Function 故事

開頭:

故事是這樣的，小明是個對寫coding有那麼點興趣的小小工程師，目前正在練習寫javasciprt，某天他寫著寫著寫出了下面這些東西:

CODE

小明寫了兩張cashCard，而且呢!都有給了個充值的方法，只要餘額(money)是大於等於0的就符合條件給充值，但小明很快了的就發現了個問題:「重複的程式碼太多了，且做了相同的事情」，重複對於程式碼來說是個不好聞的味道，小明就去山里的瀑布底下，誠心思考了七天，突然靈光乍現!「只要把相同的地方包裝起來，需要的時候再呼叫不叫好了嗎?」回家後把程式碼改成下面這個樣子:

CODE

之後如果有卡片需要充值，只要呼叫這個方法並把需要充值的卡片丟入引數內就可以拉!! 這樣程式碼就更簡便了許多!

Function 宣告

………

Function 引數:

……

* 引數 v.s 參數

後來小明覺得這樣還挺不方便的，充值的錢這樣都是固定的，所以小明就想到了再傳入一個引數是我們希望充值的錢不就得了?!於是改成了下面

Code

用著用著有次小明不小心少傳入了一個引數，也就是要充值多少錢，小明心想「完了，這肯定出問題」於是檢查了一下充值過後的cashCard，發現值變成了NaN，到底是發生了什麼事情呢?經過一翻查找資料後發現:

Function arguments

…..

　　又過了幾天，小明又發覺了一個問題，就是這個新的function看了賊煩，cashCard這個名稱太長了，想著能不能夠簡短點，於是又到了瀑布底下誠心思考，果不其然的又想出了個妙法，以後呢我們不寫上cashCard了，通通把cashCard改成this，也不傳入引數了，那這樣要怎麼把我們需要操作的值或是物件丟進去呢?沒問題，我們額外再使用call()方法，把我們需要操作的值或是物件丟進去就行，這樣function就又能夠更簡短一些:

Code

Function ‘this’

* Call()
* Apply()

Function constructor

但是呢!cashcard1 與cashcard2都是一樣的東西，只是名字或只餘額不一樣而已，所以小明想到可以用function來創立物件：像是.

Code

一改完想了一下，「恩~這樣每個new出來的物件都有toString方法，也太佔資源了吧!這個方法只要一個就好了」於是又改成:

Code

Function toString() {…};

Function CashCard(ID,Money){ };

Function 原型鍊

但其實呢!了解原型練的話，也可以;

CashCard\_pro.prototype.toString = function()

{…}