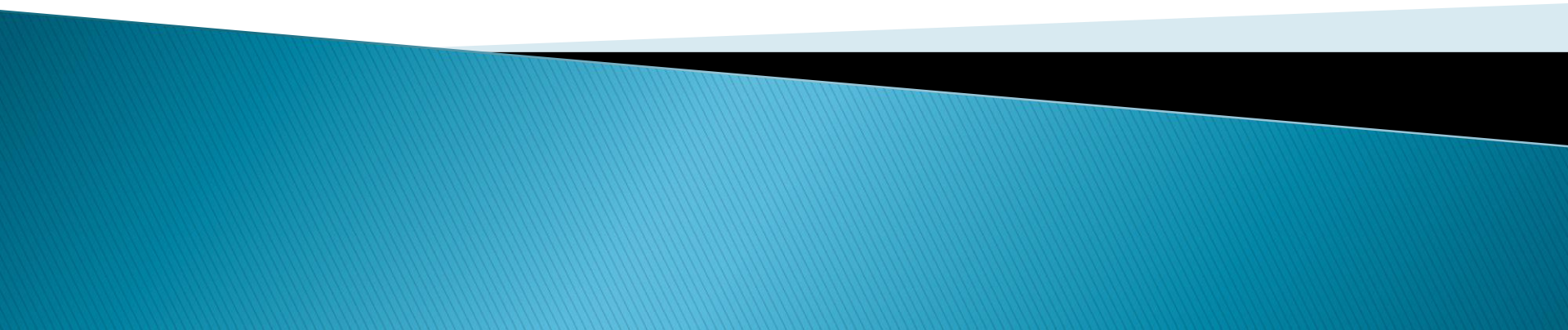
JavaScript: Introduction to Scripting



OBJECTIVES

In this chapter you will:

- Write simple JavaScript programs.
- Use input and output statements.
- Learn basic memory concepts.
- Use arithmetic operators.
- Learn the precedence of arithmetic operators.
- Write decision-making statements to choose among alternative courses of action.
- Use relational and equality operators to compare data items.

6.1 Introduction

- JavaScript
  - Scripting language which is used to 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.

6.2 Your First Script: Displaying a Line of Text with JavaScript in a Web Page

- We begin with a simple script that displays the text "`Welcome to JavaScript Programming!`" in the HTML5 document.
- All major web browsers contain JavaScript **interpreters**, which process the commands written in JavaScript.
- The JavaScript code and its result are shown in Fig. 6.1.

```
1  <!DOCTYPE html>
2
3  <!-- Fig. 6.1: welcome.html -->
4  <!-- Displaying a line of text. -->
5  <html>
6     <head>
7        <meta charset = "utf-8">
8        <title>A First Program in JavaScript</title>
9        <script type = "text/javascript">
10
11          document.writeln(
12             "<h1>Welcome to JavaScript Programming!</h1>" );
13
14       </script>
15    </head><body></body>
16 </html>
```

Fig. 6.1 | Displaying a line of text. (Part 1 of 2.)

可以想像JavaScript在背後寫了一份HTML文件，
再透過Browser顯示出來。

Fig. 6.1 | Displaying a line of text. (Part 2 of 2.)

6.2 Your First Script: Displaying a Line of Text with JavaScript in a Web Page (cont.)

- Spacing displayed by a browser in a web page is determined by the HTML5 elements used to format the page
- Often, JavaScripts appear in the <head> section of the HTML5 document (也可以出現在<body>中)
- The browser interprets the contents of the <head> section first (依照網頁由上而下的順序)
- 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 Your First Script: Displaying a Line of Text with JavaScript in a Web Page (cont.)

The `script` Element and Commenting Your Scripts

- The `<script>` tag indicates to the browser that the text which follows is part of a script.
- The type attribute specifies the MIME type of the script as well as the **scripting language** used in the script—in this case, a `text` file written in `javascript`.
- In HTML5, you can omit the `type` attribute from your `<script>` tags.
- You'll see it in legacy HTML documents with embedded JavaScripts.

Software Engineering Observation 6.1

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

6.2 Your First Script: Displaying a Line of Text with JavaScript in a Web Page (cont.)

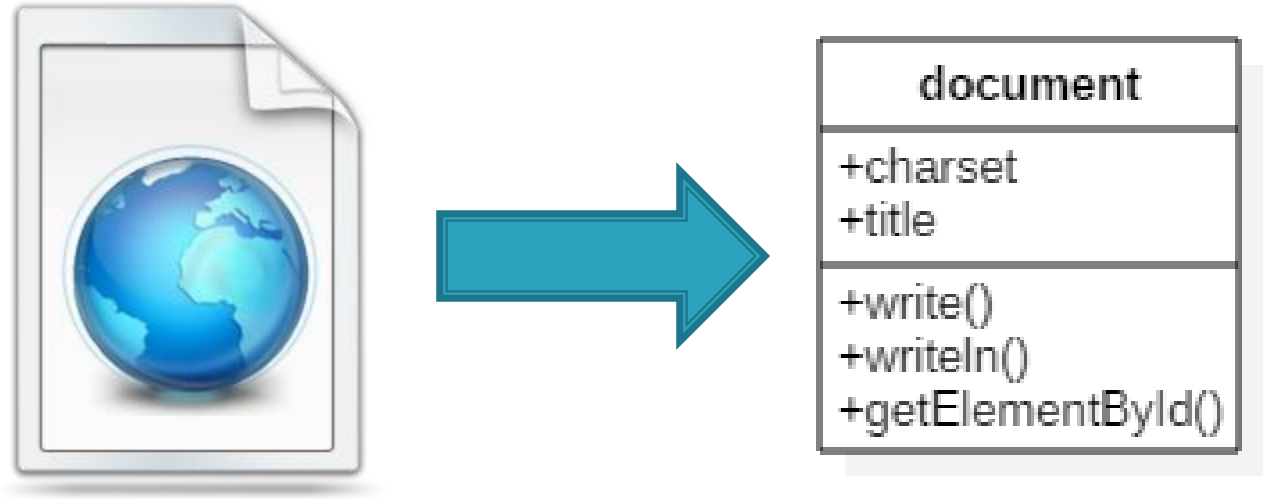
- Browser's document object represents the HTML5 document currently being displayed in the browser
  - Allows a you to specify HTML5 text to be displayed in the HTML5 document
- Browser contains a complete set of objects that allow script programmers to access and manipulate every element of an HTML5 document
- Object
  - Resides in the computer's memory and contains information used by the script
  - The term object normally implies that data (attributes) 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

Object Example (in Real World)

- Desk Lamp (檯燈)
  - **Attributes (properties)**
    - skinColor: white
    - lightColor: lightyellow
    - Status: on/off
  - **Behaviors (methods)**
    - turnOn (開)
    - turnOff (關)

Object Example (in JavaScript)

- 例如document物件就代表網頁
  - https://www.w3schools.com/jsref/dom_obj_document.asp
  - 我們可以取得網頁資訊，對網頁進行操作



6.2 Your First Script: Displaying a Line of Text with JavaScript 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
  - (HTML5 is not case sensitive)

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 HTML5 document from loading properly.

6.2 Your First Script: Displaying a Line of Text with JavaScript in a Web Page (cont.)

- The `document` object's `writeln` method
  - Writes a line of HTML5 text in the HTML5 document
  - Does not guarantee that a corresponding line of text will appear in the HTML5 document.
  - Text displayed is dependent on the contents of the string written, which is subsequently rendered by the browser.
  - Browser will interpret the HTML5 elements as it normally does to render the final text in the document

6.2 Your First Script: Displaying a Line of Text with JavaScript in a Web Page (cont.)

A Note About Embedding JavaScript Code into HTML5 Documents

- JavaScript code is typically placed in a separate file, then included in the HTML5 document that uses the script. <script src="xxxx.js"> </script>
- This makes the code more reusable, because it can be included into any HTML5 document—as is the case with the many JavaScript libraries used in professional web development today.
- We'll begin separating both CSS3 and JavaScript into separate files starting in Chapter 10.

6.3 Modifying Your First Script

- A script can display `Welcome to JavaScript Programming!` in many ways.
- Figure 6.2 displays the text in magenta, using the CSS `color` property.
- Method `write` displays a string like `writeln`, but does not position the output cursor in the HTML5 document at the beginning of the next line after writing its argument

```
1  <!DOCTYPE html>
2
3  <!-- Fig. 6.2: welcome2.html -->
4  <!-- Printing one line with multiple statements. -->
5  <html>
6     <head>
7        <meta charset = "utf-8">
8        <title>Printing a Line with Multiple Statements</title>
9        <script type = "text/javascript">
10          <!--
11          document.write( "<h1 style = 'color: magenta'>" );
12          document.write( "Welcome to JavaScript " +
13             "Programming!</h1>" );
14          // -->
15       </script>
16    </head><body></body>
17 </html>
```

Fig. 6.2 | Printing one line with separate statements. (Part 1 of 2.)

Fig. 6.2 | Printing one line with separate statements. (Part 2 of 2.)

6.3 Modifying Your First Script (Cont.)

- The + operator (called the “concatenation operator” when used in this manner) joins two strings together.
  - 與Java的規則相同
  - 與PHP規則不同

Common Programming Error 6.4

Many people confuse the writing of HTML5 text with the rendering of HTML5 text. Writing HTML5 text creates the HTML5 that will be rendered by the browser for presentation to the user.

6.3 Modifying Your First Script (Cont.)

Displaying Text in an Alert Dialog

- 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

初期可多用window.alert()除錯

也可開啟console模式，用console.log()除錯

Chrome的console模式是按下[F12]，點選[Console]

```
1  <!DOCTYPE html>
2
3  <!-- Fig. 6.3: welcome3.html -->
4  <!-- Alert dialog displaying multiple lines. -->
5  <html>
6     <head>
7        <meta charset = "utf-8">
8        <title>Printing Multiple Lines in a Dialog Box</title>
9        <script type = "text/javascript">
10          <!--
11          window.alert( "Welcome to\nJavaScript\nProgramming!" );
12          // -->
13       </script>
14    </head>
15    <body>
16       <p>Click Refresh (or Reload) to run this script again.</p>
17    </body>
18 </html>
```

Fig. 6.3 | Alert dialog displaying multiple lines. (Part 1 of 2.)

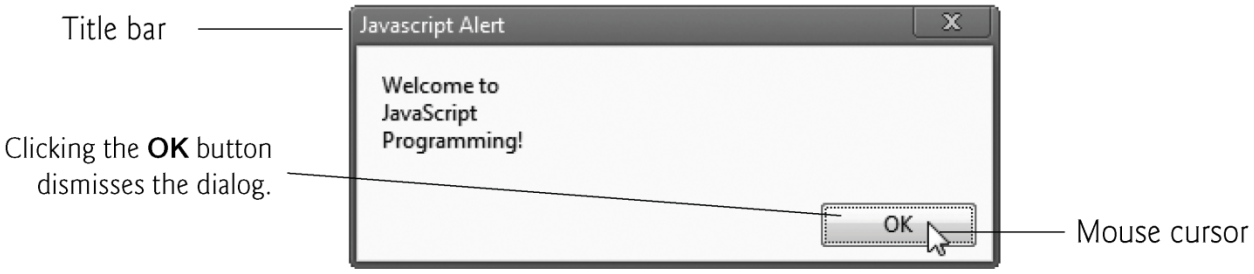


Fig. 6.3 | Alert dialog displaying multiple lines. (Part 2 of 2.)

Dialogs display plain text; they do not render HTML

Alert是純文字模式：回到你所熟悉的C的世界、要換行要加\n

6.3 Modifying Your First Script (Cont.)

Escape Sequences

- 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 HTML5 document to move to the beginning of the next line.
- 在這一章的JavaScript程式中，只有用到window.alert()時需要考慮跳脫字元。
- 其他一般在網頁上透過document.write()或DOM物件的innerHTML要輸出文字內容時，若要換行，應該插入
標籤！

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.
`\\`	*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 double quotes\"" );` displays `"in double quotes"` in an `alert` dialog.
`\'`	*Single quote*—used to represent a single-quote character in a string. For example, `window.alert( '\'in single quotes\'' );` displays `'in single quotes'` in an `alert` dialog.

Fig. 6.4 | Some common escape sequences.

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
    - 可在不透過server端處理的情況下，讓網頁產生變化。

6.4.1 Dynamic Welcome Page

- The next script creates a dynamic welcome page that obtains the user's name, then displays it on the page.
- The script uses another *predefined* dialog box from the window object—a **prompt** dialog—which allows the user to enter a value that the script can use.
- Figure 6.5 presents the script and sample output.

```
1  <!DOCTYPE html>
2
3  <!-- Fig. 6.5: welcome4.html -->
4  <!-- Prompt box used on a welcome screen -->
5  <html>
6     <head>
7        <meta charset = "utf-8">
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.writeln( "<h1>Hello " + name +
17             ", welcome to JavaScript programming!</h1>" );
18          // -->
19       </script>
20    </head><body></body>
21 </html>
```

Fig. 6.5 | Prompt box used on a welcome screen. (Part 1 of 2.)

Fig. 6.5 | Prompt box used on a welcome screen. (Part 2 of 2.)

6.4.1 Dynamic Welcome Page (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 script
  - All variables have a name, type and value, and should be declared with a var statement before they are used in a script
  - **ES6問世後，建議直接改用let關鍵字，之後的範例請各位可直接將var替換成let**
- 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.

變數名稱請又臭又長，而且能直接反應這個變數的意義。
變數建議用Camel Case (駝峰)命名

ES6 (ECMAScript 2015)

- ECMAScript 2015 was the second major revision to JavaScript.
- ECMAScript 2015 is also known as ES6 and ECMAScript 6.

https://www.w3schools.com/js/js_es6.asp

ES6: let

- The let statement allows you to declare a variable inside brackets {} scope.
  - Redeclaring a variable using var can impose problems.
  - 類似一般C與Java的變數宣告:會有block-scope

https://www.w3schools.com/js/tryit.asp?filename=tryjs_es6_let

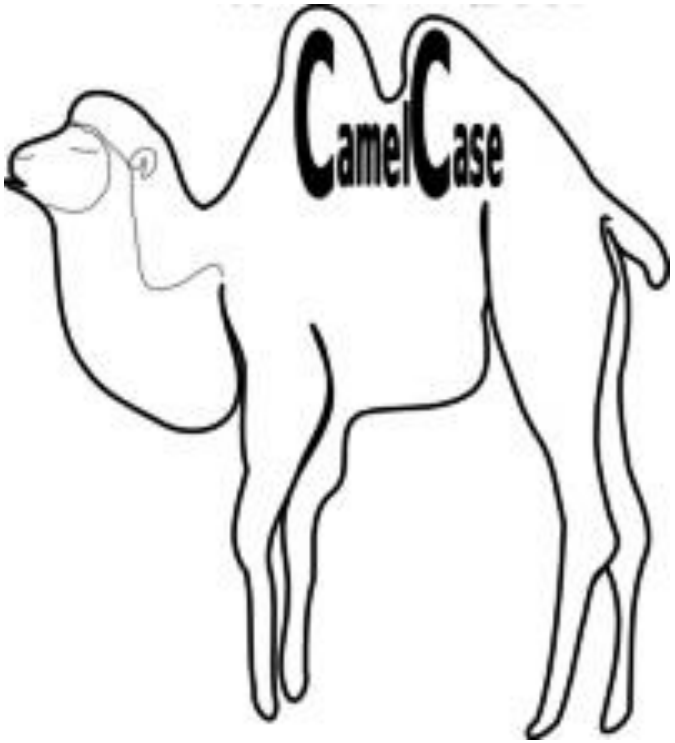
ES6: const

- Constants are block-scoped, much like variables declared using the let keyword.
- The value of a constant can't be changed through reassignment, and it can't be redeclared.

https://www.w3schools.com/js/js_const.asp

Camel Case Naming

- Examples – firstName, lastName, officehoneNumber



6.4.1 Dynamic Welcome Page (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.

6.4.1 Dynamic Welcome Page (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.

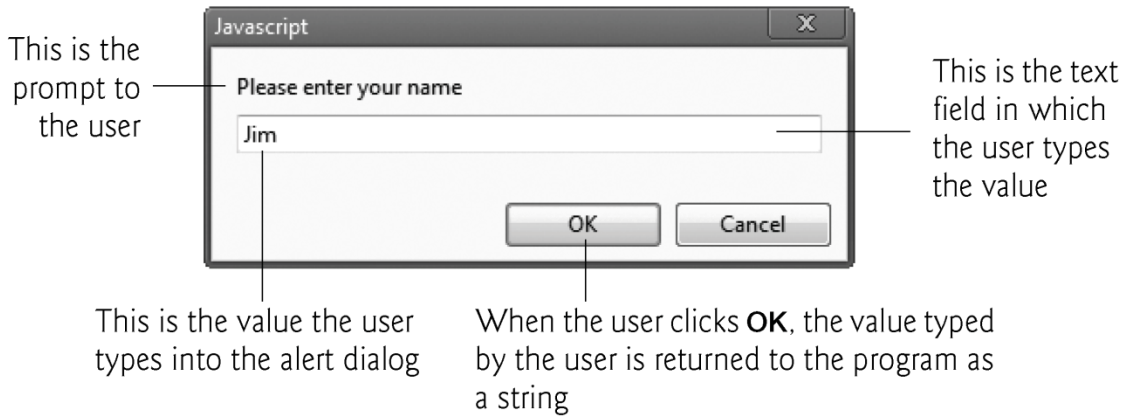


Fig. 6.6 | Prompt dialog displayed by the `window` object's `prompt` method.

6.4.1 Dynamic Welcome Page (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.

6.4.1 Dynamic Welcome Page (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

6.4.2 Adding Integers

- Our next script illustrates another use of prompt dialogs to obtain input from the user.
- Figure 6.7 inputs two *integers* (whole numbers, such as 7, –11, 0 and 31914) typed by a user at the keyboard, computes the sum of the values and displays the result.

```
1  <!DOCTYPE html>
2
3  <!-- Fig. 6.7: addition.html -->
4  <!-- Addition script. -->
5  <html>
6     <head>
7        <meta charset = "utf-8">
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 in first number from user as a string
18          firstNumber = window.prompt( "Enter first integer" );
19
20          // read in second number from user as a string
21          secondNumber = window.prompt( "Enter second integer" );
22
```

Fig. 6.7 | Addition script. (Part 1 of 3.)

```
23            // convert numbers from strings to integers
24            number1 = parseInt( firstNumber );
25            number2 = parseInt( secondNumber );
26
27            sum = number1 + number2; // add the numbers
28
29            // display the results
30            document.writeln( "<h1>The sum is " + sum + "</h1>" );
31            // -->
32         </script>
33      </head><body></body>
34   </html>
```

Fig. 6.7 | Addition script. (Part 2 of 3.)

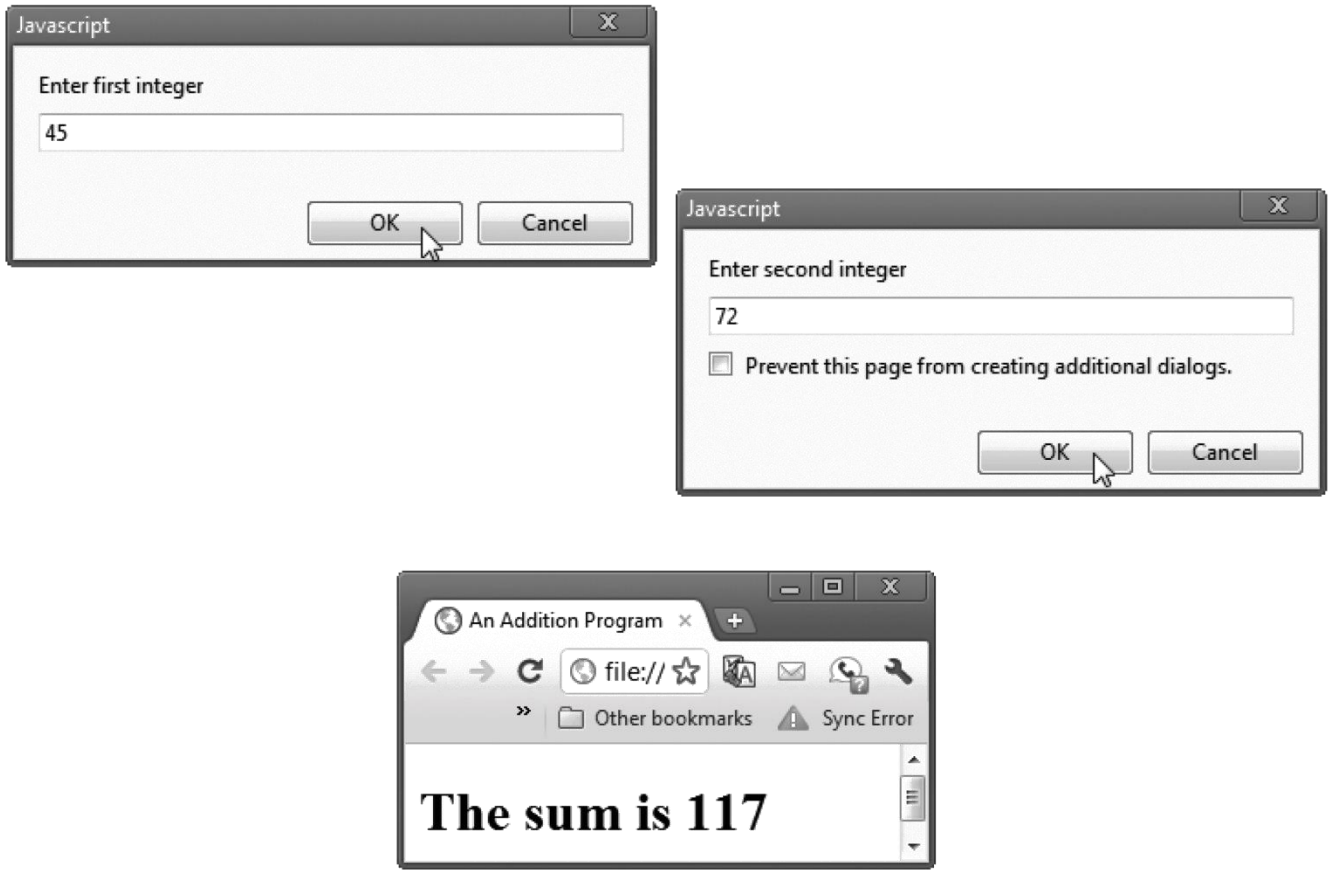


Fig. 6.7 | Addition script. (Part 3 of 3.)

6.5 Memory Concepts (Cont.)

- JavaScript does not require variables to have a type before they can be used in a script
- 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.

JavaScript Types1

- In JavaScript, a primitive (primitive value, primitive data type) is data that is not an object and has no methods.
  - There are 7 primitive data types:
    - string: a sequence of characters used to represent text
    - number: double-precision 64-bit floating point
    - boolean
    - undefined
    - null
    - bigint: can represent integers in the arbitrary precision format
    - symbol: represents a unique identifier

JavaScript Types2

- In JavaScript, objects can be seen as a collection of properties.
  - Properties can be added and removed.
  - Property values can be values of any type, including other objects.
  - Properties are identified using key values. A key value is either a String value or a Symbol value.
  - Arrays are regular objects for which there is a particular relationship between integer-key-ed properties and the 'length' property.

JavaScript Types3

- ***undefined***
  - indicates that a variable has not been assigned a value
  - https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_undefined
- ***null***
  - means "nothing“
    - but type is still an object
  - https://www.w3schools.com/js/tryit.asp?filename=tryjs_datatypes_null

JavaScript Types4

- *typeof* operator or *typeof()* function
  - returns the type of a variable or an expression
  - https://www.w3schools.com/js/tryit.asp?filename=tryjs_datatypes_typeof_string
- *isFinite()* function
  - checks whether a number is a finite, legal number.
  - https://www.w3schools.com/jsref/jsref_isfinite.asp

JavaScript Types4

- Standard built-in objects
  - *NaN* property
    - represents "Not-a-Number" value
    - https://www.w3schools.com/jsref/jsref_number_nan.asp
  - *Infinity* property
    - displays a number that exceeds the limit of a floating point number

      https://www.w3schools.com/jsref/jsref_infinity.asp

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

JavaScript operation	Arithmetic operator	Algebraic expression	JavaScript expression
Addition	`+`	$f + 7$	`f + 7`
Subtraction	`-`	$p - c$	`p - c`
Multiplication	`*`	bm	`b * m`
Division	`/`	x/y or $\frac{x}{y}$ or $x \div y$	`x / y`
Remainder	`%`	$r \bmod s$	`r % s`

Fig. 6.11 | Arithmetic operators.

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.
- 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.

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

Fig. 6.12 | Precedence of arithmetic operators.

6.7 Decision Making: Equality and Relational Operators

- `if` statement allows a script 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

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.

Common Programming Error 6.7

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."

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

Fig. 6.13 | Equality and relational operators.

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

- The script in Fig. 6.14 uses four `if` statements to display a time-sensitive greeting on a welcome page.

```
1  <!DOCTYPE html>
2
3  <!-- Fig. 6.14: welcome5.html -->
4  <!-- Using equality and relational operators. -->
5  <html>
6     <head>
7        <meta charset = "utf-8">
8        <title>Using Relational Operators</title>
9        <script type = "text/javascript">
10          <!--
11          var name; // string entered by the user
12          var now = new Date();       // current date and time
13          var hour = now.getHours(); // current hour (0-23)
14
15          // read the name from the prompt box as a string
16          name = window.prompt( "Please enter your name" );
17
18          // determine whether it's morning
19          if ( hour < 12 )
20             document.write( "<h1>Good Morning, " );
21
```

Fig. 6.14 | Using equality and relational operators. (Part 1 of 3.)

```
22          // determine whether the time is PM
23          if ( hour >= 12 )
24          {
25             // convert to a 12-hour clock
26             hour = hour - 12;
27
28             // determine whether it is before 6 PM
29             if ( hour < 6 )
30                document.write( "<h1>Good Afternoon, " );
31
32             // determine whether it is after 6 PM
33             if ( hour >= 6 )
34                document.write( "<h1>Good Evening, " );
35          } // end if
36
37          document.writeln( name +
38             ", welcome to JavaScript programming!</h1>" );
39          // -->
40       </script>
41    </head><body></body>
42 </html>
```

Fig. 6.14 | Using equality and relational operators. (Part 2 of 3.)

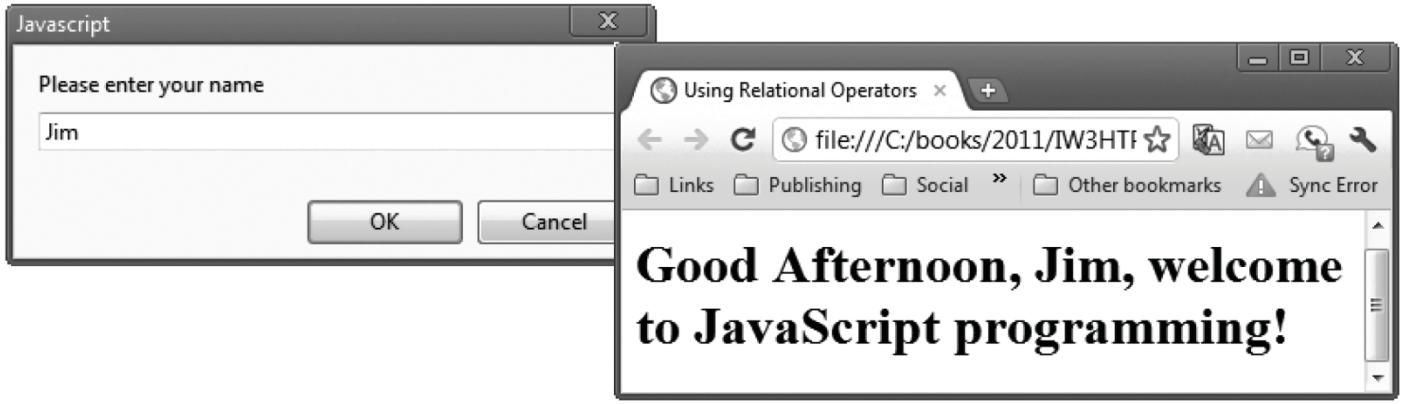


Fig. 6.14 | Using equality and relational operators. (Part 3 of 3.)

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

- 有些物件，必須把物件製造出來才能用
  - 像Date
- 有些物件，預設就能用!
  - 像document、window (大部分是BOM/DOM的一部分)

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

Fig. 6.15 | Precedence and associativity of the operators discussed so far.

Comparison Operators

- Given that x = 5

Operator	Description	Comparing	Returns
==	equal to	x == 8	FALSE
		x == 5	TRUE
		x == "5"	TRUE
===	**equal value and equal type**	x === 5	TRUE
		x === "5"	FALSE
!=	not equal	x != 8	TRUE
!==	**not equal value or not equal type**	x !== 5	FALSE
		x !== "5"	TRUE
		x !== 8	TRUE
>	greater than	x > 8	FALSE
<	less than	x < 8	TRUE
>=	greater than or equal to	x >= 8	FALSE
<=	less than or equal to	x <= 8	TRUE

https://www.w3schools.com/js/js_comparisons.asp