6

JavaScript: Introduction to Scripting

OBJECTIVES

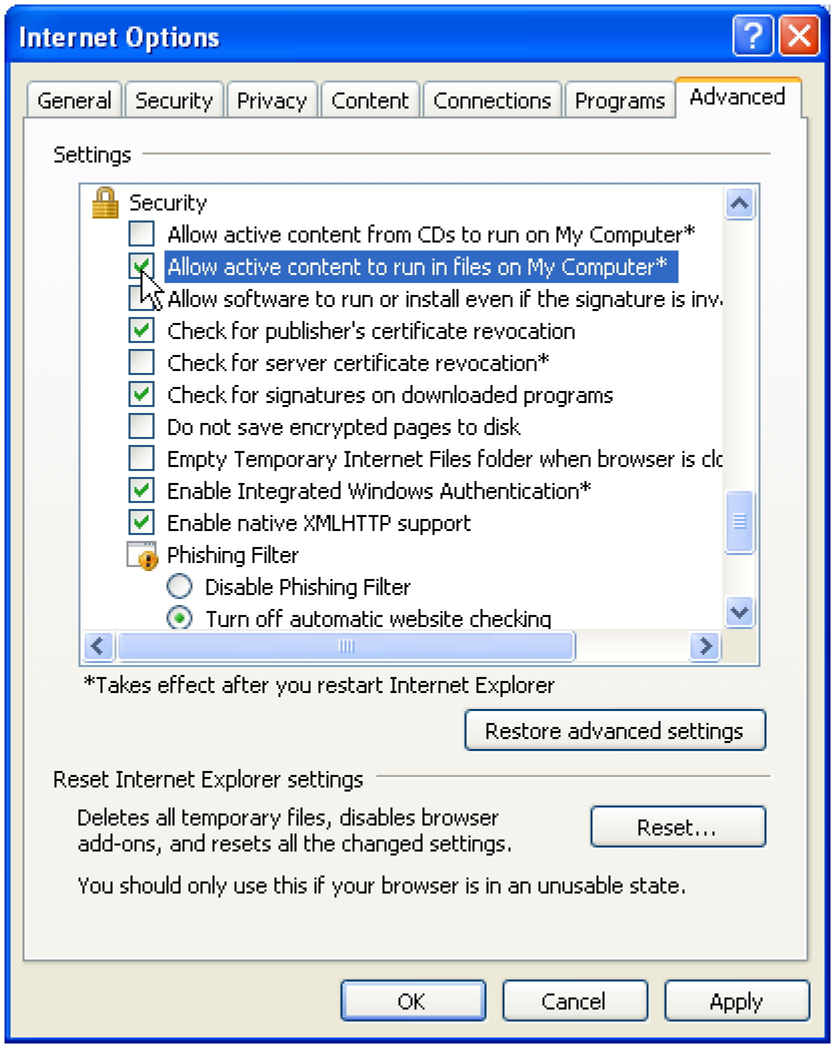
In this chapter you will learn:

- To write simple JavaScript programs.
- To use input and output statements.
- Basic memory concepts.
- To use arithmetic operators.
- The precedence of arithmetic operators.
- To write decision-making statements.
- To use relational and equality operators.

Outline

6.1 Introduction

- JavaScript
  - Scripting language that facilitates a disciplined approach to designing computer programs that enhance the functionality and appearance of web pages.
- Before you can run code examples with JavaScript on your computer, you may need to change your browser’s security settings.
  - IE7 prevents scripts on the local computer from running by default
  - FF2 enables JavaScript by default



Settings

Security
- [ ] Allow active content from CDs to run on My Computer*
- [✓] Allow active content to run in files on My Computer*
- [ ] Allow software to run or install even if the signature is inv
- [✓] Check for publisher's certificate revocation
- [ ] Check for server certificate revocation*
- [✓] Check for signatures on downloaded programs
- [ ] Do not save encrypted pages to disk
- [ ] Empty Temporary Internet Files folder when browser is cl
- [✓] Enable Integrated Windows Authentication*
- [✓] Enable native XMLHTTP support

Phishing Filter
- ○ Disable Phishing Filter
- ● Turn off automatic website checking

*Takes effect after you restart Internet Explorer

Reset Internet Explorer settings

Deletes all temporary files, disables browser add-ons, and resets all the changed settings.

You should only use this if your browser is in an unusable state.

Fig. 6.1 | Enabling JavaScript in Internet Explorer 7

6.2 Simple Program: Displaying a Line of Text in a Web Page

- **Spacing displayed by a browser in a web page is determined by the XHTML elements used to format the page**
- **Often, JavaScripts appear in the `<head>` section of the XHTML document**
- **The browser interprets the contents of the `<head>` section first**
- **The `<script>` tag indicates to the browser that the text that follows is part of a script. Attribute type specifies the scripting language used in the script—such as `text/javascript`**

6.2 Simple Program: Displaying a Line of Text in a Web Page (Cont.)

- **A string of characters can be contained between double (") or single (') quotation marks**
- **A string is sometimes called a character string, a message or a string literal**

6.2 Simple Program: Displaying a Line of Text in a Web Page (Cont.)

- The parentheses following the name of a method contain the arguments that the method requires to perform its task (or its action)
- Every statement should end with a semicolon (also known as the statement terminator), although none is required by JavaScript
- JavaScript is case sensitive
  - Not using the proper uppercase and lowercase letters is a syntax error

6.2 Simple Program: Displaying a Line of Text in a Web Page (Cont.)

- The `document` object’s `writeln` method
  - Writes a line of XHTML text in the XHTML document
  - Does not guarantee that a corresponding line of text will appear in the XHTML document.
  - Text displayed is dependent on the contents of the string written, which is subsequently rendered by the browser.
  - Browser will interpret the XHTML elements as it normally does to render the final text in the document

Fig. 6.2 | Displaying a line of text.

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- Fig. 6.2: welcome.html -->
<!-- Displaying a line of text. -->
<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>A First Program in JavaScript</title>
      <script type = "text/javascript">
         <!--
         document.writeln(
            "<h1>Welcome to JavaScript Programming!</h1>" );
         // -->
      </script>
   </head><body></body>
</html>
```

Script begins

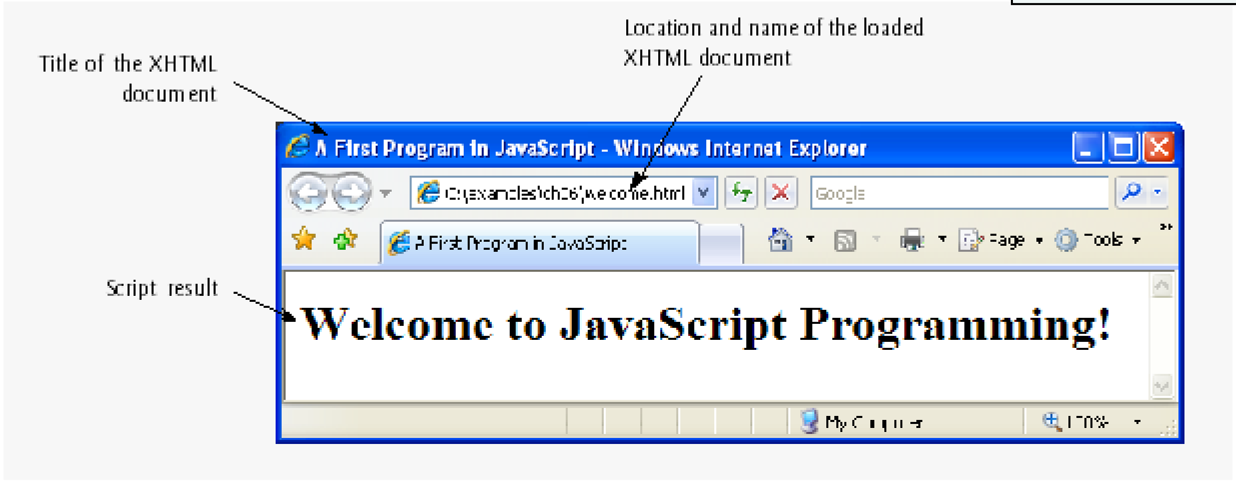
Specifies th
the JavaScri

Prevents older browsers that do not support scripting from displaying the text of the script

Writes
messa
document

XHTML comment delimiter, commented for correct interpretation by all browsers

Script ends


Title of the XHTML document
Location and name of the loaded XHTML document
Script result
A First Program in JavaScript - Windows Internet Explorer
C:\examples\ch06\welcome.html
Google
A First Program in JavaScript
Page
Tools
My Computer
100%

Welcome to JavaScript Programming!

6.3 Modifying Our First Program

- **Method `write` displays a string like `writeln`, but does not position the output cursor in the XHTML document at the beginning of the next line after writing its argument**

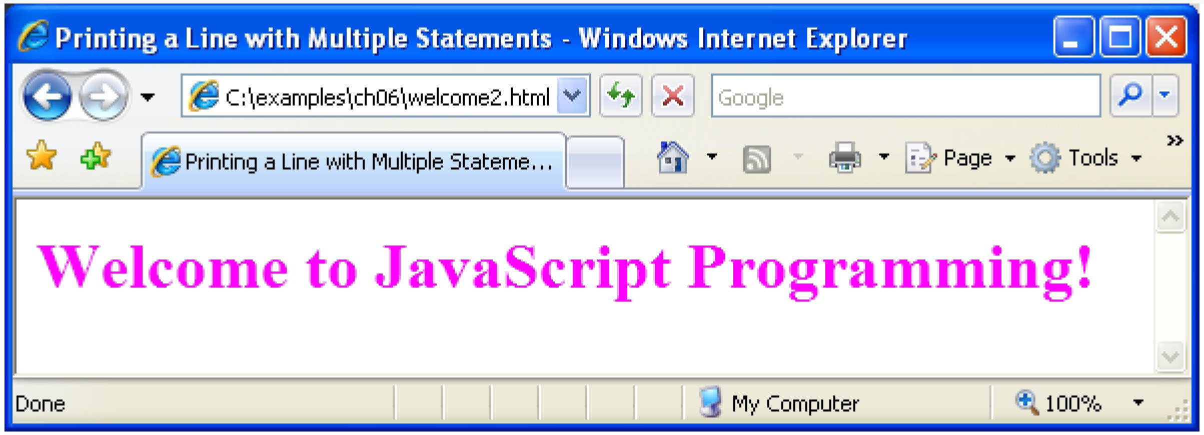
- **You cannot split a statement in the middle of a string. The + operator (called the “concatenation operator” when used in this manner) joins two strings together**

Fig. 6.3 | Printing one line with separate statements.

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- Fig. 6.3: welcome2.html -->
<!-- Printing one line with multiple statements. -->
<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Printing a Line with Multiple Statements</title>
      <script type = "text/javascript">
         <!--
         document.write( "<h1 style = \"color: magenta\">" );
         document.write( "Welcome to JavaScript " +
            "Programming!</h1>" );
         // -->
      </script>
   </head><body></body>
</html>
```

Two `write` statements create one line of XHTML text

Concatenation operator joins the string together, as it is split into multiple lines


Printing a Line with Multiple Statements - Windows Internet Explorer
C:\examples\ch06\welcome2.html
Google
Printing a Line with Multiple Stateme...
Page
Tools
Done
My Computer
100%

Welcome to JavaScript Programming!

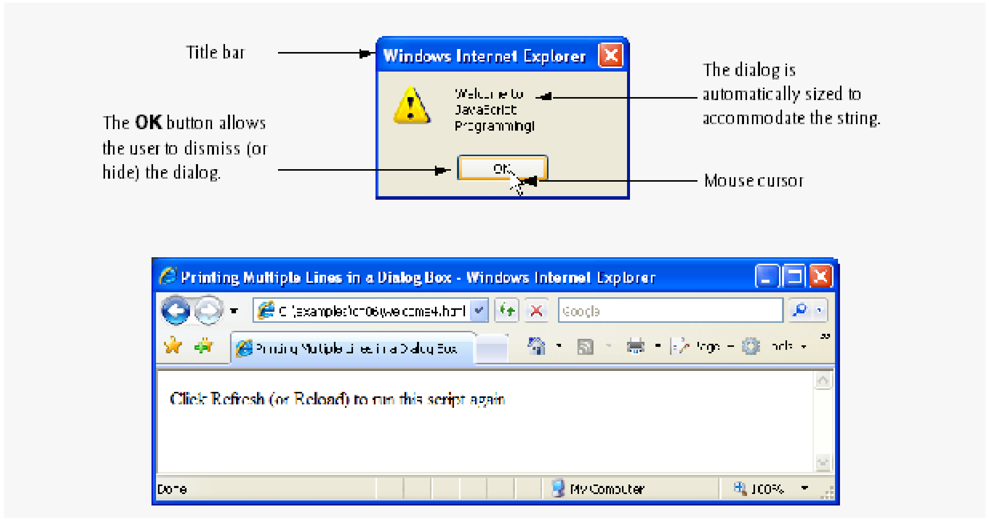
6.3 Modifying Our First Program (Cont.)

- **Dialogs**
  - **Useful to display information in windows that “pop up” on the screen to grab the user’s attention**
  - **Typically used to display important messages to the user browsing the web page**
  - **Browser’s `window` object uses method `alert` to display an alert dialog**
  - **Method `alert` requires as its argument the string to be displayed**

Fig. 6.5 | Alert dialog displaying multiple lines.

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- Fig. 6.5: welcome4.html -->
<!-- Alert dialog displaying multiple lines. -->
<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Printing Multiple Lines in a Dialog Box</title>
      <script type = "text/javascript">
         <!--
            window.alert( "Welcome to\nJavaScript\nProgramming!" );
         // -->
      </script>
   </head>
   <body>
      <p>Click Refresh (or Reload) to run this script again.</p>
   </body>
</html>
```

Creates a pop up box that alerts the welcome text to the user


Title bar
Windows Internet Explorer
The dialog is automatically sized to accommodate the string.
The **OK** button allows the user to dismiss (or hide) the dialog.
OK
Mouse cursor
Printing Multiple Lines in a Dialog Box - Windows Internet Explorer
C:\examples\ch06\welcome4.html
Google
Done
My Computer
100%

Welcome to
JavaScript
Programming!

Click Refresh (or Reload) to run this script again.

6.3 Modifying Our First Program (Cont.)

- **When a backslash is encountered in a string of characters, the next character is combined with the backslash to form an escape sequence. The escape sequence `\n` is the newline character. It causes the cursor in the XHTML document to move to the beginning of the next line.**

Fig. 6.6 | Some common escape sequences.

Escape sequence	Description
`\n`	New line. Position the screen cursor at the beginning of the next line.
`\t`	Horizontal tab. Move the screen cursor to the next tab stop.
`\r`	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.
`\\`	Backslash. Used to represent a backslash character in a string.
`\"`	Double quote. Used to represent a double-quote character in a string contained in double quotes. For example, `window.alert( "\"in quotes\"" );` displays `"in quotes"` in an `alert` dialog.
`\'`	Single quote. Used to represent a single-quote character in a string. For example, `window.alert( '\'in quotes\'' );` displays `'in quotes'` in an `alert` dialog.

6.4 Obtaining User Input with `prompt` Dialogs (Cont.)

- **Declarations end with a semicolon (`;`) and can be split over several lines, with each variable in the declaration separated by a comma (forming a comma-separated list of variable names)**
  - **Several variables may be declared in one declaration or in multiple declarations.**
- **Comments**
  - **A single-line comment begins with the characters `//` and terminates at the end of the line**
  - **Comments do not cause the browser to perform any action when the script is interpreted; rather, comments are ignored by the JavaScript interpreter**
  - **Multiline comments begin with delimiter `/*` and end with delimiter `*/`**
    - **All text between the delimiters of the comment is ignored by the interpreter.**

6.4 Obtaining User Input with `prompt` Dialogs (Cont.)

- The `window` object's `prompt` method displays a dialog into which the user can type a value.
  - The first argument is a message (called a prompt) that directs the user to take a specific action.
  - The optional second argument is the default string to display in the text field.
- Script can then use the value that the user inputs.

6.4 Obtaining User Input with `prompt` Dialogs (Cont.)

- `null` keyword
  - Signifies that a variable has no value
  - `null` is not a string literal, but rather a predefined term indicating the absence of value
  - Writing a `null` value to the document, however, displays the word "`null`"
- Function `parseInt`
  - converts its string argument to an integer
- JavaScript has a version of the + operator for string concatenation that enables a string and a value of another data type (including another string) to be concatenated

Fig. 6.7 | Prompt box used on a welcome screen (Part 1 of 2).

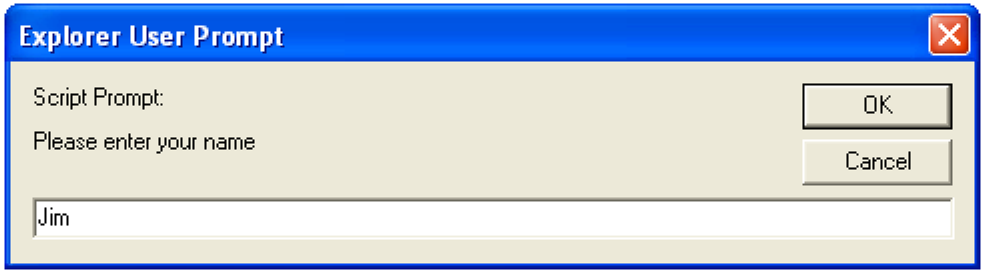
```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- Fig. 6.7: welcome5.html -->
<!-- Prompt box used on a welcome screen. -->
<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Using Prompt and Alert Boxes</title>
      <script type = "text/javascript">
         <!--
         var name; // string entered by the user

         // read the name from the prompt box as a string
         name = window.prompt( "Please enter your name" );

         document.writeln( "<h1>Hello, " + name +
            ", welcome to JavaScript programming!</h1>" );
         // -->
      </script>
   </head>
   <body>
      <p>Click Refresh (or Reload) to run this script again.</p>
   </body>
</html>
```

Declares a new variable

Sets the identifier of the variable to `name`

Inserts the value given to `name` into the XHTML text

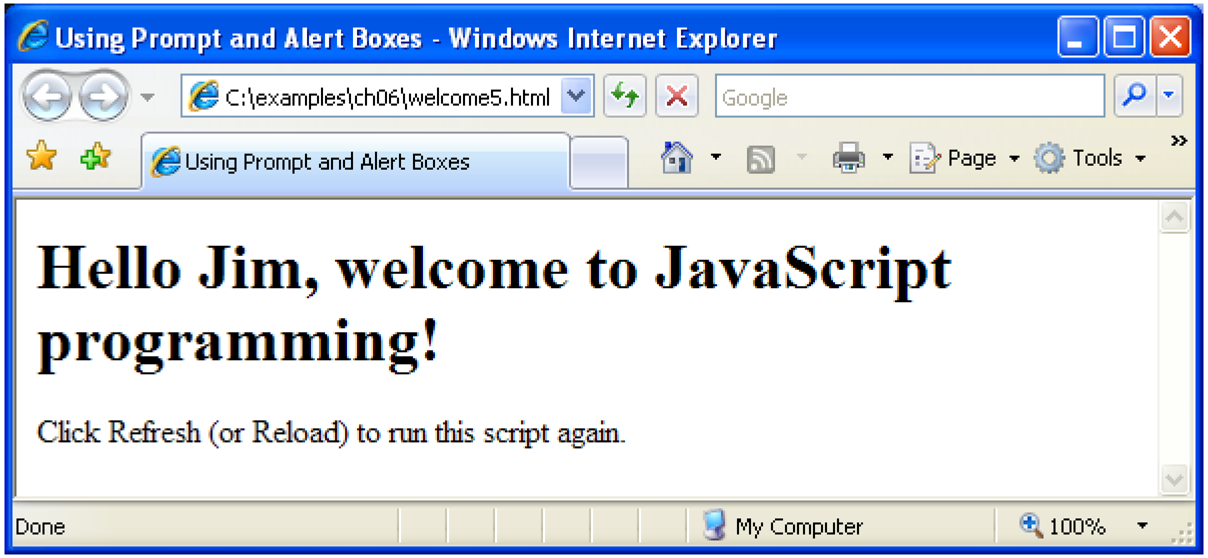
Assigns the string entered by the user to the variable `name`



Script Prompt:

Please enter your name

Jim


Using Prompt and Alert Boxes - Windows Internet Explorer
C:\examples\ch06\welcome5.html
Google
Using Prompt and Alert Boxes
Page
Tools
Done
My Computer
100%

Hello Jim, welcome to JavaScript programming!

Click Refresh (or Reload) to run this script again.

Fig. 6.7 | Prompt box used on a welcome screen (Part 2 of 2).

Fig. 6.9 | Addition script (Part 1 of 2).

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- Fig. 6.9: addition.html -->
<!-- Addition script. -->
<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>An Addition Program</title>
      <script type = "text/javascript">
         <!--
         var firstNumber; // first string entered by user
         var secondNumber; // second string entered by user
         var number1; // first number to add
         var number2; // second number to add
         var sum; // sum of number1 and number2

         // read in first number from user as a string
         firstNumber = window.prompt( "Enter first integer" );

         // read in second number from user as a string
         secondNumber = window.prompt( "Enter second integer" );

         // convert numbers from strings to integers
         number1 = parseInt( firstNumber );
         number2 = parseInt( secondNumber );

         sum = number1 + number2; // add the numbers

```

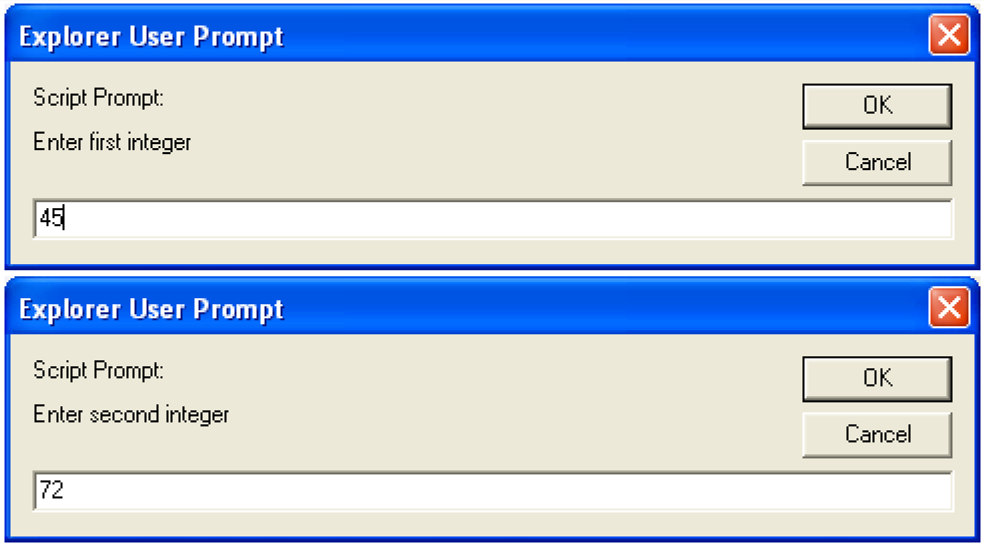
Assigns the first input from the user to the variable `firstNumber`

Assigns the second input from the user to the variable `secondNumber`

Converts the strings entered by the user into integers

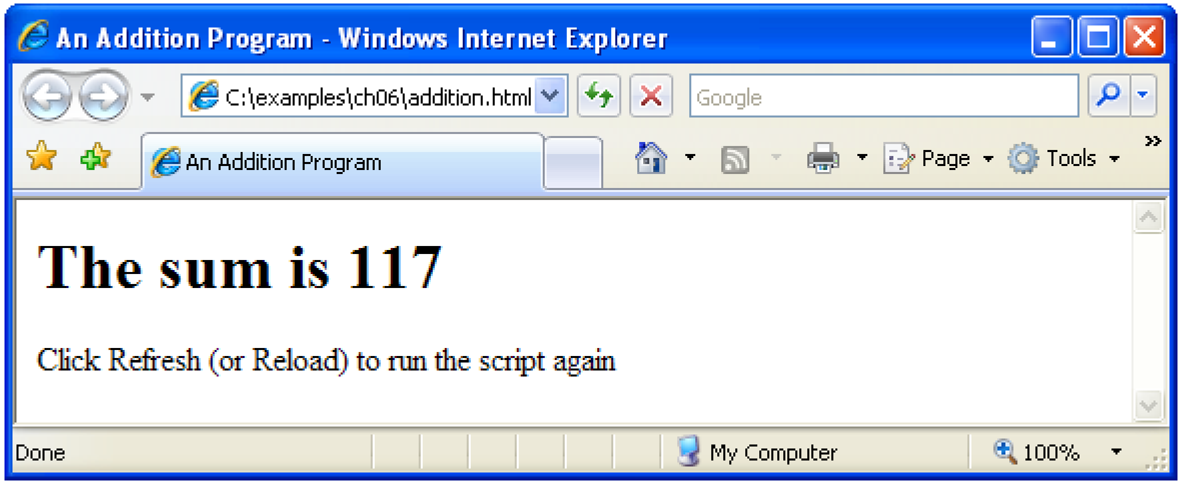
```
            // display the results
            document.writeln( "<h1>The sum is " + sum + "</h1>" );
            // -->
         </script>
      </head>
      <body>
         <p>Click Refresh (or Reload) to run the script again</p>
      </body>
   </html>
```

Fig. 6.9 | Addition script (Part 2 of 2).



Script Prompt:
Enter first integer
45

Script Prompt:
Enter second integer
72


An Addition Program - Windows Internet Explorer
C:\examples\ch06\addition.html
Google
An Addition Program
Page
Tools
Done
My Computer
100%

The sum is 117

Click Refresh (or Reload) to run the script again

6.5 Memory Concepts

- Variable names correspond to locations in the computer’s memory.
- Every variable has a name, a type and a value.
- When a value is placed in a memory location, the value replaces the previous value in that location.
- When a value is read out of a memory location, the process is nondestructive.

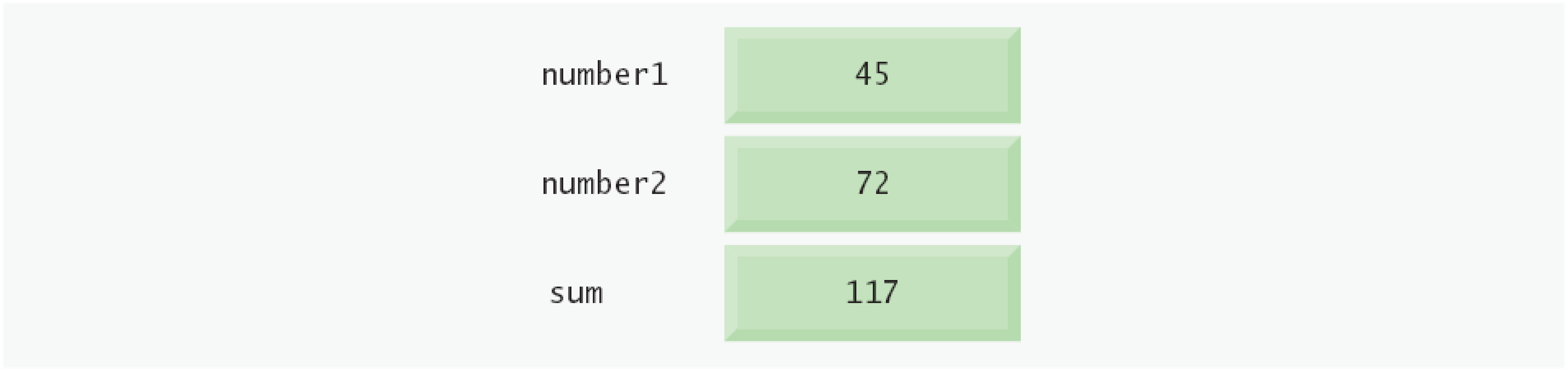


Fig. 6.12 | Memory locations after calculating the `sum` of `number1` and `number2` .

6.5 Memory Concepts (Cont.)

- **JavaScript does not require variables to have a type before they can be used in a program**
- **A variable in JavaScript can contain a value of any data type, and in many situations, JavaScript automatically converts between values of different types for you**
- **JavaScript is referred to as a loosely typed language**
- **When a variable is declared in JavaScript, but is not given a value, it has an undefined value.**
  - **Attempting to use the value of such a variable is normally a logic error.**
- **When variables are declared, they are not assigned default values, unless specified otherwise by the programmer.**
  - **To indicate that a variable does not contain a value, you can assign the value `null` to it.**

6.6 Arithmetic

- **The basic arithmetic operators (+, −, *, /, and %) are binary operators, because they each operate on two operands**
- **JavaScript provides the remainder operator, %, which yields the remainder after division**

6.6 Arithmetic (Cont.)

- **Parentheses can be used to group expressions as in algebra.**
- **Operators in arithmetic expressions are applied in a precise sequence determined by the rules of operator precedence:**
  - **Multiplication, division and remainder operations are applied first.**
  - **If an expression contains several of these operations, operators are applied from left to right.**
  - **Multiplication, division and remainder operations are said to have the same level of precedence.**
  - **Addition and subtraction operations are applied next.**
  - **If an expression contains several of these operations, operators are applied from left to right.**
  - **Addition and subtraction operations have the same level of precedence.**
- **When we say that operators are applied from left to right, we are referring to the associativity of the operators. Some operators associate from right to left.**

Fig. 6.14 | Precedence of arithmetic operators.

Operator(s)	Operation(s)	Order of evaluation (precedence)
*, / or %	Multiplication Division Remainder	Evaluated first. 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.

6.7 Decision Making: Equality and Relational Operators

- `if` statement allows a program to make a decision based on the truth or falsity of a condition
  - If the condition is met (i.e., the condition is `true`), the statement in the body of the if statement is executed
  - If the condition is not met (i.e., the condition is `false`), the statement in the body of the `if` statement is not executed
- Conditions in `if` statements can be formed by using the equality operators and relational operators

Fig. 6.16 | Equality and relational operators.

Standard algebraic equality operator or relational operator	JavaScript equality or relational operator	Sample JavaScript condition	Meaning of JavaScript condition
Equality operators			
=	`==`	`x == y`	`x` is equal to `y`
	`!=`	`x != y`	`x` is not equal to `y`
Relational operators			
>	`>`	`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`

6.7 Decision Making: Equality and Relational Operators (Cont.)

- **Equality operators both have the same level of precedence, which is lower than the precedence of the relational operators.**
- **The equality operators associate from left to right.**

6.7 Decision Making: Equality and Relational Operators (Cont.)

- `Date` object
  - Used acquire the current local time
  - Create a new instance of an object by using the `new` operator followed by the type of the object, `Date`, and a pair of parentheses

Fig. 6.17 | Using equality and relational operators (Part 1 of 3).

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- Fig. 6.17: welcome6.html -->
<!-- Using equality and relational operators. -->
<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Using Relational Operators</title>
      <script type = "text/javascript">
         <!--
         var name; // string entered by the user
         var now = new Date();      // current date and time
         var hour = now.getHours(); // current hour (0-23)

         // read the name from the prompt box as a string
         name = window.prompt( "Please enter your name" );

         // determine whether it is morning
         if ( hour < 12 )
            document.write( "<h1>Good Morning, " );

```

Set variable `now` to a new `Date` object

Assigns `hour` to the value returned by the `Date` object's `getHours` method

Conditional statement: checks whether the current value of `hour` is less than 12

This statement will execute only if the previous condition was true

Fig. 6.17 | Using equality and relational rators (Part 3).

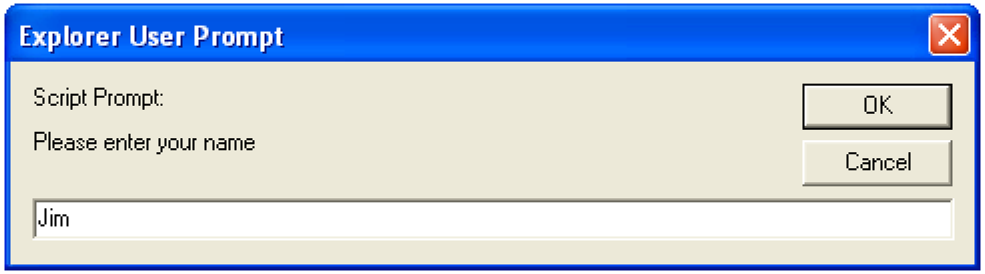
```
         // determine whether the time is PM
         if ( hour >= 12 )
         {
            // convert to a 12-hour clock
            hour = hour - 12;

            // determine whether it is before 6 PM
            if ( hour < 6 )
               document.write( "<h1>Good Afternoon, " );

            // determine whether it is after 6 PM
            if ( hour >= 6 )
               document.write( "<h1>Good Evening, " );
         } // end if

         document.writeln( name +
            ", welcome to JavaScript programming!</h1>" );
         // -->
      </script>
   </head>
   <body>
      <p>Click Refresh (or Reload) to run this script again.</p>
   </body>
</html>
```

Conditional statement: checks whether the current value of `hour` is greater than or equal to 12

If `hour` is 12 or greater (the previous condition was true), subtract 12 from the value and reassign it to `hour`

Conditional statement: checks whether the current value of `hour` is less than 6

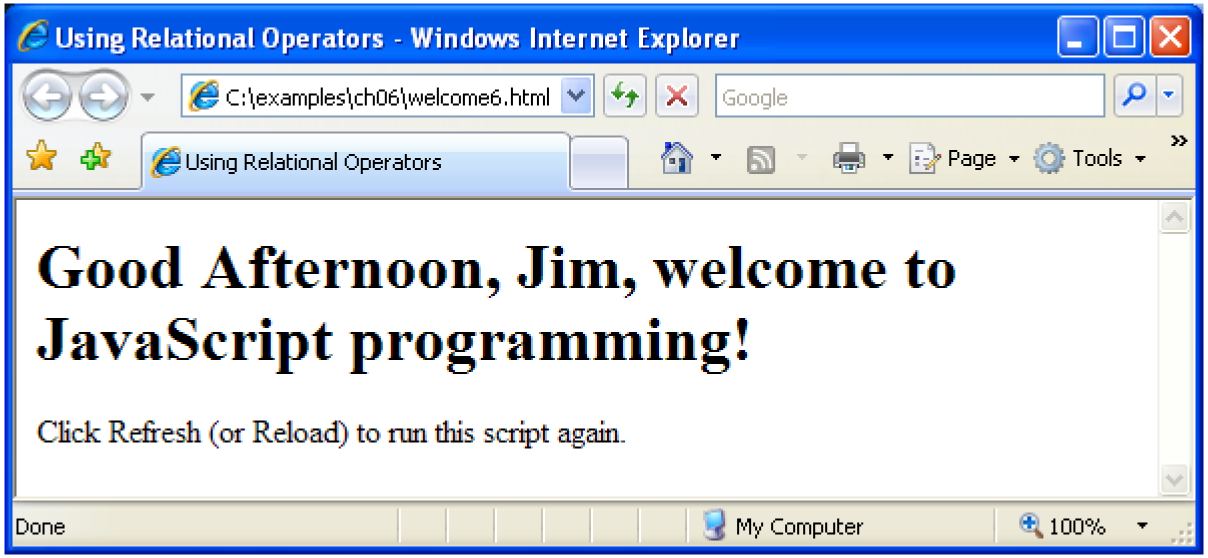
Conditional statement: checks whether the current value of `hour` is greater than or equal to 6



Script Prompt:

Please enter your name

Jim


Using Relational Operators - Windows Internet Explorer
C:\examples\ch06\welcome6.html
Google
Using Relational Operators
Page
Tools
Done
My Computer
100%

Good Afternoon, Jim, welcome to JavaScript programming!

Click Refresh (or Reload) to run this script again.

Fig. 6.17 | Using equality and relational operators (Part 3 of 3).

Fig. 6.18 | Precedence and associativity of the operators discussed so far.

Operators	Associativity	Type
`*` `/` `%`	left to right	multiplicative
`+` `-`	left to right	additive
`<` `<=` `>` `>=`	left to right	relational
`==` `!=`	left to right	equality
`=`	right to left	assignment