

Java Tutorial

Chapter 1 JAVA 基本介紹與簡單語法

1.1 什麼是 JAVA?

程式語言百百種，根據不同的情況與需求，各種程式語言都有自己不同的特色，而現今最熱門的程式語言包括了傳統的 C 與 C++，腳本語言的 Python 及 Perl，寫網頁的 JavaScript、PHP，或是做資料庫的 SQL 都在不同的用途上大放異彩。在本教學所介紹的 JAVA 更不用說，在 2015 年的調查，JAVA 成為最受歡迎的程式語言。為什麼 JAVA 會如此受人歡迎呢？JAVA 有幾項特色大概可以稍微解釋為何它是如此的受歡迎。

(1) 跨平台

首先是跨平台的特色，JAVA 從開創造以來就秉持著一個相當重要的理念：一次編寫，到處執行。不同於傳統的 C 或 C++ 只要轉換平台就必須重新編譯，JAVA 的程式只要編譯完成之後就可以在各種平台上執行，不需要去考慮平台的相容性。

(2) 支援多執行緒

多執行緒乍聽之下很深奧，說穿了就是能夠做並行處理，一個程式可以同時執行多項工作，雖然大多數的程式語言都可以實現並行處理，但對 JAVA 而言並行處理是自動控制的，而對其他語言來說，需要特別讓系統進行指派，進而造就了實作上的困難。

(3) 簡單

JAVA 是由 C 與 C++ 演變而來，然而卻不像這兩種語言有許多麻煩的地方，其中最大的一個好處就是不需要做記憶體管理，就不需要擔心程式會因為記憶體不足而導致無法執行的問題。

(4) 物件導向

以往如 C 語言是使用所謂程序型的設計方法，在程式的複雜程度開始提高之後就會開始顯得乏力與難以管理，因此物件導向的程式設計概念讓程式實作從另一個角度切入並且可以讓程式在趨於複雜與壯大的同時同時保持易於管理與實現。而針對物件導向的詳細說明會在下一個章節仔細介紹。

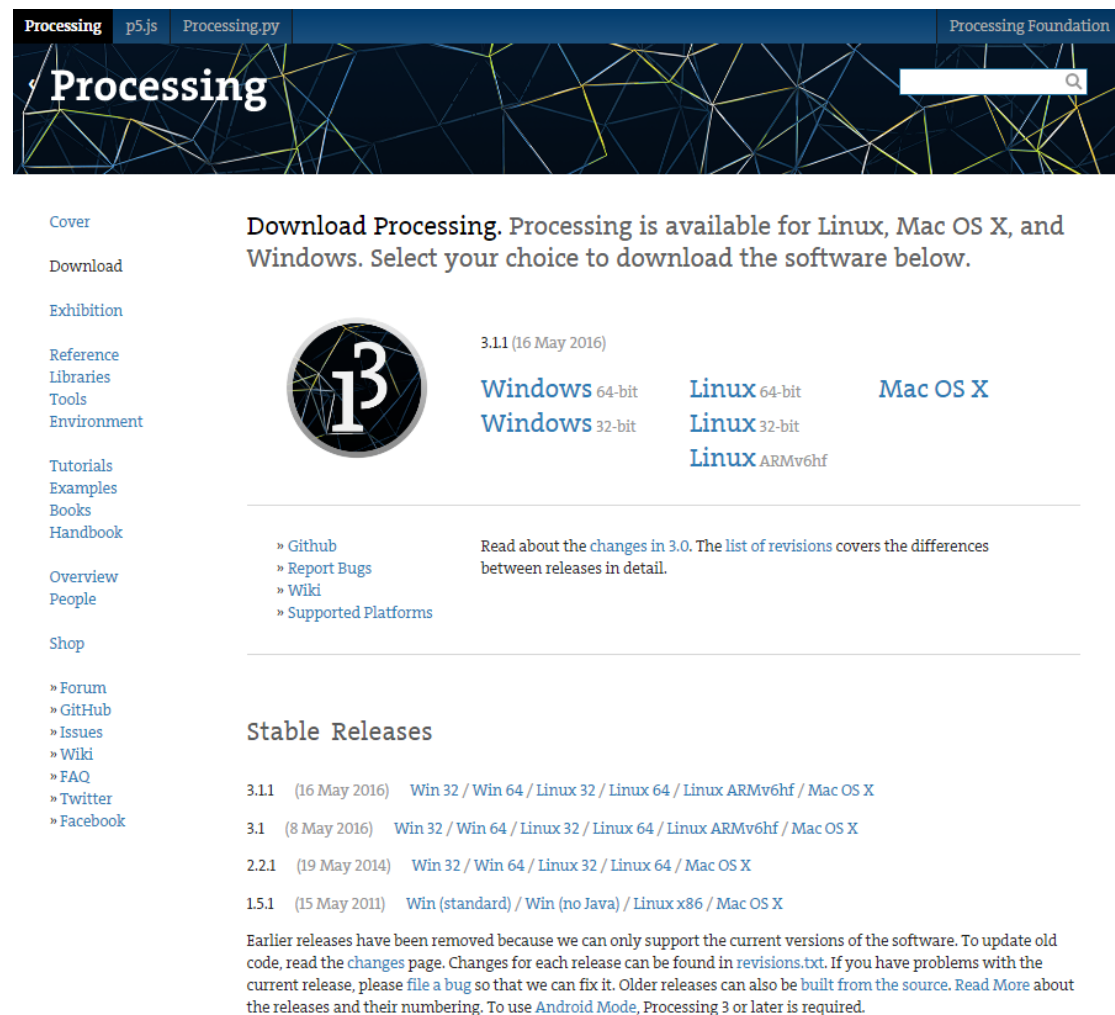
1.2 開始 JAVA

要開始實作 JAVA 程式，我們必須選擇工作環境。由於往後實作的都是與視覺有關的，所以選擇 Processing 來進行實作。Processing 不僅是開發環境，本身

是基於 JAVA 並且有自己的一些特別語法的開源語言，專門為電子藝術和視覺互動設計而創建。Processing 簡單易上手的特性也是讓新手較容易入門的一個工作環境。

1. 安裝 Processing

首先連上 <https://processing.org/download/?processing> 並選擇適合自己作業系統的版本



Processing p5.js Processing.py Processing Foundation

Download Processing. Processing is available for Linux, Mac OS X, and Windows. Select your choice to download the software below.

3.1.1 (16 May 2016)

Windows 64-bit
Windows 32-bit

Linux 64-bit
Linux 32-bit
Linux ARMv6hf

Mac OS X

» Github
» Report Bugs
» Wiki
» Supported Platforms

Read about the [changes in 3.0](#). The [list of revisions](#) covers the differences between releases in detail.

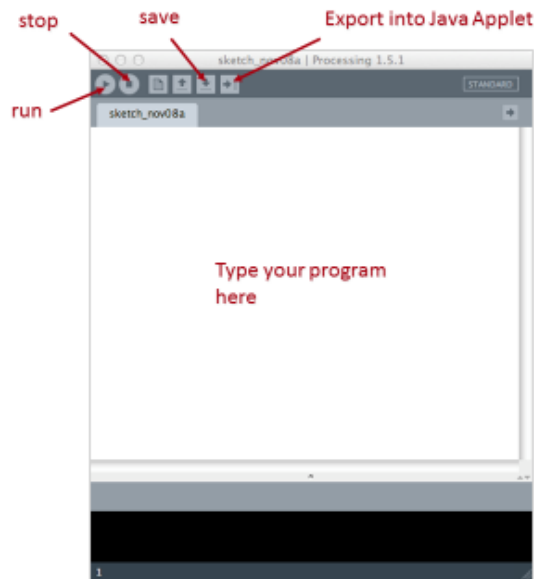
Stable Releases

Version	Date	Download Links
3.1.1	(16 May 2016)	Win 32 / Win 64 / Linux 32 / Linux 64 / Linux ARMv6hf / Mac OS X
3.1	(8 May 2016)	Win 32 / Win 64 / Linux 32 / Linux 64 / Linux ARMv6hf / Mac OS X
2.2.1	(19 May 2014)	Win 32 / Win 64 / Linux 32 / Linux 64 / Mac OS X
1.5.1	(15 May 2011)	Win (standard) / Win (no Java) / Linux x86 / Mac OS X

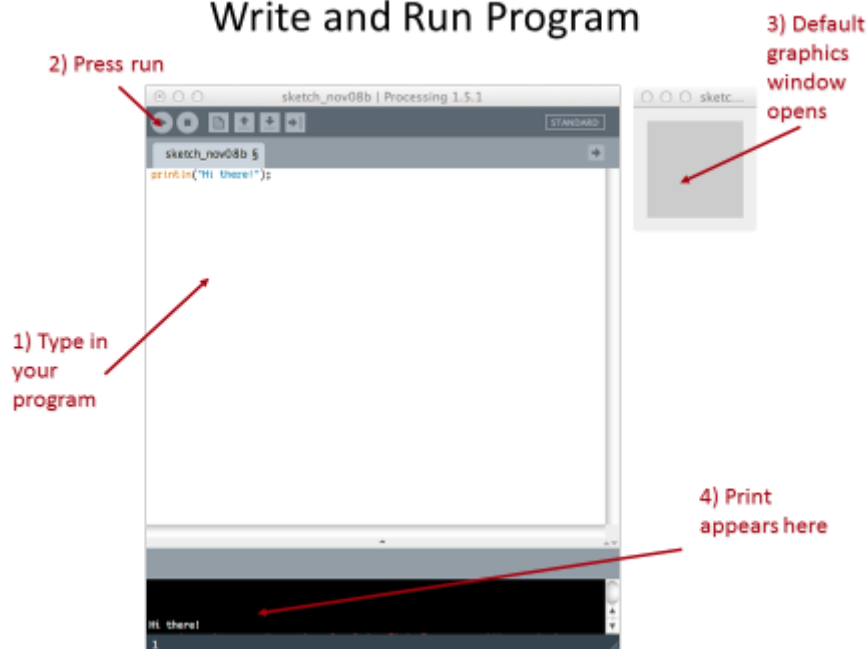
Earlier releases have been removed because we can only support the current versions of the software. To update old code, read the [changes](#) page. Changes for each release can be found in [revisions.txt](#). If you have problems with the current release, please [file a bug](#) so that we can fix it. Older releases can also be [built from the source](#). Read [More](#) about the releases and their numbering. To use [Android Mode](#), Processing 3 or later is required.

下載之後解壓縮後便可直接執行文件夾中的”Processing”應用程式即可開始準備寫第一支 JAVA 程式

PDE- Run Processing



Write and Run Program



2. Hello World

開始一個新的程式語言，當然不免俗就是要先寫出 Hello World 作為入門的第一課

```
//This prints "Hello World." to the IDE console.  
void setup() {  
    println("Hello world.");  
}
```

而 Processing 作為一個強調視覺互動的工作環境，以下的範例可以初步的跟

滑鼠做簡單的畫線互動

```
//Hello mouse.  
void setup() {  
    size(400, 400);  
    stroke(255);  
    background(192, 64, 0);  
}  
  
void draw() {  
    line(150, 25, mouseX, mouseY);  
}
```

詳細的基本語法會在下一節作介紹

1.3 基本語法與資料型態

(1) Main function

每個程式都一定會有一個函式叫做 Main，在 Processing 裡面叫做 setup。這個函式不僅僅在每個程式都一定會出現，更一定只會有一個。這個獨一無二的函式就是一切的起頭，每個程式都會從這個函式開始執行起並且只執行一次。

```
void setup() {  
    size(400, 400);  
    stroke(255);  
    background(192, 64, 0);  
    println("HAHAHA");  
}
```

此函式為固定寫法，必定為 void setup(){}，在 {} 裡面加上要執行的程式碼即可。

(2) 基本運算

首先是最基本的四則運算，加減乘除如同平時熟悉的一樣，就是使用 "+ - * /" 來進行運算。比較特殊的則是 "%"، 這個符號可以用來計算餘數，例如 5%3 得到的結果會是 2。再來另一個較特殊的是 "=" 以及 "=="。在 JAVA 裡面使用單個等號 "=" 是賦予值的意思，而雙等號 "==" 才是相等的意思。舉例來說：

A = b 是將 b 的值給 A 的意思，而

A == b 是 A 與 b 相等的意思

(3) print(), println(), printf()

在撰寫程式的時候我們常常必須確認我們所寫的邏輯是否正確，又可能需要顯示一些資訊來讓我們知道現在程式進行的過程為何，正因如此我們必須借助 Print 的 function 來把需要的資訊顯示出來才方便我們做程式的監控。

print() 與 println() 相差不大，唯一差別只在於 println() 會在結尾自動加上換行符號，而若是想要在 print() 裡換行，則必須再加上 "\n" 的換行符號。至於 printf()

則是為了一些 C/C++ 的使用者所增加的函式，其功用如同在 C/C++ 裡一樣，可以做一些格式轉換控制字元的功能的印出。

(4) 資料型態

在程式裡面，需要運算許多資訊也需要儲存資訊，這些資訊無論是由什麼方法取得，在程式的運行當中都會透過變數加以儲存。每個變數都需要被宣告指向某個記憶體空間，而在宣告的同時也必須先確立其形態，不同的資料型態理所當然地會有不同的特性並且佔有不同大小的儲存空間，因此我們在使用時候必須知道這個變數在扮演什麼樣的腳色而給予最合適的資料型態。以下將介紹幾個常用的資料型態。

a. 整數

只儲存整數數值，其中可細分 short, int, 與 long，差別在於佔用記憶體空間不一樣，也意味著可表示的整數範圍不同。而在整數系列裡 int 為最普遍使用的資料型態。

b. 位元

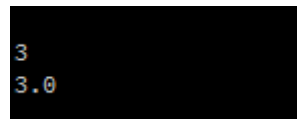
byte 資料型態，專門儲存位元資料，例如影像一個像素通常就由一個位元表示，而必要的話 byte 也可以用來儲存整數值。

c. 浮點數

主要儲存小數數值，也可以儲存範圍更大的整數，分為 float 與 double，其中差異也是佔用記憶體空間與表示範圍的不同。

以下顯示了整數與浮點數的差異，同樣被賦予 3 的值，在顯示的時候浮點數型態的 b 會被印出附帶小數點的數值。

```
void setup() {  
    int a = 3;  
    float b = 3;  
    println(a);  
    println(b);  
}
```




```
3  
3.0
```

d. 字元

char 資料型態，用來儲存字元，而 char 會以 unicode 編碼方式儲存，若是將 char 型態的變數以 int 型態表示就會顯示 unicode 的號碼。而賦予的時候只會有單一個字母或是數字，並且用單引號框起來表示。

```
void setup() {  
    char a = 'a';  
    println(a);  
    println(int(a));  
}
```



```
a  
97
```

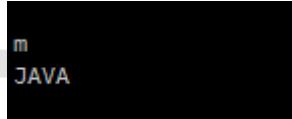
由上圖可以看到宣告了一個變數 a 為 char 型態，並且賦予了一個值為 a，而接著分別印出變數 a 的值以及 a 以 int 型態表示的值。由結果可以看到分

別印出了 a 以及 97，這也說明了不同型態之間的變化。

e. 字串

String 資料型態與字元不同，它表示的是一串字元的組合，而賦予值的時候使用雙引號表示。

```
void setup() {  
  char a = 'm';  
  String b = "JAVA";  
  
  println(a);  
  println(b);  
}
```



m
JAVA

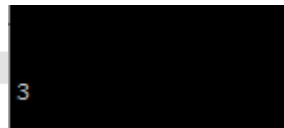
f. 布林

bool 型態只有兩種數值，分別為 true 與 false，而 true == 1 且 false == 0 為固定的定義，這種資料型態可以用來在需要做二擇判斷的時候來使用。

(5) 註解

在一個程式裡，總是需要一些註解來讓我們更好了解究竟程式碼是在進行什麼樣的計算，另一方面，我們也可以使用此方法將程式碼註記掉，使得程式執行的時候會跳過被註記的片段。在 JAVA 中，註記使用兩個斜線“//”，雙斜線後的一整行將會被程式當作是註解而不會執行。

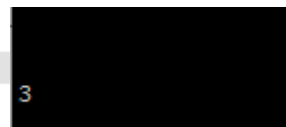
```
void setup() {  
  int a = 3;  
  float b = 3;  
  println(a);  
  //println(b);  
}
```



3

另一方面若是想要一口氣註解掉一段程式碼，則可以在開頭與結尾分別加上“/*”與“*/”。這兩個符號必定成雙成對，若是少了其中一個程式將無法執行出現錯誤。

```
void setup() {  
  int a = 3;  
  float b = 3;  
  println(a);  
  /*  
  println(b);  
  println("123123123");  
  */  
}
```



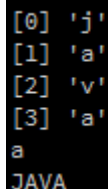
3

(6) 陣列

陣列是一種特別的資料型態，除了布林以外其餘每種資料型態都可以做成陣列，陣列可以想像成一格一格的盒子並串起來，每個盒子裡分別都裝著一個

值。宣告陣列的方法就是在資料型態後面加上中括號”[]”即可。若是要宣告一個空陣列，則必須事先配置一個大小。宣告完成之後，若需要訪問陣列裡的特定位置，則在中括號裡加上數字即可。

```
void setup() {  
    char[] a = {'j','a','v','a'};  
    char[] c = new char[4];  
    String b = "JAVA";  
  
    println(a);  
    println(a[1]);  
    println(b);  
}
```



```
[0] 'j'  
[1] 'a'  
[2] 'v'  
[3] 'a'  
a  
JAVA
```

然而陣列不只如此，在 JAVA 中陣列可以輕鬆擴充成高維度的陣列，以最常用到的二維陣列來說：

```
void setup() {  
    int[][] a = {{1,2,3},{4,5,6}};  
    String b = "JAVA";  
  
    println(a[1][2]);  
    println(b);  
}
```



```
6  
JAVA
```

值得一提的是，在程式語言中，編號的開始都是由 0 開始，因此要訪問陣列 a 的第二個值會寫成 a[1]。

(7) 函式

函式(function)在程式開始複雜之後會開始扮演重要的腳色，它可以視為一群敘述的集合，因此常會將某些經常使用的敘述直接用一個函式包裝起來進而在需要使用的時候直接呼叫函式，這樣能夠簡化重複性的程式碼片段並且較易於管理。先前所提到的 Main 即是一個函式，但由於其特殊的地位，它與一般使用的函式有著不一樣的功能。函是有點像在數學裡的函數，給予參數並經過函數的處理之後得到函數的輸出結果；同樣的在 JAVA 中，我們給予引數並經過函式的處理之後獲得我們想要的回傳值。

因此我們在建立一個函式的時候必須先確定該函式需要什麼樣的引數，並且希望經過此函式之後會有什麼樣的回傳值。舉例來說我們想要將兩個 int 變數加在一起，則我們需要分別傳入兩個 int 變數並且希望獲得的也是一個 int 變數，因此我們會將函式宣告為：



```
int add(int a , int b){ }
```

而回傳值將會在函式內最後寫出關鍵字 return 來表示此函式將回傳什麼東西。


```
int add(int a , int b){ }
    int c = a + b;
    return c;
}
```

若是我們沒有想要回傳值，只是想要將某段執行的程式碼包裝起來進而可以重複使用，則我們會在回傳值型態上寫上”void”，如同我們的 main function

```
void setup() {
```

如此一來程式碼執行到這個函式裡面並不會回傳任何東西，單純只是執行函式內的程式碼而已。

1.4 判斷式與迴圈

在撰寫程式的過程中，我們一定會遇上一些需要條件判斷的時候；另一方面，也會有需要重複執行某些東西的時候，而判斷式與迴圈就是在這個時候派上用場。

(1) 布林表示式

一般來說，我們會想要讓數值進行比大小或是判斷相等與否，則我們會有簡單的判斷符號可以進行，例如先前所提過的相等符號”==”，而不相等的符號則為”!=”。進一步來說，我們需要判斷 A 跟 B 是否同時成立，或是任一個成立，這種被稱為布林表示式。在程式的表現上我們會以”&&”、“||”與”!”來分別表示布林 and 與布林 or。

A	B	!A	A&&B	A B
true	true	false	true	true
true	false	false	false	true
false	true	true	false	true
false	false	true	false	false

(2) If else 判斷式

在知道如何使用大小比較與布林判斷之後，我們就可以將之應用到 if 判斷式以及後面會提到的迴圈裡當作條件來執行。If 判斷式非常的直觀：若符合 A 條件則執行 1，若符合 B 條件則執行 2，若都沒有則執行 3


```

int score = 88;
char level;
if(score >= 90) {
    level = 'A';
}
else if(score >= 80 && score < 90) {
    level = 'B';
}
else if(score >= 70 && score < 80) {
    level = 'C';
}
else if(score >= 60 && score < 70) {
    level = 'D';
}
else {
    level = 'E';
}

```

平時若是二擇判斷，則就只會有一組 if 與 else，若是有更多的情形會發生，就會加入 else if 來加入判斷。

(3) While 迴圈

與 if 判斷式不同，接下來的三種均為迴圈，在裡面的程式片段在符合條件之下會持續的重複執行。while 常用於沒有確定重複次數的迴圈，同時 while 後面接著迴圈結束的條件，通常是運算表示式，而在每次迴圈開始之前會進行結束條件的判斷

終止條件
↓

```

while (i <= 100) {
    sum += i++;
}

```

(4) For 迴圈

與 while 不盡相同，for 迴圈常用在有確定重複次數的迴圈，同時 for 後面會有三個參數表示重複次數與判斷基準

起始條件 終止條件 參數調整
↓ ↓ ↓

```

for (i = 1; i <= 100; i++) {
    sum += i;
}

```

從第一次進入迴圈之後就會從起始條件判斷是否符合終止條件，若符合

則執行迴圈，等到下一次再開始迴圈，就會進行參數調整之後再與終止條件判斷，迴圈將執行到不符合終止條件為止。

(5) Do while 迴圈

與 while 迴圈有些許差別，屬於後測式的迴圈，迴圈會先進行第一輪之後才進行迴圈的結束條件測試

```
do {  
    sum += i++;  
} while (i <= 100);
```

↑ 終止條件

(6) break, continue

這兩個關鍵字可以對迴圈做特殊處置，當迴圈進行到 break 的時候會即刻終止迴圈並跳出；而迴圈進行到 continue 的時候則僅僅會跳過這一輪迴圈的執行。

本章初步介紹了一些 JAVA 的基本觀念，但目前為止都還只是屬於直述程序型的程式設計，在下一章會開始進入物件導向的介紹，了解什麼是物件、類別等等較不一樣的撰寫程式的方式。