**Object Oriented Programming in JavaScript**

# 大綱

Object method, property

method: function

property: variable。

也就是說方法是一種函數，property也是一種變數。差別是，物件中的方法和變數，只有利用「.」 才可調用和存取：

例如：document.body 中，document是物件的instance ，而body 是document中的一個為property ，又 window.open(href)，中window是物件的instance 而open是方法。

## 參考

<[Object.create(): the New Way to Create Objects in JavaScript](http://www.htmlgoodies.com/beyond/javascript/object.create-the-new-way-to-create-objects-in-javascript.html) >

< [Basic Inheritance with Object.create](http://adripofjavascript.com/blog/drips/basic-inheritance-with-object-create.html) >

<[貓不會叫](http://javascript.wekeepcoding.com/article/15779199/Can%27t+access+methods+on+object%27s+prototype)>

< Data Structures: Objects and Arrays :[chapter 4](https://eloquentjavascript.net/04_data.html), >

# 基本

## Introduction

事實上，不同於其他物件導向程式語言，並沒有關鍵字class 來定義物件。一般的OOP定義物件的時候，除了利用類似class 的關鍵字以外，還會配合構建函數(constructor)。這個構建函數也是一個函數。而javascript 的物件宣告，利用的就是構建函數的觀念，從而省略class的宣告。

## 方法1：利用函數定義物件

JavaScript 利用函數function(),達成類別的宣告。如果以構建函數的眼光來看待這個函數，那麼自然而然，函數裡會有 this this 這個關鍵字，用來代表物件本身。

##### 方法定義在外部：

function Apple (type) {

this.type = type;

this.color = "red";

this.getInfo = getAppleInfo;

}

// anti-pattern! keep reading...

function getAppleInfo() {

return this.color + ' ' + this.type + ' apple';

}

上述物件變數的宣告：

var apple = new Apple('macintosh');

apple.color = "reddish";

alert(apple.getInfo());

在函數中使用屬性的時候，記得前綴詞[this](#refAnker_401516) ，否則有可能出現undefined variable 的訊息。

##### 物件的方法定義在內部

上述物件的方法定義在物件的外部，這樣就會產生一個問題，也就是在global namespece中，會有這個函數。為了避免函數名稱衝突，一般是定義在物件的內部，如下：

function Apple (type) {

this.type = type;

this.color = "red";

this.getInfo = function() {

return this.color + ' ' + this.type + ' apple';

};

}

### Methods added to the prototype

方法1的method 定義方式，我們有時候叫他靜態方法。因為每次建立物件的instance 的時候，都會有這幾個欄位()。此時，可以利用另一種方法，加入到prototype：

function Apple (type) {

this.type = type;

this.color = "red";

}

Apple.prototype.getInfo = function() {

return this.color + ' ' + this.type + ' apple';

};

<參考操作[[js]06 prototype](%5bJS%5d%2006%20prototype.pptx)>

##### static method



##### dynamic method



##### 和資料庫比較：

資料表(Monkey)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| name | height | weight | sex | needDiet() |
| Mary |  |  |  | ? |
| John |  |  |  | ? |
| Tom |  |  |  | ? |

## 方法2：Using object literals

定義物件也可以利用大刮號，例如：

**var** o **=** {};

可以取代下面的正式方法：

**var** o **=** **new** Object();

又例如陣列物件：

**var** a **=** [];

可以取代下面的正式方法：

**var** a **=** **new** Array();

下面的利用{} 省略類似類別的定義方式，並且直接建立物件的instance。

var apple = {

type: "macintosh",

color: "red",

getInfo: function () {

return this.color + ' ' + this.type + ' apple';

}

}

注意幾個關鍵字用法：屬性和方法的定義，是利用「:」（之前是「[＝](#refAnker_50863)」），欄位分隔是利用逗點「,」（之前是「[；](#refAnker_935648)」）

在這個例子中，我們並沒有經由類別的定義，宣告變數apple 而是直接就拿來使用，例如：

apple.color = "reddish";

alert(apple.getInfo());

這種物件的定義型態，有一個名詞可以對應 *singleton*. 。也就是說，這種物件只能用一次。

## 方法3：Singleton using a function

綜合上述兩種方式，可以推論也可如下定義 singleton：

var apple = new function() {

this.type = "macintosh";

this.color = "red";

this.getInfo = function () {

return this.color + ' ' + this.type + ' apple';

};

}

使用方式：

apple.color = "reddish";

alert(apple.getInfo());

從上面的範例，可以知道這是一個匿名的constructor function。然後利用關鍵字「new」來建立*instance.*

# 自行設計物件

問題：我養了一堆猴子，想在網路上建立隨時可以管理的資料，要怎樣設計「猴子」物件？

複製模版

設計物件變數（屬性）：

設計物件函數（方法）

測試

在設計變數的時候，必須常常想到這是所有猴子的共同屬性，例如名稱，重量，性別，相片等，可以知道是屬於特定一隻猴子的欄位。但是還有，其他像是平均體重等欄位，這不是屬於特定一隻猴子。

補充

# {} 和 []

Javascript（JS)中的大括弧{}和中括弧[]

一、{ } 大括弧，表示定義一個物件，大部分情況下要有成對的屬性和值，或是函式。如：

**var** employee **=** {"Name":"chung3","AGE":"28"};

上面聲明了一個名為「employee」的instance，物件內部的屬性和方法用「,」（逗號）隔開。使用時，必須透過「.」存取：例如，

employee.Name、employee.AGE。

但是除了上面的物件用法，也可以利用中刮號，以用陣列的方式來存取，如：

employee ["Name"] **===** employee.Name

employee["AGE"] **===**employee.Age。

該寫法，在JSON資料結構中經常用，除此之外，我們平時寫函式組的時候，也經常用到，如：

**var** employee **=** {

Name: **function**(){ //注意：是「：」不是「＝」

**return** "chung3";

},

Age: **function**(){

**return** "28";

}

}

呼叫方式差不多，因為是函式組，所以要加上()，如：alert( employee.Name() );

二、[ ]中括弧，表示一個陣列，也可以理解為一個陣列物件。

如：

**var** employee **=** [ "Name","employee","AGE","28" ];

很明顯，每個值或函式，都是獨立的，多個值之間只用,（逗號）隔開，因為是陣列物件，所以它等於：

**var** employee **=** Array( "Name","employee","AGE","28" );

存取時，也是和陣列一樣，alert( employee[0] );

三、{ } 和[ ] 混合使用，如：

**var** employee2 **=** { "Name":"chung3",

"boss":[ "LuLu","26" ],

"colleague":[{"Name":"one"},{"Name":"two"},{"Name":"three"}]

}

**>>**employee2["Name"]

"chung3"

**>>**employee2.Name

"chung3"

**>>**employee2["colleague"][0]

{Name: "one"}

**>>**employee2["colleague"][0]["Name"]

"one"

**>>** employee2.colleague[0].Name

"one"

注意：

如果要知道物件屬性的長度，則可以利用Object.keys()，例如

**>>** Object.keys(employee)

(2) ["name", "age"]

**>>** Object.keys(employee2)

(3) ["Name", "boss", "colleague"]

不是employee2.length 。

# 加入新函數

Math.max(1, 2, 3, 4); // 4

Math.min(1, 2, 3, 4); // 1

但是如果

x=[1,2,3,4];

Math.max(x) 無作用

Math.max.apply(null, x) // 4

Math.min.apply(null, x) // 1

更簡單的方式，透過 ES2015 的展開運算子來完成。

var numbers = [1, 2, 3, 4];

Math.max(...numbers) // 4

Math.min(...numbers) // 1

這個運算子可以在函式的參數中把陣列內的數值「展開」。

<註> 跳過

幫物件加入新函數

Function.prototype.apply() 允許你呼叫函式並給定一個 this 和一個\_陣列\_的參數。

原生的JavaScript 提供了Math.max/Math.min ，但是在Array物件中，並沒有這兩個函數，

var m=new Array(3,4,5);

console.log(Math.max(m));

但是也可以直接在Array物件中加入這兩個函數：

Array.**prototype**.max **=** **function**() {

**return** Math.max.apply(null, **this**);

};

Array.**prototype**.min **=** **function**() {

**return** Math.min.apply(null, **this**);

};

Augmenting the built-ins can cause collisions with other libraries (some see), so you may be more comfortable with just apply'ing Math.xxx() to your array directly:

**var** min **=** Math.min.apply(null, arr),

max **=** Math.max.apply(null, arr);

Alternately, assuming your browser supports ECMAScript 6, you can use the [spread operator](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Spread_operator)which functions similarly to the apply method:

**var** min **=** Math.min( ...arr ),

max **=** Math.max( ...arr );

</註>

【例】列印物件property

**function** listProp(obj, objName) {

//obj: object instance

//objName: 名稱

**for** (**var** prop **in** obj)

document.writeln(objName**+**".<font color=red>"**+**prop**+**"</font> = <font color=green>"**+**obj[prop]**+**"</font><br>");

}

部分結果：

|  |
| --- |
| document.location = http://mirlab.org/jang/books/javascript/example/docProp01.htm document.createElement = function (){var e=t.apply(this,arguments);return e&&"FORM"===e.nodeName&&e.submit&&(e.submit=u(e.submit)),e} document.fgColor =  document.linkColor =  document.vlinkColor = |