CHAPTER

5

JavaScript 內建物件

5-1 JavaScript 的內建物件

JavaScript 擁有內建物件和自訂物件，事實上，各種資料型態的變數都是物件，變數在宣告和指定值後馬上就擁有對應的方法和屬性。

5-1-1 JavaScript 內建物件的種類

JavaScript 物件依照建立方式的不同可以分為使用變數宣告的隱性物件，和使用 new 運算子建立物件的顯性物件。

隱性物件（Implicit objects）

JavaScript 的各種資料型態變數，在宣告和指定值後就是一個物件，例如：數值、字串和布林資料型態的變數等，如下所示：

```
var str="JavaScript 網頁程式設計";
```

上述程式碼宣告變數 str 是一個隱性 String 物件，雖然可以使用 String 物件的方法，不過隱性物件不支援 prototype 屬性，如下所示：

```
str.prototype.count;
```

上述程式碼會導致 JavaScript 程式執行錯誤，而且，隱性物件也無法隨意擴充物件的屬性。

顯性物件（Explicit objects）

JavaScript 物件如果是使用 new 運算子建立物件，這個物件就是一個顯性物件，如下所示：

```
var str= new String("JavaScript 網頁程式設計");
```

上述程式碼建立的也是一個字串變數，不過，這是一個 String 物件，顯性物件支援新增屬性和 Prototype 屬性。

5-1-2 JavaScript 的內建物件

JavaScript 提供十一種內建物件，在本章準備介紹常用的 String、Array、Date、Math 和 Error 物件，其他物件的簡單說明，如下所示：

Boolean 物件

Boolean 物件是一種資料型態，提供建構函數可以建立布林資料型態的物件，如下所示：

```
objBoolean = new Boolean();
```

上述程式碼使用 new 運算子建立布林物件，括號參數如為 false、0、null、NaN 或空字串的布林值為 false；否則為 true。

當使用 var 宣告布林變數且指定值後，布林變數將自動轉換成 Boolean 物件。

Function 物件

JavaScript 函數就是一個 Function 物件，在第 4 章已經說明函數建立方法，如下所示：

```
function mod(x, y) {
   return(x % y);
}
```

上述程式區塊是一個餘數函數，我們也可以使用 new 運算子建立函數的 Function 物件，如下所示：

```
var mod = new Function("x", "y", "return(x%y)");
```

上述程式碼建立 mod()函數，不論使用哪一種方法建立函數，我們都可以使用相同程式碼來呼叫它，如下所示：

```
value = mod(4, 5);
```

Function 物件是函數，如果函數擁有參數，這些傳入參數是 arguments 物件，在第 4 章已經說明過如何使用 arguments 物件來取得函數傳入的參數值。

Global 物件

Global 物件不能使用 new 運算子建立，在腳本語言引擎初始後就會自動建立此物件。在 Global 物件擁有 2 個屬性，如下表所示：

屬性	說明
Infinity	取得 Number.POSITIVE_INFINITY 的初始值
NaN	取得 Number.NaN 的初始值

Global 物件的屬性不用指名 Global 物件，直接使用屬性名稱即可，如下所示：

```
Infinity
NaN
```

Number 物件

Number 物件類似 Boolean 物件可以建立數值資料型態的變數，如下所示：

```
objnum = new Number(value);
```

上述程式碼使用 new 運算子建立 Number 物件，參數 value 為數值變數的值，通常我們使用 Number 物件的目的是為了使用 toString()方法將數值轉換成字串（詳細說明請參閱第 5-7 節）。

Number 物件屬性的語法為：Number.propertyname;，其常用屬性的說明，如下表所示：

屬性	說明
MAX_VALUE	傳回 JavaScript 數值的最大值，約 1.79E+308
MIN_VALUE	傳回 JavaScript 最接近 0 的值，約 5.00E-324
NaN	一種特殊值，表示運算式或變數值不是數值
NEGATIVE_INFINITY	傳回比 -Number.MAX_VALUE 更大的負值
POSITIVE_INFINITY	傳回比 Number.MAX_VALUE 更大的正值

Object 物件

Object 物件可以建立 JavaScript 支援的物件，如下所示：

```
Objobject = new Object(value);
```

上述程式碼使用 new 運算子建立 Object 物件，參數 value 如果是 String，它是一個字串物件，Boolean 是 Boolean 物件等，在第 4 章筆者已經使用 Object 物件來建立自訂物件。

RegExp 物件

RegExp 物件是 JavaScript「正規表達式」（Regular Expression）物件。

5-2 JavaScript 的 String 物件

字串屬於 JavaScript 基本資料型態，字串變數本身就是一種 String 物件。

5-2-1 建立 String 物件

String 物件的方法可以格式化字串或進行子字串操作，簡單的說，就是處理字串變數的資料，我們可以直接宣告字串變數或使用 new 運算子建立 String 物件，如下所示：

```
var objstr1="JavaScript";
var objstr2= new String("網頁程式設計");
```

上述程式碼都可以建立 String 物件，不論使用哪一種方法建立 String 物件，都可以使用本節方法來處理字串內容。

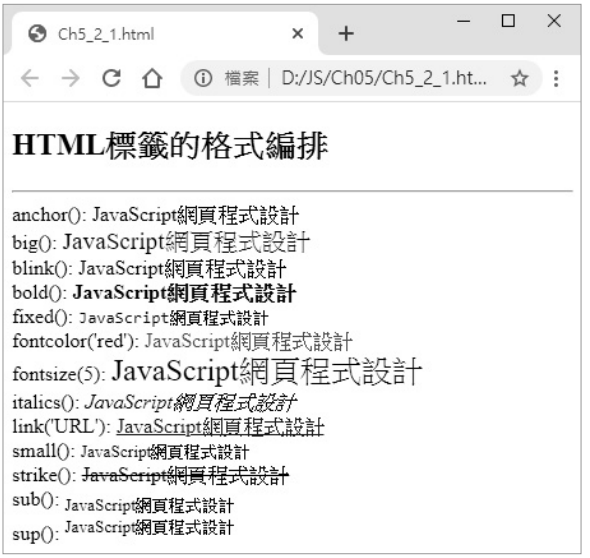
HTML 標籤的格式編排

String 物件提供一系列格式編排方法，可以將 String 物件的字串內容輸出成對應的 HTML 標籤，相關方法的說明（下表 string 代表 String 物件的字串內容），如下表所示：

方法	說明
anchor()	傳回<a>string標籤字串
big()	傳回<big>string</big>標籤字串
blink()	傳回<blink>string</blink>標籤字串
bold()	傳回string標籤字串
fixed()	傳回<tt>string</tt>標籤字串
fontcolor(color)	傳回string標籤字串
fontsize(size)	傳回string標籤字串
italics()	傳回string標籤字串
link(*url*)	傳回string標籤字串
small()	傳回string標籤字串
strike()	傳回<strike>string</strike>標籤字串
sub()	傳回_{string}標籤字串
sup()	傳回^{string}標籤字串

JavaScript 程式：Ch5_2_1.html

在 JavaScript 程式使用 String 物件的方法，可以將字串使用 HTML 標籤的格式編排來輸出內容，如右圖所示：



右述圖例顯示 String 物件格式編排方法的顯示結果，這些輸出結果都可以對應 HTML 標籤的編排效果。

程式內容

```
01: <!DOCTYPE html>
02: <html>
03: <head>
04: <meta charset="utf-8"/>
05: <title>Ch5_2_1.html</title>
06: </head>
07: <body>
08: <h2>HTML 標籤的格式編排</h2>
09: <hr/>
10: <script>
11: // 測試字串
12: var str="JavaScript 網頁程式設計";
13: document.write("anchor(): " + str.anchor() + "<br/>");
14: document.write("big(): " + str.big() + "<br/>");
15: document.write("blink(): " + str.blink() + "<br/>");
16: document.write("bold(): " + str.bold() + "<br/>");
17: document.write("fixed(): " + str.fixed() + "<br/>");
18: document.write("fontcolor('red'): " + str.fontcolor("red") + "<br/>");
19: document.write("fontsize(5): " + str.fontsize(5) + "<br/>");
20: document.write("italics(): " + str.italics() + "<br/>");
21: document.write("link('URL'): " + str.link("http://www.hinet.net") + "<br/>");
22: document.write("small(): " + str.small() + "<br/>");
23: document.write("strike(): " + str.strike() + "<br/>");
24: document.write("sub(): " + str.sub() + "<br/>");
```

```
25: document.write("sup(): " + str.sup() + "<br/>");
26: </script>
27: </body>
28: </html>
```

程式說明

第 12 列：宣告測試的字串變數且指定初值。

第 13~24 列：使用 String 物件方法顯示格式編排的字串內容。

5-2-2 字串長度與大小寫

String 物件提供方法和屬性可以取得字串長度和英文字串的大小寫轉換，相關屬性的說明，如下表所示：

屬性	說明
length	取得字串的長度

相關 String 物件的方法，如下表所示：

方法	說明
toLowerCase()	將字串的英文字母都轉換成小寫字母
toUpperCase()	將字串的英文字母都轉換成大寫字母

JavaScript 程式：Ch5_2_2.html

在 JavaScript 程式使用 String 物件的方法來取得字串長度和進行英文字母的大小寫轉換，如右圖所示：

在上述圖例最上方的 2 個字串為測試的中英文字串，接著是中英文字串的長度，最後是英文字串的大小寫轉換。

程式內容

```
01: <!DOCTYPE html>
02: <html>
03: <head>
04: <meta charset="utf-8"/>
05: <title>Ch5_2_2.html</title>
06: </head>
07: <body>
08: <h2>字串長度與大小寫</h2>
09: <hr/>
10: <script>
11: // 測試字串
12: var str1="JavaScript";
13: var str2= new String("網頁程式設計");
14: document.write("測試的英文字串: \"" + str1 + "\"<br/>");
15: document.write("測試的中文字串: \"" + str2 + "\"<br/>");
16: document.write("英文字串長度: " + str1.length + "<br/>");
17: document.write("中文字串長度: " + str2.length + "<br/>");
18: document.write("全部小寫: " + str1.toLowerCase() + "<br/>");
19: document.write("全部大寫: " + str1.toUpperCase() + "<br/>");
20: </script>
21: </body>
22: </html>
```

程式說明

第 12 列：宣告測試的英文字串變數且指定初值。

第 13 列：使用 new 運算子建立測試的中文字串物件。

第 16~17 列：分別使用 String 物件的屬性 length 取得中英文的字串長度。

第 18~19 列：進行英文大小寫字母的轉換。

5-2-3 取得字串的指定字元

在字串處理時如果需要取得字串指定位置的字元，String 物件提供 2 種方法來取得指定位置的字元，相關方法的說明，如下表所示：

方法	說明
charAt(index)	取得參數 index 位置的字元，索引值是從 0 開始
charCodeAt(index)	取得參數 index 位置的 Unicode 統一字碼

上述 2 種方法傳入參數 index 都是以 0 開始，第 1 個字元為 0，第 2 個字元為 1，依序類推。

JavaScript 程式：Ch5_2_3.html

在 JavaScript 程式使用 String 物件的方法取得指定位置的字元和 Unicode 內碼，如右圖所示：

上述圖例上方為中英文原始的測試字串，接著取得位置索引值為 4 的中英文字元，即第 5 個字元，最後取得英文字元 S 的 Unicode 字碼為 83。

程式內容

```
01: <!DOCTYPE html>
02: <html>
03: <head>
04: <meta charset="utf-8"/>
05: <title>Ch5_2_3.html</title>
06: </head>
07: <body>
08: <h2>取得字串的字元</h2>
09: <hr/>
10: <script>
11: // 測試字串
12: var str1="JavaScript";
13: var str2= new String("網頁程式設計");
14: document.write("測試的英文字串: \"" + str1 + "\"<br/>");
15: document.write("測試的中文字串: \"" + str2 + "\"<br/>");
```

```
16: document.write("英文字元 charAt(4): " + str1.charAt(4) + "<br/>");
17: document.write("中文字串 charAt(4): " + str2.charAt(4) + "<br/>");
18: document.write("英文字元 charCodeAt(4): " + str1.charCodeAt(4) + "<br/>");
19: </script>
20: </body>
21: </html>
```

程式說明

第 12 列：宣告測試的英文字串變數且指定初值。

第 13 列：使用 new 運算子建立測試的中文字串物件。

第 16~17 列：分別使用 String 物件的 charAt()方法取得中英文字串中索引值位置為 4 的字元。

第 18 列：取得英文字母的 Unicode 字碼。

5-2-4 子字串的搜尋

String 物件提供功能強大的子字串搜尋方法，可以輕鬆在字串中搜尋所需的子字串，相關方法的說明，如下表所示：

方法	說明
indexOf(string, index)	傳回第 1 次搜尋到字串的索引位置，沒有找到傳回-1，傳入參數是搜尋字串，index 為開始搜尋的索引位置
lastIndexOf(string)	如同 indexOf()方法，不過是從尾搜尋到頭的反向搜尋
match(string)	如同 indexOf()和 lastIndexOf()，不過傳回的為找到的字串，沒有找到傳回 null
search(string)	與 indexOf()的功能相似

上表 match()和 search()方法可以使用正規運算式，傳入參數是正規運算式的範本字串。

JavaScript 程式：Ch5_2_4.html

在 JavaScript 程式使用 String 物件的相關方法來搜尋指定的子字串，如右圖所示：

上述圖例由上而下分別顯示原始測試字串和測試各種方法的字串搜尋結果。

程式內容

```
01: <!DOCTYPE html>
02: <html>
03: <head>
04: <meta charset="utf-8"/>
05: <title>Ch5_2_4.html</title>
06: </head>
07: <body>
08: <h2>字串搜尋</h2>
09: <hr/>
10: <script>
11: // 測試字串
12: var str1="JavaScript";
13: var str2= new String("網頁程式設計");
14: document.write("測試的英文字串: \"" + str1 + "\"<br/>");
15: document.write("測試的中文字串: \"" + str2 + "\"<br/>");
16: document.write("英文字元 indexOf('a'): " + str1.indexOf('a') + "<br/>");
17: document.write("英文字元 indexOf('a', 2): " + str1.indexOf('a', 3) + "<br/>");
18: document.write("中文字串 indexOf('程式'): " + str2.indexOf('程式') + "<br/>");
19: document.write("英文字元 lastIndexOf('a'): " + str1.lastIndexOf('a') + "<br/>");
20: document.write("英文字元 match('Script'): " + str1.match('Script') + "<br/>");
```

```
21: document.write("中文字串 match('程式'): " + str2.match('程式') + "<br/>");
22: document.write("英文字元 search('Script'): " + str1.search('Script') + "<br/>");
23: document.write("中文字串 search('學習'): " + str2.search('學習') + "<br/>");
24: </script>
25: </body>
26: </html>
```

程式說明

第 12 列：宣告測試的英文字串變數且指定初值。

第 13 列：使用 new 運算子建立測試的中文字串物件。

第 16~23 列：分別使用 String 物件方法搜尋指定子字串是否存在。

5-2-5 子字串的處理

String 物件提供方法可以取代、分割和取出字串中所需的子字串，相關方法的說明（string1 和 string2 為子字串），如下表所示：

方法	說明
replace(string1, string2)	將找到的 string1 子字串取代成 string2
split(string)	傳回 Array 物件，使用參數 string 作為分割字串，可以將字串轉換成 Array 物件
substr(index, length)	從 index 開始取出 length 個字元
substring(index1, index2)	取出 index1 到 index2 之間的子字串
concat(string)	將 string 字串新增到 String 物件的字串後

上述 concat()方法的呼叫需要使用指定敘述，如下所示：

```
str3 = str1.concat(str2);
```

上述程式碼相當於 str3 = str1 + str2。

JavaScript 程式：Ch5_2_5.html

在 JavaScript 程式使用 String 物件的方法取出字串中的子字串，並且使用 concat()方法連接 2 個字串，如右圖：

右述圖例的前二列是中英文測試的原始字串，下方是各種子字串處理的執行結果，最後將中英文字串結合成一個字串。

程式內容

```
01: <!DOCTYPE html>
02: <html>
03: <head>
04: <meta charset="utf-8"/>
05: <title>Ch5_2_5.html</title>
06: </head>
07: <body>
08: <h2>子字串處理</h2>
09: <hr/>
10: <script>
11: // 測試字串
12: var str1="JavaScript";
13: var str2= new String("網頁程式設計");
14: document.write("測試的英文字串: \"" + str1 + "\"<br/>");
15: document.write("測試的中文字串: \"" + str2 + "\"<br/>");
16: document.write("英文 replace('Java','VB'): "+str1.replace('Java','VB')+"<br/>");
17: document.write("中文 split('程式'): " + str2.split('程式') + "<br/>");
18: document.write("英文 substr(2,4): " + str1.substr(2,4) + "<br/>");
19: document.write("中文 substring(2,5): " + str2.substring(2,5) + "<br/>");
20: // 連接 2 個字串
21: str3 = str1.concat(str2);
22: document.write("連接字串 str1.concat(str2): " + str3 + "<br/>");
23: </script>
24: </body>
25: </html>
```

程式說明

第 12 列：宣告測試的英文字串變數且指定初值。

第 13 列：使用 new 運算子建立測試的中文字串物件。

第 16~19 列： 分別使用 String 物件的 replace()、split()、substr()和 substring()方法取代和取出子字串。

第 21~22 列： 使用 concat()方法連接 2 個子字串，在第 22 列顯示連接後的字串。

5-3 JavaScript 的 Array 物件

JavaScript 資料型態並沒有陣列，而是使用 Array 物件建立陣列，每一個陣列元素事實上就是 Array 物件的屬性。

5-3-1 JavaScript 的一維陣列

基本上，JavaScript 陣列和物件的分野並不明顯，陣列擁有陣列元素如同物件擁有屬性，JavaScript 陣列事實上就是一種特殊物件。

建立一維陣列

如同 C/C++、C#、Java 或 Visual Basic 語言的陣列元素是使用數值的索引值來存取元素，JavaScript 陣列的索引值是從 0 開始，JavaScript 宣告陣列的方法就是建立 Array 物件，如下所示：

```
var username = new Array(5);
```

上述程式碼使用 new 運算子建立 Array 物件，參數 5 表示陣列有 5 個元素，索引值是從 0 開始，因為只有一個索引，所以是建立一維陣列。然後我們可以使用索引值來指定陣列的元素值，如下所示：

```
username[0] = "Joe";
username[1] = "Jane";
…
username[4] = "Merry";
```

上述程式碼指定陣列元素的內容，我們也可以在建立 Array 物件時，直接在參數指定陣列元素值，如下所示：

```
var tips = new Array(100,200,500);
```

上述兩個方法都可以建立 JavaScript 陣列，其中 username[] 陣列是一個字串陣列，tips[] 陣列是數值陣列。

走訪一維陣列

我們可以使用 for 迴圈走訪和顯示陣列元素，如下所示：

```
for (var i = 0; i < 5; i++) {
   document.write(username[i] + "<br/>");
}
```

上述程式碼使用陣列索引值取得每一個陣列元素的值，迴圈的結束條件是使用 length 屬性取得陣列尺寸。

JavaScript 程式：Ch5_3_1.html

在 JavaScript 程式使用 Array 物件建立一維的數值和字串陣列，然後使用 for 迴圈顯示陣列的所有元素，如右圖所示：

上述圖例分別顯示 2 個陣列的所有元素，上方數值為 tips[] 陣列；下方字串屬於 username[] 陣列。

程式內容

```
01: <!DOCTYPE html>
02: <html>
03: <head>
04: <meta charset="utf-8"/>
05: <title>Ch5_3_1.html</title>
06: </head>
07: <body>
08: <h2>JavaScript 的一維陣列</h2>
09: <hr/>
10: <script>
11: var tips = new Array(100,200,500);
12: var username = new Array(5);
13: username[0] = "Joe";
14: username[1] = "Jane";
15: username[2] = "Tony";
16: username[3] = "Tom";
17: username[4] = "Merry";
18: // 使用迴圈顯示陣列值
19: for(var i = 0; i < tips.length; i++){
20:    document.write(tips[i] + "<br/>");
21: }
22: // 使用迴圈顯示陣列值
23: for(var i = 0; i < 5; i++){
24:    document.write(username[i] + "<br/>");
25: }
26: </script>
27: </body>
28: </html>
```

程式說明

第 11~17 列： 分別使用 new 運算子建立數值內容的陣列 tips[]，和擁有 5 個元素的陣列物件 username[]，第 13~17 列指定陣列元素值。

第 19~21 列： 使用 for 迴圈顯示 tips[] 陣列元素，結束條件是使用 length 屬性取得陣列尺寸。

第 23~25 列： 使用 for 迴圈顯示 username[] 陣列的元素。

5-3-2 Array 物件的屬性和方法

Array 物件提供屬性和方法可以取得陣列尺寸、排序陣列元素、合併陣列和反轉陣列元素。Array 物件的屬性說明，如下表所示：

屬性	說明
length	取得陣列的元素個數，即陣列尺寸

Array 物件的相關方法說明，如下表所示：

方法	說明
join()	將陣列的元素使用字串方式顯示，每個陣列元素是使用「,」符號分隔
reverse()	將陣列元素反轉，本來是陣列的最後一個元素成為第一個元素
sort()	將陣列所有元素進行排序
concat(array)	將參數的陣列合併到目前的陣列中

JavaScript 程式：Ch5_3_2.html

在 JavaScript 程式建立 2 個測試陣列，然後分別測試 Array 物件的屬性和方法，如右圖所示：

上述圖例依序顯示陣列元素的個數，各種 Array 物件方法的執行結果，最後一列將兩個陣列結合成一個陣列。

程式內容

```
01: <!DOCTYPE html>
02: <html>
03: <head>
```

```
04: <meta charset="utf-8"/>
05: <title>Ch5_3_2.html</title>
06: <script>
07: function showArray(username) {
08:    // 使用迴圈顯示陣列值
09:    for(var i = 0; i < username.length; i++){
10:       document.write(username[i] + ",");
11:    }
12:    document.write("<br/>");
13: }
14: </script>
15: </head>
16: <body>
17: <h2>Array 物件的方法</h2>
18: <hr/>
19: <script>
20: var username = new Array("Joe","Jane","Tony","Tom");
21: var username1 = new Array("陳會安","江小魚","陳允傑");
22: document.write("陣列元素共: " + username.length + "個<br/>");
23: // 顯示陣列元素
24: document.write(username.join() + "<br/>");
25: username.reverse();  // 反轉陣列
26: showArray(username); // 顯示陣列元素
27: username.sort(); // 排序
28: showArray(username); // 顯示陣列元素
29: username = username.concat(username1); // 結合兩陣列
30: showArray(username); // 顯示陣列元素
31: </script>
32: </body>
33: </html>
```

程式說明

第 7~13 列： showArray()函數可以顯示傳入陣列的所有元素。

第 20~21 列： 使用 new 運算子建立陣列 username[]，第 21 列建立另一個陣列物件 username1[]。

第 22 列：使用 length 屬性取得陣列 username[] 的元素個數。

第 24 列：使用 join()方法顯示陣列元素。

第 25 列：使用 reverse()方法反轉陣列。

第 27~28 列：測試 sort()排序方法。

第 29~30 列：合併陣列 username[] 和 username1[]。

5-3-3 JavaScript 的多維陣列

JavaScript 的 Array 物件並不能直接建立二維或多維陣列，不過，因為 Array 物件的元素可以是另一個 Array 物件，我們仍然可以在 JavaScript 程式碼建立多維陣列，如下所示：

```
var users = new Array(5);
for(var i = 0; i < 5; i++)
   users[i] = new Array(2);
```

上述程式碼先建立擁有 5 個元素的 Array 物件 users[]，接著使用 for 迴圈將每個陣列元素分別建立成擁有 2 個元素的 Array 物件，即 5 X 2 的二維陣列，然後可以指定二維陣列的元素值，如下所示：

```
users[0][0] = "Joe";
users[0][1] = "1234";
users[1][0] = "Jane";
users[1][1] = "5678";
…
users[4][0] = "Merry";
users[4][1] = "5678";
```

上述程式碼指定二維陣列的元素值，同樣方式，我們可以將 Array 物件擴充成多維陣列。

JavaScript 程式：Ch5_3_3.html

在 JavaScript 程式使用 Array 物件建立 5 X 2 的二維陣列，當指定二維陣列值後，使用二層巢狀迴圈顯示二維陣列的值，如右圖：

上述圖例顯示二維陣列的值，共五列，每列擁有二個元素。

程式內容

```
01: <!DOCTYPE html>
02: <html>
03: <head>
04: <meta charset="utf-8"/>
05: <title>Ch5_3_3.html</title>
06: </head>
07: <body>
08: <h2>JavaScript 的二維陣列</h2>
09: <hr/>
10: <script>
11: // 建立二維陣列
12: var users = new Array(5);
13: for(var i = 0; i < 5; i++)
14:    users[i] = new Array(2);
15: users[0][0] = "Joe";
16: users[0][1] = "1234";
17: users[1][0] = "Jane";
18: users[1][1] = "5678";
19: users[2][0] = "Tony";
20: users[2][1] = "9012";
21: users[3][0] = "Tom";
22: users[3][1] = "1234";
23: users[4][0] = "Merry";
24: users[4][1] = "5678";
25: // 使用迴圈顯示陣列值
26: for(var j = 0; j < users.length; j++){
27:    for(i = 0; i < users[i].length; i++)
28:       document.write(users[j][i] + ",");
29:    document.write("<br/>");
30: }
31: </script>
32: </body>
33: </html>
```

程式說明

第 12~24 列： 使用 Array 物件建立 5 X 2 的二維陣列，第 15~24 列指定二維陣列的元素值。

第 26~30 列： 使用二層 for 迴圈顯示二維陣列的所有元素。

5-4 JavaScript 的 Date 物件

Date 物件可以取得電腦的系統時間和日期，並且提供相關方法可以將它轉換成所需的日期或時間資料。

5-4-1 取得日期和時間

Date 物件在使用 new 運算子建立物件後，就可以取得系統的時間和日期，如下所示：

```
var dttoday = new Date();
```

上述程式碼建立 Date 物件後，可以使用下表方法取得時間和日期資料，其說明如下表所示：

方法	說明
getDate()	傳回日期值 1~31
getDay()	傳回星期值 0~6，也就是星期日到星期六
getMonth()	傳回月份值 0~11，也就是一到十二月
getFullYear()	傳回完整年份，例如：2014
getYear()	傳回年份，如果在 1900~1999 年之間，傳回後兩碼，例如：1999 年傳回 99，否則傳回完整年份
getHours()	傳回小時 0~23
getMinutes()	傳回分鐘 0~59
getSeconds()	傳回秒數 0~59
getMilliseconds()	傳回千分之一秒為單位的秒數，0~999
getTime()	傳回自 1/1/1970 年開始的秒數，以千分之一秒（毫秒）為單位

JavaScript 程式：Ch5_4_1.html

在 JavaScript 程式使用 Date 物件方法取得系統時間、日期和星期，如右圖所示：

上述圖例顯示使用 Date 物件方法取得系統日期、時間和今天是星期幾。

程式內容

```
01: <!DOCTYPE html>
02: <html>
03: <head>
04: <meta charset="utf-8"/>
05: <title>Ch5_4_1.html</title>
06: </head>
07: <body>
08: <h2>取得日期時間</h2>
09: <hr/>
10: <script>
11: var weekday=new Array("星期日","星期一","星期二","星期三","星期四","星期五","星期六");
12: var dttoday = new Date();
13: // 取得系統日期
14: var output = dttoday.getDate() + "/";
15: output += (dttoday.getMonth() + 1) + "/";
16: output += dttoday.getFullYear() + "<br/>";
17: document.write("系統日期: " + output);
18: // 取得系統時間
19: output = dttoday.getHours() + ":";
20: output += dttoday.getMinutes() + ":";
21: output += dttoday.getSeconds() + "<br/>";
22: document.write("系統時間: " + output);
23: document.write(weekday[dttoday.getDay()]);
24: </script>
25: </body>
26: </html>
```

程式說明

第 11 列：建立星期字串的一維陣列。

第 12 列：使用 new 運算子建立 Date 物件。

第 14~16 列：取得系統日期，不過取得的月份需要加一才是我們慣用的月份。

第 19~23 列：取得系統時間，在第 23 列取得星期幾。

5-4-2 設定日期和時間

Date 物件提供數種方法來存取日期與時間，請注意！這些設定方法並不是修改電腦的系統時間和日期，只是設定 Date 物件儲存的時間和日期，Date 物件設定日期和時間的相關方法說明，如下表所示：

方法	說明
setDate()	設定 Date 物件的日期 1~31
setMonth()	設定 Date 物件的月份 0~11
setFullYear()	設定 Date 物件的完整年份
setYear()	設定 Date 物件的年份，在 1990~1999 間只需使用後兩位，例如：1999 使用 99，否則需要使用完整年份
setHours()	設定 Date 物件的小時 0~23
setMinutes()	設定 Date 物件的分鐘 0~59
setSeconds()	設定 Date 物件的秒數 0~59
setMilliseconds()	設定 Date 物件的秒數，以千分之一秒為單位，0~999
setTime()	設定 Date 物件的時間，自 1/1/1970 年開始，以千分之一秒為單位

Date 物件提供一組對應方法，可以設定和取得 UTC 日期和時間，例如：setDate() 方法對應 setUTCDate()；getHours()方法對應 getUTCHours()等。

Memo

「UTC」（Universal Coordinated Standard）稱為國際標準時間也就是「GMT」（Greenwich Mean Time）格林威治標準時間，如下所示：

```
Wed Nov 11 04:30:14 UTC+0800 2020
```

上述字串是本地的日期時間，UTC+0800 表示本地時間為 UTC 時間再加 8 小時，我們可以使用 toGMTString()方法將本地時間轉換成 GMT 時間，也就是 UTC 時間。

JavaScript 程式：Ch5_4_2.html

在 JavaScript 程式使用 Date 物件方法設定物件的日期和時間，如右圖所示：

上述圖例顯示使用 Date 物件方法設定的日期和時間，這是 Date 物件記錄的時間和日期，並不是電腦的系統日期和時間。

程式內容

```
01: <!DOCTYPE html>
02: <html>
03: <head>
04: <meta charset="utf-8"/>
05: <title>Ch5_4_2.html</title>
06: </head>
07: <body>
08: <h2>設定日期時間</h2>
09: <hr/>
10: <script>
11: var newdate = new Date();
12: // 設定日期
```

```
13: newdate.setDate("11");
14: newdate.setMonth("10");    // 11 月
15: newdate.setFullYear("2020");
16: newdate.setHours("4");
17: newdate.setMinutes("30");
18: document.write(newdate);
19: </script>
20: </body>
21: </html>
```

程式說明

第 11 列：建立 Date 物件。

第 13~17 列：設定 Date 物件的日期和時間。

5-4-3 日期和時間的轉換

Date 物件提供日期和時間轉換方法，可以取得時間差、轉換成千分之一秒數或輸出成字串等轉換操作，GMT 為格林威治標準時間，相關方法的說明，如下表所示：

方法	說明
getTimezoneOffset()	傳回本地時間和 GMT 的時間差，以分為單位
toGMTString()	傳回轉換成 GMT 時間的字串
toLocalString()	傳回將 GMT 轉換成本地時間的字串
parse(Date)	傳回參數 Date 物件從 1/1/1970 到本地時間的毫秒數，以千分之一秒為單位
UTC(Y,M,D,…)	傳回參數年-月-日-時-分-秒從 1/1/1970 到 GMT 時間的毫秒數，以千分之一秒為單位

上表 parse()和 UTC()方法和其他時間轉換方法在使用上有些不同，這 2 個方法不用建立物件，因為它是 Date()建構函數的方法（相當於其他語言的靜態/類別方法），如下所示：

```
document.write(Date.parse(dttoday));
document.write(Date.UTC(2020, 04, 30, 12, 1, 0));
```

上述程式碼直接使用 Date.parse()和 Date.UTC()執行方法，參數 dttoday 是 Date 物件變數 dttoday。

5-4-4 取得系統的時間

JavaScript 的 Date 物件可以取得系統時間，換句話說，只需定時執行 JavaScript 函數，就可以使用 Date 物件建立網頁小時鐘。

JavaScript 小時鐘需要使用 Window 物件的計時器方法 setTimeout()，方法參數可以設定在間隔時間後執行指定函數或網頁，對應的 clearTimeout()方法可以清除計時器。

JavaScript 程式：Ch5_4_4.html

在 JavaScript 程式使用 Date 物件和 Window 物件的計時器方法建立網頁小時鐘，使用 GIF 圖檔 0~9.gif 顯示系統時間，如右圖所示：

上述圖例是網頁小時鐘，顯示的並不是靜態時間，而是真的會走的小時鐘。

程式內容

```
01: <!DOCTYPE html>
02: <html>
03: <head>
04: <meta charset="utf-8"/>
05: <title>Ch5_4_4.html</title>
06: <script>
07: var timer = null;
08: // 顯示數字圖片
09: function displayClock(num) {
10:   var dig = parseInt(num/10);
11:   var timetag="<img src='images\\" + dig + ".gif'>";
12:   dig = num%10;
13:   timetag +="<img src='images\\" + dig + ".gif'>";
14:   return timetag;
```

```
15: }
16: // 停止計時
17: function stopClock() {
18:    clearTimeout(timer);
19: }
20: // 開始計時
21: function startClock() {
22:    var time = new Date();
23:    // 取得時間
24:    var hours = displayClock(time.getHours()) + ":";
25:    var minutes = displayClock(time.getMinutes()) + ":";
26:    var seconds = displayClock(time.getSeconds());
27:    // 顯示時間
28:    show.innerHTML = hours + minutes + seconds;
29:    timer = setTimeout("startClock()",1000);
30: }
31: </script>
32: </head>
33: <body onload="startClock()" onunload="stopClock()">
34: <div id="show"></div>
35: </body>
36: </html>
```

程式說明

第 9~15 列：displayClock()函數可以將目前時間轉換成圖片的 HTML 標籤。

第 17~19 列：stopClock()函數可以停止計時器。

第 21~30 列： startClock()函數可以取得系統時間，並且啟動 Window 物件的計時器，在第 22 列建立 Date 物件，第 24~26 列使用物件方法取得小時、分鐘和秒數，並且使用 displayClock()函數轉換成圖片的 HTML 標籤，第 28 列使用標籤物件屬性設定 標籤的內容，也就是目前的時間，在第 29 列啟動計時器來定時執行 startClock()函數。

第 33 列：在<body>標籤設定 onload 和 onunload 事件的函數。

第 34 列：顯示時間的 標籤。

5-6 JavaScript 的 Error 物件

例外處理（Exception Handling）是 JavaScript 的錯誤控制機制，當程式碼有錯誤時，可以在程式出錯時提供解決方案。

5-6-1 JavaScript 的例外處理

JavaScript 的 Error 物件可以取得執行時的錯誤資料，建立 JavaScript 程式碼的例外處理。

Error 物件

Error 物件儲存 JavaScript 執行時產生的錯誤資訊，當 JavaScript 執行階段的錯誤產生後，Error 物件會自動建立，此物件的屬性說明，如下表所示：

屬性	說明
number	錯誤碼，這是一個 32-bit 值，其中後 16-bit 才是真正的錯誤碼
message	錯誤訊息字串
description	如同 message 屬性，這也是錯誤說明的字串

JavaScript 例外處理程式敘述

JavaScript 例外處理程式敘述是：try/catch/finally，可以處理 JavaScript 執行階段的錯誤，如下所示：

```
try {
   …
}
catch(e) {
   // 例外處理
   …
}
finally {
   …
}
```

上述例外處理敘述可以分為三個程式區塊，如下所示：

- try 程式區塊：在此區塊的程式碼是 JavaScript 需要例外處理的程式碼。
- catch 程式區塊：如果 try 程式區塊的程式碼發生錯誤，在這個程式區塊傳入的參數 e 是 Error 物件，可以取得 Error 物件屬性的錯誤資訊，並且建立例外處理的程式碼。
- finally 程式區塊：這是一個選擇性的程式區塊，不論例外是否產生，都會執行此區塊的程式碼。

JavaScript 程式：Ch5_6_1.html

在 JavaScript 程式使用 try/catch/finally 程式敘述建立 JavaScript 執行階段的例外處理，如右圖所示：

上述圖例可以看到取得 Error 物件顯示的錯誤訊息，測試變數 x 的值為 10。

程式內容

```
01: <!DOCTYPE html>
02: <html>
03: <head>
04: <meta charset="utf-8"/>
05: <title>Ch5_6_1.html</title>
06: </head>
07: <body>
08: <h2>JavaScript 的例外處理</h2>
09: <hr/>
10: <script>
11: var x = 10;
12: try {
13:     x = y;  // 測試的錯誤程式碼
```

```
14: }
15: catch(e) {
16:     // 例外處理的程式碼
17:     document.write("錯誤碼: " + (e.number & 0xFFFF) + "<br/>");
18:     document.write("錯誤說明(message): " + e.message + "<br/>");
19:     document.write("錯誤說明(description): " + e.description + "<br/>");
20: }
21: finally {
22:     // 顯示測試值
23:     document.write("<hr/>測試值 x = " + x + "<br/>");
24: }
25: </script>
26: </body>
27: </html>
```

程式說明

第 11 列：產生錯誤的變數。

第 12~24 列： 例外處理的程式敘述，第 12~14 列的 try 程式區塊是在第 13 列產生 JavaScript 的程式錯誤，因為沒有宣告變數 y。

第 15~20 列： 在 catch 程式區塊使用 Error 物件取得錯誤資訊，第 17 列為錯誤碼，在第 18 和 19 列是錯誤說明。

第 21~24 列： finally 程式區塊顯示變數的測試值。

5-6-2 JavaScript 多層的例外處理架構

JavaScript 可以使用 try/catch/finally 程式敘述建立多層例外處理架構，例如：兩層例外處理架構，如下所示：

```
try {
   …
   try {
      throw 運算式
   }
   catch(e) {
      // 第二層的例外處理
      …
      throw e;  // 丟到外層的例外處理
   }
}
```

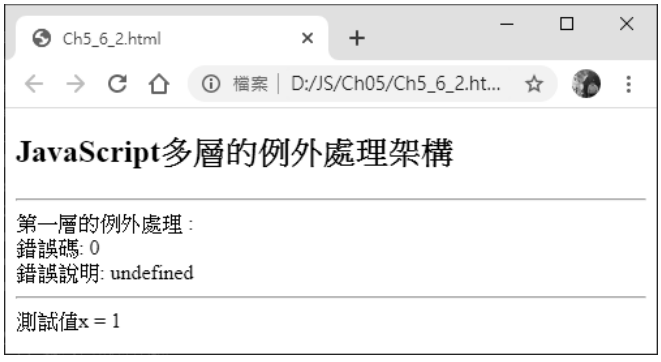
```
catch(e) {
   // 第一層的例外處理
   …
}
finally {
   …
}
```

上述程式區塊擁有兩層例外處理敘述，第二層 try 程式區塊使用 throw 關鍵字產生使用者自訂的錯誤訊息。

在第二層 catch 程式區塊可以處理第二層 try 程式區塊的錯誤，如果不屬於第二層處理的錯誤，就使用 throw 關鍵字將錯誤丟到上一層 catch 程式區塊進行處理。

JavaScript 程式：Ch5_6_2.html

在 JavaScript 程式使用兩層 try/catch/finally 程式敘述，第一層是 JavaScript 執行階段的例外處理；第二層是使用者自訂的例外處理，如右圖所示：



在上述圖例可以看到測試變數 x 的值為 1，所以顯示第一層的例外處理，錯誤訊息為 Error 物件的屬性，如果更改第 11 列程式碼為 x=0，此時的錯誤訊息是自訂的例外訊息，如右圖：

上述圖例顯示第二層例外處理的訊息文字，訊息是使用 throw 關鍵字丟出的自訂錯誤訊息。

程式內容

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8"/>
<title>Ch5_6_2.html</title>
</head>
<body>
<h2>JavaScript 多層的例外處理架構</h2>
<hr/>
<script>
var x = 1;
// 第一層例外處理
try {
    // 第二層例外處理
    try {
       if(x == 0)
          // 程式產生的錯誤訊息
          throw "x 等於零";
       else
          x = y;
    }
    catch(e) {
        // 第二層的例外處理，處理程式產生的錯誤
        if(e == "x 等於零"){
            document.write("第二層的例外處理 : <br/>");
            document.write("自訂的錯誤說明: " + e + "<br/>");
        }
        else
            // 非內部處理的例外
            throw e;  // 丟到外層的例外處理
    }
}
catch(e) {
    // 第一層的例外處理，JavaScript 的執行錯誤
    document.write("第一層的例外處理 : <br/>");
    document.write("錯誤碼: " + (e.number & 0xFFFF) + "<br/>");
    document.write("錯誤說明: " + e.description + "<br/>");
}
finally {
    // 顯示測試值
```

```
41:     document.write("<hr/>測試值 x = " + x + "<br/>");
42: }
43: </script>
44: </body>
45: </html>
```

程式說明

第 11 列： 產生錯誤的變數 x，如果變數值為 1 導致 JavaScript 執行階段的錯誤；0 是使用者的自訂錯誤。

第 13~42 列： 第一層例外處理，在第 13~32 列的 try 程式區塊擁有第二層例外處理，第 16~20 列的 if 條件產生兩種錯誤，第 18 列使用 throw 關鍵字丟出自訂錯誤訊息，第 20 列產生 JavaScript 程式錯誤，因為沒有宣告變數 y。

第 22~31 列： 第二層 catch 程式區塊，這是處理第二層例外的錯誤處理，在第 24~30 列的 if 條件判斷是否為使用者的自訂錯誤，如果是，執行 25~26 列顯示錯誤訊息，如果不是自訂錯誤，就執行第 30 列將錯誤丟到第一層的例外處理。

第 33~38 列： 第一層的 catch 程式區塊，使用 Error 物件取得錯誤資訊，第 36 列為錯誤碼，在第 37 列是錯誤說明，第 39~42 列的 finally 程式區塊顯示變數的測試值。

5-7 物件的共用屬性和方法

JavaScript 內建物件擁有一些共用屬性和方法，這些屬性和方法可以使用在大部分的內建物件。

5-7-1 JavaScript 物件的共用屬性

JavaScript 物件的 constructor 屬性是共用屬性，constructor 屬性可以取得建立物件使用的建構函數名稱，JavaScript 內建物件除了 Global 和 Math 物件外都支援此屬性。

在使用 new 運算子建立 test 物件後，就可以使用 if 條件檢查物件的建構函數，如下所示：

```
if (test.constructor == String) {
   …
}
```

上述 JavaScript 程式碼檢查物件的建構函數是否為 String()。

5-7-2 JavaScript 物件的共用方法

JavaScript 物件常用的共用方法有 toString()和 valueOf()，這兩個方法可以顯示物件的內容。

toString()方法

toString()方法可以傳回物件的內容，傳回值為字串，其語法如下所示：

```
object.toString();
```

上述程式碼可以輸出物件內容的字串，各內建物件輸出的內容，如下表所示：

物件	傳回字串
Array	將陣列元素轉換成「,」符號分隔的字串
Boolean	true 傳回字串"true"；flase 傳回字串"false"
Date	傳回日期和時間的字串
Error	傳回錯誤訊息的字串
Function	傳回字串格式"function name( ) { … }"，其中 name 為呼叫 toString() 方法的函數名稱
Number	傳回數值字串
String	傳回 String 物件的內容

valueOf()方法

valueOf()方法可以傳回物件值，不過 Math 和 Error 物件不支援 valueOf()方法，其語法如下所示：

```
object.valueOf();
```

上述程式碼可以輸出物件內容，各內建物件的輸出內容，如下表所示：

物件	傳回值
Array	將陣列元素轉換成以「,」符號分隔的字串，如同 Array.toString()和 Array.join()方法
Boolean	傳回布林值
Date	傳回前晚到現在的秒數，以千分之一秒為單位
Function	傳回函數的本身
Number	傳回數字
Object	傳回物件本身
String	傳回字串