

# 程式人

月刊  
雜誌

## Programmer



\* 0 1 2 3 7 8 \*

捐發票愛心條碼

讀書做善事、寫書做公益 – 歡迎程式人認養專欄或捐出您的網誌  
參考價：NT 50 元，如果您喜歡本雜誌，請將書款捐贈公益團體  
羅慧夫顧顏基金會 彰化銀行 (009) 帳號：5234-01-41778-800

# 程式人雜誌

2014 年 11 月

本期焦點：JsLab - JavaScript 版的科學計算平台

# 程式人雜誌

- 前言
  - 編輯小語
  - 授權聲明
- 本期焦點：JsLab -- JavaScript 版的科學計算平台
  - 科學計算軟體簡介
  - JsLab -- JavaScript 版的科學計算平台
  - 「JsLab 科學計算平台」背後的開放原始碼結構
  - R.js -- 從 jStat 延伸出的開源 JavaScript 機率統計框架
- 程式人文集
  - d3.js -- 互動式繪圖框架
  - c3.js -- 基於 d3.js 的簡易繪圖框架
  - Vis.js -- 另一個強大的 JavaScript 繪圖函式庫
  - CodeMirror -- 有 IntelliSense 功能的網頁版開源編輯器
  - Memory Sanitization (作者：研發養成所 Bridan)
  - 利用 SQL Compact Edition 免費建立擁有 DataBase 的 Azure Websites (作者：陳星銘)
  - 函數指標陣列 (Array of Function Pointers) (作者：研發養成所 Bridan)
- 雜誌訊息
  - 讀者訂閱
  - 投稿須知
  - 參與編輯
  - 公益資訊

# 前言

## 編輯小語

本期的「程式人雜誌」探討的焦點是我在建構 JsLab 這個「JavaScript 的科學計算平台」上所採用的開放原始碼專案，以及建構這個專案過程中所學到的一些經驗與心得，透過這樣的心得分享，或許可以讓「程式人」對科學計算與開放原始碼的使用有更多的瞭解也說不定。

----（「程式人雜誌」編輯 - 陳鍾誠）

## 授權聲明

本雜誌許多資料修改自維基百科，採用 創作共用：[姓名標示、相同方式分享](#) 授權，若您想要修改本書產生衍生著作時，至少應該遵守下列授權條件：

1. 標示原作者姓名 (包含該文章作者，若有來自維基百科的部份也請一併標示)。
2. 採用 創作共用：[姓名標示、相同方式分享](#) 的方式公開衍生著作。

另外、當本雜誌中有文章或素材並非採用 [姓名標示、相同方式分享](#) 時，將會在該文章或素材後面標示其授權，此時該文章將以該標示的方式授權釋出，請修改者注意這些授權標示，以避免產生侵權糾紛。

例如有些文章可能不希望被作為「商業性使用」，此時就可能會採用創作共用：[姓名標示、非商業性、相同方式分享](#) 的授權，此時您就不應當將該文章用於商業用途上。

最後、懇請勿移除公益捐贈的相關描述，以便讓愛心得以持續散播！

# 本期焦點：JsLab -- JavaScript 版的科學計算平台

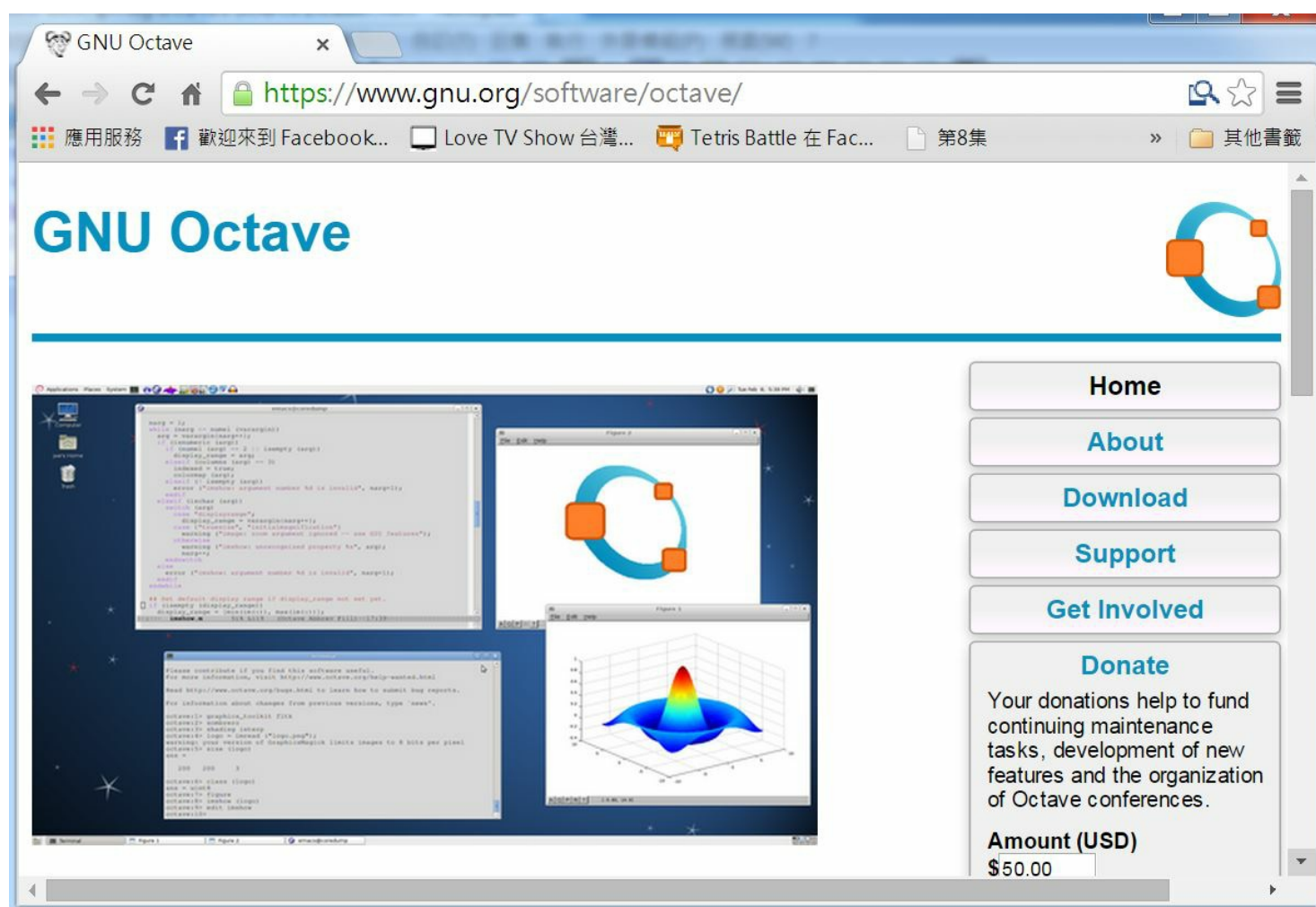
## 科學計算軟體簡介

很多人都曾經使用過「科學計算軟體」，特別是對於進行學術研究的人員而言，這些軟體可以說是不可或缺的。

在工程領域，最常被使用的科學計算軟體是 MatLab，這個軟體從「矩陣運算」出發，建構出了龐大的函式庫，讓使用者可以輕易的透過 Matlab 進行電腦實驗。

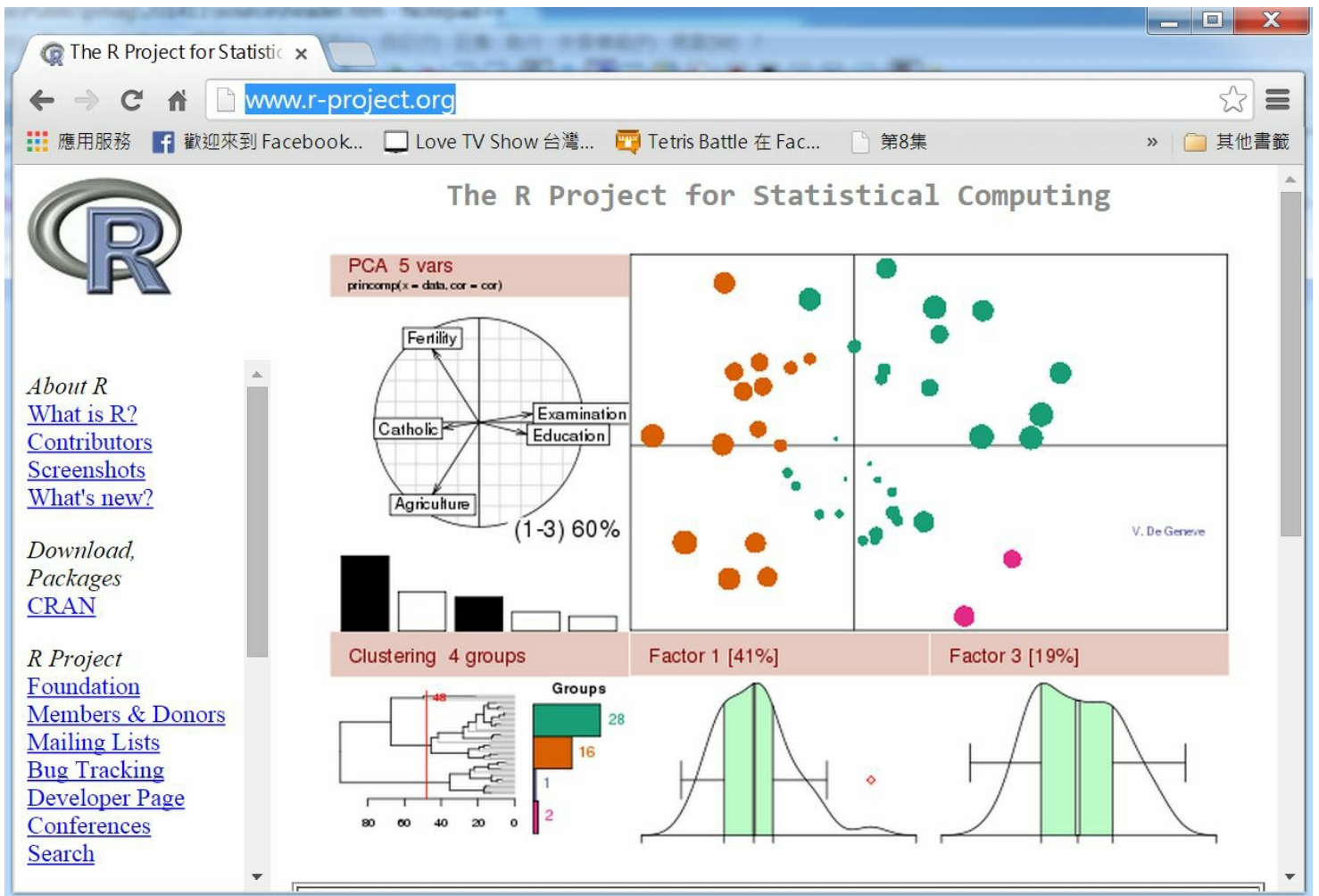
另外、在社會科學與統計學領域，很多人會用 SPSS、SAS 等軟體做統計分析，這兩個軟體是從統計出發的科學計算工具。

事實上、開放原始碼領域也有對應的科學計算軟體，像是 R 與 Octave 都是 Open Source 的科學計算軟體。



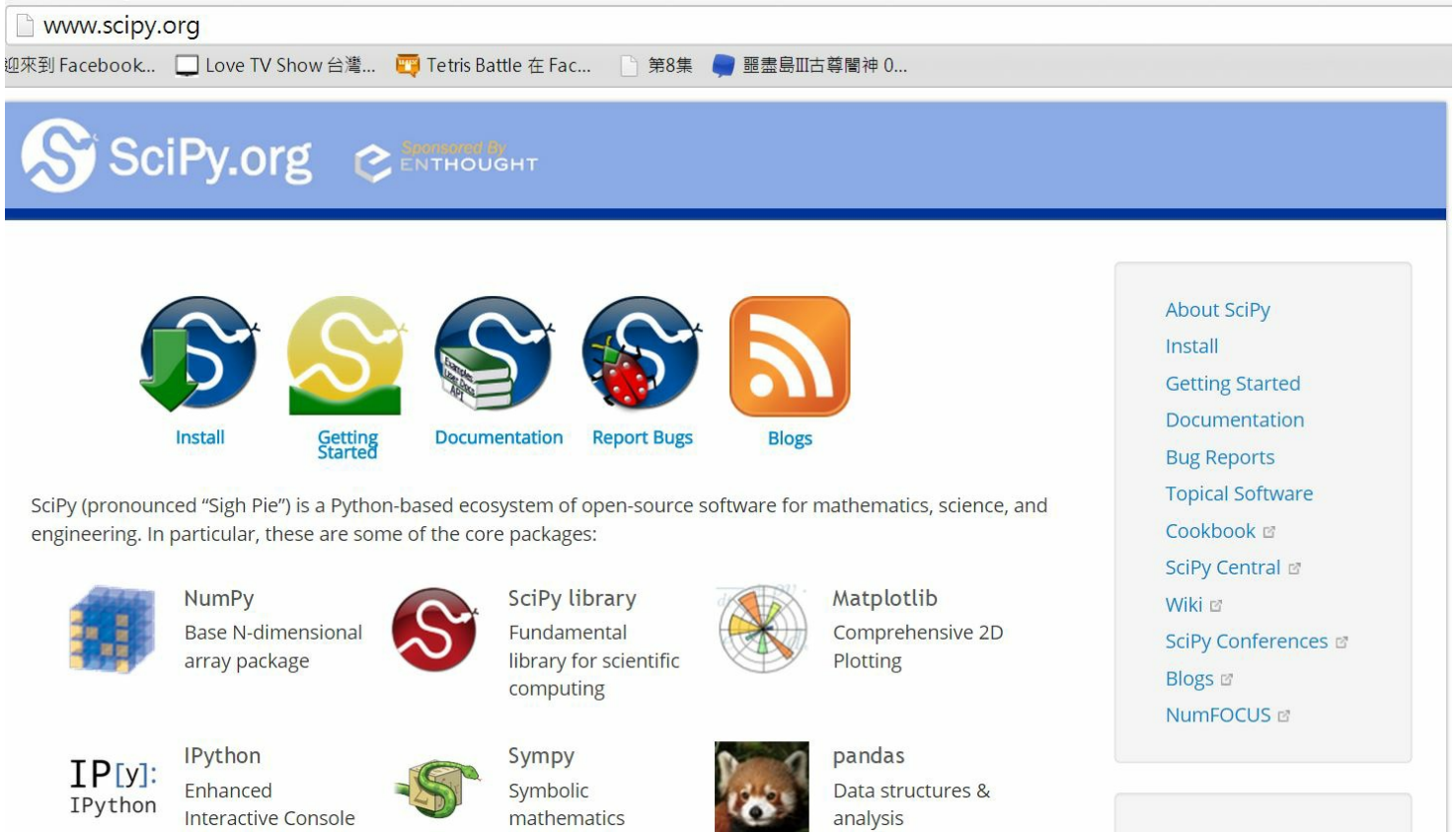
圖、Octave 軟體的官方網站

R 與 SPSS、SAS 的出發點較類似，是從機率統計領域開始建構的，而 Octave 則完全模仿 Matlab 的語法，試圖建構一個與 Matlab 語言相容的科學計算平台，讓 Matlab 的程式資源也可以被 Octave 社群所使用。



圖、R 軟體的官方網站

在開源的科學計算軟體中，R 軟體的使用者似乎是最多的，但是由於 R 採用的程式語言 S3 並非 OpenSource 程式領域的主流，因此也有人試圖用 Python 等語言去建構出科學計算的環境，像是 SciPy 就整合了 numpy、sympy、Matplotlib、



圖、SciPy 的官方網站

雖然已經有了這麼多開放原始碼的科學計算軟體，而且我本身也是 R 的使用者，但是、我仍然感到遺憾！因為我沒辦法找到建構在 JavaScript 語言上的科學計算軟體，所以、我打算自己建造一個，這個計劃就稱為 JsLab (JavaScript Laboratory)。

在下列文章中，我將介紹自己為何要建構 JsLab 專案，如何建構 JsLab 專案，並與大家分享我在建構 JsLab 專案時所學到的一些經驗與心得。

## JsLab -- JavaScript 版的科學計算平台

雖然已經有很多科學計算軟體，像是 MatLab、Mathematica、SPSS、SAS 等等，而且也有像 R、Octave、SciPy 等開放原始碼的科學計算軟體，但就是沒有以 JavaScript 為主的科學計算軟體。

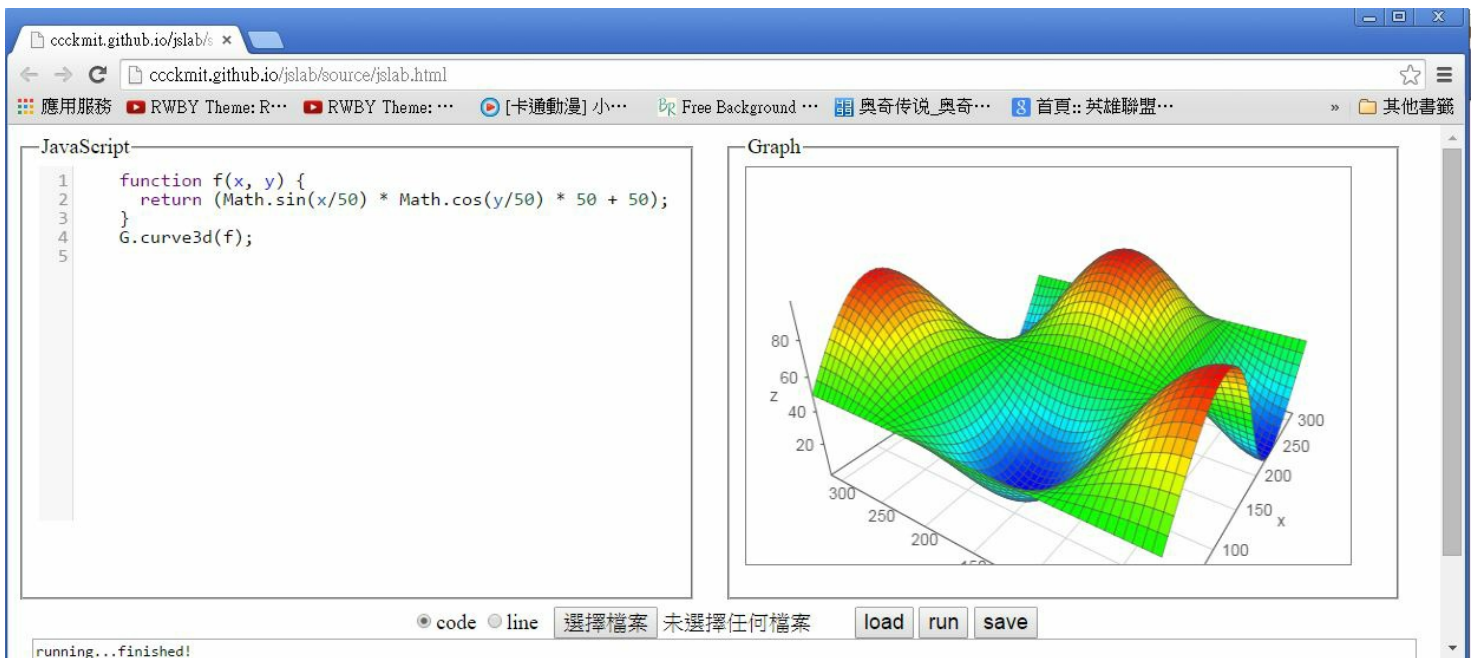
於是我們決定要用 JavaScript 建構一個科學計算軟體，所以、JsLab 計劃就誕生了，關於 JsLab 的用法，可以參考下列的展示影片。

- [展示影片：jslab -- 一個像 R 的 JavaScript 數學實驗環境](#)

下圖是我們已經上傳到 github 上的 JsLab 專案之執行畫面，該程式是一個純粹用 HTML+JavaScript 建構的網頁，網址如下：

- <http://ccckmit.github.io/jslab/source/jslab.html>

您可以連結到該網頁上直接使用該平台，雖然現在還沒有很強大，不過已經可以使用了。



圖、JsLab 的執行畫面 -- 繪製 3D 函數圖

在上述網頁中，我預設在編輯器裏放入了一個展示程式如下，JsLab 會執行該程式後將結果放在訊息視窗，並將繪圖部份顯示在右邊的窗框中。

檔案：curve3D.js

```
function f(x, y) {  
    return (Math.sin(x/50) * Math.cos(y/50) * 50 + 50);  
}  
G.curve3d(f);
```

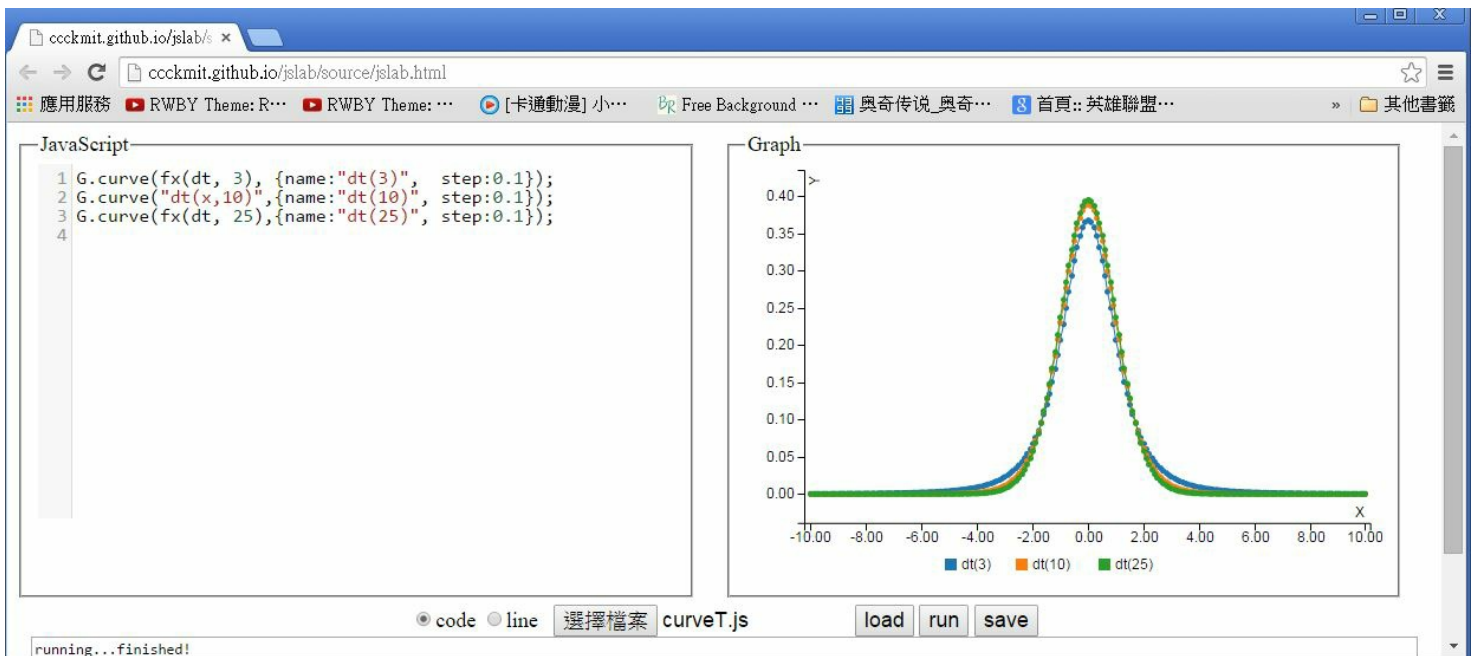
您可以將程式貼到該網頁的程式編輯區，然後按下執行即可得到執行結果。舉例而言、您可以貼上下列程式到編輯區。

檔案：curveT.js

```
G.curve(fx(dt, 3), {name:"dt(3)", step:0.1});  
G.curve("dt(x, 10)", {name:"dt(10)", step:0.1});  
G.curve(fx(dt, 25), {name:"dt(25)", step:0.1});
```

然後按下執行的 run 按鈕，就可以看到下列的曲線圖。





圖、JsLab 的執行畫面 -- 繪製 2D 曲線圖

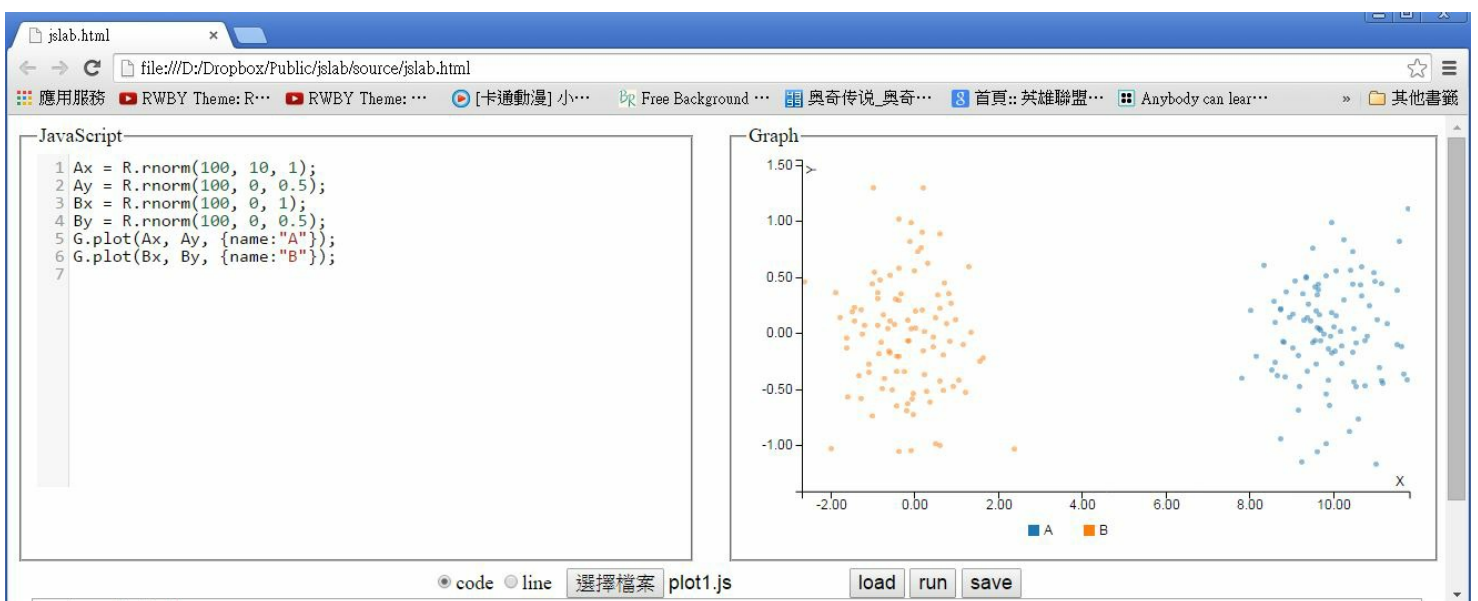
或者您也可以將下列程式貼到程式區，然後按下執行的 run 按鈕，就可以看到所繪出的散點圖。

檔案：plot1.js

```

Ax = R.rnorm(100, 10, 1);
Ay = R.rnorm(100, 0, 0.5);
Bx = R.rnorm(100, 0, 1);
By = R.rnorm(100, 0, 0.5);
G.plot(Ax, Ay, {name:"A"});
G.plot(Bx, By, {name:"B"});

```

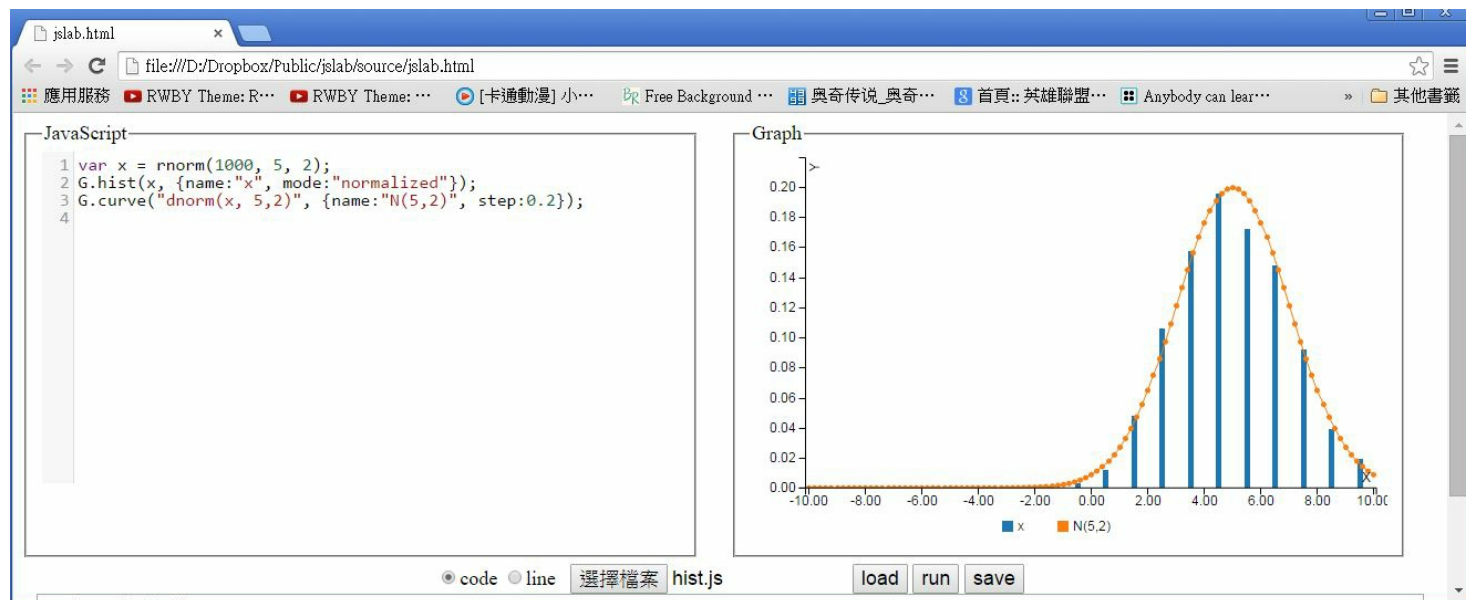


圖、JsLab 的執行畫面 -- 隨機取樣後繪製散點圖

當然、如果您將程式存檔在自己的電腦，那麼您也可以按下「選擇檔案」的按鈕，接著選取想要執行的檔案，就可以將程式上傳並執行，以下是我們透過上傳的方式執行 hist.js 程式的結果。

檔案： hist.js

```
var x = rnorm(1000, 5, 2);
G.hist(x, {name:"x", mode:"normalized"});
G.curve("dnorm(x, 5,2)", {name:"N(5,2)", step:0.2});
```



圖、JsLab 的執行畫面 -- 隨機取樣後繪製統計長條圖

如果您並不想使用瀏覽器介面，也不需要繪製圖形，那麼您也可以採用 node.js 的命令列執行方式，直接引用 JsLab 背後的函式庫，像是 R.js 進行科學計算的動作，以下是我們在 node.js 中使用 R.js 進行隨機抽樣的一個範例。

檔案： sample.js

```
var U = require("../source/U");
U.use("../source/R", "R");

log("====二項分布測試====");
log("pbinom(7, 12, 0.4)=" + pbinom(7, 12, 0.4)); // > pbinom(7, 12, 0.4) [1]
] 0.9426901
log("qbinom(0.9, 12, 0.4)=" + qbinom(0.9, 12, 0.4)); // > qbinom(0.9, 12, 0.4) [1] 7
log("qbinom(0.95, 12, 0.4)=" + qbinom(0.95, 12, 0.4)); // > qbinom(0.95, 12, 0.4) [1] 8

log("====常態分布抽樣====");
log("rnorm=" + rnorm);
log("rnorm(5, 2, 1)=" + str(rnorm(5, 2, 1)));
```

```

var y = rnorm(25, 0, 2);
log("y="+str(y));

log("=====-模擬丟銅板 100 次=-=====");
log("sample([正, 反], 100)="+sample(["正", "反"], 100));

log("=====-模擬擲骰子 100 次=-=====");
log("sample([1, 2, 3, 4, 5, 6], 100)="+sample([1, 2, 3, 4, 5, 6], 100));

```

執行結果

```

D:\Dropbox\Public\jslab\test>node sample.js
use ../source/R name=R
use ../js/jstat.js name=jStat
=====-二項分布測試=-=====-
pbinom(7, 12, 0.4)=0.9426900787200003
qbinom(0.9, 12, 0.4)=7
qbinom(0.95, 12, 0.4)=8
=====-常態分布抽樣=-=====-
rnorm=function (n, mean, sd) { return R.calls(n, jStat.normal.sample, mea
n, sd);
}
rnorm(5, 2, 1)=[2.0748, 2.3562, 3.014, 3.6833, 4.1393]
y=[-0.3756, 1.2935, -1.9635, 0.5446, 0.6514, 2.7079, -1.2434, -4.8451, 1.0937, 2.5
094, 0.1
423, 4.6915, 1.3294, -1.3275, 1.0153, 2.5058, -2.5361, 1.7784, 3.908, -2.2349, 0.18
6, -2.17
17, -1.4398, 0.0423, 1.2014]
=====-模擬丟銅板 100 次=-=====-
sample([正, 反], 100)=正, 正, 反, 反, 正, 正, 反, 正, 反, 反, 正, 正, 反, 反, 正, 正, 正,
反, 反,
正, 反, 正, 反, 反, 反, 反, 正, 正, 正, 反, 反, 反, 正, 反, 正, 反, 反, 正, 正, 正, 反, 正,
反, 反
, 正, 反, 反, 正, 正, 正, 正, 正, 正, 正, 正, 正, 反, 反, 正, 正, 反, 反, 反, 反, 反, 正, 反, 反
, 正,
反, 正, 反, 反, 正, 反, 正, 反, 反, 反, 正, 正, 反, 反, 正, 正, 正, 反, 正, 反, 反, 正, 正, 反,
反, 反
, 正

```

```
=====模擬擲骰子 100 次=====
```

```
sample([1, 2, 3, 4, 5, 6], 100)=3, 2, 4, 6, 5, 2, 4, 4, 5, 5, 1, 5, 1, 1, 2, 1, 4, 1, 1, 2, 1, 4, 1,  
6, 1, 4, 6  
, 5, 1, 4, 1, 6, 1, 1, 2, 6, 1, 5, 1, 3, 3, 1, 3, 4, 2, 5, 6, 4, 3, 5, 2, 4, 3, 1, 2, 1, 6, 3, 2, 4, 1, 2, 1,  
4, 2, 6, 3  
, 5, 4, 5, 4, 4, 1, 4, 2, 6, 2, 4, 5, 2, 3, 5, 1, 3, 3, 2, 3, 2, 1, 3, 5, 3, 2, 5, 3, 6, 2, 4, 1, 2
```

另外、目前我們也還持續的在增加 JsLab 函式庫的功能，像是我們正在為 R.js 加上各種統計檢定的功能，以下是採用 node.js 環境執行檢定的一些案例。

由於 jStat 函式庫並沒有支援這些統計檢定的函數，因此筆者只好自行撰寫，為了撰寫這些檢定的程式，我甚至將 R 的原始碼給翻了出來，網址如下：

- <https://github.com/SurajGupta/r-source/tree/master/src/library/stats/R>

以下是我們對這些檢定功能的一些測試範例，大部分的功能都有對應的 R 操作與執行結果，這樣我們就能利用 R 軟體來驗證我們所寫的檢定函數是否正確了。

檔案：rtest.js

```
var U = require("../source/U");  
U.use("../source/R", "R");  
  
// x = rnorm(10, 5, 2)  
var x = [7.169890 , 2.188864 , 2.963868 , 7.790631 , 2.474261 , 7.694849 , 1.  
585007 , 4.087697 , 3.051643 , 4.697559];  
// y = rnorm(10, 4, 2)  
var y = [4.9627295, 6.0336209, -0.4610221, 7.3744023, 2.4804347, 7.2053190, 3.  
5558563, 3.6505476, 2.2200754, 5.3021459];  
// py = x + rnorm(10, 1, 2)  
var py= [9.829046 , 2.491387 , 6.037504 , 5.709755 , 5.461208 , 7.345603 , 3.  
040538 , 4.856838 , 3.195437 , 7.079105];  
  
log("x="+str(x));  
log("y="+str(y));  
  
ttest({x:x, mu:6, alpha:0.05, op:"="}).report();  
  
/*  
> t.test(x, mu=6, alpha=0.05)
```

### One Sample t-test

```
data: x
t = -2.1732, df = 9, p-value = 0.05781
alternative hypothesis: true mean is not equal to 6
95 percent confidence interval:
 2.674155 6.066699
sample estimates:
mean of x
 4.370427 */

ttest({x:x, mu:6, alpha:0.05, op:"<"}).report();

/*
> t.test(x, mu=6, alternative="greater")
```

### One Sample t-test

```
data: x
t = -2.1732, df = 9, p-value = 0.9711
alternative hypothesis: true mean is greater than 6
95 percent confidence interval:
 2.995873      Inf
sample estimates:
mean of x
 4.370427
*/

ttest({x:x, mu:6, alpha:0.05, op:">"}).report();

/*
> t.test(x, mu=6, alternative="less")
```

### One Sample t-test

```
data: x
t = -2.1732, df = 9, p-value = 0.02891
```

alternative hypothesis: true mean is less than 6

95 percent confidence interval:

-Inf 5.744981

sample estimates:

mean of x

4.370427

\*/

```
ztest({x:x, mu:6, sd:2.5, alpha:0.05, op:"="}).report();
```

```
ttest({x:x, y:y, mu:1, alpha:0.05, varequal:true, op:"="}).report();
```

/\*

```
> t.test(x, y, mu=1, conf.level=0.95, var.equal=T, alternative="two.sided");
```

#### Two Sample t-test

data: x and y

t = -0.8012, df = 18, p-value = 0.4335

alternative hypothesis: true difference in means is not equal to 1

95 percent confidence interval:

-2.122363 2.398395

sample estimates:

mean of x mean of y

4.370427 4.232411 \*/

```
ttest({x:x, y:py, mu:-1, alpha:0.05, paired:true, op:"="}).report();
```

/\*

```
> t.test(x, py, mu=-1, conf.level=0.95, paired=T)
```

#### Paired t-test

data: x and py

t = -0.252, df = 9, p-value = 0.8067

alternative hypothesis: true difference in means is not equal to -1

```

95 percent confidence interval:
-2.33885689  0.07042649
sample estimates:
mean of the differences
      -1.134215 */

ttest({x:x, y:y, mu:1, alpha:0.05, op:"="}).report();

/*
> t.test(x, y, mu=1, conf.level=0.95, alternative="two.sided");

      Welch Two Sample t-test

data:  x and y
t = -0.8012, df = 17.985, p-value = 0.4335
alternative hypothesis: true difference in means is not equal to 1
95 percent confidence interval:
-2.122495  2.398527
sample estimates:
mean of x mean of y
 4.370427  4.232411
*/

ftest({x:x, y:y}).report();

/*
> var.test(x, y)

      F test to compare two variances

data:  x and y
F = 0.9445, num df = 9, denom df = 9, p-value = 0.9337
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.2346094  3.8026974
sample estimates:
ratio of variances

```

0.9445362

```
*/  
  
var vx = [175, 176, 173, 175, 174, 173, 173, 176, 173, 179];  
vartest({x:vx, sd:2, alpha:0.05, op:"="}).report();  
// R 軟體沒有此函數，測試請看湯銀才 143 頁  
// 信賴區間 (1.793, 12.628)
```

```
binomtest({x:7, n:12, p:0.4, op:">"}).report();
```

```
/*  
> binom.test(x=7, n=12, p=0.4, alternative="less")
```

Exact binomial test

```
data: 7 and 12  
number of successes = 7, number of trials = 12, p-value = 0.9427  
alternative hypothesis: true probability of success is less than 0.4  
95 percent confidence interval:  
 0.0000000 0.8189752  
sample estimates:  
probability of success  
 0.5833333
```

```
*/  
  
binomtest({x:7, n:12, p:0.4, op:"<"}).report();  
/*  
> binom.test(x=7, n=12, p=0.4, alternative="greater")
```

Exact binomial test

```
data: 7 and 12  
number of successes = 7, number of trials = 12, p-value = 0.1582  
alternative hypothesis: true probability of success is greater than 0.4  
95 percent confidence interval:  
 0.3152378 1.0000000  
sample estimates:
```



```
probability of success
```

```
0.5833333
```

```
*/
```

```
binomtest({x:7, n:12, p:0.4}).report(); // 有誤, p-value 與 R 不同
```

```
/*
```

```
> binom.test(x=7, n=12, p=0.4)
```

```
Exact binomial test
```

```
data: 7 and 12
```

```
number of successes = 7, number of trials = 12, p-value = 0.2417 ==> R.js
```

```
??? error : pvalue : 0.1146
```

```
alternative hypothesis: true probability of success is not equal to 0.4
```

```
95 percent confidence interval:
```

```
0.2766697 0.8483478
```

```
sample estimates:
```

```
probability of success
```

```
0.5833333
```

```
*/
```

```
proptest({x:91, n:100, p:0.9, correct:false}).report();
```

```
/* 1-sample proportions test without continuity correction
```

```
data: 91 out of 100, null probability 0.9
```

```
X-squared = 0.1111, df = 1, p-value = 0.7389
```

```
alternative hypothesis: true p is not equal to 0.9
```

```
95 percent confidence interval:
```

```
0.8377379 0.9519275
```

```
sample estimates:
```

```
p
```

```
0.91 */
```

```

proptest({x:23, n1:102, y:25, n2:135, correct:false}).report();

/*
> success = c(23, 25)
> total = c(102, 135)
> prop.test(success, total)

      2-sample test for equality of proportions with continuity correct
ion

data:  success out of total
X-squared = 0.3615, df = 1, p-value = 0.5477
alternative hypothesis: two.sided
95 percent confidence interval:
 -0.07256476  0.15317478
sample estimates:
   prop 1    prop 2 
0.2254902 0.1851852 
*/

proptest({x:8, n1:100, y:12, n2:200, op:"<", correct:false}).report();

/*
> prop.test(c(8, 12), c(100, 200), alternative="greater", correct=F)

      2-sample test for equality of proportions without continuity
correction

data:  c(8, 12) out of c(100, 200)
X-squared = 0.4286, df = 1, p-value = 0.2563
alternative hypothesis: greater
95 percent confidence interval:
 -0.03248088  1.00000000 => R.js ??? [-0.0303, Infinity]
sample estimates:
prop 1 prop 2 
 0.08  0.06

```

## 執行結果

```
D:\Dropbox\Public\jslab\test>node rtest
use ../source/R name=R
use ../js/jstat.js name=jStat
x=[7.1699, 2.1889, 2.9639, 7.7906, 2.4743, 7.6948, 1.585, 4.0877, 3.0516, 4.6976]
y=[4.9627, 6.0336, -0.461, 7.3744, 2.4804, 7.2053, 3.5559, 3.6505, 2.2201, 5.3021]
===== report =====
name      : "ttest(X)"
h         : "H0:mu=6"
alpha    : 0.05
op       : "="
pvalue   : 0.0578
ci       : [2.6742, 6.0667]
df       : 9
mean     : 4.3704
sd       : 2.3712
===== report =====
name      : "ttest(X)"
h         : "H0:mu<6"
alpha    : 0.05
op       : "<"
pvalue   : 0.9711
ci       : [2.9959, Infinity]
df       : 9
mean     : 4.3704
sd       : 2.3712
===== report =====
name      : "ttest(X)"
h         : "H0:mu>6"
alpha    : 0.05
op       : ">"
pvalue   : 0.0289
ci       : [-Infinity, 5.745]
df       : 9
mean     : 4.3704
```

sd : 2.3712

===== report =====

name : "ztest(X)"

h : "H0:mu=6 when sd=2.5"

alpha : 0.05

op : "="

pvalue : 0.0393

ci : [2.9008, 5.8401]

df : 10

===== report =====

name : "ttest(X, Y, mu=1, varequal=true) (pooled)"

h : "H0:mu1=mu2"

alpha : 0.05

op : "="

pvalue : 0.4335

ci : [-2.1224, 2.3984]

df : 18

mean : "mean(x)=4.3704 mean(y)=4.2324"

sd : "sd(x)=2.3712 sd(y)=2.4399"

===== report =====

name : "ttest(x, y, mu=-1, paired=true)"

h : "H0:mu1=mu2"

alpha : 0.05

op : "="

pvalue : 0.8067

ci : [-2.3389, 0.0704]

df : 9

mean : "mean(x-y)=-1.1342"

sd : "sd(x-y)=1.684"

===== report =====

name : "ttest(x, y, mu=1, varequal=false), Welch t-test"

h : "H0:mu1=mu2"

alpha : 0.05

op : "="

pvalue : 0.4335

ci : [-2.1225, 2.3985]

df : 17.9854

mean : "mean(x)=4.3704 mean(y)=4.2324"

sd : "sd(x)=2.3712 sd(y)=2.4399"

=====  
report  
=====

name : "ftest(X, Y)"

h : "H0:  $\sigma_1 / \sigma_2 = 1$ "

alpha : 0.05

op : "="

pvalue : 0.9337

ci : [0.2346, 3.8027]

df : [9, 9]

ratio : 0.9445

=====  
report  
=====

name : "chisqtest(X)"

h : "H0:  $\sigma_1 = \sigma_2$ "

alpha : 0.05

op : "="

pvalue : 0.9644

ci : [1.7926, 12.6278]

df : 9

=====  
report  
=====

name : "binomtest({x:7, n:12, p:0.4, op:">"})"

h : "H0:  $p > 0.4$ "

alpha : 0.05

op : ">"

pvalue : 0.9427

ci : [0, 0.819]

df : 1

p : 0.5833

=====  
report  
=====

name : "binomtest({x:7, n:12, p:0.4, op:"<"})"

h : "H0:  $p < 0.4$ "

alpha : 0.05

op : "<"

pvalue : 0.1582

ci : [0.3152, 1]

df : 1

p : 0.5833

===== report =====

name : "binomtest({x:7, n:12, p:0.4})"  
h : "H0:p=0.4"  
alpha : 0.05  
op : "="  
pvalue : 0.2417  
ci : [0.2767, 0.8483]  
df : 1  
p : 0.5833

===== report =====

name : "proptest({x:91, n:100, p:0.9, correct:false}), zprop1"  
h : "H0:p=0.9"  
alpha : 0.05  
op : "="  
pvalue : 0.7389  
ci : [0.8377, 0.9519]  
df : 1  
p : 0.91

===== report =====

name : "proptest({x:23, n1:102, y:25, n2:135, correct:false, p:0.5}), zprop2"  
h : "H0:p1-p2=0"  
alpha : 0.05  
op : "="  
pvalue : 0.4446  
ci : [-0.063, 0.1436]  
df : 1  
p : [0.2255, 0.1852]

===== report =====

name : "proptest({x:8, n1:100, y:12, n2:200, op:"<", correct:false, p:0.5}), zprop2"  
h : "H0:p1-p2<0"  
alpha : 0.05  
op : "<"  
pvalue : 0.2563  
ci : [-0.0303, Infinity]

```
df      : 1
p       : [0.08, 0.06]
```

## 「JsLab 科學計算平台」背後的開放原始碼結構

為了要用 JavaScript 建構出科學計算平台，我們創建了以下的 JsLab 開放原始碼專案。

- <https://github.com/ccckmit/jslab/tree/gh-pages>

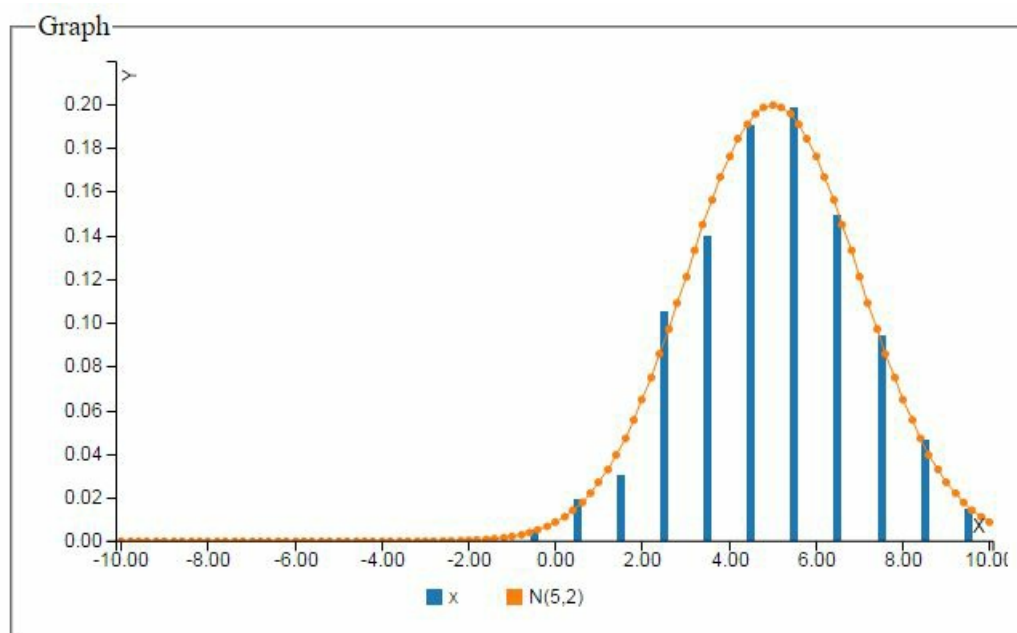
我們大量的採用了 JavaScript 的開放原始碼函式庫，像是在「機率統計」領域採用了 jStat 這個專案，在繪圖領域採用了 d3.js、c3.js、vis.js 等等，這些專案的網址如下。

- <http://d3js.org/>
- <http://c3js.org/>
- <http://visjs.org/>
- <https://github.com/jstat/jstat>

JsLab 科學計算平台的核心，是一組基於 jStat 專案的重新封裝，我們將 jStat 重新封裝成類似 R 函式庫的介面，並且加上了一些 jStat 當中所沒有的功能，特別是統計檢定的部份，形成了 R.js 這個 JavaScript 程式，讓 JavaScript 也能擁有類似 R 軟體的機率統計函式庫。

我們雖然使用了 d3.js 這個繪圖函式庫進行 2D 繪圖，但由於 d3.js 並不容易使用，所以我們採用了 c3.js 這個基於 d3 的延伸套件，簡化 d3 繪圖函式庫的使用，讓我們不需瞭解太多繪圖的細節就能寫出 JsLab 的繪圖函式庫。

舉例而言、下列的圖形就是依靠 c3.js 所繪製的，只不過我們將 c3.js 進一步封裝到 JsLab 的 G.js 這個的繪圖函式庫當中而已。



以下是我們將 c3.js 重新封裝成 G.js 的程式碼片段。

檔案：[G.js](#)

```

var C3G=function() {
  this.g = {
    data: {
      xs: {},
      columns: [ /*["x", 1, 2, 3, 4 ]*/ ],
      type: "line",
      types : {}
    },
    axis: {
      x: {
        label: 'X',
        tick: { fit: false, format:d3.format(".2f") }
      },
      y: { label: 'Y',
        tick : { format: d3.format(".2f") }
      }
    },
    bar: { width: { ratio: 0.9 } },
  };
  this.varcount = 0;
  this.xrange(-10, 10);
  this.step = 1;
  this.setrange = false;
}
....
C3G.prototype.hist = function(x, options) {
  var name = R.opt(options, "name", this.tempvar());
  var mode = R.opt(options, "mode", "");
  var step = R.opt(options, "step", this.step);
  var from = R.opt(options, "from", this.xmin);
  var to   = R.opt(options, "to", this.xmax);
  this.g.data.types[name] = "bar";
  this.g.data.xs[name] = name+"x";
  var xc = R.steps(from+step/2.0, to, step);
  var n = (to-from)/step + 1;
  var count = R.repeats(0, n);
  for (var i in x) {

```



```

    var slot=Math.floor((x[i]-from)/step);
    if (slot>=0 && slot < n)
        count[slot]++;
}
this.g.data.columns.push([name+"x"].concat(xc));
var total = R.sum(count);
if (mode === "normalized")
    count = R.apply(count, function(c) { return (1.0/step)*(c/total); });
;
this.g.data.columns.push([name].concat(count));
}
...
C3G.prototype.show = function() {
    if (typeof(module)===undefined)
        return c3.generate(this.g);
}

```

另外、由於 d3.js 並沒有支援 3D 曲面圖形的繪製，所以我們又引入了 vis.js 這個繪圖函式庫來完成 3D 圖形的繪製工作。

以下是我們將 vis.js 封裝成 G.js 的程式碼片段

檔案：[G.js](#)

```

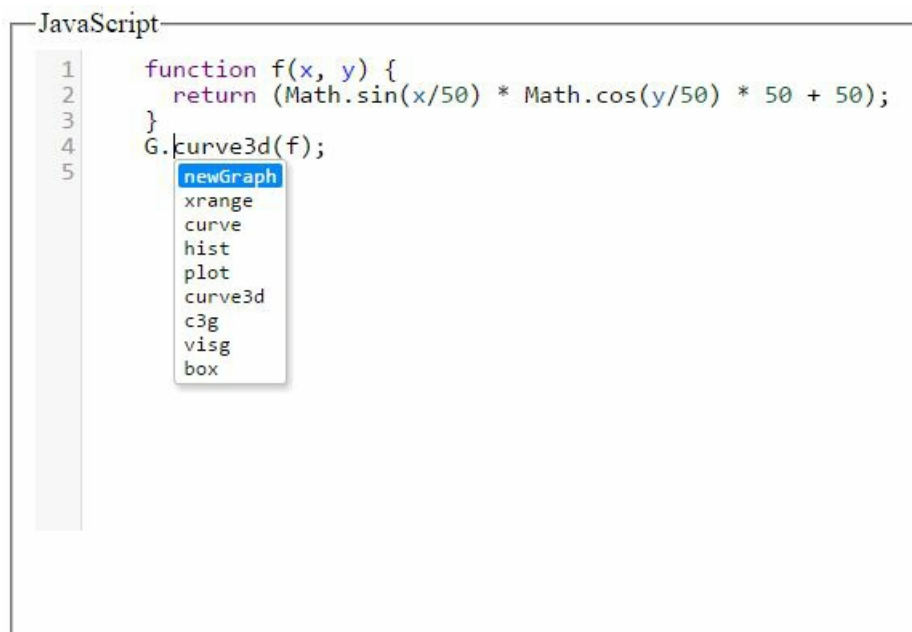
...
VISG.prototype.curve3d = function(f, box) {
    // Create and populate a data table.
    var data = new vis.DataSet();
    // create some nice looking data with sin/cos
    var counter = 0;
    var steps = 50; // number of datapoints will be steps*steps
    var axisMax = 314;
    var axisStep = axisMax / steps;
    for (var x = 0; x < axisMax; x+=axisStep) {
        for (var y = 0; y < axisMax; y+=axisStep) {
            var value = f(x,y);
            data.add({id:counter++, x:x, y:y, z:value, style:value});
        }
    }
    this.graph = new vis.Graph3d(box, data, this.options);
}

```

```
}  
  
...  
G.curve3d = function(f) {  
  G.visg.curve3d(f, G.box);  
}
```

接著、為了讓使用者編輯程式可以更容易，我們採用了這個 JavaScript 開源線上編輯器專案，該專案之援類似微軟 Visual Studio 的 IntelliSense 這樣的函數提示功能，

- <http://codemirror.net/>



```
JavaScript  
1 function f(x, y) {  
2   return (Math.sin(x/50) * Math.cos(y/50) * 50 + 50);  
3 }  
4 G.curve3d(f);  
5
```

- newGraph
- xrange
- curve
- hist
- plot
- curve3d
- c3g
- visg
- box

圖、JsLab 當中的程式碼上色與 IntelliSense 功能

以下是我們將 codemirror 封裝成 E.js (Editor) 的完整程式碼。

檔案：[E.js](#)

```
"use strict";  
  
U.use("../js/codemirror/lib/codemirror.js", "CodeMirror");  
U.use("../js/codemirror/addon/hint/show-hint.js");  
U.use("../js/codemirror/addon/hint/javascript-hint.js");  
U.use("../js/codemirror/mode/javascript/javascript.js");  
  
E = {};  
  
if (typeof module !== "undefined") module.exports = E;
```

```
E.editor = null;

E.loadEditor = function (codebox) {
  E.codebox = codebox;
  E.editor = CodeMirror.fromTextArea(codebox, {
    lineNumbers: true,
    extraKeys: {"Ctrl-." : "autocomplete"},
    lineWrapping: true,
    styleActiveLine: true,
    mode: {name: "javascript", globalVars: true}
  });

  E.editor.on('update', function (instance) {
    E.codebox.value = instance.getValue();
  });
}
```

未來、我們可能會進一步整合更多的函式庫，目前預計在「矩陣運算領域」會採用了 jStat 與 numeric.js 等函式庫，而在數位訊號語音處理領域可能會採用 DSP.js，影像處理領域可能採用 CamanJS 等等，這些函式庫的網址如下：

- <http://numericjs.com/>
- <https://github.com/corbanbrook/dsp.js/>
- <http://camanjs.com/>

目前、網路上的 JavaScript 的函式庫還持續的急速增長中，我們相信之後會有更多更好用的 JavaScript 科學計算函式庫出現，我們只要能將這些好用的函式庫整合進來，就能創建一個完整的科學計算平台了，而這也正是 JsLab 計劃所想要達成的目標。

由於 JavaScript 是瀏覽器當中唯一能使用的官方語言，因此我們認為未來 JavaScript 的發揮空間將會越來越大，就像 Jeff Atwood 在 2007 年所提出的 Atwood's Law 所說的：

Any application that can be written in JavaScript, will eventually be written in JavaScript.

換句話說 -- 「任何可以寫成 JavaScript 的應用程式，最後都會被寫成 JavaScript」。

如果 Atwood's Law 成立的話，那基於 JavaScript 的科學計算平台就應該要出現啊！

我們正在以行動來實現 Atwood 的話，這個行動的代號就是 JsLab。

## 參考文獻

- **Atwood定律**：“任何可以使用JavaScript来编写的的应用，最终会由JavaScript编写。”

## R.js -- 從 jStat 延伸出的開源 JavaScript 機率統計框架

我自從開始用 R 學習機率統計之後，就覺得這樣的科學計算平台真的很棒。

但可惜的是、我現在所使用的主力語言是 JavaScript，因為 JavaScript 是瀏覽器的唯一語言，而且有了 node.js 這樣的平台之後，可以通吃客戶端與伺服器兩方的應用。

於是我想要用 JavaScript 創造出一個類似 R 的環境，並且可以在 Web 上執行，所以就創造了 jsLab 專案。

但是、為了讓 jsLab 能支援那些機率統計功能，我必須尋找用 JavaScript 語言撰寫的機率統計函式庫。

在 2012 年我就注意到了 jStat 這個支援機率模型的函式庫，當時這個函式庫還有專屬的網站，但是現在這個函式庫連網站都不見了，還好在 github 裏還有一份原始碼，網址如下：

- <https://github.com/jstat/jstat>

jStat 在機率模型的部分支援還算完整，但是在統計檢定的部分就相當薄弱，雖然也支援矩陣運算等功能，但是在 JavaScript 語言上，jStat 的矩陣運算支援並不算特別好的。

因此、我們決定將 jStat 重新包裝，成為一個新的 javascript 檔案，稱為 R.js，這是 jsLab 專案的主要程式碼之一，R.js 的網址如下。

- <https://github.com/ccckmit/jslab/blob/gh-pages/source/R.js>

您可以看到在 R.js 檔案裏，能夠呼叫 jStat 完成的部分，我們都盡可能的交給 jStat 來做，而 jStat 在這部份也確實做得很不錯。

檔案：R.js

```
...
// 均等分布
R.runif=function(n, a, b) { return R.calls(n, jStat.uniform.sample, a, b); }
R.dunif=function(x, a, b) { return jStat.uniform.pdf(x, a, b); }
R.punif=function(q, a, b) { return jStat.uniform.cdf(q, a, b); }
R.qunif=function(p, a, b) { return jStat.uniform.inv(p, a, b); }
// 常態分布
R.rnorm=function(n, mean, sd) { return R.calls(n, jStat.normal.sample, mean, sd); }
R.dnorm=function(x, mean, sd) { return jStat.normal.pdf(x, mean, sd); }
R.pnorm=function(q, mean, sd) { return jStat.normal.cdf(q, mean, sd); }
// R.qnorm=function(p, mean, sd) { return R.q2x(p, function(q) { return R.pnorm(q, mean, sd); }); }
R.qnorm=function(p, mean, sd) { return jStat.normal.inv(p, mean, sd); }
// 布瓦松分布
```

```

R.rpois=function(n, l) { return R.calls(n, jStat.poisson.sample, l); }
R.dpois=function(x, l) { return jStat.poisson.pdf(x, l); }
R.ppois=function(q, l) { return jStat.poisson.cdf(q, l); }
R.qpois=function(p, l) { return jStat.poisson.inv(p, l); }

// F 分布
R.rf=function(n, df1, df2) { return R.calls(n, jStat.centralF.sample, df
1, df2); }
R.df=function(x, df1, df2) { return jStat.centralF.pdf(x, df1, df2); }
R.pf=function(q, df1, df2) { return jStat.centralF.cdf(q, df1, df2); }
R.qf=function(p, df1, df2) { return jStat.centralF.inv(p, df1, df2); }
// T 分布
R.rt=function(n, dof) { return R.calls(n, jStat.studentt.sample, dof); }
R.dt=function(x, dof) { return jStat.studentt.pdf(x, dof); }
R.pt=function(q, dof) { return jStat.studentt.cdf(q, dof); }
R.qt=function(p, dof) { return jStat.studentt.inv(p, dof); }
...

```

不過、在離散的機率分布上面，jStat 就支援的比較不好，而且沒有支援像 inv 這類的函數，於是我們就得自己來補足，以下是我們用 jStat 與自己寫的程式所合成的一些離散分布原始碼。

```

...
R.sample1=function(a, p) {
  var r = Math.random();
  var u = R.repeats(1.0, a.length);
  p = R.def(p, R.normalize(u));
  var psum = 0;
  for (var i in p) {
    psum += p[i];
    if (psum > r)
      return a[i];
  }
  return null;
}

R.sample=function(x, n, p) { return R.calls(n, R.sample1, x, p); }

// 二項分布

```

```

R.rbinom=function(n, N, p) { return R.calls(n, jStat.binomial.sample, N,
  p); }
R.dbinom=function(x, N, p) { return jStat.binomial.pdf(x, N, p); }
R.pbinom=function(k, N, p) { return jStat.binomial.cdf(k, N, p); }
R.qbinom=function(q, N, p) {
  for (var i=0; i<=N; i++) {
    if (R.pbinom(i, N, p) > q) return i;
  }
  return N;
}

// 負二項分布
R.rnbinom=function(n, N, p) { return R.calls(n, jStat.negbin.sample, N,
  p); }
R.dnbinom=function(x, N, p) { return jStat.negbin.pdf(x, N, p); }
R.pnbinom=function(k, N, p) { return jStat.negbin.cdf(k, N, p); }
// R.qnbinom=function(p, N, q) { return jStat.negbin.inv(p, N, q); }
R.qnbinom=function(q, N, p) {
  for (var i=0; i<N; i++) {
    if (R.pnbinom(i, N, p) > q) return i;
  }
  return N;
}
...

```

另外、由於 jStat 在統計檢定方面的函數也很薄弱，所以我們撰寫了以下這個檢定的抽象函數，實作時只要補足「q2p, o2q, h, df」等函數，就可以做出一個檢定功能了。

```

...
R.test = function(o) { // name, D, x, mu, sd, y, alpha, op
  var D      = o.D;
  var x      = o.x;
  var alpha  = R.opt(o, "alpha", 0.05);
  o.op      = R.opt(o, "op", "=");

  var pvalue, interval;
  var q1     = D.o2q(o); // 單尾檢定的 pvalue

```

```

if (o.op === "=") {
  if (q1>0.5) q1 = 1-q1; // (q1>0.5) 取右尾, 否則取左尾。
  pvalue= 2*q1; // 對稱情況: 雙尾檢定的 p 值是單尾的兩倍。
  interval = [D.q2p(alpha/2, o, "L"), D.q2p(1-alpha/2, o, "R")];
} else {
  if (o.op === "<") { // 右尾檢定 H0: q < 1-alpha,
    interval = [ D.q2p(alpha, o, "L"), Infinity ];
    pvalue = 1-q1;
  }
  if (o.op === ">") { // 左尾檢定 H0: q > alpha
    interval=[-Infinity, D.q2p(1-alpha, o, "R")];
    pvalue = q1;
  }
}

return { name: o.name,
        h: D.h(o),
        alpha: alpha,
        op: o.op,
        pvalue: pvalue,
        ci : interval,
        df : D.df(o),
        report: function() { R.report(this) }
        };
}
...

```

舉例而言，以下這個物件 t1 實作了 R.test 中所需要的函數，因此我們就可以透過「 o.D = t1; t = R.test(o); 這兩行指令呼叫 R.test 函數，完成單樣本的 t 檢定工作。

```

var t1 = { // 單樣本 T 檢定 t = (X-mu)/(S/sqrt(n))
  h:function(o) { return "H0:mu"+o.op+o.mu; },
  o2q:function(o) {
    var x = o.x, n = x.length;
    // t=(X-mu)/(sd/sqrt(n))
    var t = (R.mean(x)-o.mu)/(R.sd(x)/Math.sqrt(n));
    return R.pt(t, n-1);
  },

```

```

//  $P(X-\mu/(S/\sqrt{n}) < t) = q$  ; 信賴區間  $P(T < q)$ 
//  $P(\mu > X-t*S/\sqrt{n}) = q$  ; 這反而成了右尾檢定，所以左尾與右尾確實會
反過來
q2p:function(q, o) {
  var x = o.x, n = x.length;
  return R.mean(x) + R.qt(q, n-1) * R.sd(x) / Math.sqrt(n);
},
df:function(o) { return o.x.length-1; }
}
...

```

```

R.ttest = function(o) {
  var t;
  if (typeof o.y === "undefined") {
    o.name = "ttest(X)";
    o.D = t1;
    t = R.test(o);
    t.mean = R.mean(o.x);
    t.sd = R.sd(o.x);
  } else {

```

目前我們已經在 R.js 中加入了大部份的機率分布、還有基本的「有母數」檢定功能，之後會再加入「無母數檢定的功能」。

另外、我們也已經加入了基本的圖表繪製功能在 G.js 當中，於是形成了 jsLab 專案的基本架構，希望之後還能找到更多更棒的 JavaScript 科學計算函式庫，讓 JavaScript 語言也能成為科學計算的良好平台。

## 參考文獻

- <https://github.com/jstat/jstat>
- <https://github.com/ccckmit/jslab/tree/gh-pages>



# 程式人文集

## d3.js -- 互動式繪圖框架

d3.js 是一個在瀏覽器裏使用的互動式繪圖框架，使用 HTML、CSS、JavaScript 與 SVG (Scalable Vector Graphics) 等技術。

d3.js 專案起始於 2011 年，是從 [Protovis](#) 專案修改過來的，通常我們會用 D3 來產生 SVG 或 CSS 的繪圖結果，在瀏覽器上檢視時還可以利用 SVG 或 CSS 與使用者進行互動。

d3.js 的用法有點像 jQuery，都是透過選擇器來進行選取後操作的，舉例而言、下列指令可以選出所有 p 標記的節點並將顏色修改為 lavender (淡紫色、熏衣草)。

```
d3.selectAll("p")
  .style("color", "lavender");
```

我們可以透過「標記 tag、類別 class、代號 identifier、屬性 attribute、或位置 place 來選取節點，然後進行新增、刪除、修改等動作，然後透過設定 CSS 的轉移 (transition) 屬性，讓繪圖的結果可以和使用者進行互動，舉例而言、以下程式就會讓網頁裡的 p 標記節點逐漸地改變為紫色。(d3.js 預設的改變速度為 250ms 完成轉換)

```
d3.selectAll("p")
  .transition()
  .style("color", "pink");
```

由於 SVG 裏的標記也是 HTML 的一部分，d3 的指令也可以選取 SVG 裏的內容，以下來自 [Mike Bostock](#) 網站的 [範例](#) 顯示了這個狀況：

```
<svg width="720" height="120">
  <circle cx="40" cy="60" r="10"></circle>
  <circle cx="80" cy="60" r="10"></circle>
  <circle cx="120" cy="60" r="10"></circle>
</svg>
...
<script>
var circle = d3.selectAll("circle");
circle.style("fill", "steelblue");
circle.attr("r", 30);
</script>
```

d3.js 的主要 API 包含下列幾類：

- Selections

- Transitions
- Arrays
- Math
- Color
- Scales
- SVG
- Time
- Layouts
- Geography
- Geometry
- Behaviors

而以下的專案則是延伸自 d3.js 的套件，

- [freeDataMap](#) - Company data visualisation tool
- [dimple.js](#) - Flexible axis-based charting API
- [Cubism](#) - Time series visualisation
- [Rickshaw](#) - Toolkit for creating interactive time series graphs
- [NVD3](#) - Re-usable charts for d3
- [Crossfilter](#) - Fast Multidimensional Filtering for Coordinated Views
- [dc.js](#) - Dimensional Charting Javascript Library
- [c3.js](#) - D3-based reusable chart library

甚至還有人專門為 d3.js 寫了一本書，而且這本書還有中文版。

- [網頁互動式資料視覺化：使用D3](#), Scott Murray.

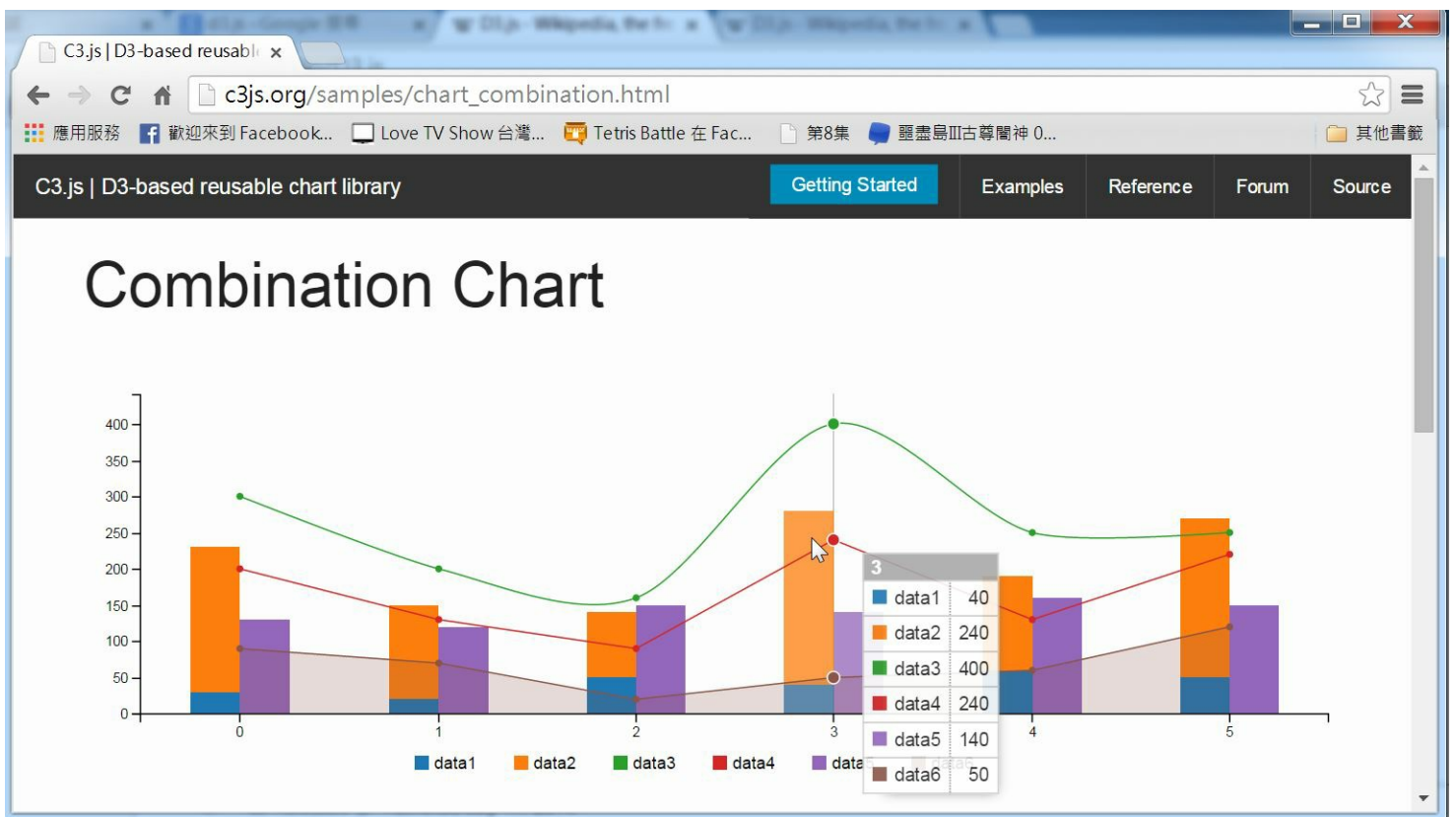
### 參考文獻

- Wikipedia: D3.js -- <http://en.wikipedia.org/wiki/D3.js>
- Mike Bostock -- <http://bost.ocks.org/mike/>
- <http://d3js.org/>
- [D3 Gallery](#)

## c3.js -- 基於 d3.js 的簡易繪圖框架

雖然 d3.js 很強大，但是卻並不容易使用，如果我們只是要畫一些簡易的圖形，可以採用延伸自 d3.js 的 c3.js。

C3.js 的使用非常的簡單，而且互動性很強大。舉例而言，以下是 C3.js 的一個範例，您將滑鼠游標移到圖形上，會顯示對應軸線的資料表格，這讓使用者可以很清楚的看到圖形所對應的數據，這是非常具有互動性的顯示方式。



圖、C3.js 的一個繪圖範例

您可以點選下列網址試試看這個範例，應該可以感覺到 C3.js 好用的地方。

- [http://c3js.org/samples/chart\\_combination.html](http://c3js.org/samples/chart_combination.html)

而且、要產生上述的圖形，也只需要短短幾行簡單的資料與程式，其程式碼如下所示：

```
var chart = c3.generate({
  data: {
    columns: [
      ['data1', 30, 20, 50, 40, 60, 50],
      ['data2', 200, 130, 90, 240, 130, 220],
      ['data3', 300, 200, 160, 400, 250, 250],
      ['data4', 200, 130, 90, 240, 130, 220],
      ['data5', 130, 120, 150, 140, 160, 150],
      ['data6', 90, 70, 20, 50, 60, 120],
    ],
  },
  type: 'bar',
  types: {
    data3: 'spline',
    data4: 'line',
    data6: 'area',
  },
},
```

```
    groups: [
      ['data1', 'data2']
    ]
  }
});
```

從以上的範例中，您應該可以看到 C3 這個架構的優點，相較於 D3 而言，C3 容易使用多了。

但是，C3 並沒有辦法完全發揮 D3 的功能，像是筆者就沒看到 C3 具有任何可以繪製統計 box chart 的功能，因此對於某些較少見的情況而言，我們還是得直接採用 D3，另外 C3 的文件說明並不完整，這是筆者所看到的 C3 框架之缺陷。

### 參考文獻

- <http://c3js.org/>

## Vis.js -- 另一個強大的 JavaScript 繪圖函式庫

雖然前述的 d3.js 與 c3.js 可以做到繪圖功能，但是這兩個函式庫強調的是互動性介面，而不是繪圖功能的部份。

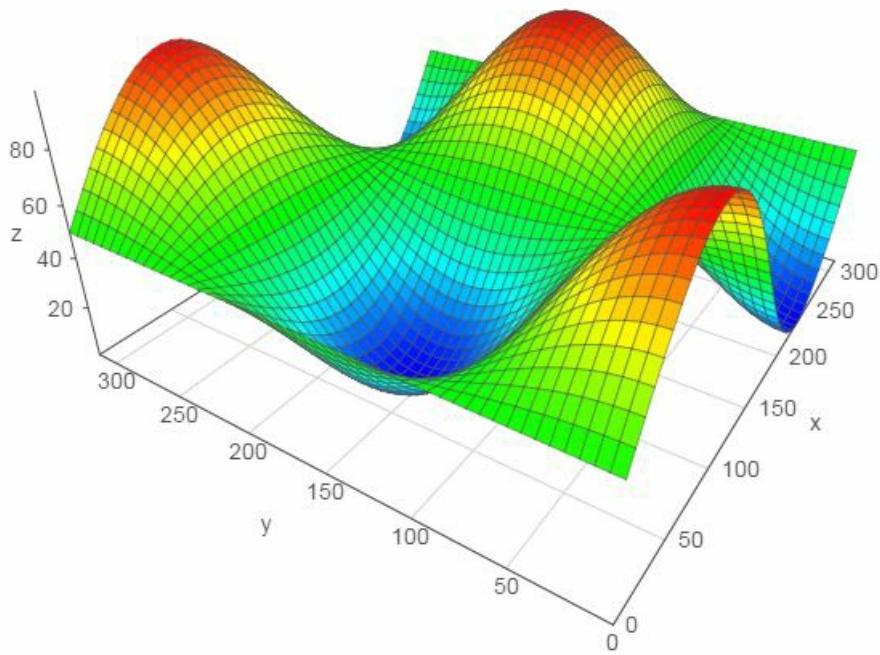
於是，我找到了 vis.js 這個專注於繪圖的函式庫，這個函式庫雖然再互動性上表現得沒有 c3 那麼好，但是在繪圖功能上卻非常的完整，該有的圖型幾乎都有，特別是有關「3D 地形圖」、「網路線圖」和的部份，是 C3 所不具備的功能，因此我們拿 vis.js 來繪製這兩類的圖形。

您可以點選下列連結，看看 vis.js 的眾多範例，相信您會對這個繪圖框架感到印象深刻的。

- <http://visjs.org/#example>

舉例而言，以下網頁是用來繪製 3D 地形圖的完整原始碼，

網址：[http://visjs.org/examples/graph3d/example01\\_basis.html](http://visjs.org/examples/graph3d/example01_basis.html)



圖、用 Vis.js 繪製 3D 圖形

以下是上述範例的完整 HTML 檔案。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
  <title>Graph 3D demo</title>

  <style>
    body {font: 10pt arial;}
  </style>

  <script type="text/javascript" src="../../dist/vis.js"></script>

  <script type="text/javascript">
    var data = null;
    var graph = null;

    function custom(x, y) {
      return (Math.sin(x/50) * Math.cos(y/50) * 50 + 50);
    }

    // Called when the Visualization API is loaded.
```

```
function drawVisualization() {
  // Create and populate a data table.
  data = new vis.DataSet();
  // create some nice looking data with sin/cos
  var counter = 0;
  var steps = 50; // number of datapoints will be steps*steps
  var axisMax = 314;
  var axisStep = axisMax / steps;
  for (var x = 0; x < axisMax; x+=axisStep) {
    for (var y = 0; y < axisMax; y+=axisStep) {
      var value = custom(x, y);
      data.add({id:counter++, x:x, y:y, z:value, style:value});
    }
  }
}
```

```
// specify options
```

```
var options = {
  width: '600px',
  height: '600px',
  style: 'surface',
  showPerspective: true,
  showGrid: true,
  showShadow: false,
  keepAspectRatio: true,
  verticalRatio: 0.5
};
```

```
// Instantiate our graph object.
```

```
var container = document.getElementById('mygraph');
graph = new vis.Graph3d(container, data, options);
}
```

```
</script>
```

```
</head>
```

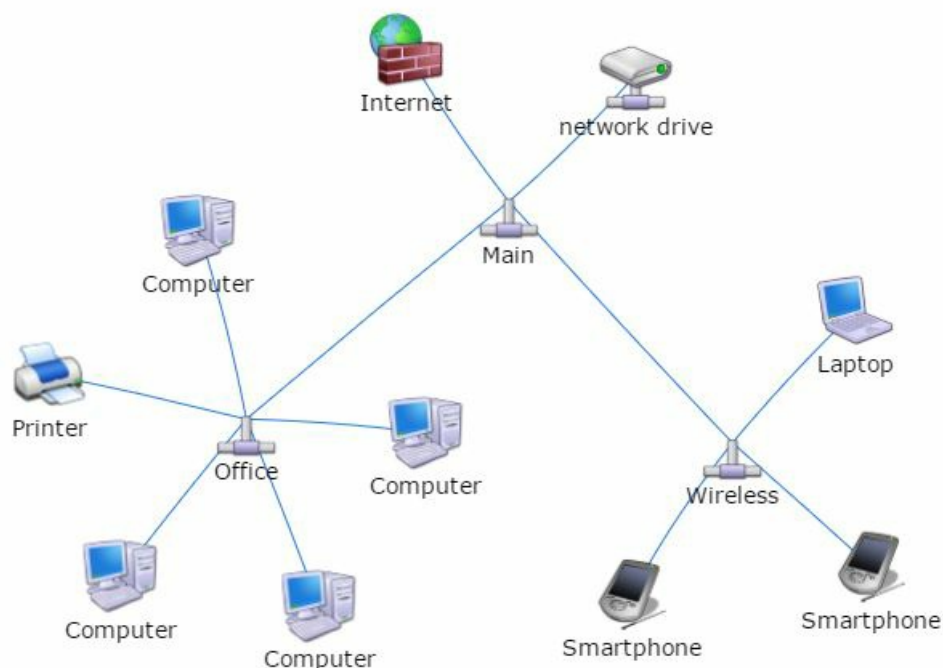
```
<body onload="drawVisualization();">
```

```
<div id="mygraph"></div>
```

```
<div id="info"></div>
</body>
</html>
```

而在網路圖方面，您甚至可以指定每個節點應該呈現的圖片，以下是一個繪製電腦網路圖的範例。

- [http://visjs.org/examples/network/03\\_images.html](http://visjs.org/examples/network/03_images.html)



圖、用 vis.js 繪製電腦網路圖

以下是上述範例的完整 HTML 檔案。

```
<!doctype html>
<html>
<head>
  <title>Network | Images</title>

  <style type="text/css">
    body {
      font: 10pt arial;
    }
    #mynetwork {
      width: 600px;
      height: 600px;
      border: 1px solid lightgray;
    }
  </style>
</head>
</html>
```

```
</style>
```

```
<script type="text/javascript" src="../../dist/vis.js"></script>
```

```
<link href="../../dist/vis.css" rel="stylesheet" type="text/css" />
```

```
<script type="text/javascript">
```

```
var nodes = null;
```

```
var edges = null;
```

```
var network = null;
```

```
var DIR = 'img/refresh-cl/';
```

```
var LENGTH_MAIN = 150;
```

```
var LENGTH_SUB = 50;
```

```
// Called when the Visualization API is loaded.
```

```
function draw() {
```

```
    // Create a data table with nodes.
```

```
    nodes = [];
```

```
    // Create a data table with links.
```

```
    edges = [];
```

```
    nodes.push({id: 1, label: 'Main', image: DIR + 'Network-Pipe-icon.png', shape: 'image'});
```

```
    nodes.push({id: 2, label: 'Office', image: DIR + 'Network-Pipe-icon.png', shape: 'image'});
```

```
    nodes.push({id: 3, label: 'Wireless', image: DIR + 'Network-Pipe-icon.png', shape: 'image'});
```

```
    edges.push({from: 1, to: 2, length: LENGTH_MAIN});
```

```
    edges.push({from: 1, to: 3, length: LENGTH_MAIN});
```

```
    for (var i = 4; i <= 7; i++) {
```

```
        nodes.push({id: i, label: 'Computer', image: DIR + 'Hardware-My-Computer-3-icon.png', shape: 'image'});
```

```
        edges.push({from: 2, to: i, length: LENGTH_SUB});
```

```
    }
```



```
nodes.push({id: 101, label: 'Printer', image: DIR + 'Hardware-Printer-Blue-icon.png', shape: 'image'});
edges.push({from: 2, to: 101, length: LENGTH_SUB});

nodes.push({id: 102, label: 'Laptop', image: DIR + 'Hardware-Laptop-1-icon.png', shape: 'image'});
edges.push({from: 3, to: 102, length: LENGTH_SUB});

nodes.push({id: 103, label: 'network drive', image: DIR + 'Network-Drive-icon.png', shape: 'image'});
edges.push({from: 1, to: 103, length: LENGTH_SUB});

nodes.push({id: 104, label: 'Internet', image: DIR + 'System-Firewall-2-icon.png', shape: 'image'});
edges.push({from: 1, to: 104, length: LENGTH_SUB});

for (var i = 200; i <= 201; i++ ) {
    nodes.push({id: i, label: 'Smartphone', image: DIR + 'Hardware-My-PDA-02-icon.png', shape: 'image'});
    edges.push({from: 3, to: i, length: LENGTH_SUB});
}

// create a network
var container = document.getElementById('mynetwork');
var data = {
    nodes: nodes,
    edges: edges
};
var options = {
    stabilize: false // stabilize positions before displaying
};
network = new vis.Network(container, data, options);
}
</script>
</head>

<body onload="draw()">
```

```
<div id="mynetwork"></div>

</body>
</html>
```

透過這兩個範例，相信您應該可以感覺到 vis.js 的誠意，這真的是一個相當棒的繪圖函式庫啊！

### 參考文獻

- <http://visjs.org/>

## CodeMirror -- 有 IntelliSense 功能的網頁版開源編輯器

在建構 jsLab 科學計算平台的過程中，由於要讓使用者在 JavaScript 程式編輯上有更好的感受，而不是只能依賴死板板的 textarea 區塊，所以我們找了幾個用 javascript 建構的網頁版程式碼編輯器，像是 Ace、Atom、EditArea 與 CodeMirror 等專案，最後我們認為 CodeMirror 最適合我們使用，因為 CodeMirror 的資源完整，而且具有支援 JavaScript 語言 IntelliSense 功能的插件。

- <https://atom.io/>
- <http://ace.c9.io/>
- <http://www.cdolivet.com/editarea/>
- <http://codemirror.net/>

CodeMirror 支援超過六十種語言的上色與編輯功能，包含 HTML、XML、Javascript、Python、Ruby、C/C++/C#、... 等等，您可以參考下列網頁。

- <http://codemirror.net/mode/index.html>

另外，CodeMirror 還支援了下列的特色功能。

- A powerful, composable language mode system
- Autocompletion (XML)
- Code folding
- Configurable keybindings
- Vim, Emacs, and Sublime Text bindings
- Search and replace interface
- Bracket and tag matching
- Support for split views
- Linter integration
- Mixing font sizes and styles
- Various themes
- Able to resize to fit content
- Inline and block widgets
- Programmable gutters

- Making ranges of text styled, read-only, or atomic
- Bi-directional text support
- Many other methods and addons...

其中我們最需要的是 JavaScript 的 Intellisense 功能，該功能的範例網址如下：

- <http://codemirror.net/demo/complete.html>

可惜的是，該範例使用 Ctrl-Space 做為 Intellisense 的功能鍵，但是這個按鍵與繁體中文 windows 的輸入法切換功能相衝，所以沒辦法正常運作，因此我們將該範例的「Ctrl-Space」改為「Ctrl-。」，以避免這種衝突的情況，修改版的範例的網址與程式碼如下。

- <http://ccckmit.github.io/jslab/js/codemirror/demo/jscomplete.html>

```
<!doctype html>

<title>CodeMirror: Autocomplete Demo</title>
<meta charset="utf-8"/>
<link rel=stylesheet href="../doc/docs.css">

<link rel="stylesheet" href="../lib/codemirror.css">
<link rel="stylesheet" href="../addon/hint/show-hint.css">
<script src="../lib/codemirror.js"></script>
<script src="../addon/hint/show-hint.js"></script>
<script src="../addon/hint/javascript-hint.js"></script>
<script src="../mode/javascript/javascript.js"></script>

<div id=nav>
  <a href="http://codemirror.net"></a>

  <ul>
    <li><a href="../index.html">Home</a>
    <li><a href="../doc/manual.html">Manual</a>
    <li><a href="https://github.com/marijnh/codemirror">Code</a>
  </ul>
  <ul>
    <li><a class=active href="#">Autocomplete</a>
  </ul>
</div>
```

```
<article>
```

```
<h2>Autocomplete Demo</h2>
```

```
<form><textarea id="code" name="code">
```

```
function getCompletions(token, context) {
  var found = [], start = token.string;
  function maybeAdd(str) {
    if (str.indexOf(start) == 0) found.push(str);
  }
  function gatherCompletions(obj) {
    if (typeof obj == "string") forEach(stringProps, maybeAdd);
    else if (obj instanceof Array) forEach(arrayProps, maybeAdd);
    else if (obj instanceof Function) forEach(funcProps, maybeAdd);
    for (var name in obj) maybeAdd(name);
  }

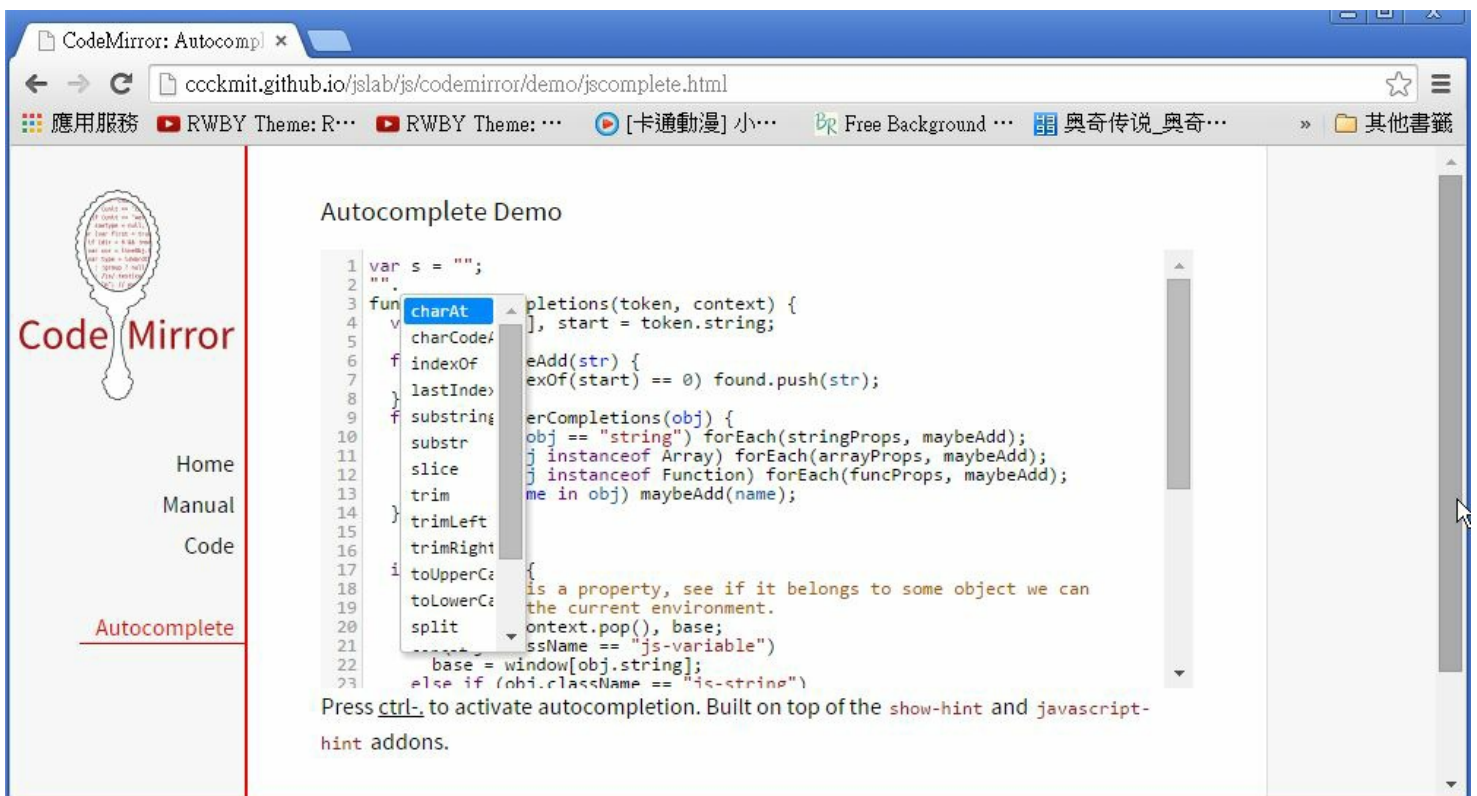
  if (context) {
    // If this is a property, see if it belongs to some object we can
    // find in the current environment.
    var obj = context.pop(), base;
    if (obj.className == "js-variable")
      base = window[obj.string];
    else if (obj.className == "js-string")
      base = "";
    else if (obj.className == "js-atom")
      base = 1;
    while (base != null && context.length)
      base = base[context.pop().string];
    if (base != null) gatherCompletions(base);
  }
  else {
    // If not, just look in the window object and any local scope
    // (reading into JS mode internals to get at the local variables)
    for (var v = token.state.localVars; v; v = v.next) maybeAdd(v.name);
    gatherCompletions(window);
    forEach(keywords, maybeAdd);
  }
  return found;
}
```

```
}  
</textarea></form>
```

Press **ctrl-.** to activate autocompletion. Built on top of the [`show-hint`](../doc/manual.html#addon_show-hint) and [`javascript-hint`](../doc/manual.html#addon_javascript-hint) addons.

```
<script>  
  var editor = CodeMirror.fromTextArea(document.getElementById("code"), {  
    lineNumbers: true,  
    extraKeys: {"Ctrl-." : "autocomplete"},  
    mode: {name: "javascript", globalVars: true}  
  });  
</script>  
</article>
```

不過、雖然 CodeMirror 支援了 JavaScript 的 Intellisense 功能，但是卻不徹底，對於像是字串之類的語法，他可以顯示提示函數與訊息，但對於變數的部分，卻無法正確提示，以下是一個可以正確提示的範例。



圖、CodeMirror 可以正確提示的 JavaScript 編輯範例

如果需要更進一步的正確提示功能，恐怕還是得修改 CodeMirror 的 `../addon/hint/javascript-hint.js` 模組才行。

雖然 CodeMirror 已經算不錯了，不過我想還有進步空間，但大體來說 CodeMirror 已經是很好的網頁程式編輯器了。

不過、如果您需要的是可以加上「粗體、斜體、字型、超連結、....」等功能，那就不是 CodeMirror 所具備的功能了，這種功能是 RichText Editor 才具備的，您可以參考下列文章中的 RichText 編輯器。

- [5 Free JavaScript Libraries to Add Text Editing to Your Web Application](#)

#### 參考文獻

- <http://codemirror.net/>
- <http://codemirror.net/demo/complete.html>

## Memory Sanitization (作者：研發養成所 Bridan)

什麼叫記憶體衛生處理？讓我們先看個故事再做說明。

阿誠是某高科技公司的工程師，他負責使用儀器量測新產品實驗數據，有位同事阿堅和他很要好，常常去實驗室找他聊天，其實阿堅是另一家公司派來長期臥底の間諜，專門探詢公司最新研發的產品。公司有一台極輕巧的溫度監控儀，阿誠常使用它對新產品零件溫度監控，實驗完畢後，資料當然改存到電腦中，並且依據儀器說明書上，資料清除步驟清資料。一天阿堅向阿誠借用這儀器，其實他是把儀器拿給儀器駭客進行逆向工程，去解碼原始實驗資料。

從這故事，你學到了什麼？有許多科技公司對資訊安全非常在意，連清理資料也要徹底破壞！

大家都知道在 [電腦上刪除檔案](#)，只不過是把檔案搬移到資源回收筒，隨時可以檔案還原，稍微注意的人，還會清除資源回收筒的檔案，更高竿的會再進行衛生處理，徹底洗掉不要的資料。

一般資料庫，清除資料的做法是破壞資料索引，簡單的說，利用一個索引指示有多少資料在資料庫中，如果索引為N，表示有N筆資料在其中，使用者可以循序用指令將資料取出。當索引為零時，也表示資料庫資料清除。這方法雖然快速簡單，可是欠缺資料安全性，為避免上述事件發生，除了一般清除，還要另外設計徹底清除功能。換個例子說明，也就是上完廁所，除了擦屁股，沖水之外，不要忘記馬桶順便刷一刷洗一洗，徹底做個衛生處理。

## 利用 SQL Compact Edition 免費建立擁有 DataBase 的 Azure Websites (作者：陳星銘)

在只有免費服務才使用的這個世代，如果只是一個Demo的小型網站自然不想使用到雲端的SQL DB來做為 DataBase (其實只是不想花一個月150左右的DB費用XD)

鑑於想要使用免費Azure Websites，但又想要連接資料庫的人要怎麼做呢？

只能每個月砸150台幣買DB了嗎！？

當然是 **NO!**

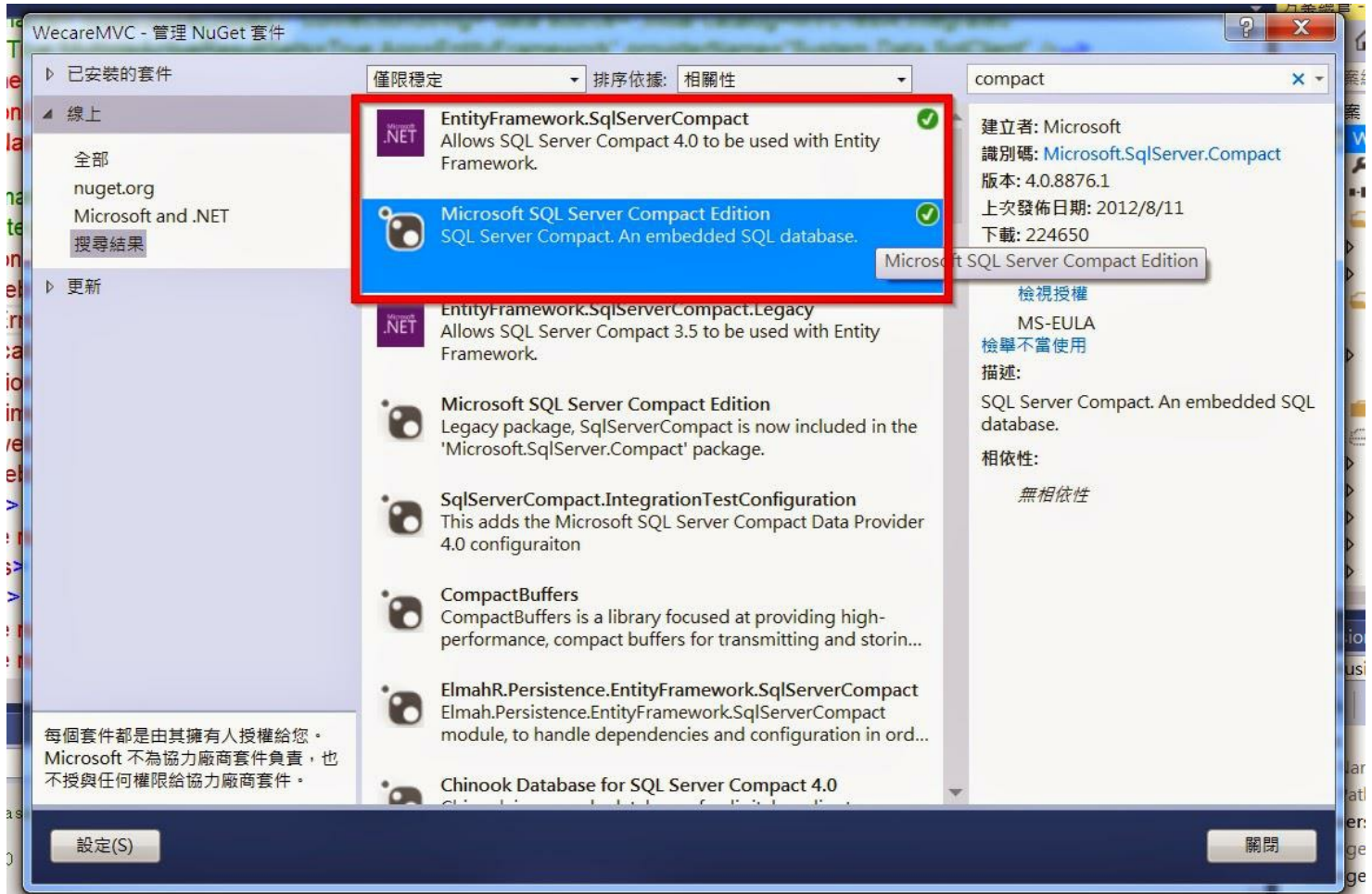
今天就來教大家利用SQL Compact Edition不花一毛錢使用擁有 DataBase 的 Azure Websites 吧！

以下看圖說故事開始：

第一步：打開你的VS安裝兩個Nuget套件

為你的專案加入兩個Nuget套件，分別是

1. EntityFramework.SqlServerCompact
2. Microsoft SQL Server Compact Edition



第二步：加入以下連線字串至Web.Config <connectionStrings> </connectionStrings> 區段中

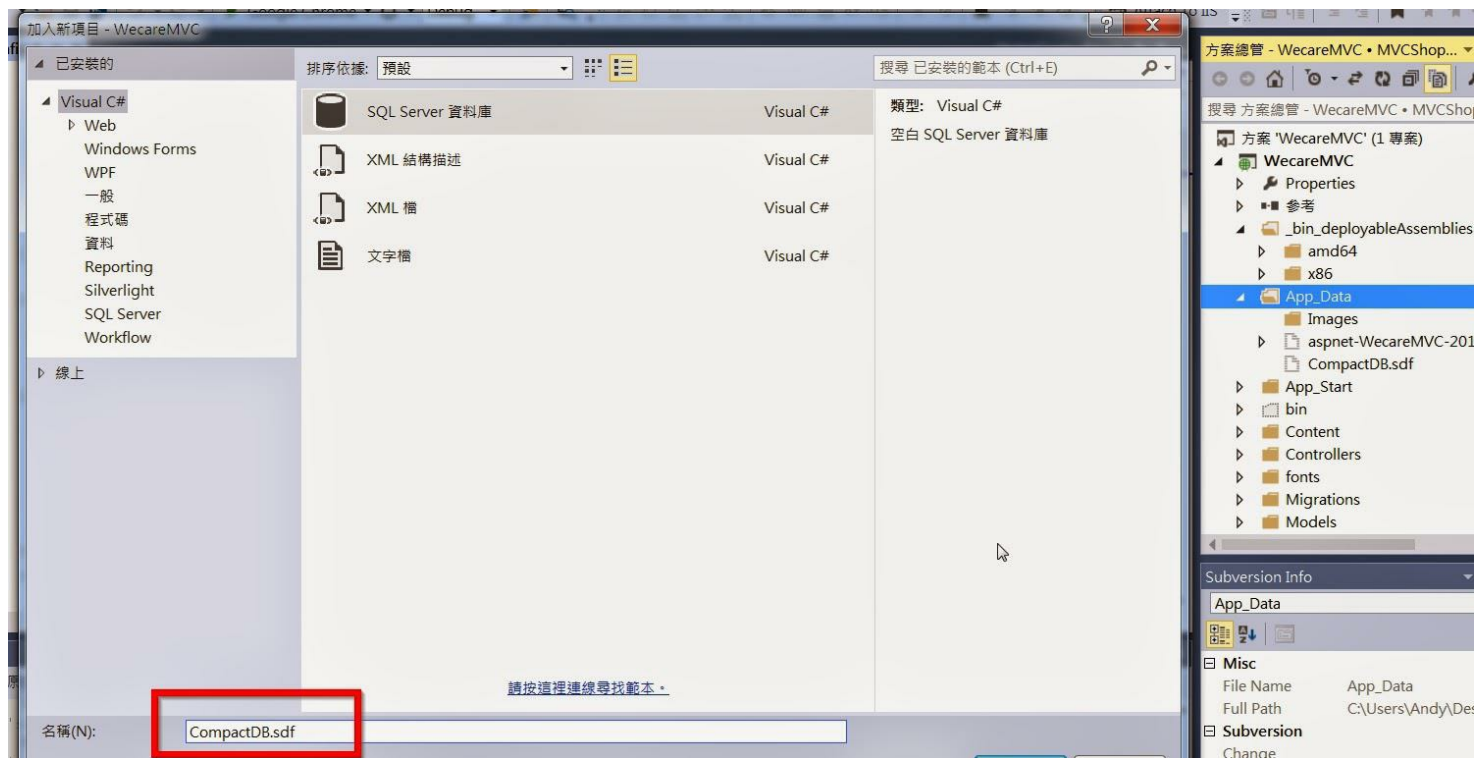
```
<add name = "DefaultConnection" connectionString = "Data Source=|DataDirectory|CompactDB.sdf" providerName = "System.Data.SqlServerCe.4.0" />
```

其中 Data Source=|DataDirectory|CompactDB.sdf 可得到相對路徑的 App\_Data.sdf

```
<connectionStrings>  
  <add name = "DefaultConnection" connectionString = "Data Source=|DataDirectory|CompactDB.sdf" providerName = "System.Data.SqlServerCe.4.0" />  
</connectionStrings>
```

第三步：在App\_Data中右鍵加入→新增項目→Sql Server 資料庫

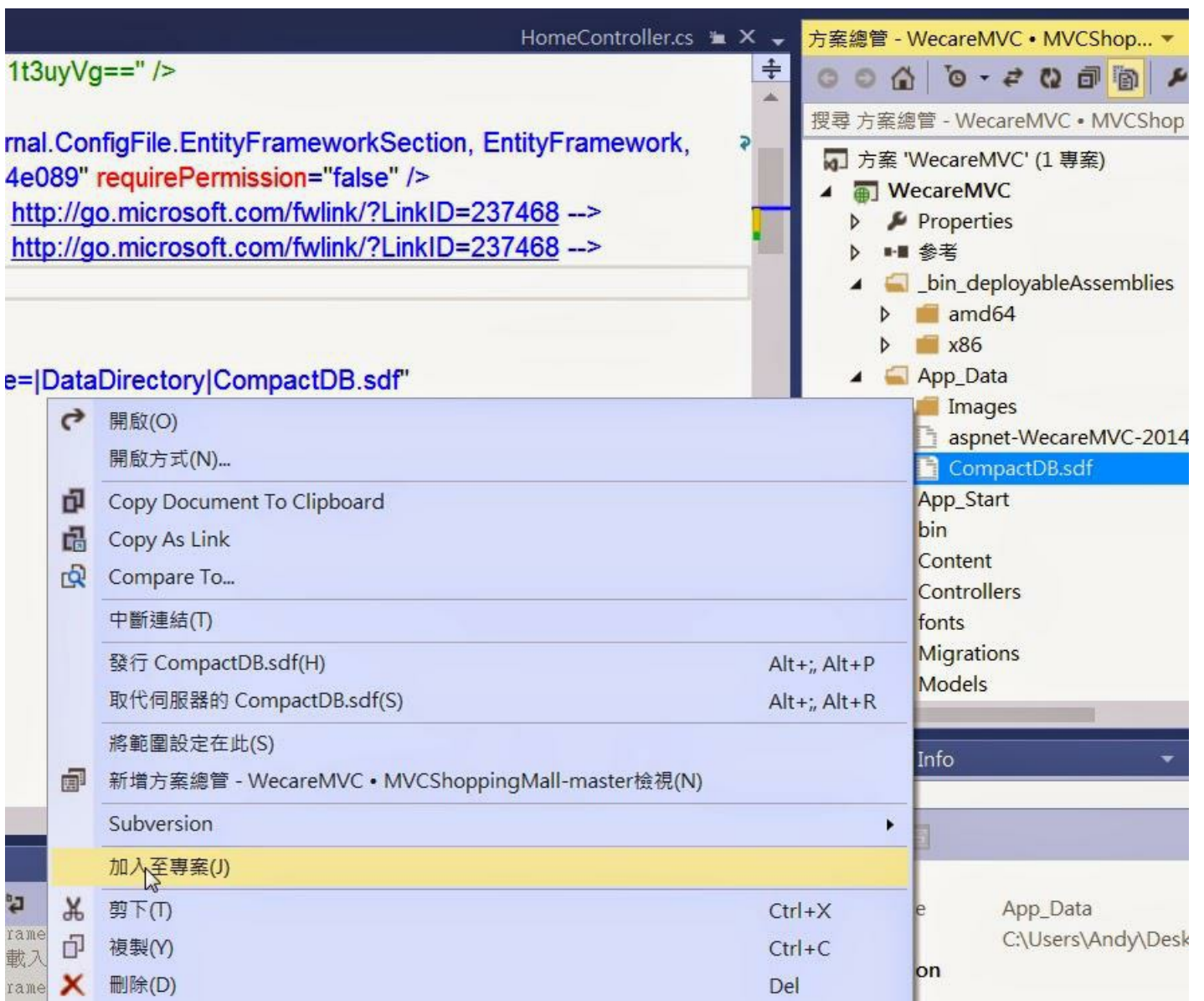
注意：這裡我將檔名改為.sdf檔，因為.sdf 很適合小型專案使用，不需要用到.mdf



若你用的和我一樣是MVC的CodeFirst則是改掉你的連線字串後，讓DB自己產生出來，但是這之後有一個很重要的步驟！真的很重要！我卡在這邊很久！操作如下：

產生DB後→點選右上角顯示所有檔案→找到你的CompactDB.sdf→右鍵加入至專案





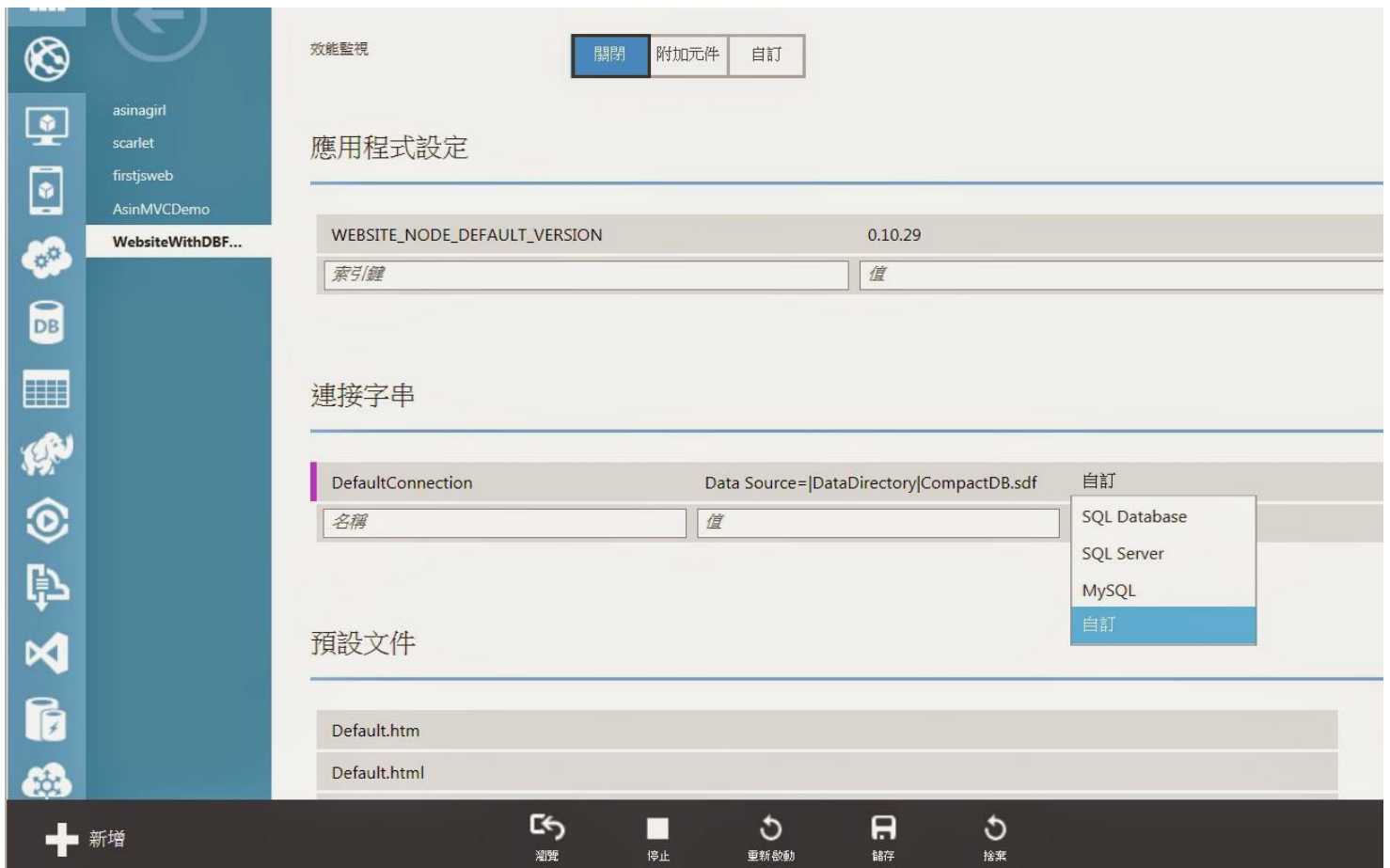
第四步：到你的Azure建立WebSites

左下新增→接著如圖選擇建立網站



第五步：

建好網站後，點選你的網站，點選上方設定，拉到下面，填入剛剛的连接字串，選擇"自訂"，按下方儲存



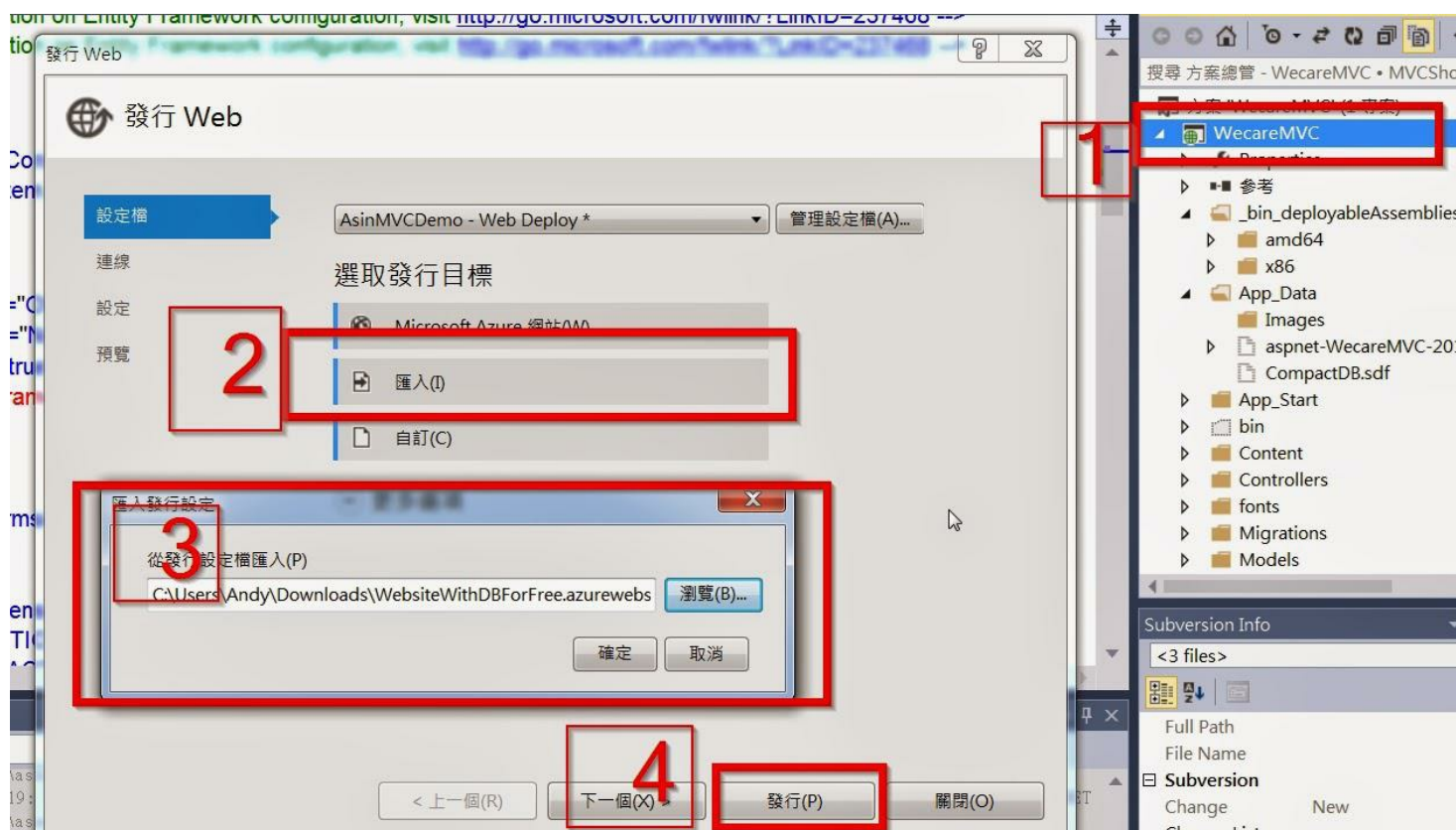
第六步：

到儀表板，點選下載發行設定檔，將其儲存在電腦中



第七步：

對你的專案按右鍵→發行→匯入→選到剛剛的設定檔→確定→發行



然後就發行成功囉！完成 [網址參考](#)

希望有幫助到大家的錢包！XD

授權說明：

- 本著作係採用姓名標示-非商業性-相同方式分享 3.0 台灣授權。欲查看本授權條款副本，請到 <http://creativecommons.org/licenses/by-nc-sa/3.0/tw/>，或寫信至 Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, USA.

## 函數指標陣列 (Array of Function Pointers) (作者：研發養成所 Bridan)

函數指標陣列，這是一種 C/C++ 程式語言的高階設計技巧，希望能有較高的執行效能。我以 Arduino 當作測試平台，比較兩種程式設計技巧，發現與我的認知有些差異。

先看傳統設計方式，用 switch case 執行不同功能：

```
//
// Author: Bridan
//      http://4rdp.blogspot.com
// Date: 2014/09/27
//
// Brief: Test switch case
//

void setup() {
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for Leonardo only
  }

  TCCR1A = 0x00;           // Normal mode, just as a Timer
  TCCR1B &= ~_BV(CS12);    // no prescaling
  TCCR1B &= ~_BV(CS11);
  TCCR1B |= _BV(CS10);
}

void loop() {
  byte i;

  TCNT1 = 0; // reset timer
  for (i=0 ; i<3 ; i++) {
    switch (i) {
      case 0:
```

```

    Serial.println("CASE 0");
    break;
case 1:
    Serial.println("CASE 1");
    break;
case 2:
    Serial.println("CASE 2");
    break;
}
}
Serial.println(TCNT1);
}

```

switch case 3 個時，編譯 2410 bytes，執行 6092 ~ 6100 timer clock switch case 4 個時，編譯 2430 bytes，執行 8136 ~ 8146 timer clock switch case 5 個時，編譯 2458 bytes，執行 10185 ~ 10195 timer clock

將上面程式修改成函數指標陣列，以查表方式直接跳到執行的程式：

```

//
// Author: Bridan
//      http://4rdp.blogspot.com
// Date: 2014/09/27
//
// Brief: Test Array of Function Pointers
//

void setup() {
    Serial.begin(9600);
    while (!Serial) {
        ; // wait for serial port to connect. Needed for Leonardo only
    }

    TCCR1A = 0x00; // Normal mode, just as a Timer
    TCCR1B &= ~_BV(CS12); // no prescaling
    TCCR1B &= ~_BV(CS11);
    TCCR1B |= _BV(CS10);
}

void FUNC0(void) {

```

```

    Serial.println("CASE 0");
}
void FUNC1(void) {
    Serial.println("CASE 1");
}
void FUNC2(void) {
    Serial.println("CASE 2");
}

void (*TABLE_JUMP[]) (void) = {
    FUNC0,
    FUNC1,
    FUNC2
};

void loop() {
    byte i;

    TCNT1 = 0;    // reset timer
    for (i=0 ; i<3 ; i++) {
        (*TABLE_JUMP[i]) ();
    }
    Serial.println(TCNT1);
}

```

TABLE 3 個時，編譯 2438 bytes，執行 6082 ~ 6096 timer clock  
 TABLE 4 個時，編譯 2470 bytes，執行 8130 ~ 8142 timer clock  
 TABLE 5 個時，編譯 2504 bytes，執行 10176 ~ 10194 timer clock

以往我所用過的 compiler，switch case 相當於很多 if ... else ... 的組合，條件一個一個比較，數值越大的條件，花費比較的時間越多，以上面的例子在比較方面所費的時間 = 1 + 2 + 3 + ... + N，而函數指標陣列查表時間約 = 1 x N，比 switch case 有效率，這部分與結果相符 (比較條件太少，不易看出差異)。

至於程式碼大小，發現越多條件狀況，以函數指標陣列方式設計比 switch case 程式碼多？因為不清楚 Arduino compiler 如何設計，無法進一步評論，但直覺 Arduino compiler 缺少這方面最佳化處理。

# 雜誌訊息

## 讀者訂閱

程式人雜誌是一個結合「開放原始碼與公益捐款活動」的雜誌，簡稱「開放公益雜誌」。開放公益雜誌本著「讀書做善事、寫書做公益」的精神，我們非常歡迎程式人認養專欄、或者捐出您的網誌，如果您願意成為本雜誌的專欄作家，請加入 [程式人雜誌社團](#) 一同共襄盛舉。

我們透過發行這本雜誌，希望讓大家可以讀到想讀的書，學到想學的技術，同時也讓寫作的朋友的作品能產生良好價值 – 那就是讓讀者根據雜誌的價值捐款給慈善團體。讀雜誌做公益也不需要壓力，您不需要每讀一本就急著去捐款，您可以讀了十本再捐，或者使用固定的月捐款方式，當成是雜誌訂閱費，或者是季捐款、一年捐一次等都 OK！甚至是單純當個讀者我們也都很歡迎！

本雜誌每期參考價：NT 50 元，如果您喜歡本雜誌，請將書款捐贈公益團體。例如可捐贈給「羅慧夫顏面基金會 彰化銀行(009) 帳號：5234-01-41778-800」。(若匯款要加註可用「程式人雜誌」五個字)

## 投稿須知

**給專欄寫作者：**做公益不需要有壓力。如果您願意撰寫專欄，您可以輕鬆的寫，如果當月的稿件出不來，我們會安排其他稿件上場。

**給網誌捐贈者：**如果您沒時間寫專欄或投稿，沒關係，只要將您的網誌以 [創作共用的「姓名標示、非商業性、相同方式分享」授權] 並通知我們，我們會自動從中選取需要的文章進行編輯，放入適當的雜誌當中出刊。

**給文章投稿者：**程式人雜誌非常歡迎您加入作者的行列，如果您想撰寫任何文章或投稿，請用 markdown 或 LibreOffice 編輯好您的稿件，並於每個月 25 日前投稿到 [程式人雜誌社團](#) 的檔案區，我們會盡可能將稿件編入隔月 1 號出版程式人雜誌當中，也歡迎您到社團中與我們一同討論。

如果您要投稿給程式人雜誌，我們最希望的格式是採用 markdown 的格式撰寫，然後將所有檔按壓縮為 zip 上傳到社團檔案區給我們，如您想學習 markdown 的撰寫出版方式，可以參考 [看影片學 markdown 編輯出版流程](#) 一文。

如果您無法採用 markdown 的方式撰寫，也可以直接給我們您的稿件，像是 MS. Word 的 doc 檔或 LibreOffice 的 odt 檔都可以，我們會將這些稿件改寫為 markdown 之後編入雜誌當中。

## 參與編輯

您也可以擔任程式人雜誌的編輯，甚至創造一個全新的公益雜誌，我們誠摯的邀請您加入「開放公益出版」的行列，如果您想擔任編輯或創造新雜誌，也歡迎到 [程式人雜誌社團](#) 來與我們討論相關事宜。

## 公益資訊

公益團體	聯絡資訊	服務對象	捐款帳號

財團法人羅慧夫顱顏基金會	<a href="http://www.mncf.org/">http://www.mncf.org/</a> <a href="mailto:lynn@mncf.org">lynn@mncf.org</a> 02-27190408分機 232	顱顏患者(如唇顎裂、小耳症或其他罕見顱顏缺陷)	銀行：009彰化銀行民生分行 帳號：5234-01-41778-800
社團法人台灣省兒童少年成長協會	<a href="http://www.cyga.org/">http://www.cyga.org/</a> <a href="mailto:cyga99@gmail.com">cyga99@gmail.com</a> 04-23058005	單親、隔代教養、弱勢及一般家庭之兒童青少年	銀行：新光銀行 戶名：台灣省兒童少年成長協會 帳號：103-0912-10-000212-0