**Summary p614**

P564

JS不支援類別，物件係從其他物件繼承行為，我們稱之為原型繼承(prototypal inheritance)或是**基於原型繼承**。 相較典型物件導向程式設計，基於原型繼承更具彈性、效率、和表達力。

繼承原型物件的好處

省記憶體空間(不用建立大量含有function的物件, 只需仰賴原型裡建好的)

降低執行時間

可快速組裝重用程式碼的物件 p572

**JS中, 函式也是一個物件**。 事實上，一切皆為物件，即使是陣列。 p575

如何設置原型 p576

透過建構程序的prototype屬性來設定。

**>>>>> 非常重要的觀念 <<<<<**

於 **實例** 中呼叫 **原型** 的function, 若function內有 this.name, 則此this 是指向 實例中的 name屬性(如果實例中有name這個屬性的話, 否則會往上找, 也就是往原型找) p580

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* P583練習(解答於p617) \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**|**

**|** Game.prototype.unlock = function () {

**|** // 此處this指向 Game 的 level 屬性

**|** If (this.level === 42) {

**|** Robot.prototype.deployLaser = function (){

**|** // **此處 this 指向叫用 deployLaser 的實例物件**

**|**  Console.log(this.name + “ is blasting you with laser beams.”);

**|** }

**|** }

**|** }

**|** Robby.deployLaser(); // deployLaser裡的 this 指向 Robby

**|**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

P586

Obj.hasOwnProperty(“attribute”); 若attribute是Obj實例的屬性 則會回傳ture.

**設置原型鏈 (chain of prototypes)** p591

**A繼承B** : 把B物件**實例**賦值給A**建構程序的prototype** p596

A.prototype = new B();

繼承其它物件後 習慣上 要把 constructor 設為自己 p602

Obj.prototype = new ObjInheritance();

Obj.**prototype.constructor** = Obj;

(不設定的話 執行Obj.constructor 印出結果可以看到是繼承來的 function ObjInheritance，而非自己 )

**逐步分析 Constructor.Call** p604

Function ShowDog(name, breed, weight, handler) {

Dog.call(**this**, name, breed, weight);

(Dog為物件原型 i.e function Dog (){this.name, …})

…

}

**// this 會指向以new來叫用ShowDog時所建立的物件**

**// 且它也是Dog 之程式碼主體中的this**

任何物件最初都繼承自 Object p607

意即原型練將會結束於Object.

即使未叫用toString(); 也會被叫用的情況:

Console.log(“Robot is: ”+ toy);

// toy在連接到其他字串之前，會叫用toy.toString() 將toy物件轉為// 字串