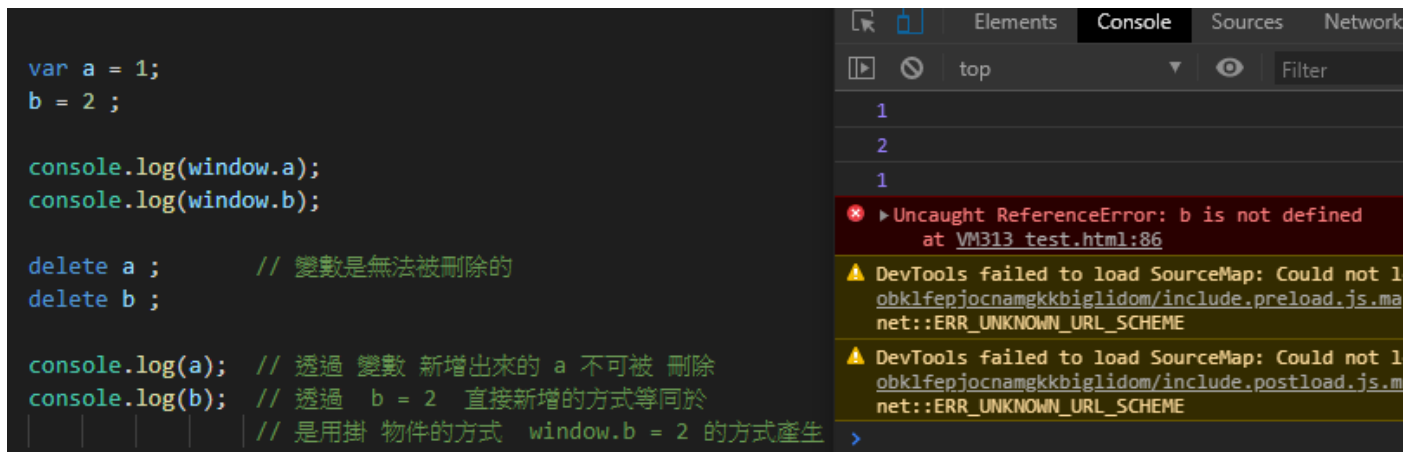


JS 核心篇 Table of Contents

- 4-34 物件的參考特性 傳值 or 傳址
- 4-31 變數及物件屬性的差異 變數無法被刪除
- 5-45 閉包 Closure
- 5-46 閉包進階：工廠模式及私有方法
- 5-47 最常見的 this：物件的方法調用
- 5-53 原型在哪裡？Prototype & __proto__ 比較
- 6-52 原型鍊的概念 - 為什麼有原型 class 歷史講解
- 6-56 使用 Object.create 建立多層繼承 Object.create
- 7-59 屬性特徵是什麼？ Object.defineProperty() / for-in
- 5-60 物件屬性不可寫入？物件擴充的修改與調整 .preventExtensions() / .seal() / .freeze()
- 5-61 屬性列舉與原型的關係 .hasOwnProperty()
- 5-62 Getter 與 Setter，賦值運算不使用函式

4-31 變數及物件屬性的差異

物件可以透過 `delete` 被刪除，但是變數無法



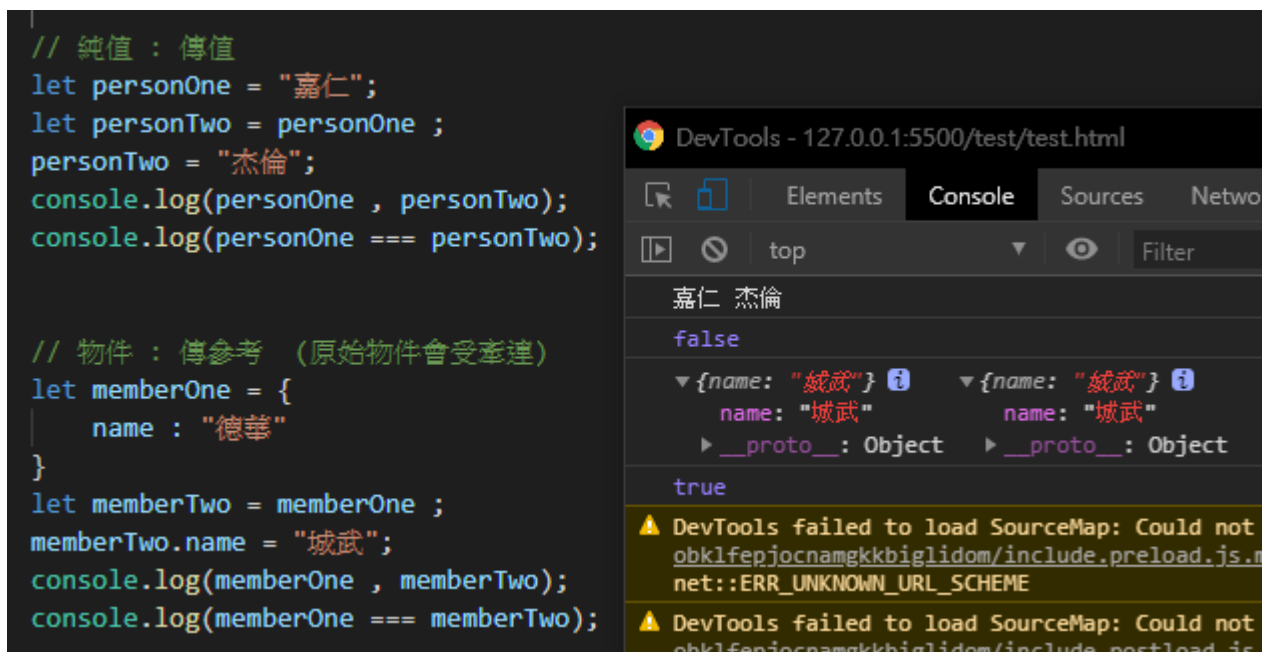
補充：如果在最後 `console.log(window.b) // undefine`

如果是透過 `window` 來尋找 那 可以針對為定義的物件屬性 顯示出 `undefine` 而非 `is not defined` 而導致 程式中斷

4-34 物件的參考特性 傳值 or 傳址

Javascript 在賦予一值變數時會有兩種特性，傳值以及傳參考

如果是物件有傳參考特性，因為物件會共用同一參考 所以參考被改寫



Call by value vs call by reference

傳值	傳參考
<ul style="list-style-type: none">- Boolean- Null- Undefined- Number- String- ...	<ul style="list-style-type: none">- 物件 (陣列, 函式)

5-45 閉包 Closure

錢包範例

```
let wallet = () => {  
  let money = 1000;  
  return (price) => {  
    money = money + price;  
    return money  
  }  
}  
  
console.log(wallet()(100)); // 這裡的 wallet() 裡的 money 將會被釋放  
console.log(wallet()(100)); // 所以不會進行累加  
  
console.log("*****")  
  
let leon_wallet = wallet();// 這裡的 wallet() 是一個表達式 所以他會有return值，那今天它被賦予到leon 的錢包 後，裏頭的 money 將不會被釋放  
console.log(leon_wallet(200)); // 以下會持續累加  
console.log(leon_wallet(200));  
console.log(leon_wallet(200));  
  
console.log("*****")  
  
let teddy_wallet = wallet();// 這裡又重新宣告一個獨立的 teddy 的錢包 將與 leon 的錢包 是不一樣的  
console.log(teddy_wallet(1000)); // 以下會持續累加  
console.log(teddy_wallet(1000));  
console.log(teddy_wallet(1000));
```

Elements	Console	Sources	Network	Performance	Memory
top	Filter	Default levels			
1100					VM722
1100					VM722
*****					VM722
1200					VM722
1400					VM722
1600					VM722
*****					VM722
2000					VM722
3000					VM722
4000					VM722
⚠ DevTools failed to load SourceMap: Could not load content for chrome-ss					

46. 閉包進階：工廠模式及私有方法

以下三種範例 展示閉包

範例一 的迴圈內 用 var 將使 全域變數 3 溢出

```
48
49 function arrFunction() {
50     let arr = [];
51     for (var i = 0; i < 3; i++) {
52         arr.push(function () {
53             console.log(i) // var 出來的 i 會變成全域變數
54         }) // 將溢出for 迴圈 然後依值被記住 成 3
55     }
56     console.log("這裡是溢出來的 i : " + i)
57     return arr;
58 }
59 let fn = arrFunction();
60 console.log(fn)
61 fn[0]();
62 fn[1]();
63 fn[2]();
64
```

DevTools - 127.0.0.1:5500/te... —

Elements Console >> 1 hidden

這裡是溢出來的 i : 3 VM774 test.html:56

VM774 test.html:60

▼ (3) [f, f, f] ⓘ

- ▶ 0: f ()
- ▶ 1: f ()
- ▶ 2: f ()
- length: 3
- ▶ __proto__: Array(0)

3 VM774 test.html:53

3 VM774 test.html:53

3 VM774 test.html:53

⚠ DevTools failed to load SourceMap: Could not load content for chrome-extension://gighmmpiobklfepjocnamgkkbiglidom/include.p

```
65
66 // let 改善
67 function arrFunction2() {
68     let arr = [];
69     for (let i = 0; i < 3; i++) {
70         arr.push(function () {
71             console.log(i)
72         })
73     }
74     console.log("這裡抓不到 i ")
75     return arr;
76 }
77 let fn2 = arrFunction2();
78 console.log(fn2)
79 fn2[0]();
80 fn2[1]();
81 fn2[2]();
82
```

DevTools - 127.0.0.1:5500/te... —

Elements Console >> 1 hidden

這裡抓不到 i VM800 test.html:74

VM800 test.html:78

▼ (3) [f, f, f] ⓘ

- ▶ 0: f ()
- ▶ 1: f ()
- ▶ 2: f ()
- length: 3
- ▶ __proto__: Array(0)

0 VM800 test.html:71

1 VM800 test.html:71

2 VM800 test.html:71

⚠ DevTools failed to load SourceMap: Could not load content for chrome-extension://gighmmpiobklfepjocnamgkkbiglidom/include.p

```
83
84 // 用立即函式限制作用域
85 function arrFunction3() {
86     let arr = [];
87     for (var i = 0; i < 3; i++) {
88         (function (j) {
89             arr.push(function () {
90                 console.log(j)
91             })
92         })(i);
93     }
94     console.log("這裡抓不到 i ")
95     return arr;
96 }
97 let fn3 = arrFunction3();
98 console.log(fn3)
99 fn3[0]();
100 fn3[1]();
101 fn3[2]();
```

DevTools - 127.0.0.1:5500/te... —

Elements Console >> 1 hidden

這裡抓不到 i VM826 test.html:94

VM826 test.html:98

▼ (3) [f, f, f] ⓘ

- ▶ 0: f ()
- ▶ 1: f ()
- ▶ 2: f ()
- length: 3
- ▶ __proto__: Array(0)

0 VM826 test.html:90

1 VM826 test.html:90


2 VM826 test.html:90

⚠ DevTools failed to load SourceMap: Could not load content for chrome-extension://gighmmpiobklfepjocnamgkkbiglidom/include.p

函式工廠：給予不同的 參數值 而做相同的處理，就類似工廠的概念

```
function storeMoney(par = 1000) { // 這裡設店裡本來有多少資本（沒帶參數就用預設值1000）
  let money = par
  return (price = 100) => { // 然後每天 賺多少錢（沒帶參數就用預設值100）
    money = money + price;
    return money;
  }
}

let jiaAllMoney = storeMoney(2000) // 這裡參數給 店裡本來有多少資本
console.log(`第一天賺 ${jiaAllMoney(300)}`) // 這裡參數給 每日 賺多少錢
console.log(`第二天賺 ${jiaAllMoney(500)}`)
console.log(`第三天賺 ${jiaAllMoney(250)}`)
```

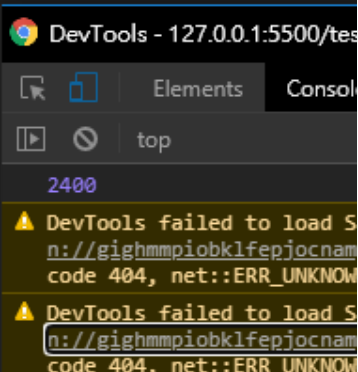


私有方法：不像函式工廠只能不斷存入 參數，所以其函示結果只有一種（只進行 return 函數），而是透過 return 物件 進行多種方法調配，那物件裡就可以有多種函式了

```
function storeMoney(par = 1000) { // 這裡設店裡本來有多少資本（沒帶參數就用預設值1000）
  let money = par
  return {
    increase: (price = 100) => { money += price }, // 每次賺多少（沒帶參數就預設 +100）
    decrease: (price = 100) => { money -= price }, // 每次花多少（沒帶參數就預設 -100）
    total: () => money
  }
}

let jiaAllMoney = storeMoney(2000) // 這裡參數給 店裡本來有多少資本
jiaAllMoney.increase(200); // 這裡參數給 賺多少錢
jiaAllMoney.increase(300); // 這裡參數給 賺多少錢
jiaAllMoney.decrease(100); // 這裡參數給 花多少錢

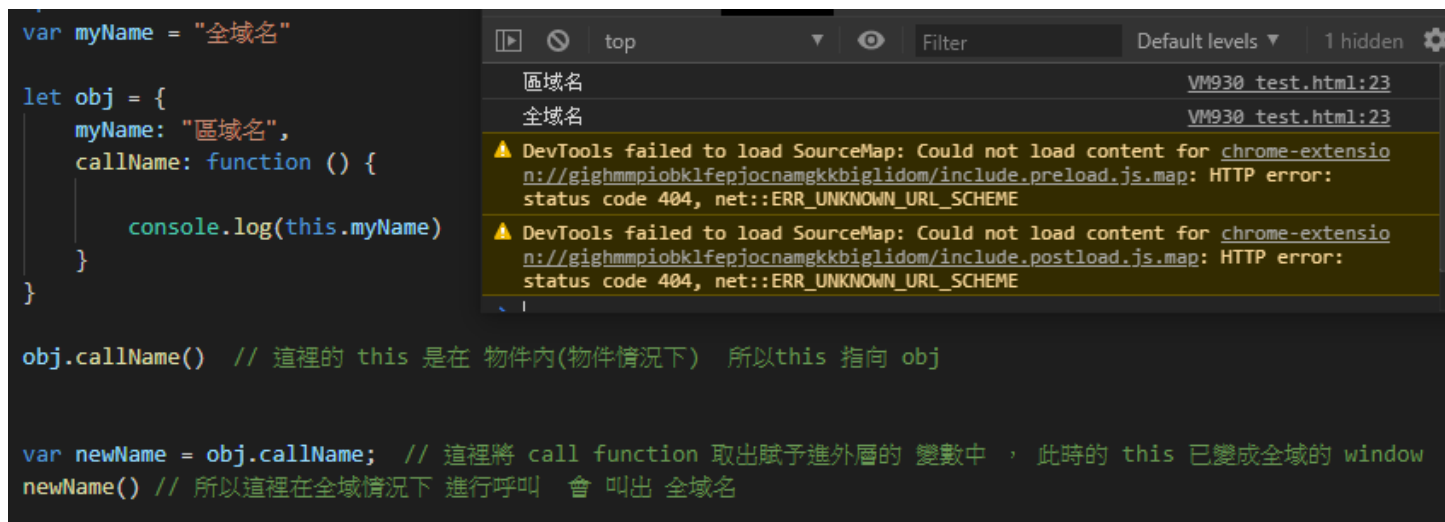
console.log(jiaAllMoney.total()) // 秀出餘額
```



5-47. 最常見的 this：物件的方法調用

如果要知道 this 指向誰，完全不需要理會他是如何被定義，只要知道他在呼叫的時候是在物件時被呼叫還是在全域情況下被呼叫

關鍵在於 "()" 的時機 !!!



The screenshot shows a code editor with the following JavaScript code:

```
var myName = "全域名"

let obj = {
  myName: "區域名",
  callName: function () {
    console.log(this.myName)
  }
}

obj.callName() // 這裡的 this 是在 物件內(物件情況下) 所以this 指向 obj

var newName = obj.callName; // 這裡將 call function 取出賦予進外層的 變數中，此時的 this 已變成全域的 window
newName() // 所以這裡在全域情況下 進行呼叫 會 叫出 全域名
```

On the right, the browser's DevTools console is open, showing two error messages:

- 區域名 VM930 test.html:23
- 全域名 VM930 test.html:23
- DevTools failed to load SourceMap: Could not load content for chrome-extension://gighmmpiobkfepjocnamgkbbiglidom/include.preload.js.map: HTTP error: status code 404, net::ERR_UNKNOWN_URL_SCHEME
- DevTools failed to load SourceMap: Could not load content for chrome-extension://gighmmpiobkfepjocnamgkbbiglidom/include.postload.js.map: HTTP error: status code 404, net::ERR_UNKNOWN_URL_SCHEME

6-52 原型鍊的概念 - 為什麼有原型 `class` 歷史講解

Java 的物件導向 有著名的 Class - 類別繼承：

`new` 出來的 新類別 可以繼承 原始類別的 屬性(顏色、外觀..)與 方法(某種動作)

那 Javascript 他本身其實都是用 `Object` 建構出來的 於是模仿了 此種類別繼承，在 ES6 的時候使用語法糖 建立出了獨有的 原型繼承(`prototype`)，但要注意 他並不是另外加入了此種類別語法，而只是語法糖而已（背後運作根本還是去使用 `prototype` 的方法運行 ex. 當使用 `babel` 去轉譯 ES6 寫出來的 `Class` 時會發現他還是 把其轉為 `prototype`）

5-53 原型在哪裡？ `Prototype` & `__proto__` 比較

`Prototype` & `__proto__` 比較

函式本身就是個物件，那其中會有個特別的屬性叫 `Prototype` 那透過這個 `Prototype` 屬性新增出來的就會形成 原型上的方法

那 `__proto__` 與 `Prototype` 兩者其實 用起來差不多，只是 前者是物件上的原型 後者是 函式上的原型，一般不建議 操作 `__proto__` 來新增 方法

6-56 使用 Object.create 建立多層繼承 Object.create

ES6 的繼承透過 class 的 extend

ES6 的 建構式 透過 constructor

ES5 的繼承 透過 Object.creat()

ES5 的的建構式 透過 .call(this,)

7-59 屬性特徵是什麼？ Object.defineProperty ()

一般的物件屬性與物件的值都是可以被調整的，但如果遇及有需要限制的情況則使用：

.defineProperty () 是一個物件方法，可用於調整限制物件的屬性：

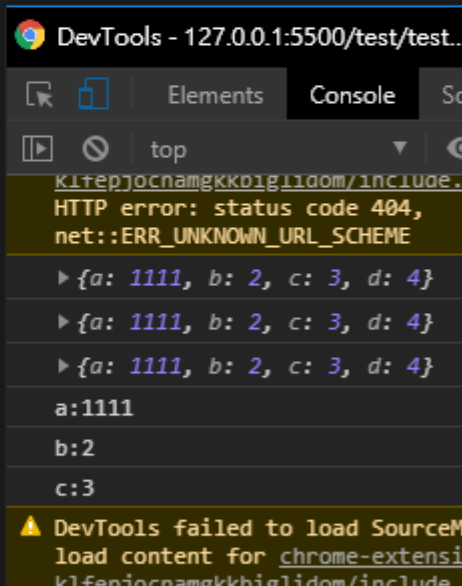
.defineProperty () 只針對物件的屬性作操作

```
let person = {
  a : 1,
  b : 2,
  c : 3,
  d : 4,
}
// 三個參數：物件本身，屬性(用字串方式傳入)，參數(預設都是true)
Object.defineProperty( person, 'a', {
  value: 1111,           // 值
  writable: true,       // 是否允許寫入
  configurable: true,   // 是否允許刪除
  enumerable: true,     // 是否允許被列舉 (一一成列)
})
console.log(person) // 這裡的 a 已被改寫

Object.defineProperty(person, 'b', {
  writable: false,
})
person.b = 2222;
console.log(person) // 這裡的 b 無法寫入 維持原樣

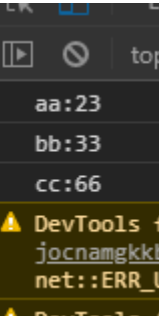
Object.defineProperty(person, 'c', {
  configurable: false,
})
delete person.c;
console.log(person) // 這裡 刪除不掉 c

Object.defineProperty(person, 'd', {
  enumerable: false,
})
for(let index in person){
  console.log(` ${index}: ${person[index]} `) // 這裡列舉不出 d
}
```



for-in 簡單範例：可列舉所有陣列或是物件的值 (一一列出來)

```
let arr = {
  aa : 23,
  bb : 33,
  cc : 66,
};
for (let index in arr){
  console.log(index + ":" + arr[index])
}
```



5-60 物件屬性不可寫入？物件擴充的修改與調整

`.preventExtensions()`、`.seal()`、`.freeze()` 三者 直接針對物件本身做操作

以下均只能針對物件本身屬性做調整，無法對 其巢狀屬性調整

```
// 以下均只能針對物件本身屬性做調整，無法對 其巢狀屬性調整
// ( 如果是物件中的值 又是物件 那其物件無法 調整 )

Object.preventExtensions() // 防止擴充
Object.isExtensible()      // 用於檢查 某物件可否被擴充 (會回傳 true / false 表示)
Object.getOwnPropertyDescriptor(); // 用於看 特定屬性 的特徵是什麼

Object.seal() // 進行封裝
Object.isSealed() // 用於檢查 某物件可否被封裝 (會回傳 true / false 表示)

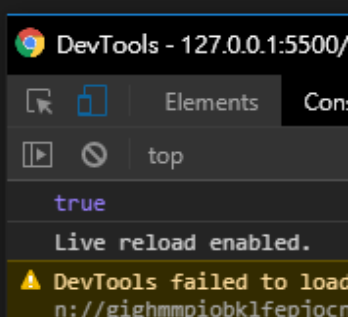
Object.freeze() // 凍結 當其一執行 物件會自動加上 seal 並無法調整
Object.isFrozen() // 用於檢查 某物件可否被凍結 (會回傳 true / false 表示)
```

5-61 屬性列舉與原型的關係 `.hasOwnProperty()`

以下定義一個 建構式 然後透過 `.hasOwnProperty()` 檢查其屬性 在不再

```
function Person(){
}
let jiaRen = new Person();
jiaRen.age = 23;

// hasOwnProperty() 將回傳 該屬性在不在
console.log( jiaRen.hasOwnProperty('age') )
```

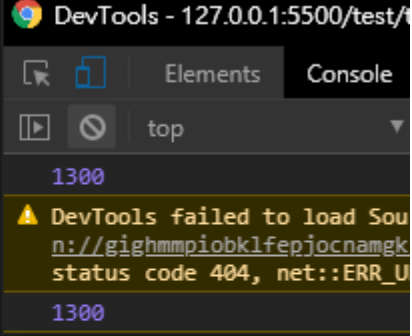


5-62 Getter 與 Setter，賦值運算不使用函式

當你在賦予值 想做運算 或是 取值的時候不要那麼直接取值 就可以使用 get & set

```
let wallet = {
  total: 1000,
  // 透過 set 可以 直接放一個函式在 object 裡面
  set save(price){
    this.total = this.total + price ;
  },
  get end_total(){
    return this.total
  }
}

wallet.save = 300; // 這裡的 300 或透過 傳參數的方式傳進去 save(price)
// 這裡 是使用 " = " 來賦予參數 並非用 wallet.save(300)
console.log(wallet.total) // 這裡予以下 total 相同
console.log(wallet.end_total) // 取得 get 的值
```



The screenshot shows the Chrome DevTools interface. The left pane displays the JavaScript code for the 'wallet' object and its usage. The right pane shows the 'Console' tab with a log entry of '1300'. Above the log, there is a warning message: 'DevTools failed to load Source map: Error: Can't load source map: http://localhost:5500/test/... status code 404, net::ERR_HTTP_404'.