



# Chapter 04

## JavaScript的函數與物件

中原大學 資訊管理學系

賴錦慧 老師

[chlai@cycu.edu.tw](mailto:chlai@cycu.edu.tw)

# 大綱

- JavaScript的函數
- JavaScript函數的變數範圍
- JavaScript的物件
- 自訂JavaScript的物件
- JavaScript的Prototype物件

# JavaScript的函數



- JavaScript的內建函數
- 建立JavaScript自訂函數
- 擁有參數的JavaScript函數
- JavaScript函數的傳回值
- JavaScript函數的傳值或傳址參數
- JavaScript函數的參數陣列

# JavaScript的函數



## □ JavaScript函數

- 程式中一些共用程式碼獨立成**程式區塊**，能夠傳入參數和傳回執行結果
- 函數是將程式區塊的程式碼隱藏起來，**使用函數名稱進行呼叫和傳遞參數**
- 函數也是一種**物件**
- 一種JavaScript的「全域方法」（Global Methods）。

## □ JavaScript擁有兩種函數

- 內建函數
- 自訂函數

# JavaScript內建函數



## □ JavaScript擁有一些內建函數

- `parseInt()`和`parseFloat()`兩個函數（或稱為方法）來轉換變數的資料型態 (ch2)

## □ `escape()`函數

- 使用URL編碼傳入的參數字串，可以傳回加碼後的字串

```
strURLcode = escape(strMsg);
```

## □ `unescape()`函數

- 解碼參數的URL編碼字串，可以傳回還原成編碼前的原始字串

```
strOriginal = unescape(strURLcode);
```

# JavaScript自訂函數



- JavaScript函數是由**function**關鍵字、函數名稱和程式區塊組成

```
function writeString() {  
    document.write("歡迎使用JavaScript!<br/>");  
}
```

- 呼叫函數只需使用函數名稱  
writeString();

# 自訂函數-Example

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8"/>
<title>Ch4_1_2.html</title>
<script>
    function writeString(){
        document.write("歡迎使用JavaScript!<br/>");
    }
</script>
</head>
<body>
<h2>使用函數顯示文件內容</h2>
<hr/>
<script>
    // 呼叫函數
    writeString();
</script>
</body>
</html>
```

# 擁有參數的JavaScript函數



## □ JavaScript函數可以傳入1至多個參數

- 在呼叫函數時，只需傳入不同參數值就可以產生不同的執行結果

```
function writeNString(strMsg, intnumber) {  
    for(var i=1; i<=intnumber; i++) {  
        document.write(strMsg + "<br/>");  
    }  
}
```

- 參數如果不只一個，請使用「,」符號分隔
- 傳入參數的變數可以在函數的程式區塊中使用



# JavaScript函數的傳回值



## □ 傳回函數的執行結果

```
function sumToN(intNumber) {  
    var intSum = 0;  
    for(var i=1; i<=intNumber; i++) {  
        intSum += i;  
    }  
    return intSum;  
}
```

## □ 使用return關鍵字傳回函數的執行結果

## □ 函數有傳回值，在呼叫時通常是使用指定敘述來取得傳回值

```
var intSum = sumToN(10);
```

# JavaScript函數的傳值或傳址參數



## □ JavaScript函數的傳入參數擁有兩種參數傳遞方式

傳遞方式	說明
傳值	將變數值傳入函數，函數會另外配置記憶體空間來儲存參數值，所以不會變更原變數值
傳址	將變數實際儲存的記憶體位址傳入，如果在函數中變更參數值，也會同時變動原變數值

不同資料型態有不同的參數傳遞方式:

- 傳值: 數值、字串、布林
- 傳址: 物件、陣列、函數、字串物件



```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8"/>
<title>Ch4_1_5.html</title>
<script>
// number和boolean參數為傳值
function funcA(c, b){
    c++;
    b = false;
    document.write("在funcA
為 :"+c+"/"+b+"<br/>");
}

// object為傳址和字串參數為傳值
function funcB(objA, a){
    objA.name = "江小魚";
    a = "陳允傑";
    document.write("在funcB為 :
"+objA.name+"/"+a+"<br/>");
}
</script>
</head>
```

```
<body>
<h2>測試傳值和傳址的函數呼叫</h2>
<hr/>
<script>
// 宣告變數
var c = 1;      // 數值
var b = true;   // 布林
```

```
var a = "陳會安"; // 字串
var objA = new Object(); // 建立物件實例
objA.name = "陳會安";
document.write("呼叫funcA前 : "+c+"/"+b+"<br/>");
```

**傳值** funcA(c,b); // 呼叫函數

```
document.write("呼叫funcA後 : "+c+"/"+b+"<br/>");
document.write("呼叫funcB前 : "+objA.name+"/"+a+"<br/>");
傳址 funcB(objA, a); // 呼叫函數
document.write("呼叫funcB後 : "+objA.name+"/"+a+"<br/>");
```

```
</script>
</body>
</html>
```

## 測試傳值和傳址的函數呼叫

```
呼叫funcA前 : 1/true
在funcA為 :2/false
呼叫funcA後 : 1/true
呼叫funcB前 : 陳會安/陳會安
在funcB為 : 江小魚/陳允傑
呼叫funcB後 : 江小魚/陳會安
```

# 練習1

- 請設計JavaScript的函數, 並利用傳值方式傳遞參數, 計算 $N!$ 的結果
  - \*\*注意: 函數中要傳入參數, 並傳回計算結果

# JavaScript函數的參數陣列-說明



- 「參數陣列」 ( Arguments Array ) 物件
  - 當呼叫函數傳入參數時，函數就算沒有指明參數名稱，一樣可以使用參數陣列的物件取得參數個數和個別參數值

```
function sumInt() {  
    ...  
}
```

- 可以在呼叫時傳遞參數

```
sumInt(100,45,567,234);
```

# JavaScript函數的參數陣列-取出



- 使用arguments物件的length屬性取得傳遞多少個參數  
`sumInt.arguments.length;`
- 使用陣列索引取得傳入函數的個別參數  
`sumInt.arguments[0];`  
`sumInt.arguments[1];`  
`sumInt.arguments[2];`  
`sumInt.arguments[3];`
- 陣列索引是以0開始，以上述函數呼叫為例，取得的參數值依序為100、45、567和234。

# JavaScript函數的變數範圍



- 變數範圍會影響程式碼的變數存取
- JavaScript擁有兩種變數範圍
  - 區域變數（ Local Variables ）：在函數內宣告的變數，變數只能在函數程式區塊之中使用，函數之外的程式碼並無法存取此變數。
  - 全域變數（ Global Variables ）：如果變數是在函數外宣告，整個JavaScript程式檔的函數和程式碼都可以存取此變數。

# JavaScript的物件



- 物件導向程式語言
- JavaScript的物件、屬性和方法
- JavaScript支援的物件



# 物件導向程式語言-1

- 「物件導向程式語言」 ( Object-oriented Language ) 的三種特性

## 封裝 ( Encapsulation )

- 封裝是將資料和函數建立成物件，這些函數稱為方法 ( Methods )
- 在物件導向程式語言定義物件是使用「類別」 ( Class )，即建立一種抽象資料型態
- JavaScript並沒有類別，可以使用建構函數來建立物件。



〔物件〕至少有表示性質或狀態的資訊，也就是屬性(property)與方法(method)。

以程式面來看，花的屬性與方法

屬性(property)	方法(method)
花朵的顏色	花隨風擺動
花朵的大小	行光合作用
花瓣的數目	發出新芽
葉子的形狀	.....
.....	



# 物件導向程式語言-2



## 繼承 ( inheritance )

- 繼承是物件的再利用
- 當定義一個類別後，其他類別可以繼承此類別的屬性和方法，並且新增或取代繼承物件的屬性和方法來擴充其功能。
- JavaScript是使用Prototype物件來實作繼承。

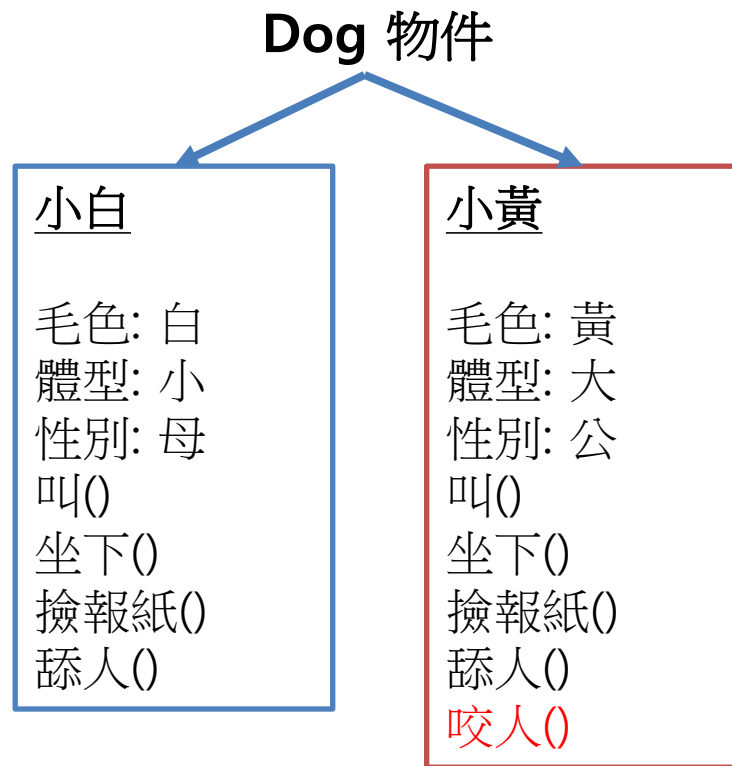
## 多形 ( Polymorphism )

- 多形是物件導向最複雜的特性
- 類別如果需要處理各種不同資料型態，只需繼承基礎資料型態的類別，擴充此類別建立同名方法來處理各種不同資料型態，因為方法的名稱相同，只是參數和程式碼不同，所以也稱為同名異式。

# 繼承- Example

## Dog 物件定義

屬性	方法 (函數)
毛色	叫 ()
體型	坐下()
性別	撿報紙()
	舔人()





# 參考資料—物件導向基礎

- 物件導向基礎概念
  - <http://expect7.pixnet.net/blog/post/38682120>
- Javascript物件導向
  - [https://developer.mozilla.org/zh-TW/docs/JavaScript\\_%E7%89%A9%E4%BB%B6%E5%B0%8E%E5%90%91%E4%BB%8B%E7%B4%B9](https://developer.mozilla.org/zh-TW/docs/JavaScript_%E7%89%A9%E4%BB%B6%E5%B0%8E%E5%90%91%E4%BB%8B%E7%B4%B9)
- Javascript 物件導向式程式設計基礎講解
  - <http://inspiregate.com/programming/javascript/291-javascript-based-object-oriented-programming-explained.html>
- Javascript物件導向基礎概念 | 物件、屬性和方法
  - <http://www.ucamc.com/e-learning/javascript/47-javascript-object-oriented.html>

# JavaScript的物件、屬性和方法-1



## 物件 ( Objects )

- 物件是資料 ( **Data** ) 和處理資料函數的綜合體
- 只需知道物件提供那些屬性 ( 資料 ) 、方法 ( 處理資料的函數 ) 、和如何使用這些屬性和方法即可。

- JavaScript物件只是名稱和值成對的集合，即「物件文字值」 ( **Object Literals** )

```
var objStudent = {  
    name : "陳允傑",  
    age : 5  
};
```

# JavaScript的物件、屬性和方法-2



## 屬性 ( Properties )

- 物件屬性可以存取物件儲存的資料
  - 例如：String物件的String.length屬性，可以取得字串長度
- 存取物件屬性是使用「.」運算子  
`objName.propertyName;`

## 方法 ( Methods )

- JavaScript物件的方法是用來處理物件儲存資料的函數
    - 例如：String物件擁有String.substr()方法，其處理的就是字串物件的內容
- `objName.methodName();`

# JavaScript支援的物件



## □ 內建物件 ( Intrinsic Objects )

- JavaScript提供十一種內建物件Array、Boolean、Date、Function、Global、Math、Number、Object、RegExp、Error和String物件(請參閱第5章)

## □ 自訂物件 ( Custom Objects )

- JavaScript能夠建立使用者自訂的物件，擴充JavaScript的功能(ch4-4)

## □ 瀏覽器物件 ( Host Objects )

- 瀏覽器物件是由瀏覽器提供的物件
  - 以Internet Explorer來說，稱為「BOM」( Browser Object Model )，這是一種階層架構的物件模型。





# 自訂JavaScript的物件

- 使用Object物件建立自訂物件
- with程式區塊
- 使用建構函數來建立物件
- 物件的階層架構
- 新增物件的方法

# 使用Object物件建立自訂物件-1



- 在JavaScript可以直接建立Object物件後，新增所需的屬性和方法來建立自訂物件

```
var objCard = new Object();  
objCard.name = "陳會安";  
objCard.age = 42;  
objCard.phone = "02-22222222";  
objCard.email = "hueyan@ms2.hinet.net";
```

# 使用Object物件建立自訂物件-2



- 可以使用物件文字值（ Object Literal ）建立物件

```
var objCard = {  
  name : "陳會安",  
  age : 42,  
  phone: "02-22222222",  
  email: "hueyan@ms2.hinet.net"  
};
```

# with程式區塊

- with程式區塊能夠針對物件建立程式區塊，在程式區塊的程式碼不需要指明物件名稱，即可新增屬性和顯示屬性內容

```
var objCard = new Object();
with(objCard) {
    name = "陳會安";
    age = 42;

    .....
    document.write("姓名：" + name + "<br/>");
    document.write("年齡：" + age + "<br/>");
    .....
}
```

# 使用建構函數來建立物件-說明



- 「建構函數」 ( Constructor Function )
  - 定義物件的屬性和方法，在程式範例Ch4\_4\_1.htm的自訂物件是使用內建建構函數Object()，
  - 所謂JavaScript內建物件就是一些預設的建構函數
    - 例如：String物件就是String()；Array物件是Array()建構函數等。

# 使用建構函數來建立物件-步驟1



## 步驟一：使用建構函數宣告物件

□ 定義物件的建構函數，建構函數的語法是一個 JavaScript 函數，在 **建構函數** 可以定義物件屬性和方法

□ 此為一個物件宣告（但它並不是類別）

```
function nameCard(name,age,phone,email) {  
    this.name = name;  
    this.age = age;  
    this.phone = phone;  
    this.email = email;  
}
```

1. 參考物件本身
2. 表示指定物件

# 使用建構函數來建立物件-步驟2



## 步驟二：使用new運算子建立物件

- 在定義宣告物件的建構函數後，就可以使用new運算子建立物件

```
var objMyCard = new nameCard("陳會安", 42,  
    "02-22222222","hueyan@ms2.hinet.net");
```

- 可以在建立後再指定物件的屬性值：

```
var objCard = new nameCard();  
objCard.name = "江小魚";  
objCard.age = 35;  
objCard.phone = "03-33333333";  
objCard.email = "hueyan@yahoo.com.tw";
```

# 物件的階層架構-說明

- 「物件的階層架構」 ( Object Hierarchy )
  - 物件屬性可以是另一個子物件，可以建立階層關係的物件架構
  - 例如：nameCard物件擁有子物件phoneList，這個子物件是用來儲存住家電話和手機電話號碼。




# 物件的階層架構-範例

- 例如：nameCard物件擁有子物件phoneList，它是使用phoneList()建構函數，如下所示：

```
function nameCard(name,age,phone,email) {  
    this.name = name;  
    this.age = age;  
    this.phone = new phoneList(phone, "N/A");  
    this.email = email;  
}
```

```
function phoneList(homephone,cellphone) {  
    this.homephone = homephone;  
    this.cellphone = cellphone;  
}
```



使用物件:

```
var objMyCard = new nameCard();  
objMyCard.cellphone= " 090-666666 ";
```

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8"/>
<title>Ch4_4_3.html</title>
<script>
// 物件的建構函數
function nameCard(name,age,phone,email) {
    this.name = name;
    this.age = age;
    this.phone = phone;
    this.email = email;
}
</script>
</head>
```

```
<body>
<h2>使用建構函數建立物件</h2>
<hr/>
<script>
// 建立自訂物件
var objMyCard = new nameCard("陳會安", 42,
                              "02-22222222","hueyan@ms2.hinet.net");

var objCard = new nameCard(); // 建立物件
// 設定屬性
objCard.name = "江小魚";
objCard.age = 35;
objCard.phone = "03-33333333";
objCard.email = "hueyan@yahoo.com.tw";

// 顯示objMyCard物件屬性
document.write("姓名：" + objMyCard.name + "<br/>");
document.write("年齡：" + objMyCard.age + "<br/>");
document.write("電話：" + objMyCard.phone + "<br/>");
document.write("郵件：" + objMyCard.email + "<br/> <hr/>");

// 顯示objCard物件屬性
document.write("姓名：" + objCard.name + "<br/>");
document.write("年齡：" + objCard.age + "<br/>");
document.write("電話：" + objCard.phone + "<br/>");
document.write("郵件：" + objCard.email + "<br/>");
</script>
</body>
</html>
```

# 新增物件的方法

## □ 以新增物件方法來顯示物件的屬性值

- 例如：在nameCard物件新增print()方法顯示名片資料：

```
function nameCard(name,age,phone,email) {  
    this.name = name;  
    this.age = age;  
    this.phone = phone;  
    this.email = email;  
    this.print = printCard;  
}
```

- 上述建構函數nameCard()最後的print是一個方法，值printCard就是指向參考的printCard()函數。

```
function printCard() {  
    document.write("姓名：" + this.name + "<br/>");  
    document.write("年齡：" + this.age + "<br/>");  
    document.write("電話：" + this.phone + "<br/>");  
    document.write("郵件：" + this.email + "<br/><hr/>");  
}
```

# 練習2

- 請使用建構函數和物件文字值來建立圖書資料物件book,其中包含bookid, title, author,price屬性來儲存書號、書名、作者和書價 (P.4-44)
  - 請使用物件儲存至少2筆資料,並把物件內容顯示於網頁中

---

書籍ID : A12345  
書名 : 小王子  
作者 : prince  
價格 : \$350

---

---

書籍ID : B12255  
書名 : 飢餓遊戲  
作者 : David  
價格 : \$490

---

# JavaScript的Prototype物件



- 類別基礎和原型基礎程式語言
- 新增Prototype物件的屬性
- 新增Prototype物件的方法
- 擴充JavaScript內建物件的方法
- Prototype物件的繼承

# JavaScript的Prototype物件



## □ JavaScript支援Prototype物件

- 新增物件的屬性或方法
- 實作Prototype物件的繼承

## □ JavaScript是一種「原型基礎」( Prototype-based ) 程式語言，不同於C++、Java或C#的「類別基礎」( Class-based ) 程式語言。

# 類別基礎和原型基礎程式語言-1



## 類別基礎和原型基礎程式語言

### □ 類別基礎程式語言

- 類別 ( Class ) 是一種抽象資料型態，它和物件實例 ( Instance ) 是不同的，我們使用類別的藍圖來建立物件實例

### □ 原型基礎程式語言

- 類別和物件之間分野並不明顯，類別事實上就是物件。
- 物件在原型基礎程式語言屬於一個實際的實體，可以使用現成的物件作為原型 ( Prototype ) 來建立其他物件，或使用Prototype物件來繼承其他物件

# 類別基礎和原型基礎程式語言-2



## 物件的prototype屬性

- JavaScript的每一個物件都擁有prototype屬性，  
這個屬性是一個Prototype物件
- Prototype物件的屬性會被所有物件所繼承
- prototype屬性的優點
  - 使用prototype屬性擴充物件可以大量減少物件使用的記憶體空間
  - 不論是否已經建立物件，都可以使用prototype屬性來擴充物件的屬性和方法



# 新增Prototype物件的屬性

- JavaScript的prototype屬性能夠擴充JavaScript內建物件或自訂物件的屬性
  - 例如：在自訂物件circle建立PI屬性  
`circle.prototype.PI = 3.1415926;`
  - prototype屬性在所有建立的物件都會新增PI屬性

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8"/>
<title>Ch4_5_2.html</title>
<script>
// 物件的建構函數
function circle(r, color) {
    this.r = r;
    this.color = color;
    this.display = showCircle;
}

// 物件方法
function showCircle() {
    document.write("半徑：" + this.r + "<br/>");
    document.write("色彩：" + this.color +
"<br/>");
    document.write("圓周率：" + this.PI +
"<br/><hr/>");
}
</script>
</head>
```

```
<body>
<h2>新增Prototype物件的屬性</h2>
<hr/>
<script>
// 建立自訂物件
var objCircle1 = new circle(2, "red");
var objCircle2 = new circle(3, "green");

// 新增Prototype物件的屬性
circle.prototype.PI = 3.1415926;

// 執行物件方法
objCircle1.display();
// 執行物件方法
objCircle2.display();

</script>
</body>
</html>
```

### 新增Prototype物件的屬性

半徑：2  
色彩：red  
圓周率：3.1415926

半徑：3  
色彩：green  
圓周率：3.1415926

# 新增Prototype物件的方法

- 使用prototype屬性新增area()方法(程式範例 Ch4\_5\_2.htm) ，計算圓面積的getArea()函數

```
function getArea() {  
    var result = this.PI * this.r * this.r;  
    document.write("圓面積：" + result + "<br/><hr/>");  
}
```

- 使用prototype屬性新增的方法：  
circle.prototype.area = getArea;

# 擴充JavaScript內建物件的方法



- 對於JavaScript內建物件，使用Prototype物件新增物件的方法


```
var objMessage=new String("JavaScript網頁程式設計");
```

- 只需使用prototype屬性就可以新增String物件的方法

```
String.prototype.reverse = reverse_string;
```

```
String.prototype.even = even_string;
```

- 程式碼新增String物件的reverse()和even()方法



```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8"/>
<title>Ch4_5_4.html</title>
<script>
// 新增的物件方法
function reverse_string() {
    for (var i = (this.length-1); i >= 0; i--)
        document.write(this.charAt(i));
    document.write("<br/>");
}
// 新增的物件方法
function even_string() {
    var output = "";
    for (var i = 0; i < this.length; i+=2)
        output += this.charAt(i);
    return output;
}
// 擴充物件方法
String.prototype.reverse = reverse_string;
String.prototype.even = even_string;
</script>
</head>
```

```
<body>
<h2>擴充JavaScript內建物件的方法</h2>
<hr/>
<script>
// 建立內建物件String
var objMessage = new String("JavaScript網頁程式設計");
document.write("原始字串: " + objMessage + "<br/>");

// 執行物件方法
objMessage.reverse();
strOutput = objMessage.even();
document.write(strOutput + "<br/>");
</script>
</body>
</html>
```

## 擴充JavaScript內建物件的方法

原始字串: JavaScript網頁程式設計  
計設式程頁網tpircSavaJ  
JvSrp網程設

# Prototype物件的繼承-父



- JavaScript物件的繼承可以將一個物件擴充成其他物件

- 例如：position物件的建構函數：

```
function position(x, y, color) {  
    this.x = x;  
    this.y = y;  
    this.color = color;  
}
```

- position()建構函數定義圖形的基本資料，包含位置x、y和色彩color屬性。

# Prototype物件的繼承-子



## □ 建立circle物件繼承position物件

```
function circle(r) {  
    this.r = r;  
    this.info = showCircleInfo;  
  
    function showCircleInfo() {  
        var result = 3.1415926 * this.r * this.r;  
        document.write("半徑：" + this.r + "<br/>");  
        document.write("X座標：" + this.x + "<br/>");  
        document.write("Y座標：" + this.y + "<br/>");  
        document.write("圖形色彩：" + this.color + "<br/>");  
        document.write("圓面積：" + result + "<br/>");  
    }  
}
```

# Prototype物件的繼承-繼承



- 在circle物件使用prototype屬性繼承position物件

`circle.prototype = new position();`

- circle物件就可以繼承position物件的屬性和方法。



# 練習3

- 延續練習2, (1)請利用prototype新增書籍的數量屬性-amount, 以及(2)新增showTotal()方法以計算總金額(price\*amount)