# JS高级

* 面向对象与面向过程
* 面向过程：分析解决过程的**步骤**，使用函数一步步实现。使用时再一一调用
* 面向对象OOP：把事物分解成一个个**对象**，由对象之间分工合作

封装性 继承性 多态性：同一个对象具有多个形态

* 面向对象的思维特点：

1. 抽取对象共用的属性和行为，封装成一个类
2. 对类进行实例化，获取类的对象

* JS中，万物皆对象
* 对象是由属性和方法构成的：属性描述事物的特征，方法是指事物的行为
* 类：使用关键字class声明一个类，可以使用这个类来实例化对象

一个类抽象了对象的公共部分，泛指具有共同特征的某一大类

对象特指某一个事物，通过类，可将一个具体的对象实例化

1. ES中的类和对象
2. 类的创建和实例化

* 类的创建： class name { //class body }
* 对象实例化： var 对象= **new** name ( ) ;
* 先有类，才能有实例化。因为ES6中没有对类进行提升

1. constructor 构造函数：

***constructor ( )*** 方法是类的默认构造函数，用于**传递参数**、返回实例对象。new命令生成对象实例时，会自动调用该方法。**类里面必须有该方法。**



1. 类中的函数不需要写function
2. 多个函数、方法之间不需要加逗号
3. 类的继承

* 语法

class FatherName { }

class SonName **extends** FatherName { } //子类继承父类

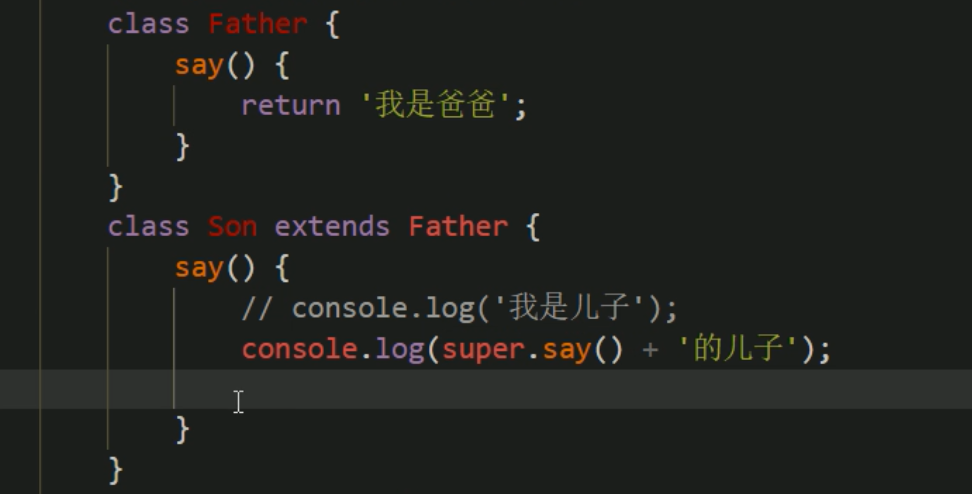
* 就近原则：父类子类拥有同名方法、属性，默认优先调用的是子类的函数。
* super关键字

子类继承父类之后，在子类的constructor中直接使用this来传递参数会报错。因为子类的this指向子类，而父类的this指向父类，两者不能等同。

但是可以通过在子类的我constructor中使用super( )来调用父类的构造函数，从而实现参数的传递。此处的super必须在所有this的前面。



super除了可以调用父类的constructor函数，还可以调用其它函数



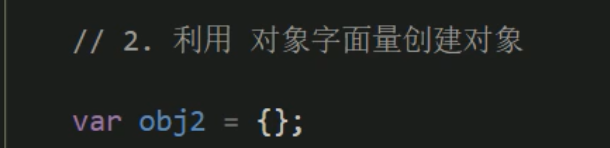
1. 类使用的注意事项
2. 先定义类，才能有实例化
3. 

**该类一经new创建后，此处的方法就会被立即调用**

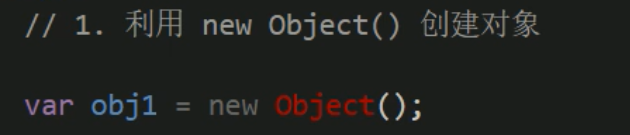
1. 类中共有的属性和方法一定要加this使用
2. constructor中的this指向的是**实例对象**
3. 类中其他属性、方法的this指向 **属性、方法的调用者**
4. 函数名加括号会被直接调用 比如init()
5. 构造函数和原型ES5

* ES6之前对象并非基于类来创建，而是使用**构造函数**这一特殊函数来定义对象和特征
* 构建对象的三种方式

1. 对象字面量



1. new Object ( )



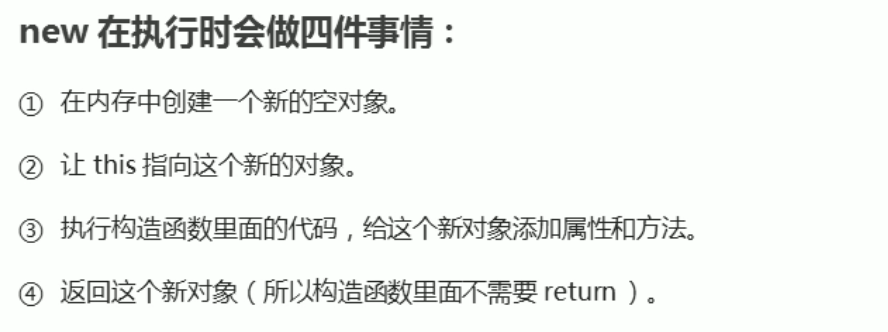
1. 自定义构造函数：构造函数是一种特殊的函数，主要用于初始化对象，为对象成员变量赋初始值，。我们可以把对象中的公共属性和方法抽取出来，封装到构造函数中。



构造函数

实例化

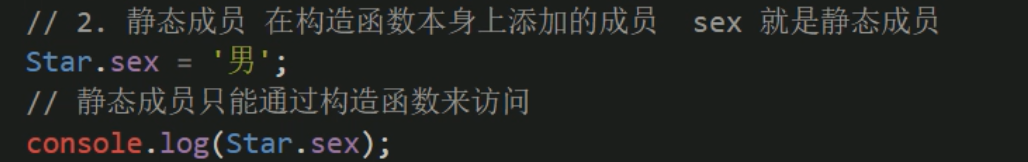
* 构造函数创建某一类对象，首字母要大写。
* 构造函数要和new一起使用。



* 成员——构造函数中的属性和方法

实例成员：构造函数内部**通过this添加的成员，只能通过实例化的对象来访问**，例如上图中ldh.uname

静态成员：在构造函数本身直接添加的成员，只能通过构造函数来访问，不能通过实例化对象来访问



* 构造函数存在的问题：每实例化一个对象，会开辟一个新的内存空间，浪费内存空间。解决方法：构造函数原型
* 构造函数原型prototype：
* 构造函数通过原型分配的函数是所有对象所共享的
* js规定每一个构造函数都有一个prototype属性，指向另一个对象。这**prototype就是一个对象**，该对象的所有属性和方法都被构造函数所拥有。
* **把不变的方法直接定义在prototype上，这样所有对象的实例就可以共享这些方法**



调用方法sing，与之前一样

**把方法sing，定义在prototype上**