Javaweb-js加强

# 闭包

函数对象可以通过作用域链相互关联起来，函数体内部的变量都可以保存在函数作用域内，这种特性在计算机科学文献中称为”闭包”。这个术语是指函数变量可以被隐藏于作用域链内，因此看起来是函数将变量”包裹”了起来。使用闭包可以简化代码，但是会将子函数作用域链中所有对象保存在内存中，内存消耗比较大。

如果要理解闭包概念，我们需要明确以下两个概念

什么是变量作用域

什么是作用域链？

## 变量作用域

一个变量的作用域是指程序中定义这个变量的区域，简单分为全局变量与局部变量。

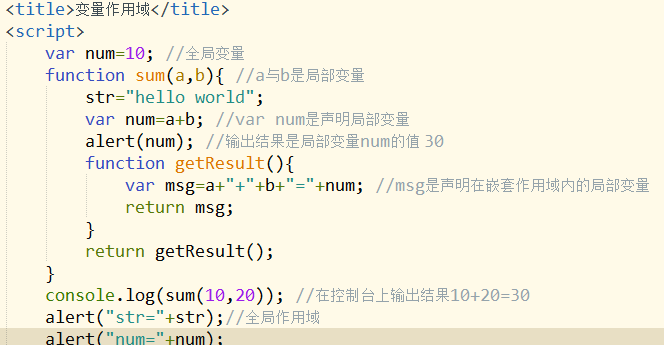
全局变量拥有全局的作用域，简单说，就是在代码中任何地方都可以访问到的对象。所有window对象的属性具有全局作用域。

在函数内部声明的变量只能在函数体内有定义，它们是局部变量，作用域是局部性的，函数参数也是局部变量，它们只在函数体内有定义。

注意：

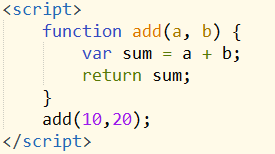
在函数体内，局部变量的优先级高于同名的全局变量

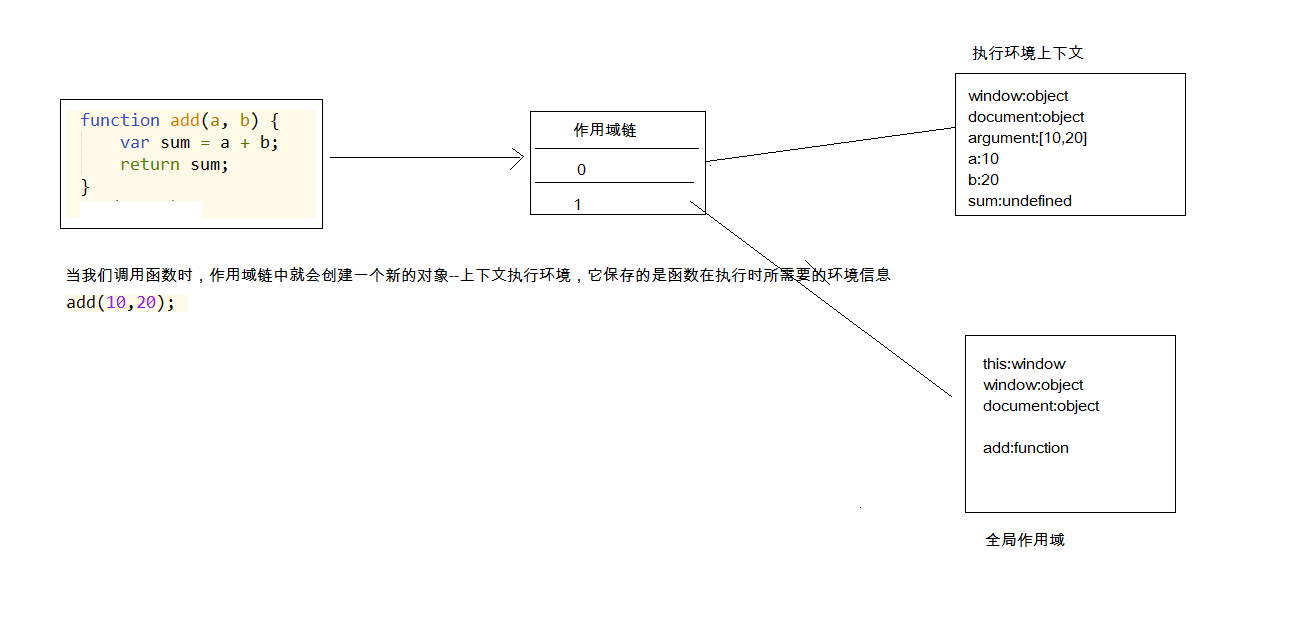
声明局部变量时必须使用var语句，全局变量无要求。



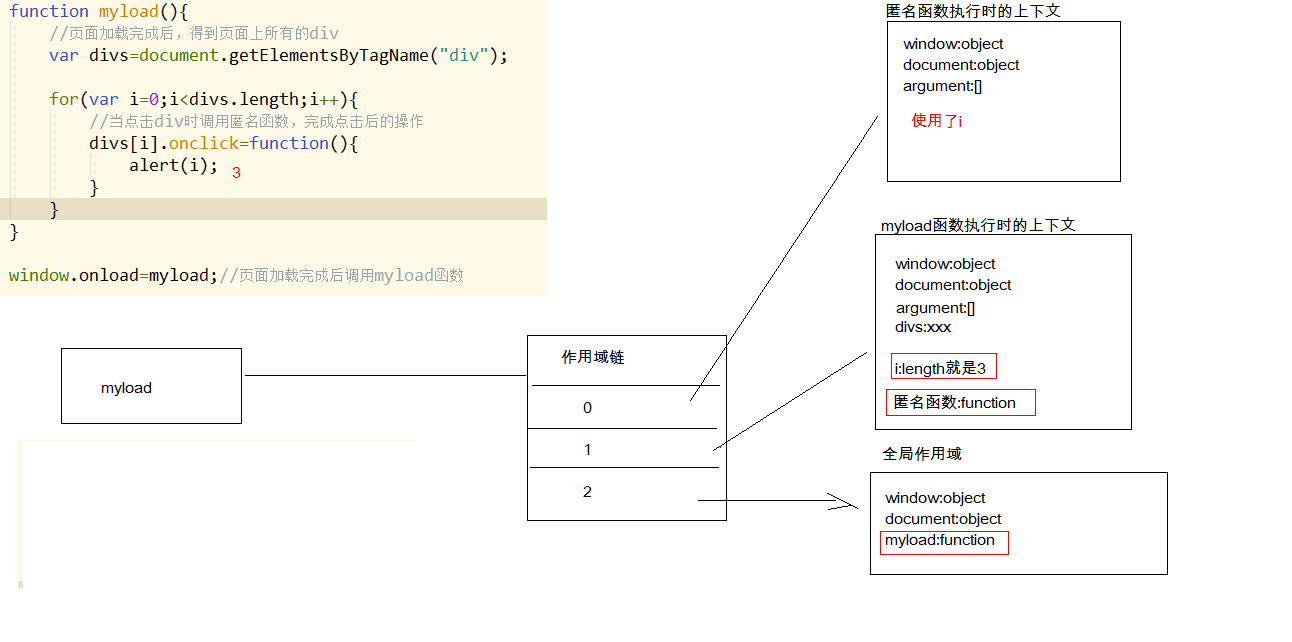
## 作用域链

作用域链由一个全局对象组成。在不包含嵌套的函数体内，作用域链上有两个对象，第一个是定义函数参数与局部变量的对象，第二个是全局对象。在一个嵌套的函数体内，作用域链上至少有三个对象。当定义一个函数时，它实际上保存一个作用域链。当调用这个函数时，它创建一个新的对象来存储它的局部变量，并将这个对象添加至保存的那个作用域链上，同时创建一个新的更长的表示函数调用作用域的“链”。





嵌套函数作用域链



# javaScript自定义对象

在javascript中可以通过对象直接量，关键字new 和Object.create()函数来创建对象

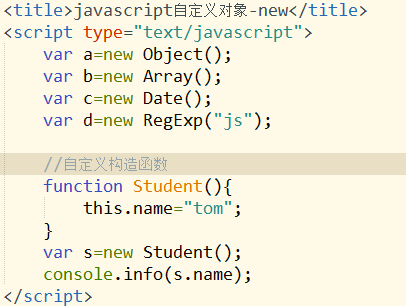
## 对象直接量

对象直接量是由若干名/值对组成的映射表，名/值对中间用冒号分隔，名/值对之间用逗号分隔，整个映射表用花括号括起来。



## 关键字new

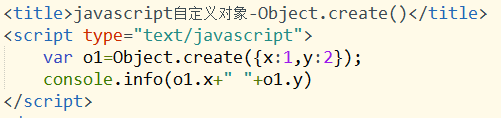
new运算符创建并初始化一个新对象。关键字new后跟随一个函数调用，这里的函数称做构造函数，构造函数用以初始化一个新创建的对象.javascript语言核心中的原始类型都包含内置构造函数。



## Object.create()

ECMAScript5定义了一个名为Object.create()方法，它创建一个新的对象，其中第一个参数是这个对象的原型。Object.create()提供第二个可选参数，用以对对象的属性进行进一步描述。

Object.create()是一个静态函数，而不是提供给某个对象调用的方法，使用它的方法很简单，只需要传入所需的原型对象即可



# javaScript函数

函数就是完成特定功能的代码块。

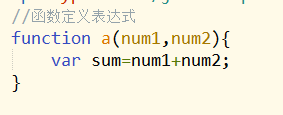
## 函数声明

函数使用function关键字来定义，它可以用在函数定义表达式，函数声明语句中，在这两种形式中，函数定义都从function关键字开始，其后跟随以下组成部分

* 函数名称标识符：它是函数声明语句必需的组成部分
* 一对圆括号：包含0个或多个用逗号隔开的标识符组成的列表，简单说就是函数的参数名称
* 一对花括号：共中包含0条或多条javascript语句，这些语句构成了函数体，一旦调用函数，就会执行这些语句

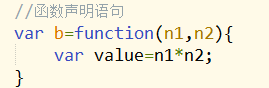
### 函数定义表达式

function 函数名称(参数){}



### 函数声明语句

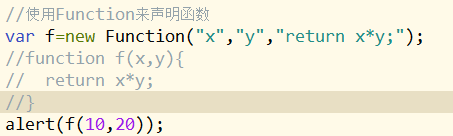
Var fun=function(){}



### Function声明函数

不管是使用函数声明语句还是使用函数表达式，函数的定义都需要使用functionn关键字，但函数的定义还可以使用Function()构造函数来定义

Var f=new Function(“x”,”y”,”return x\*y;”);



## 构造函数与prototype属性

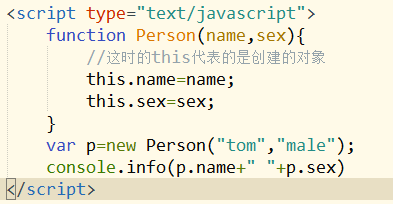
所谓构造函数，就是提供了一个生成对象的模板并描述对象的基本结构的函数。一个构造函数，可以生成多个对象，每个对象都有相同的结构。总的来说，构造函数就是对象的模板，对象就是构造函数的实例。

构造函数的特点有：

a：构造函数的函数名首字母必须大写。

b：内部使用this对象，来指向将要生成的对象实例。

c：使用new操作符来调用构造函数，并返回对象实例。



构造函数的缺点就是：同一个构造函数的对象实例之间无法共享属性或方法

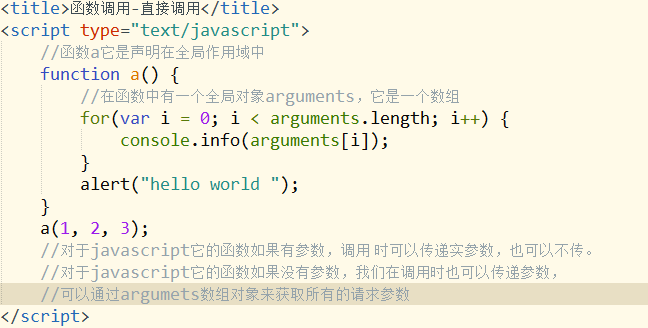
为了解决构造函数的对象实例之间无法共享属性的缺点，js提供了prototype属性

原型对象上的所有属性和方法，都会被对象实例所共享。



## 函数调用

### 直接调用



### 方法调用



### 构造器调用



### call()方法和apply()方法

我们可以将call()和apply()看做是某个对象的方法，通过调用方法的形式来间接调用函数。

call()和apply()的第一个实参是要调用函数的母对象，它是调用上下文，在函数体内通过this来获得对它的引用

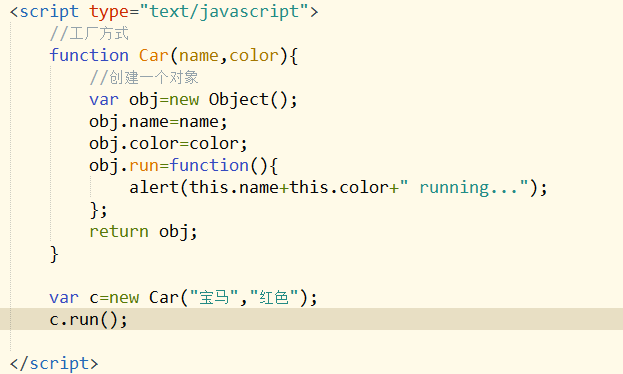


# javaScript类与继承

## javaScript中类定义

javascript中类的概念是通过函数来模拟的。

### 工厂方式



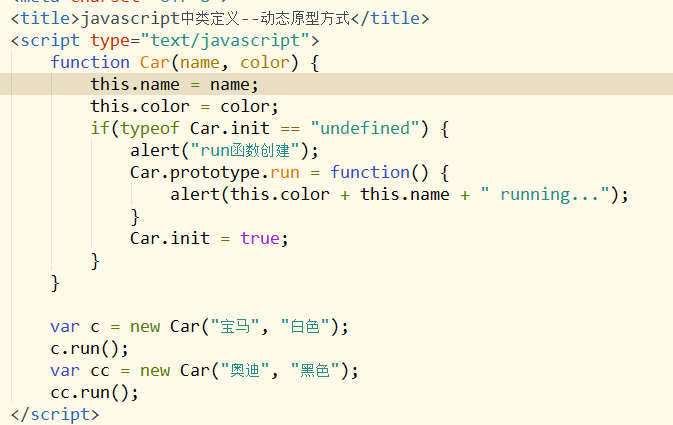
### 构造函数方式



### 原型方式



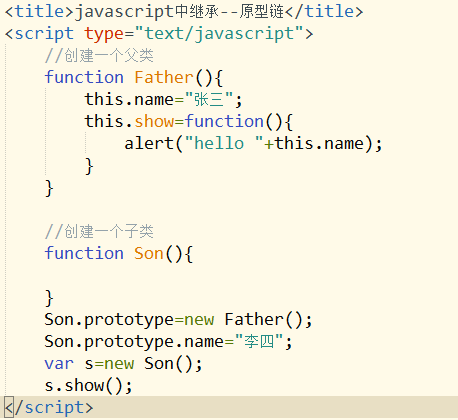
### 动态原型方式



## javaScript中继承实现

### 原型链继承

将父类的实例作为子类的原型



优点:使用简单易懂

缺点：子类添加新的属性/方法必须在new Father（）后。

子在创建时不能向父传递参数

### 构造继承

使用父类的构造函数来增强子类实例，等于是复制父类的实例属性给子类（没用到原型）



优点:创建子类时可以向父类传递参数

缺点：子只能继承父的实例属性/方法，不能继承原型属性/方法

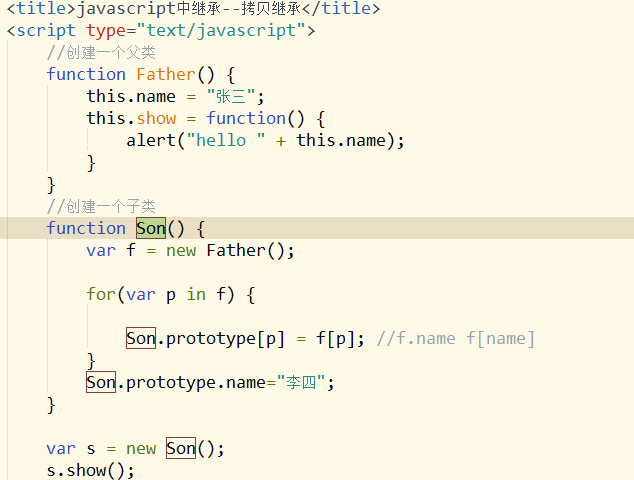
### 实例继承

为父类实例添加新特性，作为子类实例返回



返回的不是真正的子类对象，而是父类对象

### 拷贝继承



内存消耗比较大

### 组合继承

通过调用父类构造，继承父类的属性并保留传参的优点，然后通过将父类实例作为子类原型，实现函数复用



这种方式，会执行两次父类构造函数.

### 寄生组合继承

通过寄生方式，砍掉父类的实例属性，这样，在调用两次父类的构造的时候，就不会初始化两次实例方法/属性，避免的组合继承的缺点

这种方式比较完美，但是它相对比较繁琐

