ECMAScript / JavaScript

Getting Start

課程大綱 1/2

- JavaScript Fundamental
  - 基本語法
  - Basic Data Types
    - char, byte/short/int/long, float/double, boolean
  - Variable
  - Reference Types
    - Array, String 陣列與字串
  - Operators
    - assignment, increment, decrement, compare, logical, bitwise (complement, shift, logical)

課程大綱 2/2

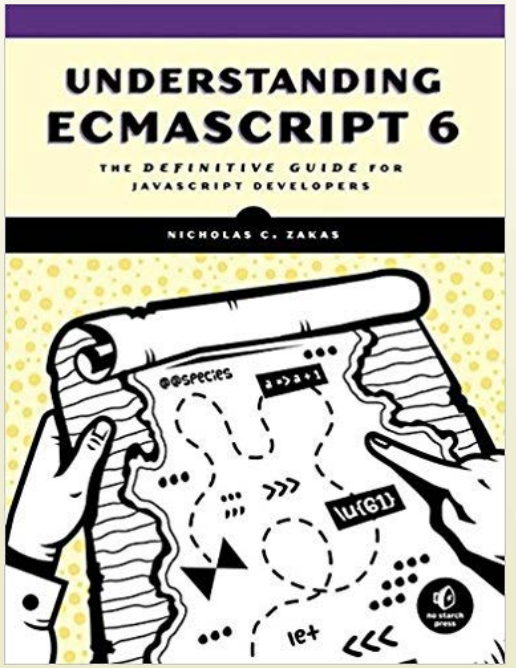
- Expression
- Flow Control
  - if…else…, switch…case…, while, do…while, for
- Function 函數
- DOM 物件 (Document Object Model)
  - window 物件, document 物件, Form 物件, 各種 element 物件
- Event Handling
- Cookie/Session
- AJAX (Asynchronous JavaScript and XML)
- (option) Regular Expression

Reference

- Professional JavaScript for Web Developers
  - Nicholas C. Zakas
  - 3rd Edition
  - January 18, 2012; Wrox

簡中版：JavaScript 高級程序設計

- Understanding ECMAScript 6: The Definitive Guide for JavaScript Developers
  - Nicholas C. Zakas
  - September 3, 2016



- 免費電子書
https://leanpub.com/understandinges6/read
- 有簡中版

What is JavaScript

- JavaScript 是一種 script 程式語言，用來在網頁上加上動態的行為。
  - e.g. 檢驗使用者輸入的帳號密碼是否是空的
  - e.g. 動態改變網頁上文字的顏色與字型大小
- e.g. 使用 Javascript 輸出字串至畫面上

```
document.write("This is my first JavaScript!");
```

- e.g. 設定某個物件為隱藏 (變更 CSS 的 visibility 屬性)

```
$("layer").style.visibility="hidden";  /* jQuery */
```

[note 1] Dart : Google 發展的新語言(2011/10)，欲取代 Javascript
[note 2] WebAssembly (Brendan Eich, 網頁的二進制格式)
[note 3]TypeScript

Java versus JavaScript

- Java and JavaScript are two completely different languages.
- Java (developed by Sun Microsystems) is a powerful and much more complex programming language - in the same category as C and C++.
- JavaScript is an interpreted, lightweight programming language

JavaScript Version

- The language was invented by Brendan Eich at Netscape (Navigator 2.0)
- JavaScript 1.0
  - 1995 Nov., Netscape 公佈 Live Script
  - 之後取得 Sun 授權,改名為 JavaScript
  - appeared in all Netscape and Microsoft browsers since 1996

ECMAScript Version

- 1997 June, ECMA-262 藍皮書 (known as JavaScript 1.3)
  - ECMA-262 is the official JavaScript standard.
  - ECMAScript is developed and maintained by the ECMA organization TC39 (European Computer Manufacturer's Association 歐洲電腦製造商協會, Technical Committee 39)
- 1998 2nd Edition
- 1999 3rd Edition(JavaScript 1.5)
  - 2004 ECMA-357, an extension to ECMAScript
    - 增加對 XML 的支援
- 2009 ECMA-262 5th Edition
  - 2011 Edition 5.1, revision of 5th Edition
- June 2015 ECMA-262 6th edition (ES6-Harmony)
- June 2016 ECMA-262 7th edition (ES7)
- June 2017 ECMA-262 8th edition (ES8)
- June 2018 ECMA-262 9th edition (ES9)

Development Tools

- 選擇 JavaScript Editor
  - 編寫 JavaScript 程式碼可以選擇普通的文字編輯器，如 NotePad++、Sublime Text 、UltraEdit…等。
- Integrated Development Environment (IDE)
  - aptana http://www.aptana.com/
  - Eclipse + JSEclpse
  - JavaBean
  - Spket http://spket.com/download.html
  - …

JavaScript 寫在那裏 1/4

- 直接內嵌在在 <script> 與 </script> tag內
- e.g. j1-first1.html

e.g. <html><head>
<meta charset="utf-8">
<title>Say Hi</title>
</head>
<body>
<script type="text/javascript">
  document.write("Hello World!");
</script>
</body>
</html>

JavaScript 寫在那裏 2/4

- 透過 <script> 的 src 屬性引入

  <script type="text/javascript" src="xxx.js"></script>
  - 程式碼可放在另一檔案內，然後以 src 屬性引用
  - 副檔名 js 只是一般習慣，也可以選用其它附檔名
  - <script>與</script>間不可以再撰寫其他程式碼
  - e.g.
    - 首先編輯 j1-first.js

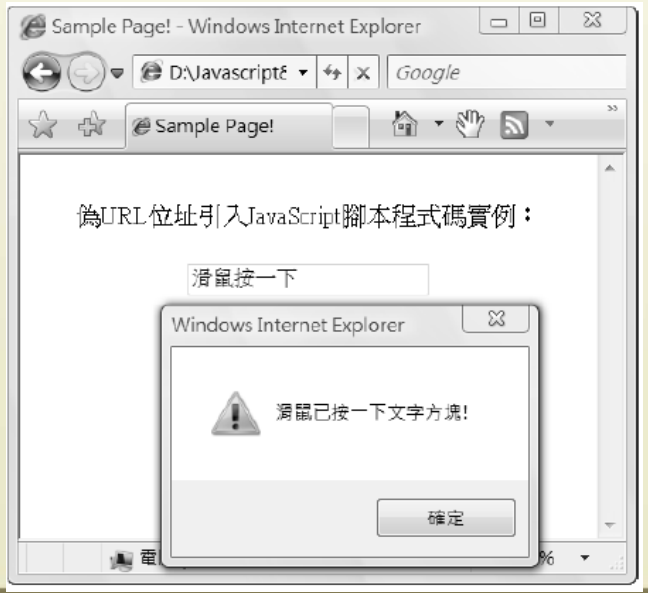
      document.write("Hello World!");
    - 接著使用 src 屬性

      <script type="text/javascript" src="j1-first.js"></script>

JavaScript 寫在那裏 3/4

- 透過網頁事件處理常式引入
- e.g. j1-first2.html

```
<body>
<form name="MyForm">
<input type="text" name="MyText" value="滑鼠按一下"
      onclick="alert('滑鼠已按一下文字方塊')">
</form>
</center>
</body>
```



JavaScript 寫在那裏 4/4

- 也可以直接在 URL 上使用 javascript，
  e.g. j1-first3.html

- 註:
  - <script> 可以放在< head> 與 </head> 之間
  - 也可以放在 <body> 與 </body> 之間
  - JavaScript is case-sensitive.
  - The semicolon ; is a statement terminator.

Comment

- ECMAScript uses C-style comments
- block comment
  - /* multi-line comment */
- line comment
  - // line comment

Example - 基本輸出

- e.g. j1-write.html 利用 JavaScript 來印出 "Hello World!"
- 程式碼重點

```
<script language="text/javascript">
        document.write("Hello World!");
</script>
```

- 說明
  - document 則是一個物件，代表程式碼所在的文件
  - write 則是 document 的一個方法，可將一個字串印出於瀏覽器

Example – write(), writeln()

- 主題：document.write() 和 document.writeln() 的差別
  - 連結：j1-writeln.html
  - 程式碼重點
    - <script>document.write("Good"); document.write("Bye!");</script>
    - <script>document.writeln("Good"); document.writeln("Bye!");</script>
- 說明
  - writeln() 和 write() 的最大差別在於 writeln() 在列印完畢後會換列，但 write() 不會。
  - 如果連續呼叫 document.write("Good") 和 document.write("Bye!")，在網頁會呈現連在一起的 "GoodBye!"，但是如果連續呼叫 document.writeln("Good") 和 document.writeln("Bye!")，則在網頁會呈現中間有空格的 "Good Bye!"，

Alert Window 警示視窗

- 主題：顯示網頁載入時間
- 連結： j2-alert.html
- 程式碼重點

```
<script> today = new Date();
hour = today.getHours();
minute = today.getMinutes();
second = today.getSeconds();
string = "網頁載入時間是"+hour+"點"+minute+"分"+second+"秒“;
</script>
...
<a href="javascript:alert(string)">網頁載入時間</a>
```

- 說明
  - 我們產生字串 string，當連結被按下去時，會以警告視窗顯示載入時間

ECMAScript Language Basics

Variable

- ECMAScript variables are loosely typed.
  - 在宣告時不需要指定資料型態
  - can hold any type of data, and change type anytime
- define variable
  - var xxx; // define a variable
  - var xxx, yyy; // define more than one variable
  - var message = "hi"; // define a variable with initial value

Strict Mode

- The "use strict" directive is new in JavaScript 1.8.5 (ECMAScript version 5).
- It is not a statement, but a literal expression, ignored by earlier versions of JavaScript.
- The purpose of "use strict" is to indicate that the code should be executed in "strict mode".
- Strict mode makes it easier to write "secure" JavaScript.
- With strict mode, you can not, for example, use undeclared variables.

Strict Mode

- variable/property/object must be declared, e.g. "use strict"; x = 1; // cause an error
- delete a variable/function/argument, is not allowed
  - delete operator removes a property from an object
- defining a property more than once, is not allowed
  - e.g. var x = {p1:10, p1:20}; // error
- duplicating a parameter name is not allowed
- Octal numeric literals and escape characters are not allowed
- …
- http://www.w3schools.com/js/js_strict.asp

Identifier 識別字

- [a-zA-Z_$\x7f-\xff][a-zA-Z0-9_$\x7f-\xff]*
- The first character must be a letter, underscore _, or dollar sign $ (第一個字必須是文字或 _ $兩個符號)
- All other characters may be letter, underscore, dollar sign, or numbers (第二個字起可加用數字)
- case sensitivity 大小寫有別
- *keywords* and *reserved words*, true, false, and null cannot be used. e.g. if, else, var...
- e.g. legal identifier
  - TRUE, True, _test, $unitPrice, x7, …

JavaScript 關鍵字一覽表

abstract	boolean	break	byte	case
catch	char	class	const	continue
debugger	default	delete	do	double
else	enum	export	extends	false
final	finally	float	for	function
goto	if	implements	import	in
instanceof	int	interface	long	native
new	null	package	private	protected
public	return	short	static	super
switch	synchronized	this	throw	throws
transient	true	try	typeof	undefined
var	void	volatile	while	with

http://www.quackit.com/javascript/javascript_reserved_words.cfm

identifier 命名習慣

- By convention, ECMAScript use camel case, meaning that first letter is lowercase and each additional word is offset by a capital letter, like this:
  - topSecret
  - myCar
  - doSomething
- upper case for symbolic constants
  - var PI = 3.1415926
  - (ES6) const PI = 3.1415926
    - const 具有 block scope，類似 let
- 雖然有些中文字可以用，建議還是不要用

Data Types

- three simple types (primitive types, scalar types)
  - number (double precision floating-point number, 64-bit)
  - boolean (true/false)
  - string
- two special types (特殊資料型態)
  - undefined (not initialized variable)
  - null (empty object pointer)
- compound types (組合資料型態)
  - array
  - Objects
    - [Built-in] Date、Math、Number…etc.
- operator typeof

typeof

- e.g. j3-typeof.html
  - 用 typeof 到各種變數的資料型態。
- 說明
  - 使用 “==”, “>”,”<”,…這類的判斷運算元，會回傳 true 或 false 的布林型態。

number 數值

- 為保留最高的精確度，JavaScript 內部把所有的數值均表示成雙倍精準浮點數 (IEEE 754 Double Precision)
- 浮點數，其內部儲存和運算方式都是以雙倍精準的浮點數來進行。
- 整數內部以雙倍精準的浮點數儲存。運算時會先轉為整數，計算後再轉為浮點數儲存。
- 在 PC 上，一般浮點數都是佔用 4 bytes，但雙倍精準浮點數會佔用 8 bytes，以提升數值運算的精確度。(常用的科學計算軟體 MATLAB，它的預設數值型態也是雙倍精準浮點數。)

number base 數值基底

- ECMAScript 的整數大部分是以十進位來表示，但若有需要，也可以使用不同的基底 (Base) 來表示，例如八進位和十六進位：
  - 八進位的數字以 0 開頭，只包含數字 0 到 7。(但是如果有一個數字的開頭是 0，但卻也包含數字 8 或 9，或包含小數點，那麼它就會被認定成是一個十進位數字。)
  - 十六進位的數字以 "0x"為字首，只包含了數字 0 到 9，和字母 A 到 F (不論大小寫) 。字母 A 到 F 分別代表 10 到 15。也就是說，0xA 等於 10，而 0xF 等於 15。
  - 八進位和十六進位的數字可以是負數，但是不能有小數部分，而且也不能以科學記號法 (指數) 來表示。

number example 1/2

數字	說明	十進位表示法
.0001, 0.0001, 1e-4, 1.0e-4	4 個浮點數，值皆相等。	0.0001
3.45e2	一個浮點數。	345
42	一個整數。	42
0378	一個整數。雖然看起來像八進位數字（以0開頭），8 不是正確的數字，所以這個數字是十進位數字。	378
0377	一個八進位整數。注意它雖然只比上面的數字小1，它真正的值卻截然不同。	255
0.0001	一個浮點數字。即使以0開頭，它卻不是八進位數字，因為它有一個小數點。	0.0001

number example 2/2

數字	說明	十進位表示法
00.0001	這是一個錯誤。開頭的兩個0顯示它是個八進位數，可是八進位數字是不能有小數點的。	(編譯器錯誤)
0Xff	十六進位的整數。	255
0x37CF	十六進位的整數。	14287
0x3e7	十六進位的整數。注意 "e" 並不是指數。	999
0x3.45e2	這是一個錯誤。十六進位數字不能有小數部分。	(編譯器錯誤)

- 說明
  - 浮點數可用小數點或用科學記號法來表示。若用科學記號法，大寫或小寫的 “e” 都可以表示「10的次方」。
  - IEEE 754 雙倍精準浮點數
    - max: $1.7976931348623157 \times 10^{308}$ (Number.MAX_VALUE)
    - min: 5.00×10^{-324} (Number.MIN_VALUE)

special number

- 產生 JavaScript 的特殊數值
  - NaN（Not a Number，非數字）：用不正確的資料來執行數學運算時，或是執行無意義的數學運算，就會產生這個數值。
  - 正的無限大（Infinity）：當正數大到無法顯示在 JavaScript 中時，就會使用這個數值。
  - 負的無限大（-Infinity）：當負數大到無法顯示在 JavaScript 中時，就會使用這個數值。
  - 為什麼「Math.pow(0, 0) = 1」？理論上應該是 NaN，但不知 JavaScript 為何產生 1，這可能是 JavaScript 內部的一個 bug。
- e.g. j3-numberSpecial.html

about NaN

- NaN == NaN // false
  - NaN is not equal to any value
- isNaN(*number*)
  - 如果傳回值為 NaN，則 isNaN 函式會傳回 true，否則會傳回 false。
  - isNaN(10) // false, 10 is a number
  - isNaN("10") // false,
    - "10" can be converted to number 10
  - isNaN(true) // false
    - true can be converted to number 1

parseInt()

- 若要將字串轉成數值，可用 parseInt() 或是 parseFloat() 這兩個函數
- parseInt(numString, [radix])
  - 傳回一個從字串 numString 轉換而來的整數，其中 radix 是介於 2 和 36 之間的值，用來指出包含在 numString 中的數字基底（Base）。
    - 如果未提供，則字首為 0x 的字串會視為十六進位
    - 字首為 0 的字串則會視為八進位的數字。
    - 其他所有的字串則會視為十進位的數字。
  - 若轉換不成功，則傳回 NaN

parseFloat()

- parseFloat(numString)
  - 傳回一個從字串 numString 轉換而來的浮點數。
  - 若轉換不成功，則傳回 NaN
- e.g. j3-parseInt.html
- 通常會搭配 isNaN() 來測試 parseInt 和 parseFloat 方法的傳回值。

boolean 範例

- j3-boolean.html
  - 有關 Boolean 資料型態的測試
- 程式碼重點

```
document.write("x==y ==> " + (x==y) + "<br>");
document.write("y==z ==> " + (y==z) + "<br>");
document.write("x=z ==> " + (x=z) + "<br>");
```

- 說明
  - JavaScript 中的布林 (Boolean) 資料型態的值只有兩種：true 和 false。
  - 在上述範例中，(x==y) 和 (y==z) 都會測試兩個數目是否相等，因此會回傳布林常數 true 和 false，但是 (x=z) 是一個指派敘述，因此回傳的值就是被指派的值。

boolean 的注意事項

- 0, "", NaN, undefined, null 在邏輯判斷時都算 false
- 其他值則為 true

字串表式方式

- 字串資料型態可以用來表示一列文字內容。我們只要把文字括在相符的單括號或雙括號裡，就可以形成一個字串。
  - "This is a string"
  - 'This is a string quoted by single quotes'
  - "This is a string with 'single' quotes"
  - 'This is a string with "double" quotes'
  - "This another string with \"double\" quotes"
- 說明
  - 為了避掉雙引號的原來用途（標示字串的開始和結束），我們要在雙引號前加上反斜線(\)。
  - 雙引號和單引號可以互相夾雜，但是如："a 'b "c"d ' e"這種過多層的夾雜就會出問題，建議分成多行來寫作，或在引號前使用反斜線(\)。

字串注意事項

- 字串一旦建立就不能再改變
  - e.g. 下例第二行 JavaScript 會重新產生一個足夠儲存新字串的空間來用，原字串所用空間會消除

    ```
    var lang="java";
    lang = lang + "Script";
    ```
- 存取字串第 i 個字元
  - Firefox 可用 str[i], 像是把 str 當 pointer, IE 不可如此用. 所以建議不要這麼用

Template Literals (ES6)

- 使用 back-tick ` (反單引號)
- e.g. var s = `hello, world`;
- 用於多行字串, .e.g.
  `Template literals are string literals allowing embedded expressions.
  You can use multi-line strings and string interpolation features with them.
  They were called "template strings" in prior editions of the ES2015 specification.`
- 變數代換, e.g.
  var name = "John";
  var s = `hello, ${name}`;

undefined

- undefined 是一種特殊的資料值，通常用來判斷：
  - 變數是否未宣告
  - 或 已宣告 但沒有初始值
- 未宣告的變數無任存取，所以存取前可以用 typeof() 函數檢查，未宣告的變數會回傳 "undefined" 字串。
- 如果已宣告但尚未初始值，存取時會得到 undefined (非字串!)。
  - 此類變數亦可經由 typeof() 回傳 "undefined" 字串
- e.g. j3-undefined.html

null

- null 是一種特殊的資料值，指的是空物件指標 (empty object pointer)
- null 是小寫，Null 或 NULL 都是錯誤的
- null, 0, false 在條件判斷時都是不成立的條件。
- 以 typeof() 檢查是回傳 "object"
- e.g.：j3-null.html

Operator

Operators 1/4

- Arithmetic Operators + - * / %
  - % mod
  - The + operator can also be used to concatenate string variables or text values together.
  - If you add a number and a string, the result will be a string.
- Increment Operator ++, Decrement Operator --
- e.g. j4-operator.html

Operators 2/4

- Bitwise Operators & | ^ ~ << >> >>> [note]
  - e.g. n = n & ~077 /* set last 6 bits to zero */
  - >> shift right with signed bit extension
  - >>> shift right without signed bit extension (filled by 0)

- [note] All number in ECMAScript are stored in IEEE-754 64-bit format, but bitwise operations do not work directly on the format. instead, the value is converted into a 32-bit 2's complement integer, then operation takes place, and the result is converted back into IEEE-754 64-bit format.

Operator notes

- e.g.

```
var x = 9007199254740992;  // 2^53
var y = -x;
x == x + 1; // true !
y == y - 1; // also true !
x / 2; // 4503599627370496
x >> 1; // 0
x | 1; // 1
```

Operators 3/4

- Assignment Operators = += -= *= /= %= &= ^= |= <<= >>=
- Relational Operators > >= < <= == === != !
  - === exactly equal (value and type)
    - e.g. x=5 so x == “5” is true
      x === “5” is false
- Logical Operators && ||
  - e.g. 判斷閏年
    (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)

Operators 4/4

- Conditional Operator (implicit If) ?:
  - e.g. z= (a > b) ? a : b /* z = max(a, b) */
  - e.g. j4-implicitIf.html
    - 取得現在時間，並用 ?: 判斷是上午或下午

      prepand = (hour>=12)? "下午":"上午";

      hour = (hour>=12)? (hour-12):hour;
- Comma Operator ,
  - e.g.

    var num = (5, 1, 4, 8, 0); →num=0
  - e.g.

    data = (a=1, b=2, c=3) → Q: data = ?
- Spread Operator … (ES6)

Precedence	Operator	Description	Example	Associated
20	( )	Expression grouping	(3 + 4)	
19	.	Member	person.name	
19	[]	Member	person["name"]	
19	()	Function call	myFunction()	
19	new	Create	new Date()	
17	++ --	Postfix Increment/Dec	i++	
16	++ --	Prefix Increment/Dec	++i	
16	!	Logical not	!(x==y)	
16	typeof	Type	typeof x	right to left
15	**	Exponentiation (ES2016)	10 ** 2	right to left
14	* / %	Multiplication/Div/Mod	10 * 5	left to right
13	+ -	Addition/Subtraction	10 + 5	left to right
12	<< >> >>>	Shift	x << 2	left to right
11	< <= > >=	Relation	x < y	left to right
11	in	Property in Object	"PI" in Math	left to right
11	instanceof	Instance of Object	instanceof Array	left to right
10	== === != !==	Equal	x == y	left to right
9	&	Bitwise AND	x & y	left to right
8	^	Bitwise XOR	x ^ y	left to right
7	|	Bitwise OR	x | y	left to right
6	&&	Logical AND	x && y	left to right
5	||	Logical OR	x || y	left to right
4	? :	Condition	? "Yes" : "No"	left to right
3	+= /= -= *= %= <<= >>= >>>= &= ^= |=	Assignment	x += y	right to left
2	yield	Pause Function	yield x	
1	,	Comma	5 , 6	left to right

Flow Control

if-else敘述

- if-else基本結構

```
if (條件句) {
        程式碼 1
} else {
        程式碼 2
}
```

- 說明
  - 當條件句的值是true或非零，就會執行程式碼 1，剩下的情況就會執行程式碼 2
  - 這樣的架構只會執行程式碼 1 或 2，一組if-else敘述只會執行其中一段程式碼。

if 範例

- e.g. j5-ifElse.html
  - 利用if-else敘述，判斷使用者輸入的值。
- 重點程式碼

```
if (a<30) {
        alert("您只有 "+a+ " 歲，真是青年才俊啊！");
} else {
        alert("您年過30，想必是事業有成了！");
}
```

- 說明
  - 如果在 if-else 程式碼只有單行，可以省略{ }符號。
  - 如果需要判斷很多種可能，可以用 if…else if…else，其中 else if 的個數視需求而定。

if 範例

- e.g. j5-leapYear.html
  - 利用 if-else 敘述，判斷是否為閏年
- 重點程式碼

```
if ((year % 4 == 0 && year % 100 != 0) || year % 400 ==
0) {
    document.writeln(year + " is a leap year");
} else {
    document.writeln(year + " is not a leap year");
}
```

- 說明

switch-case 敘述

- switch-case基本結構

```
switch(var){
        case 1:
                程式碼1; break;
        case 2:
                程式碼2; break;
        default:
                程式碼3;
}
```

- 說明
  - var的值如果等於case後面放的數字，就會從該case開始執行程式碼。
  - 如果不用break，會造成程式碼循序往下執行，使用break會跳出switch-case這個區塊。
  - 當var的值沒有相對應的case時，就會執行default。

switch 範例

- e.g. j5-switch.html
  - 利用switch-case敘述，根據星期替換網頁內容。
- 說明
  - day 的值是從 0 到 6，分別代表星期日、星期一、星期二、...、星期六。
  - default 之後的敘述，只會在所有條件均不符合時，才會被執行。
  - 如果不加上 break，則系統會在符合某一個特定條件後，繼續執行後續其他條件的敘述，產生不是我們要的結果。
  - switch-case 的行為和 C/C++ 中的 switch-case 相同

邏輯判斷

- 若是判斷條件較複雜，我們也可以使用「且」、「或」、「否定」等方式來產生複合條件。

說明	符號
且	&&
或	||
否定	!

- 範例：判斷是否「a 大於零，或 b 和 c 均不小於零」。

```
if ((a>0)|| (!(b<0) &&!(c<0))){
    ...
}
```

true/false 範例

- e.g. j5-testIf.html
  - 判斷條件的真偽。
- 說明
  - 當運算結果是一個數值時，若此數值等於 0，則是 false，其他則是 true。
  - 當運算結果是一個字串時，若此字串等於空字串“”，則是 false，其他則是 true。

條件敘述	判定結果
0	false
5	true
-3	true
""	false
"0"	true
"00"	true
"0.0"	true

for 迴圈

- 基本結構

```
for (計數變數的初值;判斷式;更改計數變數的值) {
        迴圈內部敘述
}
```

- 說明
  - 在每一次執行前都會先檢查判斷式，成立就執行迴圈，不成立就跳出迴圈。
  - 更改計數變數的值，是在每次執行完一輪迴圈後，計數變數值的變化。

for 範例

- e.g. j5-forLoop.html
  - 由 for 迴圈來產生 5 個由小變大的 "Hello World!
- 重點程式碼

```
for (i=1; i<=5; i++) {
      document.write("Font size = " + i + "em ===> ");
      document.write("<span style=\"font-size:"+i+"em;\"
>Hello World!</span><br>");
}
```

- 說明
  - 利用變數 i 和 html 字串組合，跑出五種不同大小的字。

for-in 迴圈

- 基本結構

```
for (變數 in 物件){
    …
}
```

- 說明
  - 這個變數代表物件中每一個屬性的屬性名稱，我們可以用”物件[變數]”，這樣的方式來取得該屬性的值。

for-in 範例

- e.g. j5-forInLoop.html
  - 使用 for-in 迴圈來列舉 window 物件的屬性
- 重點程式碼

```
for (propName in window)
    document.write( propName + "<br>" );
```

- 說明

while 迴圈

- 基本結構

```
while(條件式){
    程式碼
}
```

- 說明
  - 程式會先判斷條件式是否成立，再決定是否要繼續迴圈，當條件式成立時，會執行{ }中的程式碼，不成立時則跳出。

while 範例

- e.g. j5-while.html
  - 用 while 迴圈亂數產生許多 0~1 之間的數字。
- 重點程式碼

```
x=Math.random();
while (x<=0.8){
    document.write("<br>"+x); x=Math.random();
}
```

- 說明
  - 只要條件式為真，while 迴圈的內部敘述就會反覆一再被執行。利用 while 迴圈來反覆印出隨機變數值，直到所遇到的隨機變數值大於 0.8 才停止。

do-while 迴圈

- 基本結構

```
do{
    程式碼
} while (條件式);
```

- 說明
  - 和 while 不同地方在於 do-while 會先執行一次再判斷，而 while 是先判斷一次再執行。
  - 在 while (條件式) 的後面切記要加上”;”

do-while 範例

- e.g. j5-doWhile.html
  - 利用 do-while 迴圈產生許多 0~1 之間的數字。
- 重點程式碼

```
do{
   x=Math.random(); document.write("<br>"+x);
} while (x<=0.8);
```

- 說明：
  - 和範例 j5-while.html 差別在於 do-while 最後一個輸出的值會大於 0.8。