// 5. 回乎函式Callback!

// add: 用來作加法

// n1: 數字

// n2: 數字

// callback:函式(加法的結果)

function add(n1, n2, callback /\* 預期callback是一個函式\*/){ //用callback就不會寫死，每次add可以做不同的事

callback(n1 + n2);

}

function handler(result){ //handler代表函式的本體!!

alert(result)

}

add(3, 4, handler); //可以這樣!

add(10, 5 , function(result){ //更可以這樣! 直接把"匿名函式"放在這裡

document.title = result; //讓標題變成15

});

add(11, 6 , (result)=>{ //更可以這樣! 直接把"箭頭函式"放在這裡

document.title = result; //讓標題變成17

});

// 6. JSON(Javascript Object Notation)

// 以下表p 有x, y, get三個成員

let p = new Object();

p.x = 10;

p.y = 5;

p.get = function(){

alert(this.x + ',' + this.y);

}

p.get();

// 7. 用JSON語法完成物件設計 (物件) (在大量使用物件會比較輕鬆)

// JSON >　製造物件的語法！

let p={x:10, y:5, get: function(){

alert(this.x + ',' + this.y);

}};

p.get();

// 8. add: 用來作加法

// args: 物件

// n1: 數字

// n2: 數字

// callback: 函式(加法的結果)

// 但這裡很冗長很累

function add(args){

args.callback(args.n1+args.n2)

}

let data = new Object();

data.n1 = 3;

data.n2 = 5;

data.callback = function(result){

alert(result);

}

add(data);

<很棒的寫法!!>

// 9. add: 用來作加法

// args: 物件

// n1: 數字

// n2: 數字

// mycallback: 函式(加法的結果)

// 這樣比較輕鬆

function add(args){

args.mycallback(args.n1+args.n2)

}

add({n1:3, n2:4, mycallback:function(result){

alert(result);

}});

11. (Javascript沒有類別的概念，是用建構式!!!)

建立建構子, 快速建立點物件

// 11. 建立建構子, 快速建立點物件

function Point(x, y){ // 建構式，快速建立大量類似的東西(即對程式做的包裝)

this.x = x;

this.y = y;

this.get = function(){

alert(this.x + ',' + this.y);

};

};

let p1 = new Point(3, 4);

let p2 = new Point(5, 6);

p1.get();

p2.get();

12. 可在上新增方法

// 12. 可在上新增方法

function Point(x, y){

this.x = x;

this.y = y;

this.get = function(){

alert(this.x + ',' + this.y);

};

this.equals = function(p){

return this.x == p.x && this.y == p.y;

};

};

let p1 = new Point(3, 4);

let p2 = new Point(3, 4);

p1.get();

p2.get();

let result = p1.equals(p2); // 會顯示True

alert(result);

接下來慢慢拉到ES6(多了些語法糖(syntax sugar))

像是新增類別樣子的寫法，但背後還是一樣的建構式!!

13.

// 13. 類別 (ES6的syntax sugar)

class Point{ // 類別的名稱

constructor(x, y){ // 建構式constructor 是關鍵字

this.x = x;

this.y = y;

}

get(){

alert(this.x + ',' + this.y);

}

equals(p){

return this.x == p.x && this.y == p.y;

};

}

let p1 = new Point(3, 4);

let p2 = new Point(3, 4);

p1.get();

p2.get();

let result = p1.equals(p2); // 會顯示True

alert(result);

<my: 我問 那剛的建構式寫法和下面這個ES6的寫法完全依樣嗎?

彭: 還是有那麼一點點不一樣，但要講很長，可以搜尋”原型鏈”不同

>

My: <https://pjchender.blogspot.tw/2016/06/javascriptprototypeprototype.html>