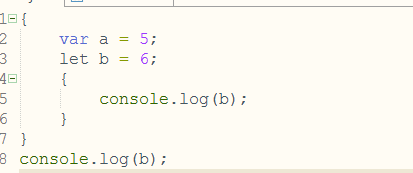
# JavaScript Es6学习

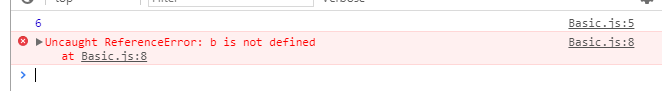
## let 与 var

* let是块级作用域
* var 作用域不定

列子：

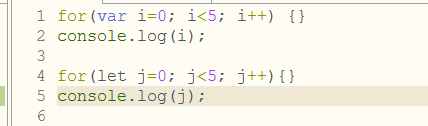


运行结果：

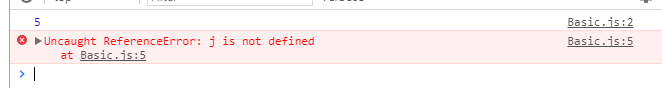


### 泄密

* + 代码：

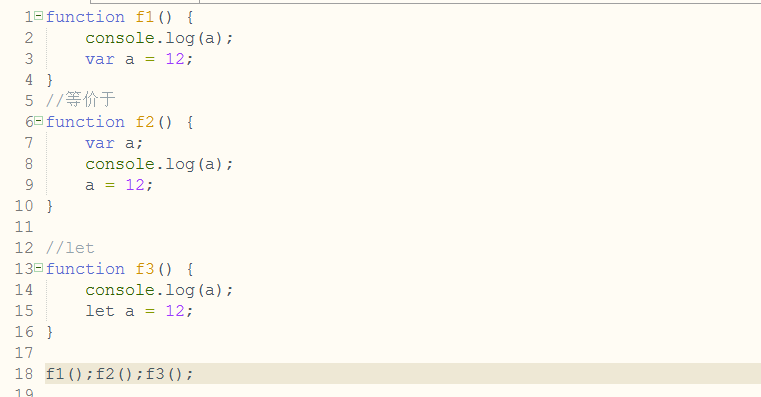


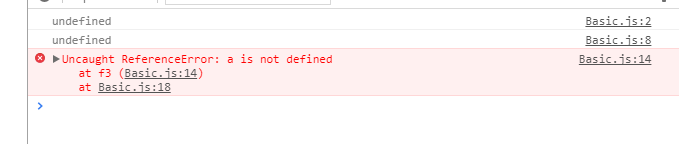
* 结果



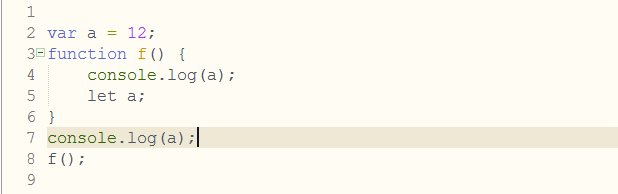
* 结论：使用let变量只在循环作用域优效

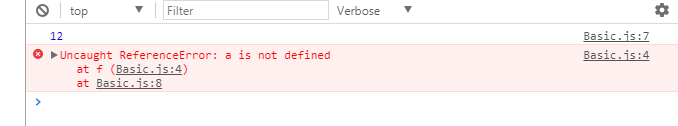
### 提升问题



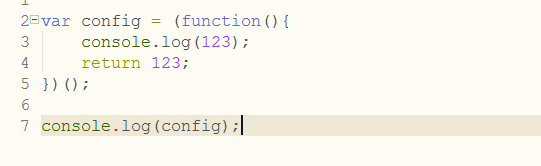


### 临时失效



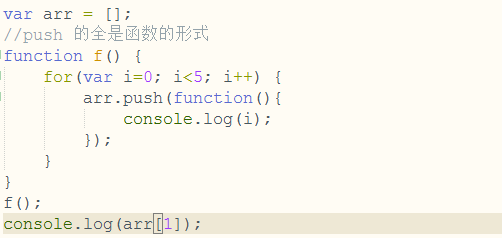


### 立即执行

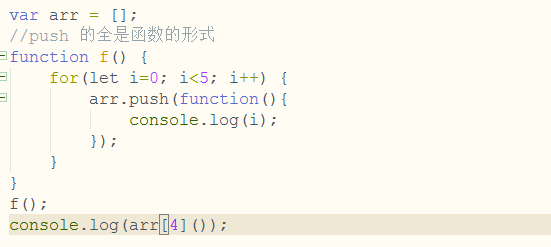


### 数组函数

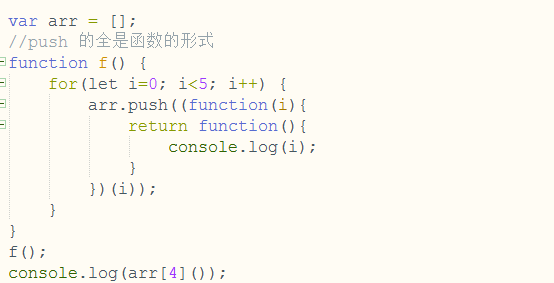
此处push的全是函数形式，要想左后打印结果是每次的i，则需要换一种写法



使用let解决



原始解决方案：

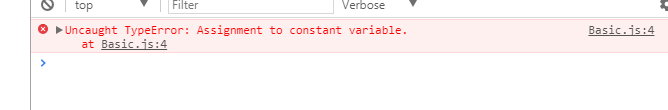


### 常量 const

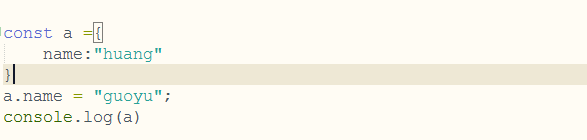
作用域与let相同

案列一：





案列二：





Const的意思不是说所指对象不可以修改，而是所指的内存不可修改；

解决方案：

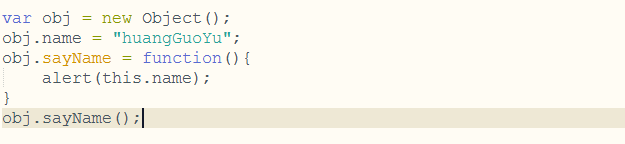
使用严格模式： ‘use strict’，定义好对象后使用Object.freeze(a);

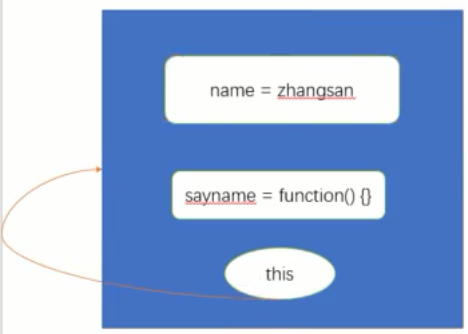
# 面向对象

## 2.1 定义对象

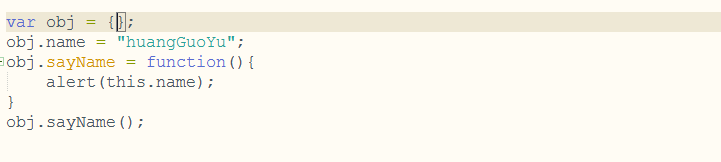
## 2.1.1创建对象

#### new Object()





#### {}



注意：以上方式定义一个对象是没有具体类型的对象，如果要定义类似的对象必须再次重复编写代码。

* 使用字面量

#### factory

解决了代码重复的问题，但是并有没有创建出不同的类型，并且函数还是定义了多个。



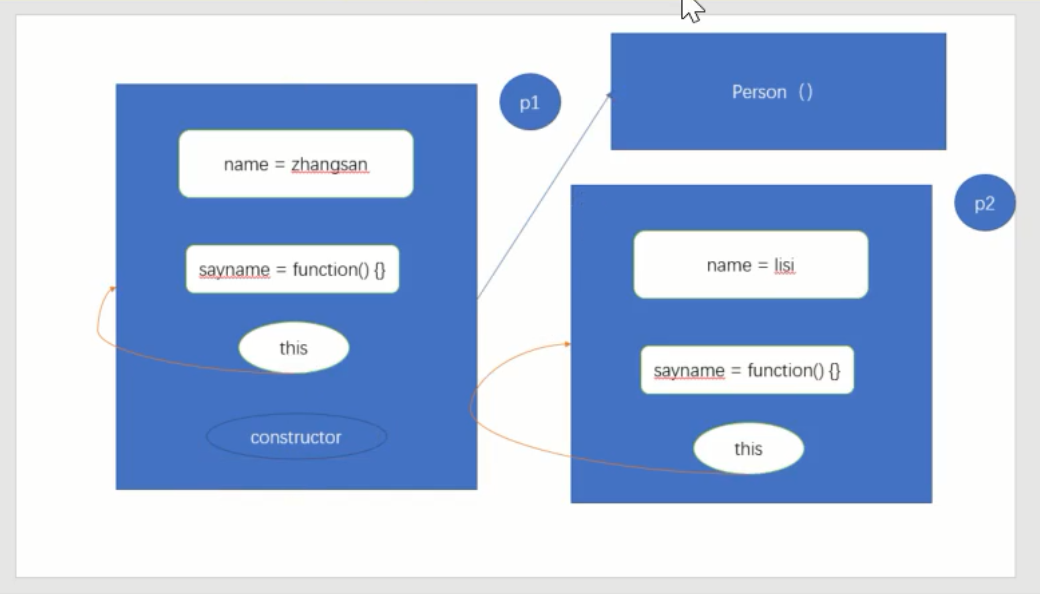
#### 造函数模式

约定俗成：构造函数的方式构造函数首字母大写。不可直接调用方法需要使用new关键字进行创建对象。其中的函数任然不同因为是不同的内存空间。可以使用全局函数来解决，但是这样解决破坏了封装性。

每个new出来的对象，都有一个引用这个引用指向函数

所有的对象都是继承Object





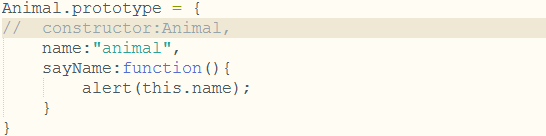
#### 原型模式

任何一个函数都是一个函数对象，其中有一个prototype的指针，指向这个函数的prototype对象。构造函数模式每次创建一个对象都会新创建一个空间，只是每个对象的构造方法相同。在原型模式中属性以及方法都是放在函数的原型对象中，所以其所有的对象的属性和方法也是指向原型空间的。原型空间就相当于java中的Class本身。

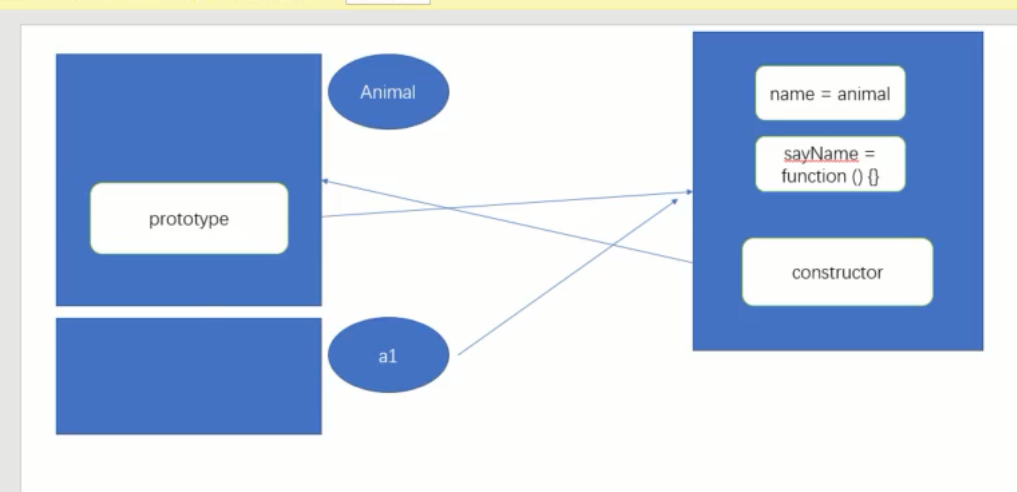


再此处使用的是给原型空间追加属性，但是在直接给原型引用分配新对象时一定要注意constructor的指向位置（不重要可以不管）。

列如：



注意：原型空间中有一个constructor又执行原来的函数。

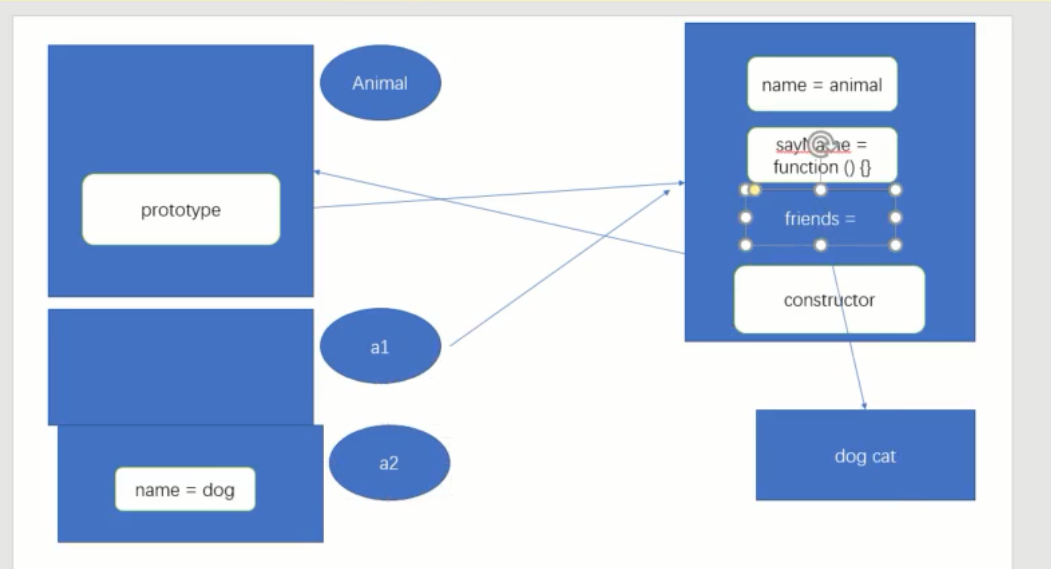


a1.name = “dog”name此刻a1会在自己的空间创建一个name属性，而不是在函数的原型空间去改。

通过Animal对象new出来的对象查找属性过程：

1. 在本身去找
2. 去Animal的原型空间找

**缺点：**



**在使用原型模式的时候如果原型空间中有一个属性是一个引用类型，这时候一个对象改变了这个类型的值，其余所有对象也会改变。如上所述如果是直接**

**Obj.属性 = 值**

**这样是改变了内存地址，这样还是相当于在自己的空间添加了属性，不会改变原型空间，只有在拿着引用对象进行引用属性的方法操作时才会影响所有对象。**

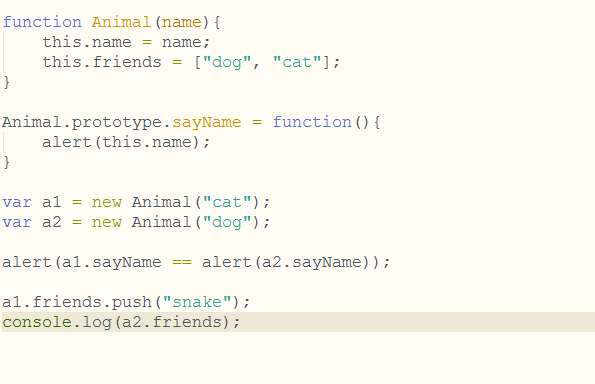
**列子如下：**



#### 构造函数+原型

原理：

1. 将每个对象私有的属性放在构造方法中
2. 将每个对象可以共用的方法放在原型中



sayName

constructor

Name

Friends

Prototype

Name

Friends

Constructor

## 继承实现

### 原型链实现继承



注意：没有对象应该都有指针指向同类的原型空间，在实现继承的时候一定要创建一个对象出来这样才分配了一个空间，不然是直接指向了Animal函数的内存地址。

Prototype

Animal 实列

Animal function

Dog对象

prototype prototype

缺点：任然和原型链对象是相同的，如果在原型中（Animal）

存在引用类型那么还是会出现在dog中一个对象改了，所有对象都会被修改。

总结：

构造方法对象实际上就是复制了一份构造方法中的内容

原型链就是共享原型链中的内容

### 构造函数实现继承



缺点：这种继承方式（call）只能是简单的拷贝出函数中的属性，但是子类的原型空间并没有指向父类。同构造函数的对象一样的问题（冗余）。

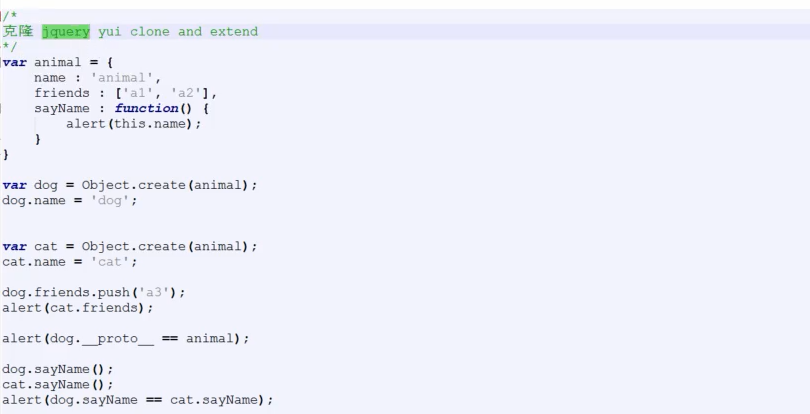
### 组合继承（空间浪费但常用）



缺点：父类属性都会在子类属性中重写。就像当于执行了n + 1次父类构造。

### 克隆和扩展实现

注意：克隆是直接克隆对象。不需要构造函数。Object.create(obj)，相当于把当前对象的原型空间指针指向obj.不会重新创建一个obj空间，所以所有对象的原型空间都指向同一个animal。



### 最理想的方式

也就是将组合继承中的原型空间直接利用object。Create指向animal的原型空间。



### Jquery extends’

extends（bool， cur， copyed）

### javaScript 闭包

function outer(){

var count = 1000;

function inner() {

count++;

alert(count);

}

return inner;

}

var innerFuc = outer();