

# Introduction to Scripting

## Outline

- 7.1 Introduction
- 7.2 Simple Program: Printing a Line of Text in a Web Page
- 7.3 Obtaining User Input with prompt Dialogs
  - 7.3.1 Dynamic Welcome Page
  - 7.3.2 Adding Integers
- 7.4 Memory Concepts
- 7.5 Arithmetic
- 7.6 Decision Making: Equality and Relational Operators
- 7.7 Web Resources

# Objectives

- In this lesson, you will learn:
  - To be able to write simple JavaScript programs.
  - To be able to use input and output statements.
  - To understand basic memory concepts.
  - To be able to use arithmetic operators.
  - To understand the precedence of arithmetic operators.
  - To be able to write decision-making statements.
  - To be able to use relational and equality operators.

# 7.1 Introduction

- JavaScript scripting language
  - Enhances functionality and appearance
  - Client-side scripting
    - Makes pages more dynamic and interactive
  - Foundation for complex server-side scripting
  - Program development
  - Program control

## 7.2 Simple Program: Printing a Line of Text in a Web Page

- Inline scripting
  - Written in the `<body>` of a document
  - `<script>` tag
    - Indicate that the text is part of a script
    - `type` attribute
      - Specifies the type of file and the scripting language use
    - `writeln` method
      - Write a line in the document
    - Escape character ( `\` )
      - Indicates “special” character is used in the string
    - `alert` method
      - Dialog box

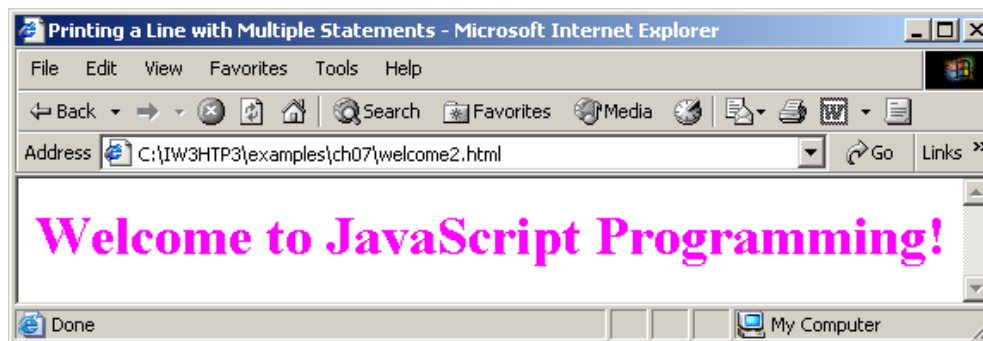
```
1 <?xml version = "1.0"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 7.1: welcome.html   -->
6 <!-- Displaying a line of text -->
7
8 <html xmlns = "http://www.w3.org/1999/xhtml">
9   <head>
10     <title>A First Program in JavaScript</title>
11
12     <script type = "text/javascript">
13       <!--
14       document.writeln(
15         "<h1>welcome to JavaScript Programming!</h1>" );
16       // -->
17     </script>
18
19   </head><body></body>
20 </html>
```



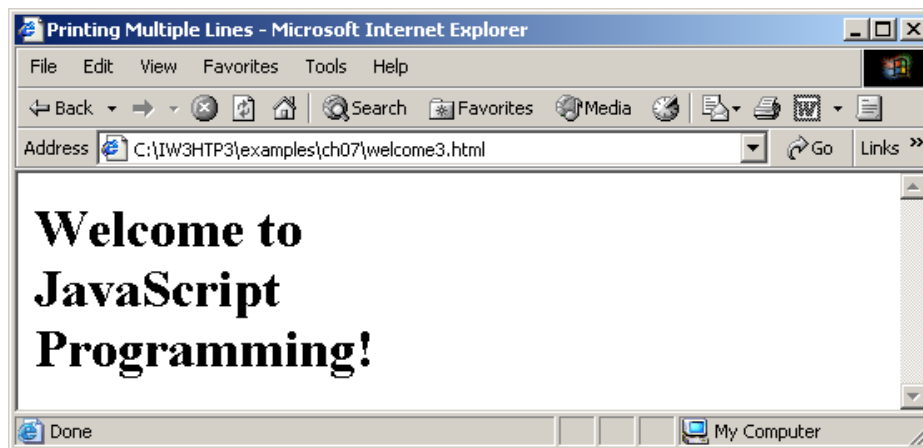
```

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. 7.2: welcome2.html          -->
6 <!-- Printing a Line with Multiple Statements -->
7
8 <html xmlns = "http://www.w3.org/1999/xhtml">
9   <head>
10    <title>Printing a Line with Multiple Statements</title>
11
12    <script type = "text/javascript">
13      <!--
14      document.write( "<h1 style = \"color: magenta\">" );
15      document.write( "welcome to JavaScript " +
16        "Programming!</h1>" );
17      // -->
18    </script>
19
20  </head><body></body>
21 </html>

```

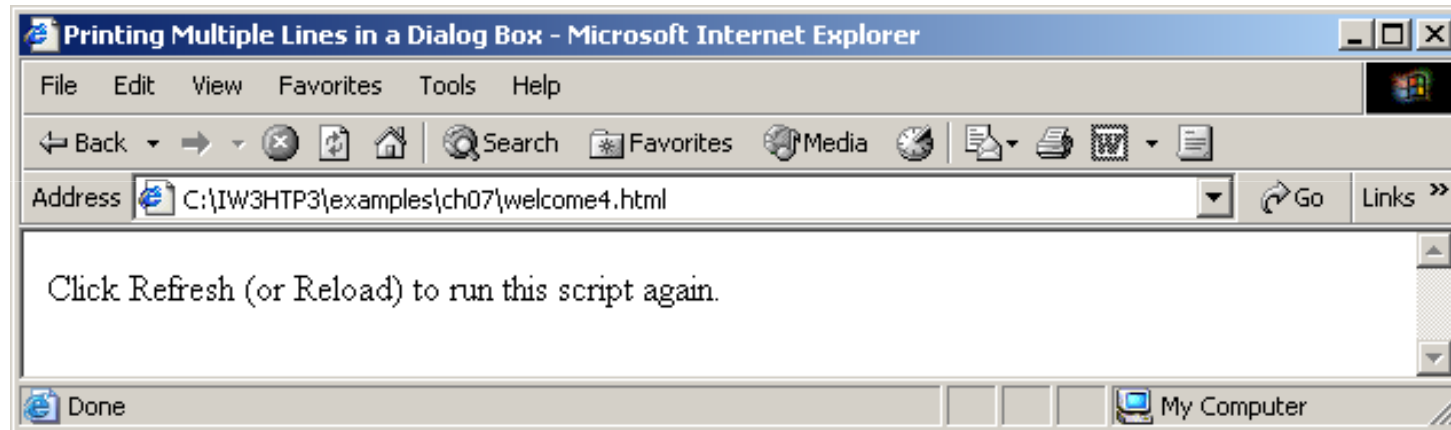


```
1 <?xml version = "1.0"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 7.3: welcome3.html  -->
6 <!-- Printing Multiple Lines  -->
7
8 <html xmlns = "http://www.w3.org/1999/xhtml">
9   <head><title>Printing Multiple Lines</title>
10
11     <script type = "text/javascript">
12       <!--
13       document.writeln( "<h1>welcome to<br />JavaScript" +
14         "<br />Programming!</h1>" );
15       // -->
16     </script>
17
18   </head><body></body>
19 </html>
```



```
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. 7.4: welcome4.html -->
6 <!-- Printing multiple lines in a dialog box -->
7
8 <html xmlns = "http://www.w3.org/1999/xhtml">
9   <head><title>Printing Multiple Lines in a Dialog Box</title>
10
11     <script type = "text/javascript">
12       <!--
13         window.alert( "welcome to\nJavaScript\nProgramming!" );
14       // -->
15     </script>
16
17 </head>
18
19 <body>
20   <p>Click Refresh (or Reload) to run this script again.</p>
21 </body>
22 </html>
```





## 7.2 Simple Program: Printing a Line of Text in a Web Page

Escape sequence	Description
<code>\n</code>	Newline. Position the screen cursor at the beginning of the next line.
<code>\t</code>	Horizontal tab. Move the screen cursor to the next tab stop.
<code>\r</code>	Carriage return. Position the screen cursor to the beginning of the current line; do not advance to the next line. Any characters output after the carriage return overwrite the characters previously output on that line.
<code>\\</code>	Backslash. Used to represent a backslash character in a string.
<code>\"</code>	Double quote. Used to represent a double quote character in a string contained in double quotes. For example, <pre>    window.alert( "\"in quotes\"" );</pre> displays "in quotes" in an alert dialog.
<code>\'</code>	Single quote. Used to represent a single quote character in a string. For example, <pre>    window.alert( '\'in quotes\'' );</pre> displays 'in quotes' in an alert dialog.

Fig. 7.5 Some common escape sequences.

## 7.3.1 Dynamic Welcome Page

- A script can adapt the content based on input from the user or other variables

```
1 <?xml version = "1.0"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
3   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
4
5 <!-- Fig. 7.6: welcome5.html -->
6 <!-- Using Prompt Boxes      -->
7
8 <html xmlns = "http://www.w3.org/1999/xhtml">
9   <head>
10     <title>Using Prompt and Alert Boxes</title>
11
12     <script type = "text/javascript">
13       <!--
14       var name; // string entered by the user
15
16       // read the name from the prompt box as a string
17       name = window.prompt( "Please enter your name", "GalAnt" );
18
19       document.writeln( "<h1>Hello, " + name +
20         ", welcome to JavaScript programming!</h1>" );
21       // -->
22     </script>
```

```
23
24     </head>
25
26     <body>
27     <p>Click Refresh (or Reload) to run this script again.</p>
28     </body>
29 </html>
```



(2012)



## 7.3.1 Dynamic Welcome Page

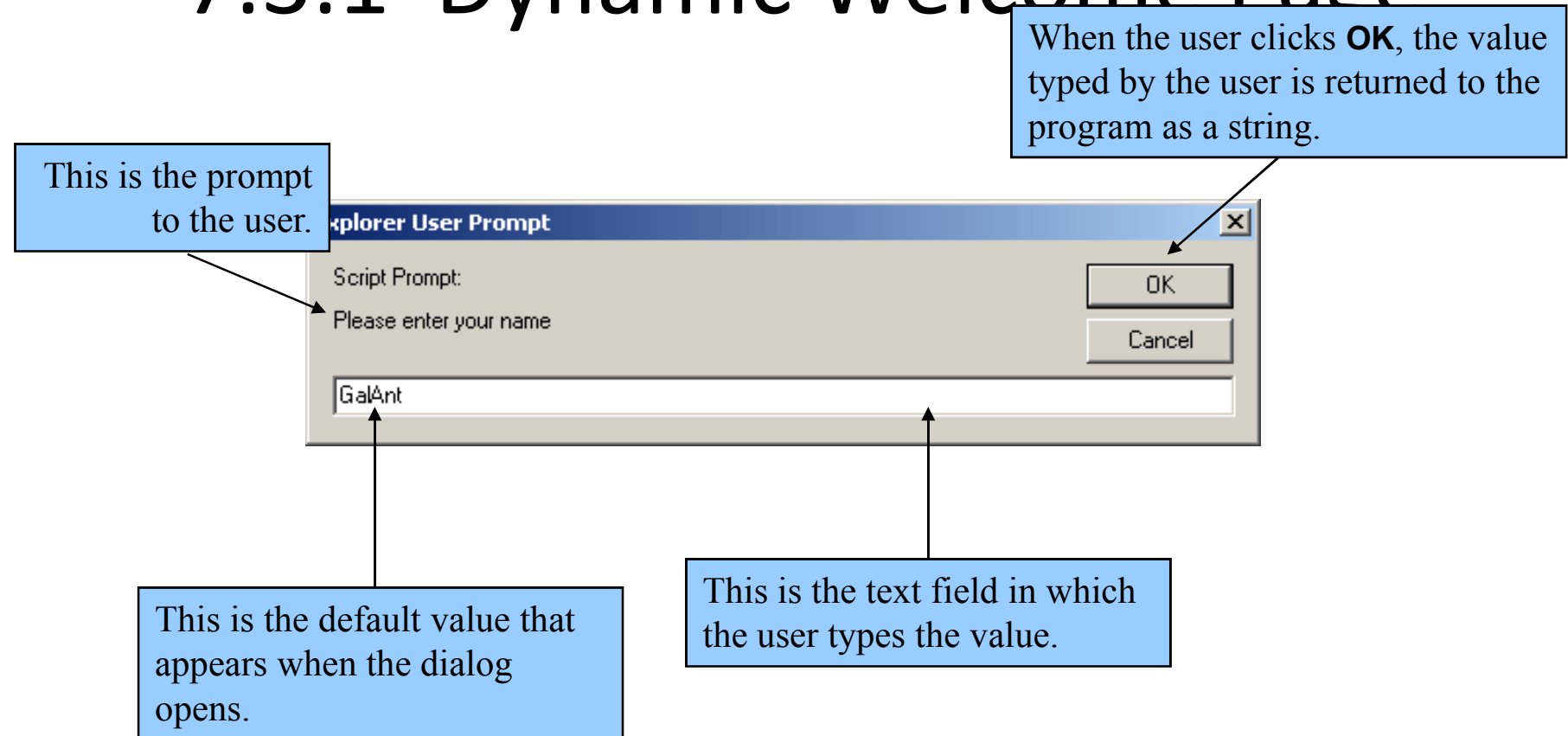


Fig. 7.7 Prompt dialog displayed by the window object's prompt method.

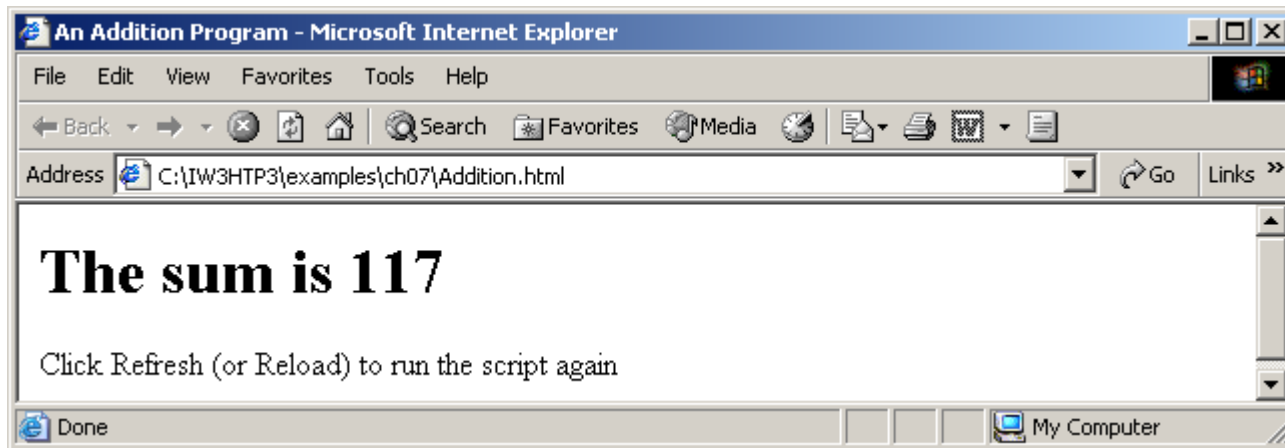
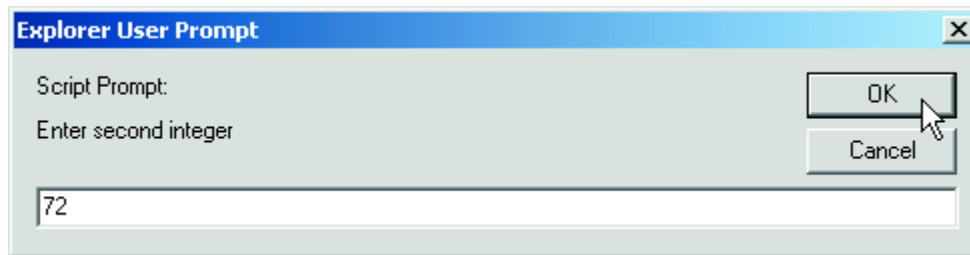
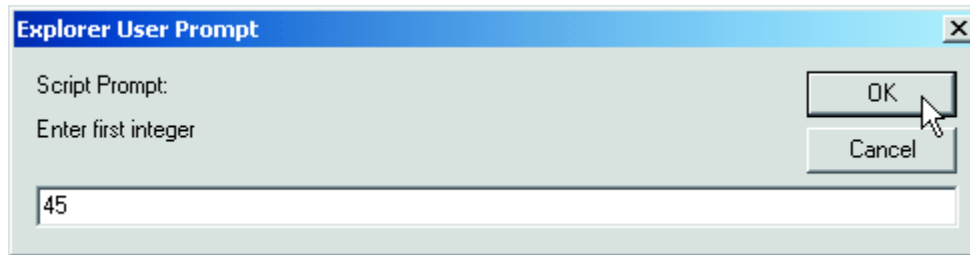
## 7.3.2 Adding Integers

- Prompt user for two integers and calculate the sum (Fig. 7.8)
- NaN (not a number)
- `parseInt`
  - Converts its string argument to an integer

```
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. 7.8: Addition.html -->
6 <!-- Addition Program      -->
7
8 <html xmlns = "http://www.w3.org/1999/xhtml">
9   <head>
10     <title>An Addition Program</title>
11
12     <script type = "text/javascript">
13       <!--
14         var firstNumber,    // first string entered by user
15             secondNumber,  // second string entered by user
16             number1,       // first number to add
17             number2,       // second number to add
18             sum;           // sum of number1 and number2
19
20         // read in first number from user as a string
21         firstNumber =
22             window.prompt( "Enter first integer", "0" );
23
```



```
24 // read in second number from user as a string
25 secondNumber =
26     window.prompt( "Enter second integer", "0" );
27
28 // convert numbers from strings to integers
29 number1 = parseInt( firstNumber );
30 number2 = parseInt( secondNumber );
31
32 // add the numbers
33 sum = number1 + number2;
34
35 // display the results
36 document.writeln( "<h1>The sum is " + sum + "</h1>" );
37 // -->
38 </script>
39
40 </head>
41 <body>
42     <p>Click Refresh (or Reload) to run the script again</p>
43 </body>
44 </html>
```



## 7.4 Memory Concepts

- Variable names correspond to locations in the computer's memory
- Every variable has a name, a type, and a value
- Read value from a memory location
  - nondestructive

# 7.4 Memory Concepts

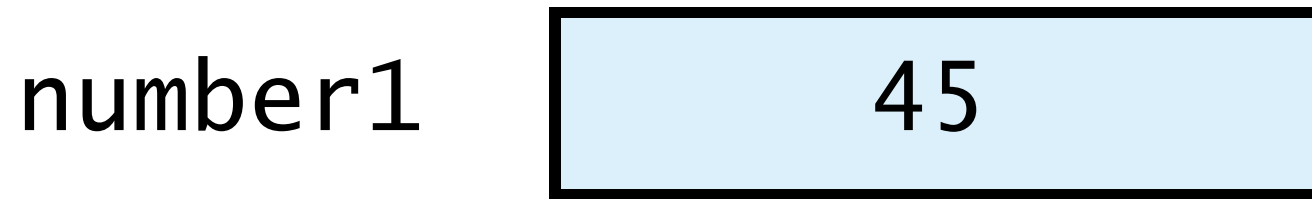


Fig. 7.9 Memory location showing the name and value of variable number1.

## 7.4 Memory Concepts



Fig. 7.10 Memory locations after values for variables number1 and number2 have been input.

## 7.4 Memory Concepts

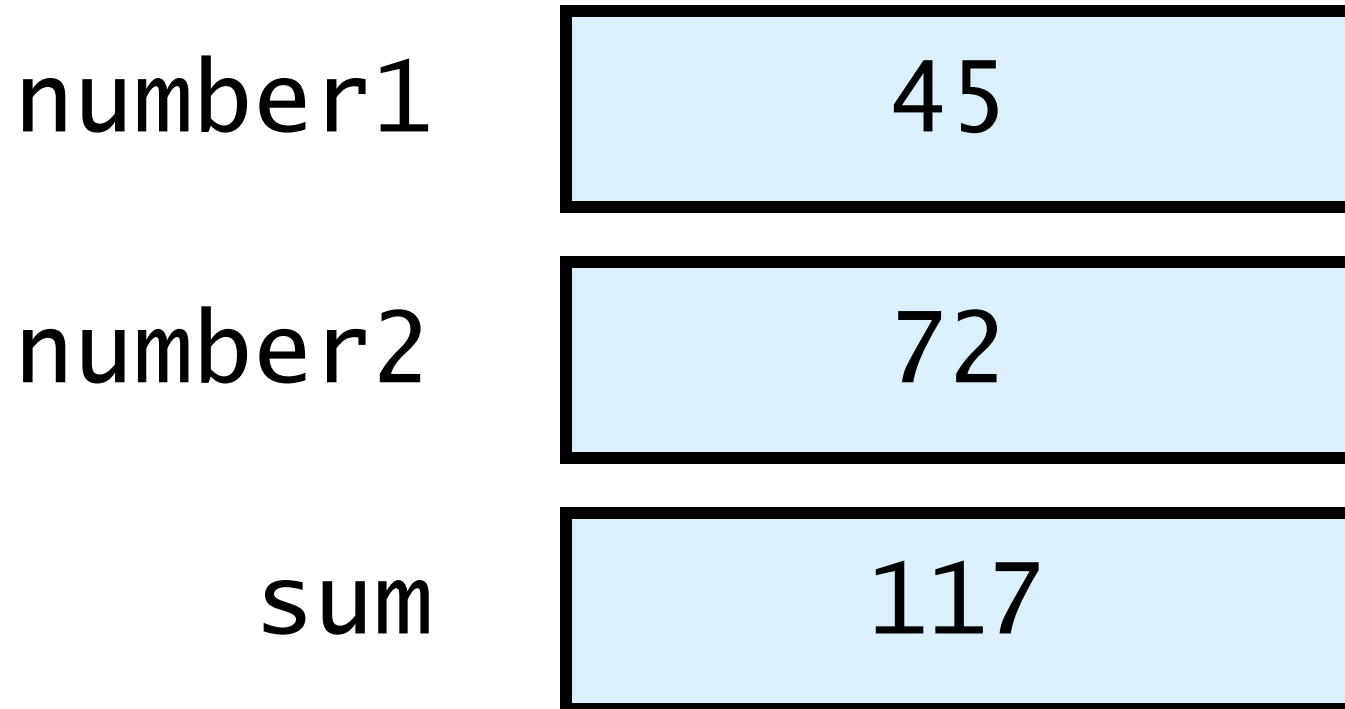


Fig. 7.11 Memory locations after calculating the sum of number1 and number2.

## 7.5 Arithmetic

- Many scripts perform arithmetic calculations
  - Expressions in JavaScript must be written in straight-line form

# 7.5 Arithmetic

JavaScript operation	Arithmetic operator	Algebraic expression	JavaScript expression
Addition	+	$f + 7$	<code>f + 7</code>
Subtraction	-	$p - c$	<code>p - c</code>
Multiplication	*	$bm$	<code>b * m</code>
Division	/	$x / y$ or $\frac{x}{y}$ or $xy$	<code>x / y</code>
Remainder	%	$r \text{ mod } s$	<code>r % s</code>

Fig. 7.12 Arithmetic operators.

Operator(s)	Operation(s)	Order of evaluation (precedence)
*, / or %	Multiplication Division Modulus	Evaluated second. If there are several such operations, they are evaluated from left to right.
+ or -	Addition Subtraction	Evaluated last. If there are several such operations, they are evaluated from left to right.

Fig. 7.13 Precedence of arithmetic operators.



# 7.5 Arithmetic

Step 1.  $y = 2 * 5 * 5 + 3 * 5 + 7;$

$2 * 5$  is **10**

*(Leftmost multiplication)*

Step 2.  $y = 10 * 5 + 3 * 5 + 7;$

$10 * 5$  is **50**

*(Leftmost multiplication)*

Step 3.  $y = 50 + 3 * 5 + 7;$

$3 * 5$  is **15**

*(Multiplication before addition)*

Step 4.  $y = 50 + 15 + 7;$

$50 + 15$  is **65**

*(Leftmost addition)*

Step 5.  $y = 65 + 7;$

$65 + 7$  is **72**

*(Last addition)*

Step 6.  $y = 72;$

*(Last operation—place 72 into Y )*

Fig. 7.14 Order in which a second-degree polynomial is evaluated.

## 7.6 Decision Making: Equality and Relational Operators

- Decision based on the truth or falsity of a condition
  - Equality operators
  - Relational operators

# 7.6 Decision Making: Equality and Relational Operators

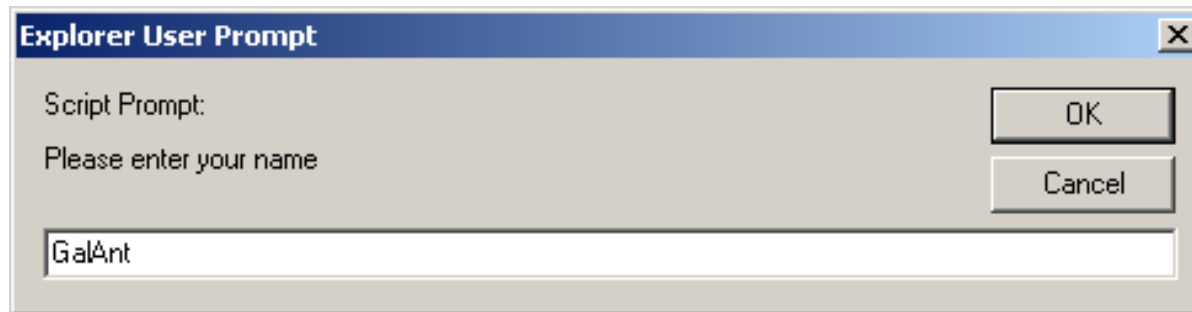
Standard algebraic equality operator or relational operator	JavaScript equality or relational operator	Sample JavaScript condition	Meaning of JavaScript condition
<i>Equality operators</i>			
=	==	x == y	x is equal to y
?	!=	x != y	x is not equal to y
<i>Relational operators</i>			
>	>	x > y	x is greater than y
<	<	x < y	x is less than y
≥	>=	x >= y	x is greater than or equal to y
≤	<=	x <= y	x is less than or equal to y

Fig. 7.15 Equality and relational operators.

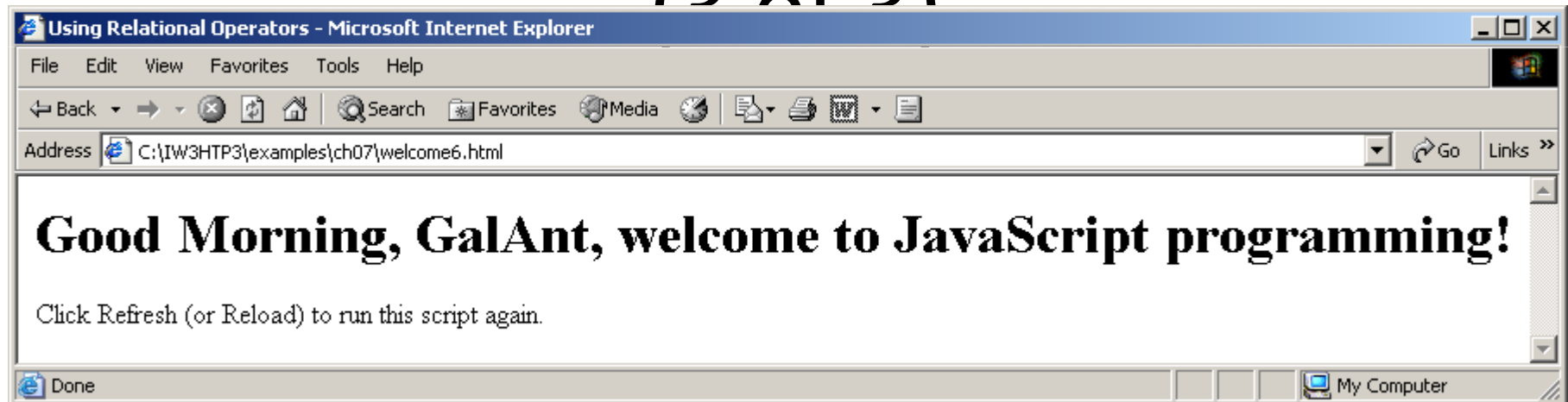
```
1 <?xml version = "1.0"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
3   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
4
5 <!-- Fig. 7.16: welcome6.html -->
6 <!-- Using Relational Operators -->
7
8 <html xmlns = "http://www.w3.org/1999/xhtml">
9   <head>
10     <title>Using Relational Operators</title>
11
12     <script type = "text/javascript">
13       <!--
14       var name, // string entered by the user
15           now = new Date(), // current date and time
16           hour = now.getHours(); // current hour (0-23)
17
18       // read the name from the prompt box as a string
19       name = window.prompt( "Please enter your name", "GalAnt" );
20
21       // determine whether it is morning
22       if ( hour < 12 )
23         document.write( "<h1>Good Morning, " );
24
```

```
25     // determine whether the time is PM
26     if ( hour >= 12 )
27     {
28         // convert to a 12 hour clock
29         hour = hour - 12;
30
31         // determine whether it is before 6 PM
32         if ( hour < 6 )
33             document.write( "<h1>Good Afternoon, " );
34
35         // determine whether it is after 6 PM
36         if ( hour >= 6 )
37             document.write( "<h1>Good Evening, " );
38     }
39
40     document.writeln( name +
41         ", welcome to JavaScript programming!</h1>" );
42     // -->
43 </script>
44
45 </head>
46
```

```
47 <body>
48   <p>Click Refresh (or Reload) to run this script again.</p>
49 </body>
50 </html>
```



(2 of 2)



## 7.6 Decision Making: Equality and Relational Operators

Operators	Associativity	Type
* / %	left to right	multiplicative
+ -	left to right	additive
< <= > >=	left to right	relational
== !=	left to right	equality
=	right to left	assignment
Fig. 7.17 Precedence and associativity of the operators discussed so far.		