

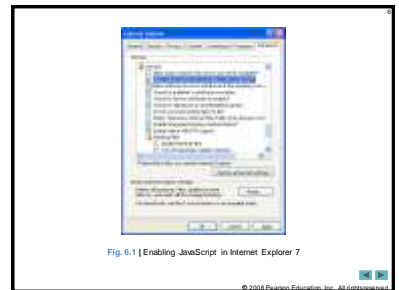
# 6 JavaScript: Introduction to Scripting

*Comment is free, but facts are sacred.*  
—C. P. Scott  
*The creditor hath a better memory than the debtor.*  
—James Howell  
*When faced with a decision, I always ask, “What would be the most fun?”*  
—Peggy Walker  
*Equality, in a social sense, may be divided into that of condition and that of rights.*  
—James Fenimore Cooper

- ### 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.

|     |   |
|-----|---|
| 6.1 | Introduction  |
| 6.2 | Simple Program: Displaying a Line of Text in a Web Page |
| 6.3 | Modifying Our First Program                             |
| 6.4 | Obtaining User Input with prompt Dialogs                |
| 6.5 | Memory Concepts   |
| 6.6 | Arithmetic  |
| 6.7 | Decision Making: Equality and Relational Operators      |
| 6.8 | Wrap-Up   |
| 6.9 | Web Resources   |

- ### 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



- ### 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**

### Portability Tip 6.1

Some browsers do not support the **<script>** **</script>** tags. If your document is to be rendered with such browsers, enclose the script code between these tags in an XHTML comment, so that the script text does not get displayed as part of the web page. The closing comment tag of the XHTML comment (**</-->**) is preceded by a JavaScript comment (**//**) to prevent the browser from trying to interpret the XHTML comment as a JavaScript statement.

### Common Programming Error 6.1

Forgetting the ending **</script>** tag for a script may prevent the browser from interpreting the script properly and may prevent the XHTML document from loading properly.

## 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

© 2008 Pearson Education, Inc. All rights reserved.

## Software Engineering Observation 6.1

Strings in JavaScript can be enclosed in either double quotation marks (") or single quotation marks (').

© 2008 Pearson Education, Inc. All rights reserved.

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

- Browser's **document** object represents the XHTML document currently being displayed in the browser
  - Allows a script programmer to specify XHTML text to be displayed in the XHTML document.
- Browser contains a complete set of objects that allow script programmers to access and manipulate every element of an XHTML document
- **Object**
  - Resides in the computer's memory and contains information used by the script
  - The term object normally implies that attributes (data) and behaviors (methods) are associated with the object
  - An object's methods use the attributes' data to perform useful actions for the client of the object—the script that calls the methods

© 2008 Pearson Education, Inc. All rights reserved.

## 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

© 2008 Pearson Education, Inc. All rights reserved.

## Good Programming Practice 6.1

Always include a semicolon at the end of a statement to terminate the statement. This notation clarifies where one statement ends and the next statement begins.

© 2008 Pearson Education, Inc. All rights reserved.

## Common Programming Error 6.2

JavaScript is case sensitive. Not using the proper uppercase and lowercase letters is a syntax error. A syntax error occurs when the script interpreter cannot recognize a statement. The interpreter normally issues an error message to help you locate and fix the incorrect statement. Syntax errors are violations of the rules of the programming language. The interpreter notifies you of a syntax error when it attempts to execute the statement containing the error. The JavaScript interpreter in Internet Explorer reports all syntax errors by indicating in a separate popup window that a "runtime error" has occurred (i.e., a problem occurred while the interpreter was running the script). Note: To enable this feature in IE7, select *Internet Options* from the *Tools* menu. In the *Internet Options* dialog that appears, select the *Advanced* tab and click the checkbox labeled *Display a notification about every script error* under the *Browsing* category. Firefox has an error console that reports JavaScript errors and warnings. It is accessible by choosing *Error Console* from the *Tools* menu.

© 2008 Pearson Education, Inc. All rights reserved.

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

- The **document** object's **write()** 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

© 2008 Pearson Education, Inc. All rights reserved.

Fig. 6.2 | Displaying a line of text.

Script begins

Specifies the JavaScript

Prevents other browsers that do not support scripting from displaying across the text of the script.

Writes

XHTML comment delimiter, commented for correct interpretation by all browsers.

Script ends

© 2008 Pearson Education, Inc. All rights reserved.

## Error-Prevention Tip 6.1

When the interpreter reports a syntax error, sometimes the error is not on the line number indicated by the error message. First, check the line for which the error was reported. If that line does not contain errors, check the preceding several lines in the script.

© 2008 Pearson Education, Inc. All rights reserved.

### 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

### 6.3 Modifying Our First Program (Cont.)

- 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

### Common Programming Error 6.3

Splitting a statement in the middle of a string is a syntax error.

```
1 <html version="1.0" encoding="UTF-8" >
2 <script src="http://www.js.com/js/1.0/src/js.js"
3 </script> </html>
4
5 <!-- Fig. 6.3: welcome.html -->
6 <!-- Printing one line with multiple statements -->
7 <html title = "http://www.csl.org/2008/js01/" >
8 <head>
9 <!-- Printing a line with multiple statements -->
10 <script type = "text/javascript">
11 <code>document.write("Welcome to JavaScript Programming!");
12 </code>
13 </script>
14 </head>
15 </html>
```

**Fig. 6.3 | Printing one line with separate statements.**

Two `write` statements create one line of XHTML text.

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

### Common Programming Error 6.4

Many people confuse the writing of XHTML text with the rendering of XHTML text. Writing XHTML text creates the XHTML that will be rendered by the browser for presentation to the user.

### 6.3 Modifying Our First Program (Cont.)

- A single statement can cause the browser to display multiple lines by using line-break XHTML tags (`<br/>`) throughout the string of XHTML text in a `write` or `writeLn` method call

```
1 <html version="1.0" encoding="UTF-8" >
2 <script src="http://www.js.com/js/1.0/src/js.js"
3 </script> </html>
4
5 <!-- Fig. 6.4: welcome.html -->
6 <!-- Printing on multiple lines with a single statement -->
7 <html title = "http://www.csl.org/2008/js01/" >
8 <head>
9 <!-- Printing multiple lines -->
10 <script type = "text/javascript">
11 <code>document.write("Welcome to JavaScript Programming!");
12 </code>
13 </script>
14 </head>
15 </html>
```

**Fig. 6.4 | Printing on multiple lines with a single statement.**

Inserts line-break

### 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

```
1 <html version="1.0" encoding="UTF-8" >
2 <script src="http://www.js.com/js/1.0/src/js.js"
3 </script> </html>
4
5 <!-- Fig. 6.5: welcome.html -->
6 <!-- Alert dialog displaying multiple lines -->
7 <html title = "http://www.csl.org/2008/js01/" >
8 <head>
9 <!-- Printing multiple lines -->
10 <script type = "text/javascript">
11 <code>document.write("Welcome to JavaScript Programming!");
12 </code>
13 </script>
14 <!-- Click here to run this script again -->
15 </html>
```

**Fig. 6.5 | Alert dialog displaying multiple lines.**

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

### Common Programming Error 6.5

Dialogs display plain text; they do not render XHTML. Therefore, specifying XHTML elements as part of a string to be displayed in a dialog results in the actual characters of the tags being displayed.

### Common Programming Error 6.6

XHTML elements in an alert dialog's message are not interpreted as XHTML. This means that using `<br />`, for example, to create a line break in an alert box is an error. The string `<br />` will simply be included in your message.

### 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.

| Escape sequence | Description   | Fig. 6.6   Some common escape sequences |
|-----------------|---|---|
| <code>\n</code> | New line. 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 contained in double quotes. For example, <code>window.alert( "\\\" in quotes\" );</code> displays "in quotes" in an alert dialog.                            |   |
| <code>\"</code> | Double quote. Used to represent a double-quote character in a string contained in double quotes. For example, <code>window.alert( '\" in quotes\" );</code> displays "in quotes" in an alert dialog.                        |   |
| <code>'</code>  | Single quote. Used to represent a single-quote character in a string. For example, <code>window.alert( \'\' in quotes\'\' );</code> displays "in quotes" in an alert dialog.  |   |

© 2008 Pearson Education, Inc. All rights reserved.

### 6.4 Obtaining User Input with prompt Dialogs

- Scripting
  - Gives you the ability to generate part or all of a web page's content at the time it is shown to the user
  - Such web pages are said to be dynamic, as opposed to static, since their content has the ability to change

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

- Keywords are words with special meaning in JavaScript
- Keyword **var**
  - Used to declare the names of variables
  - A variable is a location in the computer's memory where a value can be stored for use by a program
  - All variables have a name, type and value, and should be declared with a **var** statement before they are used in a program
- A variable name can be any valid identifier consisting of letters, digits, underscores (`_`) and dollar signs (`$`) that does not begin with a digit and is not a reserved JavaScript keyword.

### Good Programming Practice 6.2

Choosing meaningful variable names helps a script to be "self-documenting" (i.e., easy to understand by simply reading the script, rather than having to read manuals or extended comments).

### Good Programming Practice 6.3

By convention, variable-name identifiers begin with a lowercase first letter. Each subsequent word should begin with a capital first letter. For example, identifier `itemPrice` has a capital P in its second word, `Price`.

### Common Programming Error 6.7

Splitting a statement in the middle of an identifier is a syntax error.

## Good Programming Practice 6.4

Some programmers prefer to declare each variable on a separate line. This format allows for easy insertion of a descriptive comment next to each declaration. This is a widely followed professional coding standard.

© 2008 Pearson Education, Inc. All rights reserved.

## 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 over 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.

© 2008 Pearson Education, Inc. All rights reserved.

## Common Programming Error 6.8

Forgetting one of the delimiters of a multiline comment is a syntax error.

© 2008 Pearson Education, Inc. All rights reserved.

## Common Programming Error 6.9

Nesting multiline comments (i.e., placing a multiline comment between the delimiters of another multiline comment) is a syntax error.

© 2008 Pearson Education, Inc. All rights reserved.

## 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.

© 2008 Pearson Education, Inc. All rights reserved.

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

- A variable is assigned a value with an assignment statement, using the assignment operator, =.
- The = operator is called a binary operator, because it has two operands.

© 2008 Pearson Education, Inc. All rights reserved.

## Good Programming Practice 6.5

Place spaces on either side of a binary operator. This format makes the operator stand out and makes the program more readable.

© 2008 Pearson Education, Inc. All rights reserved.

## 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 pre-defined 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

© 2008 Pearson Education, Inc. All rights reserved.

## Common Programming Error 6.10

Confusing the + operator used for string concatenation with the + operator used for addition often leads to undesired results. For example, if integer variable **y** has the value 5, the expression "**y + 2 = 7**", because first the value of **y** (i.e., 5) is concatenated with the string "**y + 2 =** ", then the value 2 is concatenated with the new, larger string "**y + 2 = 5**". The expression "**y + 2 =** " + (**y + 2**) produces the string "**y + 2 = 7**" because the parentheses ensure that **y + 2** is executed mathematically before it is converted to a string.

© 2008 Pearson Education, Inc. All rights reserved.

```

1 <html version="1.0" encoding="utf-8" >
2 <script src="http://www.w3schools.com/js/lib/jquery.js" ></script>
3 <script src="http://www.w3schools.com/js/lib/jquery.cookie.js" ></script>
4 <!-- Fig. 6.7: welcome.html -->
5 <!-- Prompt box used as a welcome screen -->
6 <html lang="en" >
7 <head >
8 <title>Using Prompt and Alert Boxes</title>
9 <script type="text/javascript">
10
11 // var name // string entered by the user
12
13 // read the name from the prompt box as a string
14 name = window.prompt("Please enter your name");
15
16 document.write("Hello, " + name +
17               ", welcome to w3schools programming!");
18 // -->
19 </script>
20 </head>
21 <body >
22 <script>
23 <!--Click Refresh (or reload) to run this script again-->
24 </script>
25 </body>
26 </html>

```

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

- 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

© 2008 Pearson Education, Inc. All rights reserved.

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

© 2008 Pearson Education, Inc. All rights reserved.

**Fig. 6.8 | Prompt dialog displayed by the window object's prompt method.**

© 2008 Pearson Education, Inc. All rights reserved.

```

1 <html version="1.0" encoding="utf-8" >
2 <script src="http://www.w3schools.com/js/lib/jquery.js" ></script>
3 <script src="http://www.w3schools.com/js/lib/jquery.cookie.js" ></script>
4 <!-- Fig. 6.8: add10a.html -->
5 <!-- Addition script -->
6 <html lang="en" >
7 <head >
8 <title>An Addition Program</title>
9 <script type="text/javascript">
10
11 // var firstNumber // first string entered by user
12 // var secondNumber // second string entered by user
13 // var number1 // first number to add
14 // var number2 // second number to add
15 // var sum // sum of number1 and number2
16
17 // read the first number from user as a string
18 firstNumber = window.prompt("Enter a number");
19
20 // read the second number from user as a string
21 secondNumber = window.prompt("Enter a number");
22
23 // convert numbers from strings to integers
24 number1 = parseInt(firstNumber);
25 number2 = parseInt(secondNumber);
26
27 sum = number1 + number2; // add the numbers

```

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

- 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

© 2008 Pearson Education, Inc. All rights reserved.

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

© 2008 Pearson Education, Inc. All rights reserved.

## 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.

© 2008 Pearson Education, Inc. All rights reserved.

**Fig. 6.10 | Memory location showing the name and value of variable number1.**

© 2008 Pearson Education, Inc. All rights reserved.

**Fig. 6.11 | Memory locations after inputting values for variables number1 and number2.**

© 2008 Pearson Education, Inc. All rights reserved.

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

© 2008 Pearson Education, Inc. All rights reserved.

## 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
- Arithmetic expressions in JavaScript must be written in straight-line form to facilitate entering programs into the computer

Fig. 6.13 | Arithmetic operators.

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

## 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.

## Good Programming Practice 6.6

Using parentheses for complex arithmetic expressions, even when the parentheses are not necessary, can make the arithmetic expressions easier to read.

Fig. 6.14 | Precedence of arithmetic operators.

| Operator(s) | Operation(s)                            | Order of evaluation (precedence)  |
|-------------|---|---|
| *, / or %   | Multiplication<br>Division<br>Remainder | Evaluated first. If there are several such operations, they are evaluated from left to right. |
| + or -      | Addition<br>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 |
|---|--|-----------------------------|---------------------------------|
| <b>Equality operators</b>                                   |  |                             |                                 |
| =   | ==   | <code>x == y</code>         | x is equal to y                 |
| !=  | !=   | <code>x != y</code>         | x is not equal to y             |
| <b>Relational operators</b>                                 |  |                             |                                 |
| >   | >  | <code>x &gt; y</code>       | x is greater than y             |
| <   | <  | <code>x &lt; y</code>       | x is less than y                |
| ≥   | >=   | <code>x &gt;= y</code>      | x is greater than or equal to y |
| ≤   | <=   | <code>x &lt;= y</code>      | 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.

© 2008 Pearson Education, Inc. All rights reserved.

## Common Programming Error 6.11

It is a syntax error if the operators `==`, `!=`, `>=` and `<=` contain spaces between their symbols, as in `= =`, `! =`, `> =` and `< =`, respectively.

© 2008 Pearson Education, Inc. All rights reserved.

## Common Programming Error 6.12

Reversing the operators `!=`, `>=` and `<=`, as in `=!`, `=>` and `=<`, respectively, is a syntax error.

© 2008 Pearson Education, Inc. All rights reserved.

## Common Programming Error 6.13

Confusing the equality operator, `==`, with the assignment operator, `=`, is a logic error. The equality operator should be read as “is equal to,” and the assignment operator should be read as “gets” or “gets the value of.” Some people prefer to read the equality operator as “double equals” or “equals equals.”

© 2008 Pearson Education, Inc. All rights reserved.

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

- If more than one variable is declared in a single declaration, the names are separated by commas ( , )
  - This list of names is referred to as a comma-separated list

© 2008 Pearson Education, Inc. All rights reserved.

## 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

© 2008 Pearson Education, Inc. All rights reserved.

```

<html version="1.0" >
<script src="http://www.w3schools.com/js/tryit.asp?filename=try_js_slides_6_7_13.js" type="text/javascript">
</script>
</html>
<!-- Fig. 6.13 | Using equality and relational operators -->
<script src="http://www.w3schools.com/js/tryit.asp?filename=try_js_slides_6_7_13.js" type="text/javascript">
</script>
</html>
    
```

**Fig. 6.13 | Using equality and relational operators (Part 1 of 3).**

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

Assign `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

© 2008 Pearson Education, Inc. All rights reserved.

```

// determine whether the clock is AM
if (hour < 12) {
    // convert to a 12-hour clock
    hour = hour < 10 ?
        // determine whether it is before 6 AM
        // determine whether it is after 6 AM
    } else if (
        // determine whether it is after 6 AM
        // determine whether it is before 6 AM
    ) {
        // welcome to JavaScript programming!
    }
}
</script>
</html>
    
```

**Fig. 6.17 | Using equality and relational operators (Part 2).**

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

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

This statement will execute only if the previous condition was true

© 2008 Pearson Education, Inc. All rights reserved.

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

© 2008 Pearson Education, Inc. All rights reserved.

### Good Programming Practice 6.7

Include comments after the closing curly brace of control statements (such as `if` statements) to indicate where the statements end, as in line 36 of Fig. 6.17.

© 2008 Pearson Education, Inc. All rights reserved.

### Good Programming Practice 6.8

Indent the statement in the body of an `if` statement to make the body of the statement stand out and to enhance program readability.

© 2008 Pearson Education, Inc. All rights reserved.

### Good Programming Practice 6.9

Place only one statement per line in a program. This enhances program readability.

© 2008 Pearson Education, Inc. All rights reserved.

### Common Programming Error 6.14

Forgetting the left and/or right parentheses for the condition in an `if` statement is a syntax error. The parentheses are required.

© 2008 Pearson Education, Inc. All rights reserved.

### Common Programming Error 6.15

Placing a semicolon immediately after the right parenthesis of the condition in an `if` statement is normally a logic error. The semicolon would cause the body of the `if` statement to be empty, so the `if` statement itself would perform no action, regardless of whether its condition was true. Worse yet, the intended body statement of the `if` statement would now become a statement in sequence after the `if` statement and would always be executed.

© 2008 Pearson Education, Inc. All rights reserved.

### Common Programming Error 6.16

Leaving out a condition in a series of `if` statements is normally a logic error. For instance, checking if `hour` is greater than 12 or less than 12, but not if `hour` is equal to 12, would mean that the script takes no action when `hour` is equal to 12. Always be sure to handle every possible condition.

© 2008 Pearson Education, Inc. All rights reserved.

### Good Programming Practice 6.10

A lengthy statement may be spread over several lines. If a single statement must be split across lines, choose breaking points that make sense, such as after a comma in a comma-separated list or after an operator in a lengthy expression. If a statement is split across two or more lines, indent all subsequent lines.

© 2008 Pearson Education, Inc. All rights reserved.

### Good Programming Practice 6.11

Refer to the operator precedence chart when writing expressions containing many operators. Confirm that the operations are performed in the order in which you expect them to be performed. If you are uncertain about the order of evaluation in a complex expression, use parentheses to force the order, exactly as you would do in algebraic expressions. Be sure to observe that some operators, such as assignment (`=`), associate from right to left rather than from left to right.

© 2008 Pearson Education, Inc. All rights reserved.

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     |

© 2008 Pearson Education, Inc. All rights reserved.