行動網頁前端開發技術

JavaScript 程式設計 – 基本入門 (上)

講師：陳俊吉
ccckluke@gmail.com
2016

課程大綱

- 網頁技術架構與MVC架構
- JavaScript概要
- 程式設計基本觀念
- 變數、變數型別
- 基本演算
- 函數基本概念
- 條件判斷指令
- 演算結果的三種輸出方式

網頁技術架構

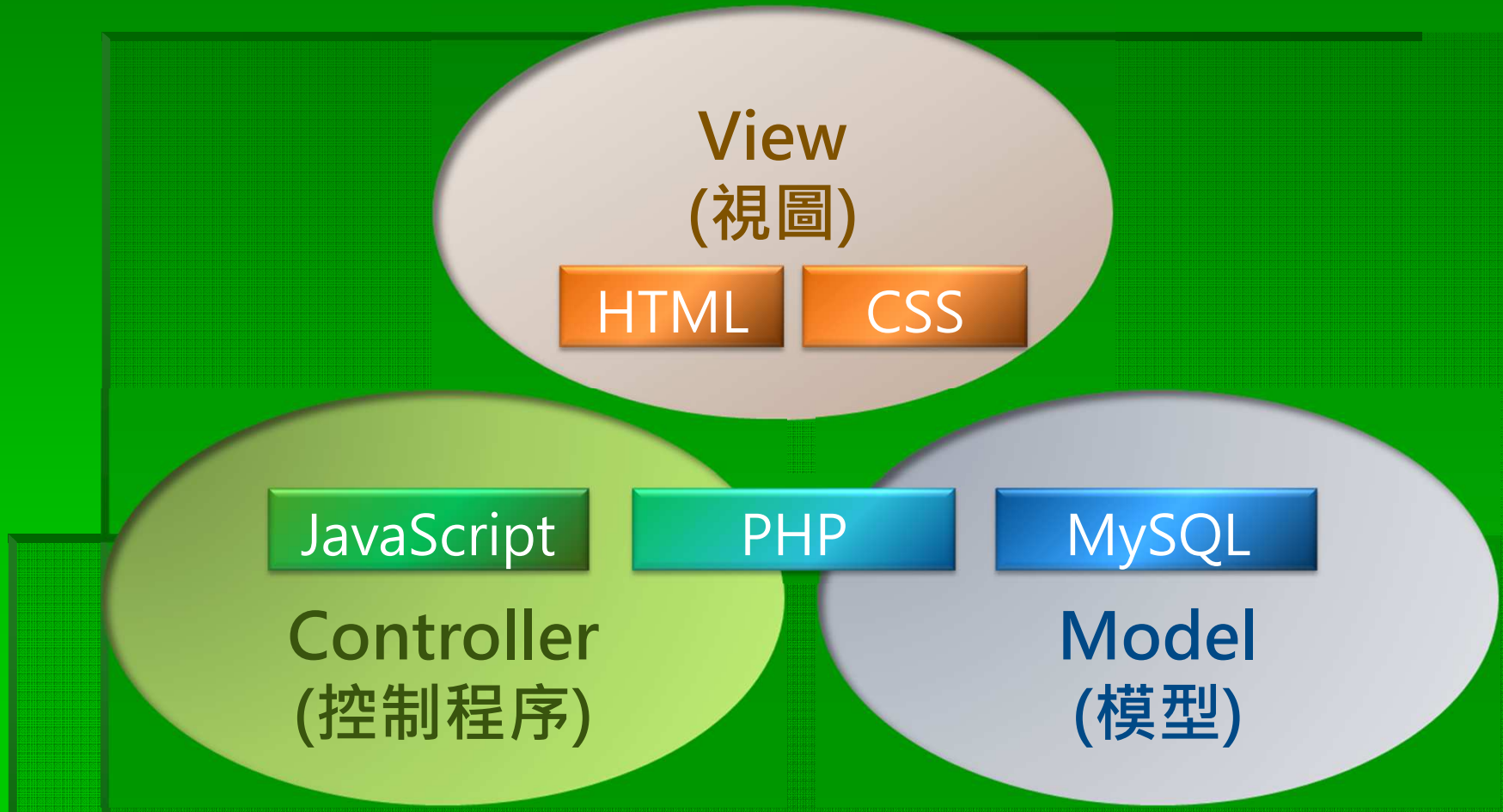
前端（使用者端）

- 內容組成
  - HTML
- 外觀呈現
  - CSS
- 互動處理
  - JavaScript

後端（伺服器端）

- 網頁伺服器（Web Server）
  - Apache
- 腳本程式引擎（Script Engine）
  - PHP、JSP、ASP... (__P)
- 關連式資料庫（RDB, Relation Database）
  - MySQL、MS SQL... (__SQL)

網頁技術MVC架構


View
(視圖)
HTML
CSS
JavaScript
PHP
MySQL
Controller
(控制程序)
Model
(模型)

JavaScript概要

- 當前世界上，可攜性最高、執行環境最廣泛和普及的程式語言。
- 作用與目的
  - 畫面動態（動畫）
  - 互動操作（事件驅動）
  - 資料處理（驗證、演算...）
  - 後端通訊（資料存取）
    - 後兩者為HTML5（Web App）強化與改良重點

JavaScript概要

- 最新趨勢：
  - 行動平台互動、後端伺服器程式、系統硬體控制
  - 除了如jQuery、Ext JS（Sencha）、AngularJS這類前端程式庫或程式框架外，還有具備後端功能的NodeJS。
  - 參考網站：
    - jQuery Mobile：http://jquery.com/
    - Sencha Touch：http://www.sencha.com/
    - AngularJS：http://angularjs.org/
    - NodeJS：http://nodejs.org/
- 為何學？
  - 目的：網頁應用程式、行動裝置Apps、Unity…

程式設計基本概念

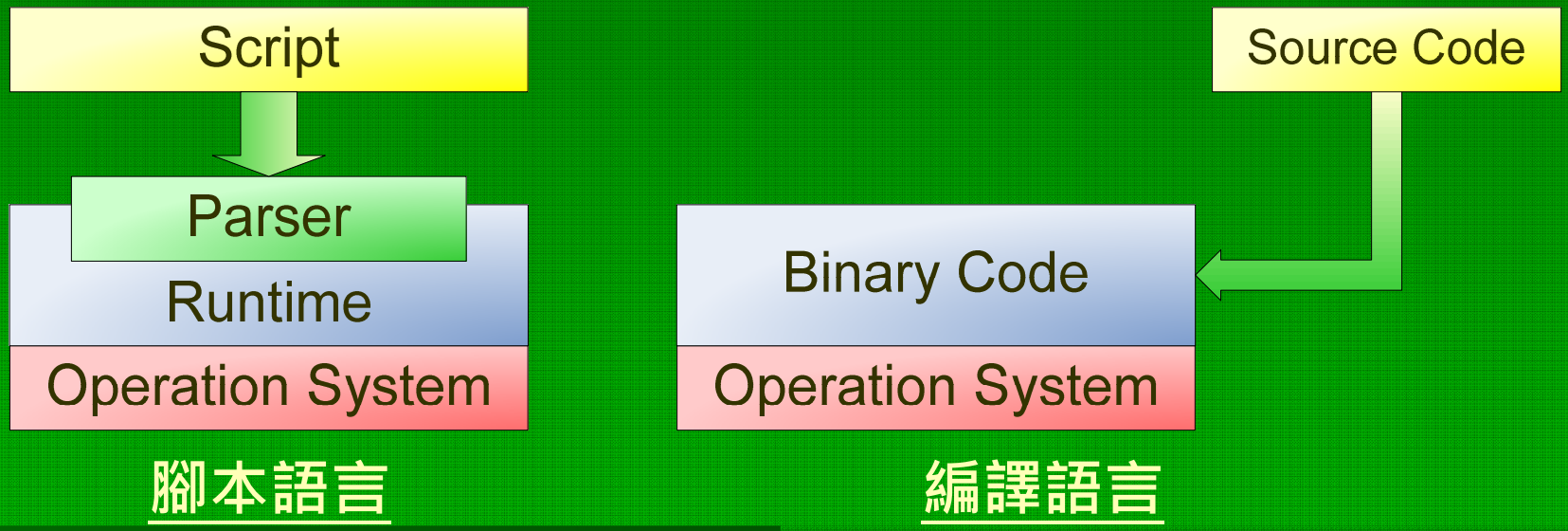
- 程式（Program）
  - [名] 節目、大綱、計劃
  - [動] 安排節目、規劃大綱、做計畫
  - 以某種程式語言，依據處理「程」序所編寫出來的「式」子，就是「程式」。

腳本語言v.s編譯語言

- 腳本語言（Script）：
  - 以原始碼存在，直接執行於適當的腳本執行環境（runtime）上的一種語言形式。
  - ex: JavaScript、ActionScript、Lua...
- 語言/編譯語言（Compiled Language）：
  - 用來表達程式命令與處理流程的形式。
  - 編寫完成後，會被轉換/編譯（compile）成專屬的機器執行代碼。
  - ex: C、Java、VB...

腳本語言v.s編譯語言

- 執行情況示意圖：



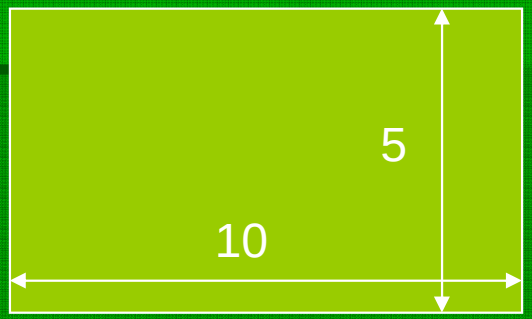
- 概念釐清：
  - JavaScript是一種腳本語言
  - JavaScript ≠ Java

電腦強大威力的核心概念

- 電腦（Computer）的兩力：
  - 記憶力（Remember）：
    - 保存資料，以便存取使用。
  - 計算力（Calculator）：
    - 進行演算與流程處理。
  - 電腦所能完成的任何工作，都是靠這兩力的交互作用來達成。
- 案例：如何求得物體的面積大小？

案例：如何求得物體的面積大小？

- 問題：
  - 長度為 10 個單位，寬度為 5 個單位的矩形範圍面積是多少平方單位？
- 公式：
  - 面積 = 長 x 寬 = 10 x 5



開始寫程式 - 以 JavaScript 來實現

- 編輯與執行：
  - 編輯程式
    - 使用DW開啟新的網頁檔案，並編輯右側範例內容：
  - 執行程式
    - 使用瀏覽器直接開啟該網頁。
- 討論：
  - 兩種形式的差別？

形式一

```
<html>
<head>
</head>
<body>
  <script>
    var size;
    size = 10 * 5;
    alert(size);
  </script>
</body>
</html>
```

形式二

```
<html>
<head>
  <script>
    var size;
    size = 10 * 5;
    alert(size);
  </script>
</head>
<body>
</body>
</html>
```

程式基礎概念

- 陳述式（Expression）
  - 以符合文法格式的方式，將需要進行的演算作業給陳述表達而成的式子。相當於文章中的每個句子一樣。
  - 每個式子最後，都會以一個「;」（分號字元）為結束。相當於文章中，每個句子最後的「。」（句號）。
- 文法（Syntax）
  - 針對每一種語言，都會有其規範的表達格式，如此，執行環境才有辦法正確解讀並執行陳述式當中的內容。

```
<html>
<head>
</head>
<body>
  <script>
    var size;
    size = 10 * 5;
    alert(size);
  </script>
</body>
</html>
```

陳述式

程式基礎概念

- 變數（Variable）
  - 在電腦程式裡頭，可用來保存資料內容的記憶體空間。
  - 由於該記憶體空間所保存的內容，可以被隨時放入其他不同資料。因此，該記憶體內容就是屬於「可變化的數據資料」，故被簡稱為「變數」。
- 宣告（Declare）：
  - 指示電腦系統配置適當記憶體空間，以便進行資料保存作業的動作，稱之為「宣告」。

運算方式

- 運算
  - 透過「運算子」（operator）來對資料或變數進行演算。
- 基本運算：
  - 加減乘除：+、-、*、/
- 遞增運算：
  - 變數前面或後面加上這類運算子，就會在該陳述式演算前或演算後，對該變數數值進行遞增或遞減。
  - 數值遞增 1 ：++　　本身數值遞減 1： --
- 字串串接：
  - 使用「+」字元來把前後字串給串接成一個單一字串。
  - 例如：”abc” + “CDE” => “abcCDE”

變數

- 變數宣告：
  - var 變數名稱;
- 變數使用：
  - 指派（儲存）：變數 = 資料內容;
  - 取出（讀取）：變數名稱
  - 範例：

```
var size;          // 變數宣告
size = 10 * 5;     // 變數儲存
alert(size);       // 變數取出
```

- 練習：
  - 熟悉變數宣告、存取、演算，並顯示結果。

關於變數名稱

- 被配置好的記憶體空間，需要起個名字，才能夠被正確存取。
- 基本命名規則
  - 可用字元：英數字、底線字元
  - 第一個字不得為數字
  - 不可使用關鍵字或保留字
- 命名慣例
  - 以有意義的英文字命名
  - 第一個字小寫，後面每個詞的第一個字母大寫
    - 例如：myAge、todayHour...

變數與資料型別

- 型別（type）
  - 在變數裡頭，可儲存數值、字串、物件等類型的資料。而這些資料的「類型」，就被稱之為「型別」。
- 常見型別：
  - 數值（integer, float）
    可用來進行加減乘除等算數運算的資料。
    - 例如：10, 3.14, -128 …
  - 字串（string）
    以單純的字元內容所構成的一串文字，而在程式碼裡頭，一段字串的兩邊必須用雙引號（”）或單引號（’）字元刮住。
    - 例如：”文字內容”，“text string” …

變數與資料型別

- 常見型別（續）：
  - 布林值（Boolean）
    邏輯判斷後，結果不是 true，就是 false。
    這種特殊值就是所謂的「布林值」。
    - 例如：true, false …
  - 物件（object）
    由各種變數和函數所組成的一種整體結構。
    - 例如：document, Array …

功能/函數（Function）- 概念

- 在數學當中，「函數」代表的是一段複雜數學式子的集合。
- 例如：
  - f(x) = x2 + 2x + 1
  - f(x, y) = x2 + 2xy + y2
- 在程式當中，「函數」就是一連串複雜陳述式的集合。由於這樣的函數通常都能完成某些特定功能，也因此也被通稱為「功能函數」。

呼叫功能/函數（Function）

- 函數寫法就跟傳統數學一樣，直接在函數名稱後面加上一組小括弧即可。
- 簡易呼叫：
  - 直接寫出（呼叫）要使用的功能函數。
    - 函數名稱( );
- 參數呼叫：
  - 寫出要使用的功能函數，並在括弧中填入需要搭配處理的參數（parameter）。
    - 函數名稱(參數1, 參數2, ...);
- 範例：
  - alert("顯示訊息");

```
var size;          // 變數宣告
size = 10 * 5;     // 變數儲存
alert(size);       // 呼叫（使用）函數
```

演算結果的三種輸出方式

- 提醒視窗顯示（對話方塊輸出）：
  - alert(輸出內容);
- 文件內容輸出（網頁直接輸出）：
  - document.write(輸出內容);
- 主控台紀錄（背景紀錄輸出）：
  - console.log(輸出內容);
- 備註：
  - 每種輸出方式有不同的實際效益，未來應該根據不同需求來選用適當方式。
- 練習：
  - 實際體驗各種輸出效果。

案例：喜歡就帶回家！(但額度夠嗎？)

- 思考方向：
  - 假如金額在額度上限以內，
    就購買；
    否則無法購買。
    （相反邏輯：假如超過額度，就無法購買）
- 關鍵：
  - 依據某些邏輯條件狀況，
    來決定要處理的方式或程序。

條件判斷指令 Part 1：if

- 基本語法：

```
if( 邏輯條件式 ) {
  // 條件成立時的處理作業
}
```

- 運作方式：
  - 假如當下的邏輯條件式（通常是一種比較運算）正確（成立），就會執行緊接在後的處理作業；要是條件式的情況錯誤（不成立），則會忽略（跳過）緊接在後的處理作業。

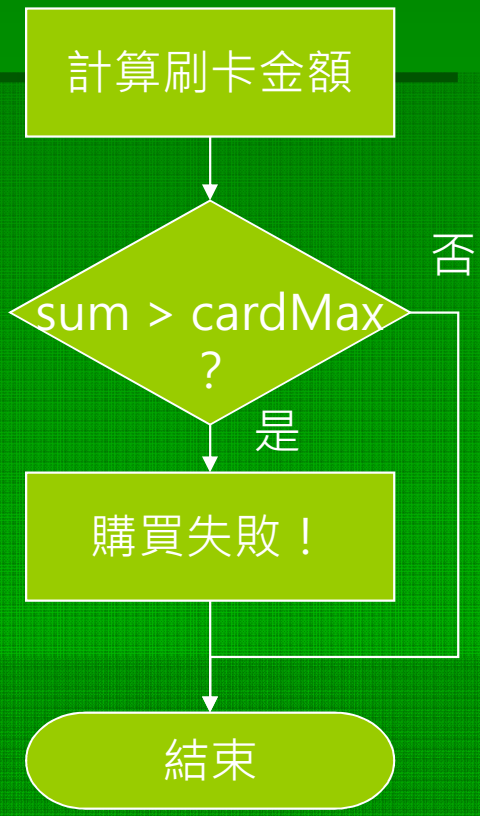
條件判斷指令 Part 1：if

- 比較運算：
  - >（大於）、>=（大於或等於）
  - <（小於）、<=（小於或等於）
  - ==（內容值相等）、!=（內容值不相等）
  - ===（完全相等）、!==（完全不相等）
- 比較結果：
  - 邏輯條件正確，結果即為 true（真/是）
  - 邏輯條件錯誤，結果即為 false（偽/否）
- 練習：
  - 測試各種比較方式，並將結果輸出。
  - 數值1和字串“1”比較結果為何？

條件判斷指令 Part 1：if

- 實作：

```
var sum;// 購物總金額
var cardMax;       // 卡片上限
sum = 100 + 200;
cardMax = 500;
if(sum > cardMax) {
  alert("卡刷爆了！購買失敗！");
}
```



條件判斷指令 Part 2：else、else if

- 當狀況和條件更為複雜時，「假如...則...」的形式就不夠用了。可能需要「假如...則...，否則...」，甚至是「假如...則...，或者假如...，則...」之類的邏輯判斷處理。
- 基本語法：

```
if( 邏輯條件式 ) {
  // 條件成立時的處理作業
}
else {
  // 條件不成立時的處理作業
}
```

```
if( 邏輯條件式1 ) {
  // 條件式1成立時的處理作業
}
else if (邏輯條件式2) {
  // 條件式2成立時的處理作業
}
    :
else {
  // 條件式n不成立時的處理作業
}
```

程式區塊：{和}

- 當需要以一行以上（多行）陳述式為一整個群組（或處理範圍）時，可以使用程式區塊起始字元（{，左大括弧）和程式區塊結束字元（}，右大括弧 ）來圍住某範圍的多行陳述式。
- 範例：

```
var sum;              // 購物總金額
var cardMax;          // 卡片上限
sum = 100 + 200;
cardMax = 500;
If (sum > cardMax) {
  sum = 0;            // 清除購物金額
  alert("卡刷爆了！購買失敗！");
}
```

註解文字

- 程式碼裡頭，若要加入註記文字，以便之後能輕易解讀原始碼的話，通常會以「//」或「/* ... */」符號，來標示註解的說明文字。
- 註解只是單純的註記文字，並不會被執行，因此可以隨心所欲編寫任何內容。
- 範例：

```
var sum;              // 購物總金額
var cardMax;          // 卡片上限
sum = 100 + 200;
cardMax = 500;
If (sum > cardMax) {
  sum = 0;            // 清除購物金額
  alert("卡刷爆了！購買失敗！");
}
```

註解文字

- 單行註解：
  - 陳述式中，從「//」開始，到該行最後字元為止的內容，全都是不會被執行的註解。
- 多行註解：
  - 在註解開始的「/*」符號，到註解結束的「*/」之間的所有內容，都是註解文字。而裡頭同時可以包含多行內容。

```
/*
  計算刷卡情況
  if … else … 示範
*/
var sum = 100 + 200;          // 購物總金額
var cardMax = 500;            // 卡片上限
If (sum > cardMax) {
  sum = 0;                    // 清除購物金額
  alert("卡刷爆了！購買失敗！");
}
else
  alert(“順利購買完成！” );
```

常用內建函數

- 轉換函數：
  - 字串轉成整數：
    - parseInt(整數字串);
  - 字串轉成小數：
    - parseFloat(小數字串);
- 物件（或數值）轉成字串：
  - String(物件);
- 練習：
  - 測試並檢查內建函數的輸出結果。