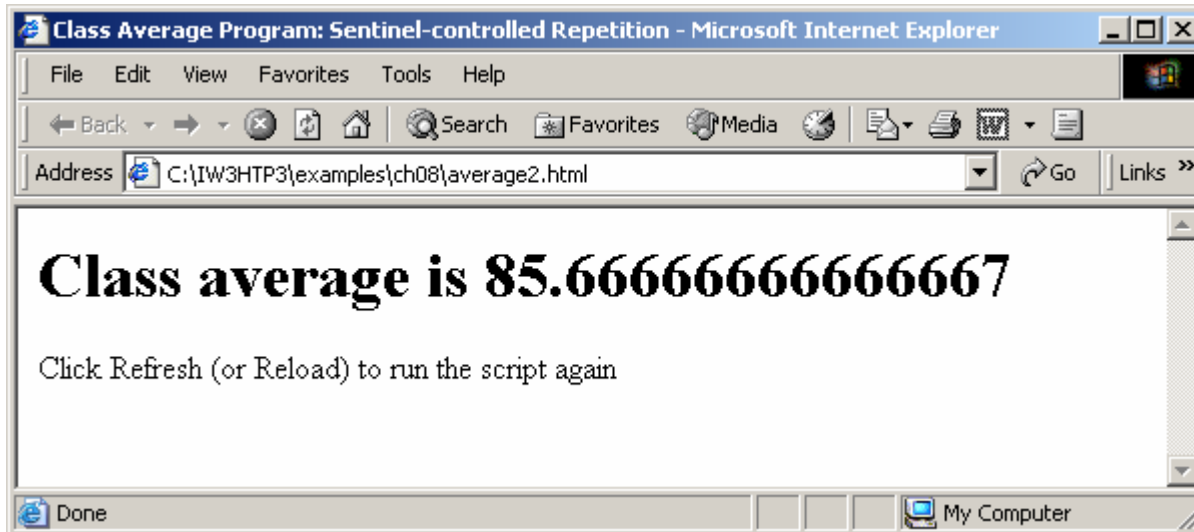
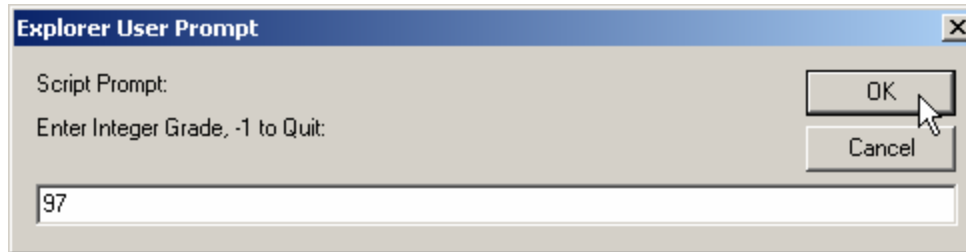


**Formulating Algorithms with Top-Down,
Stepwise Refinement:
Case Study 2 (Sentinel-Controlled Repetition)**

```
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. 8.9: 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
```

```
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
```

```
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>
```





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

```
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
```

```
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>
```



Explorer User Prompt [X]

Script Prompt:
Enter result (1=pass,2=fail)

1

OK Cancel

Analysis of Examination Results - Microsoft Internet Explorer [Min] [Max] [Close]

File Edit View Favorites Tools Help

Back Forward Stop Home Search Favorites Media Print Copy Paste

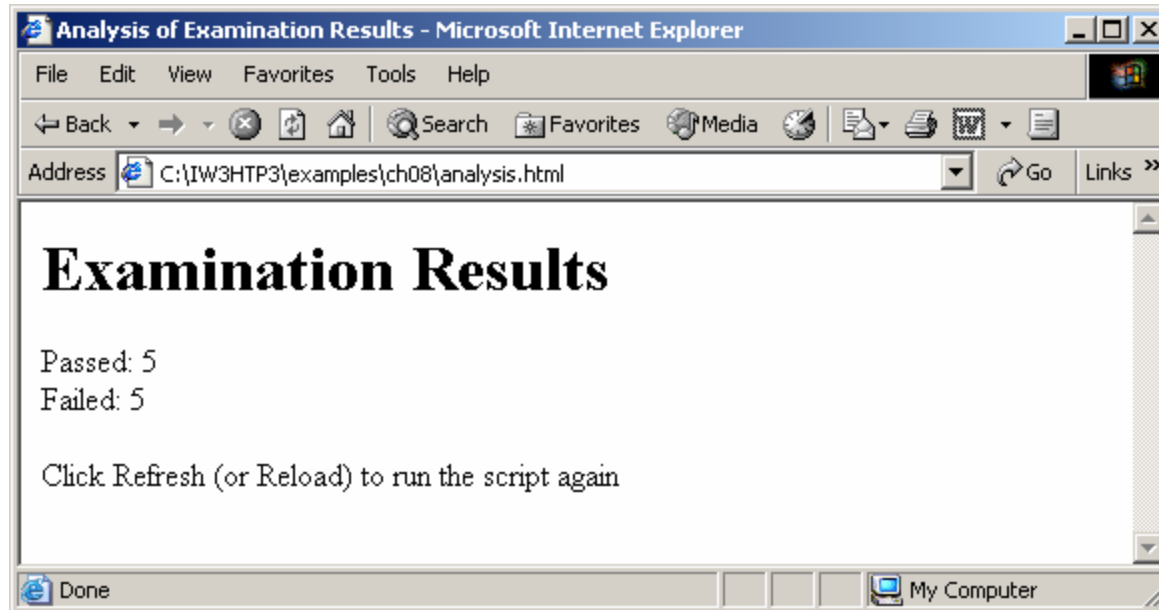
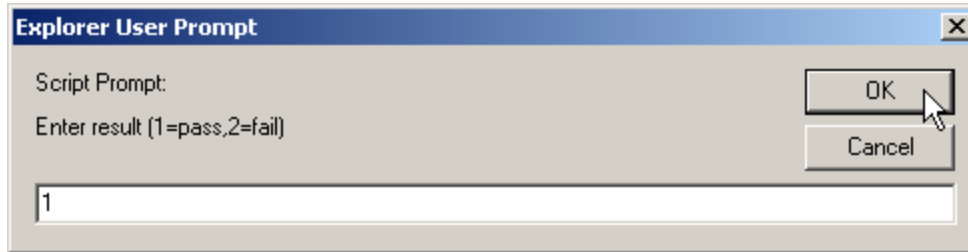
Address C:\IW3HTP3\examples\ch08\analysis.html Go Links >>

Examination Results

Passed: 9
Failed: 1
Raise Tuition

Click Refresh (or Reload) to run the script again

Done My Computer



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.

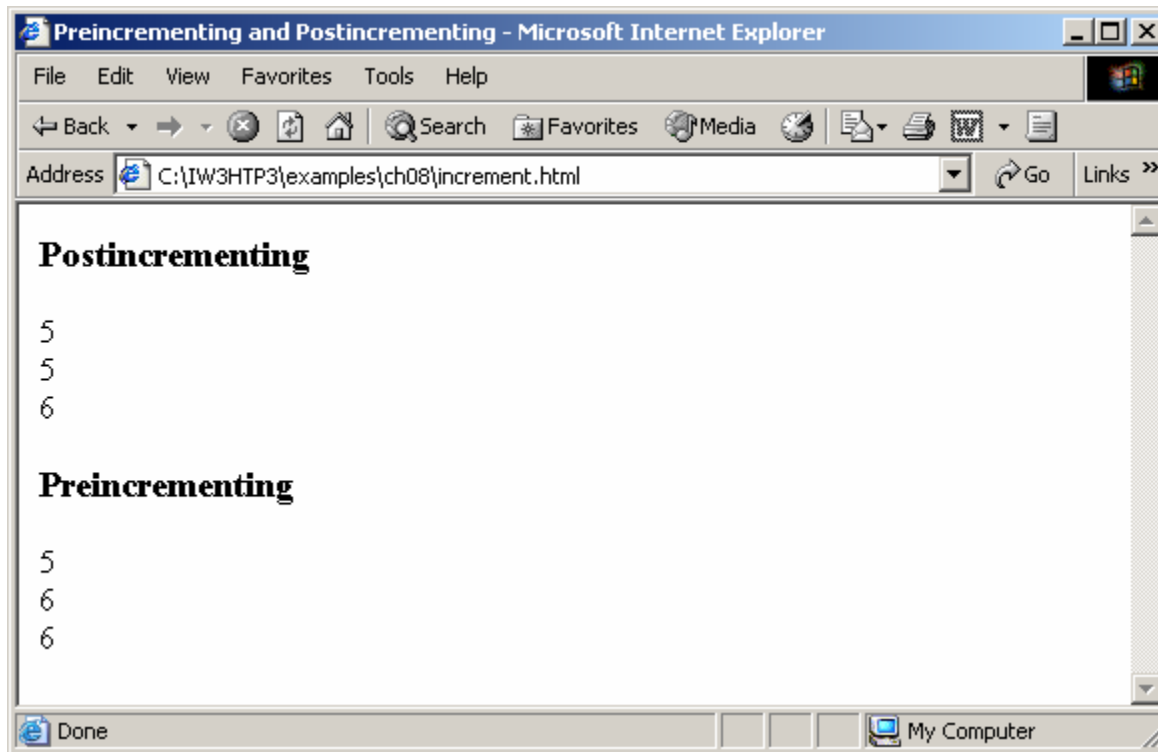
Increment and Decrement Operators

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
```

```
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 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.