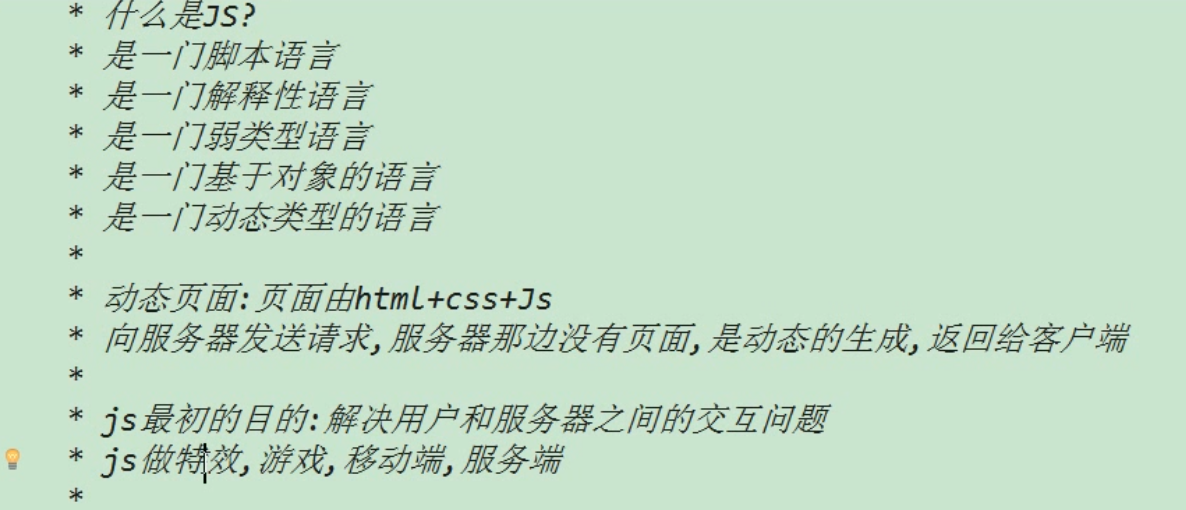
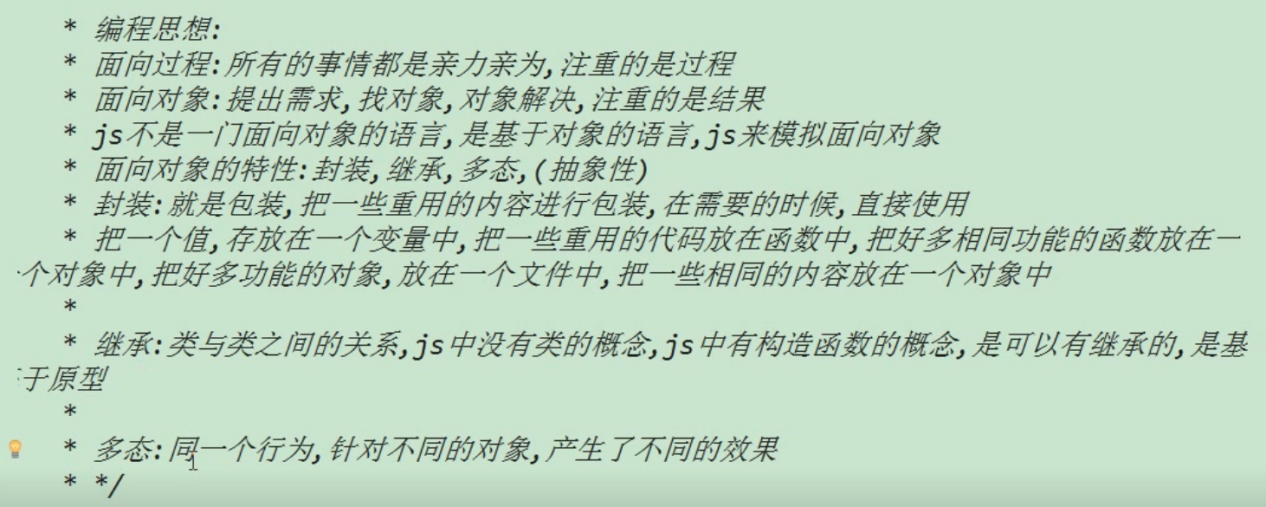
## Javascript的模拟性面向对象编程

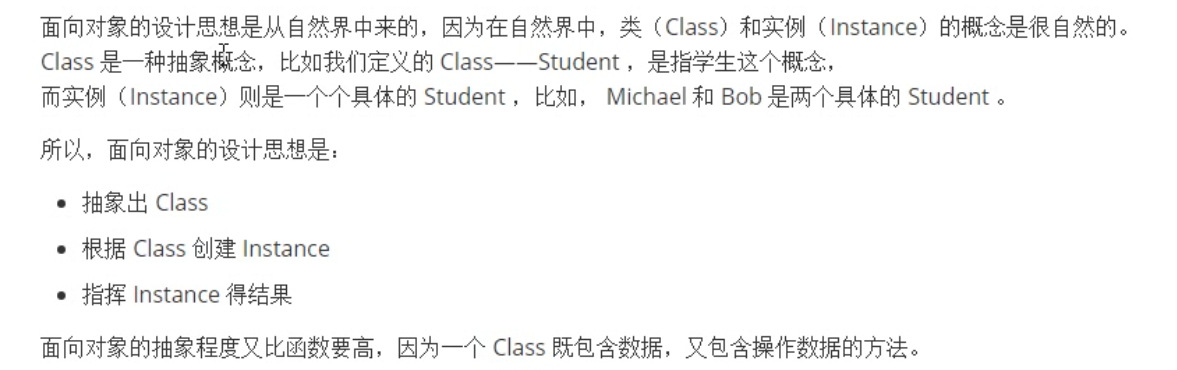


什么是js

什么是动态的

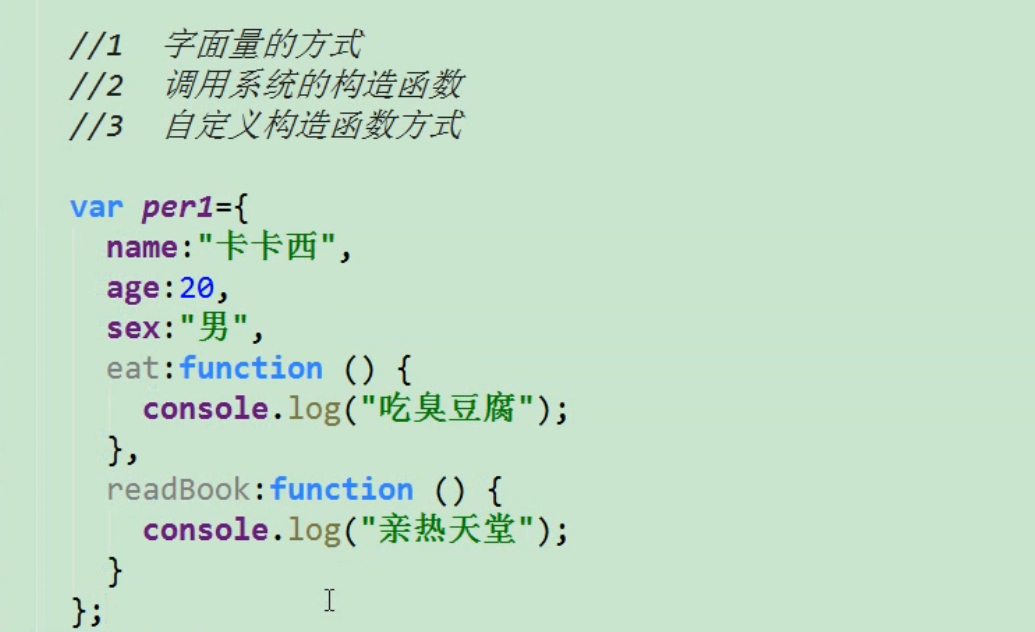
以及js的存在意义



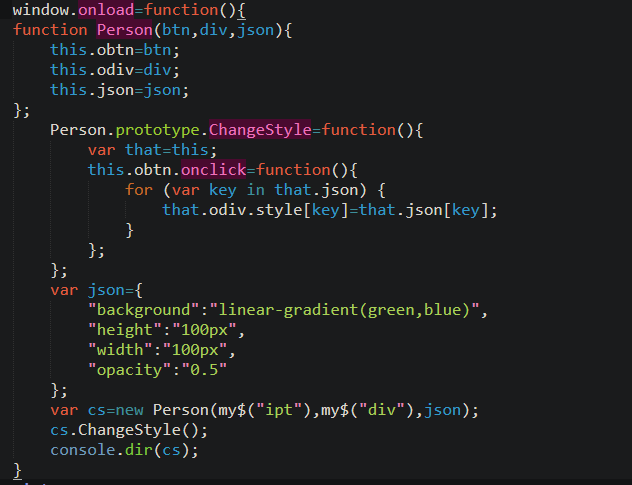


想象：人类里的对象人

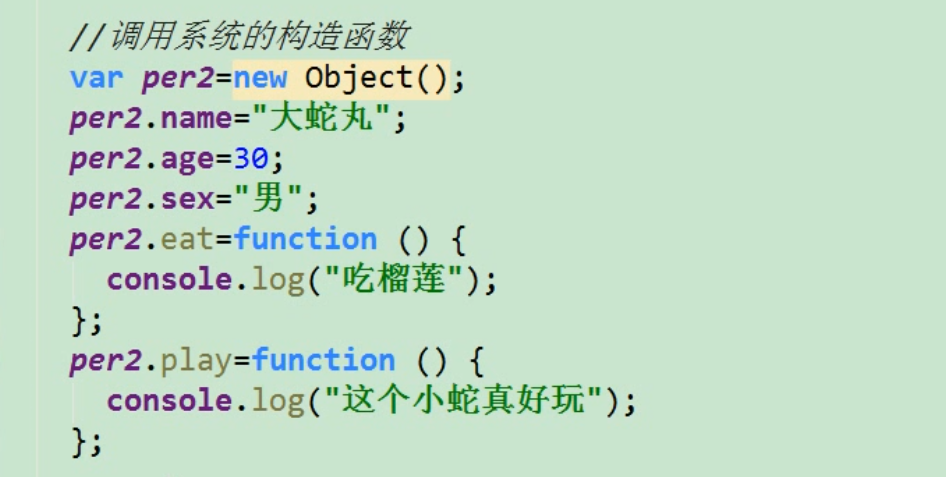
## 字面量



字面量和json的区别，json是自己定义下标的数组，而字面量是对象。



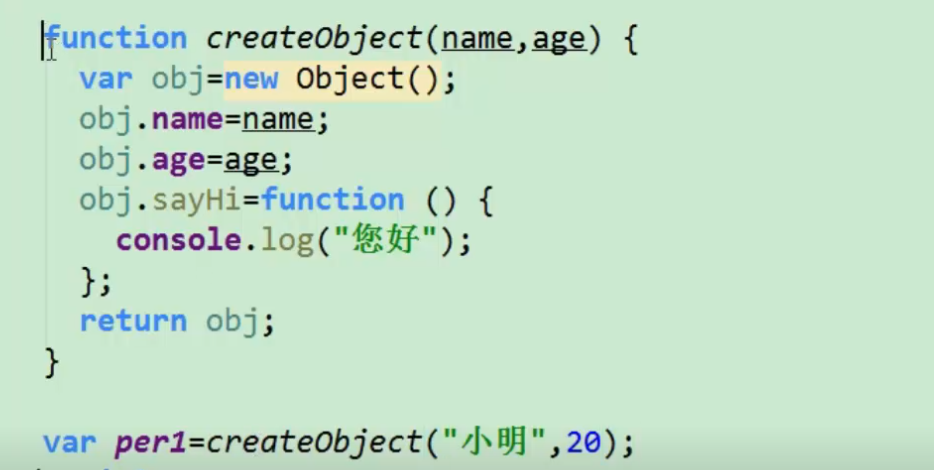
## 调用系统函数



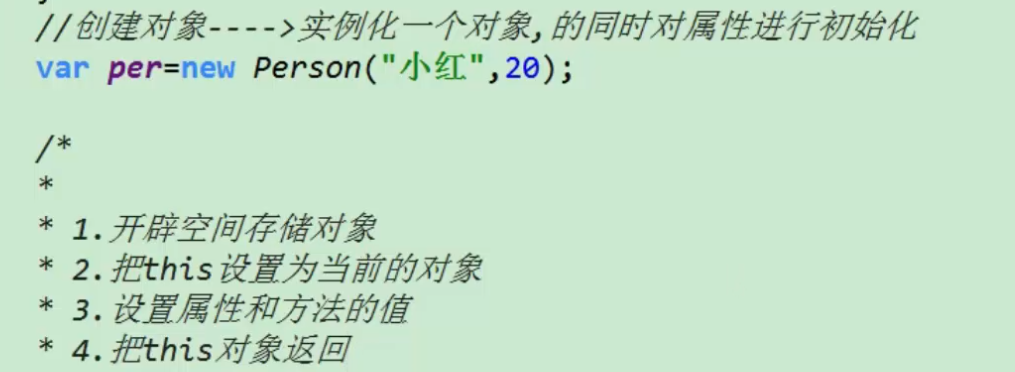
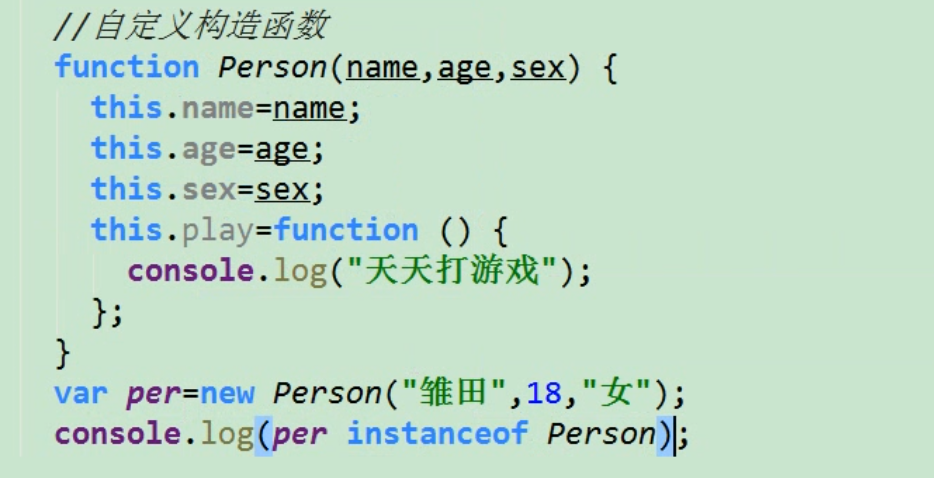


系统函数都是属于objsct的对象，所以没有用。

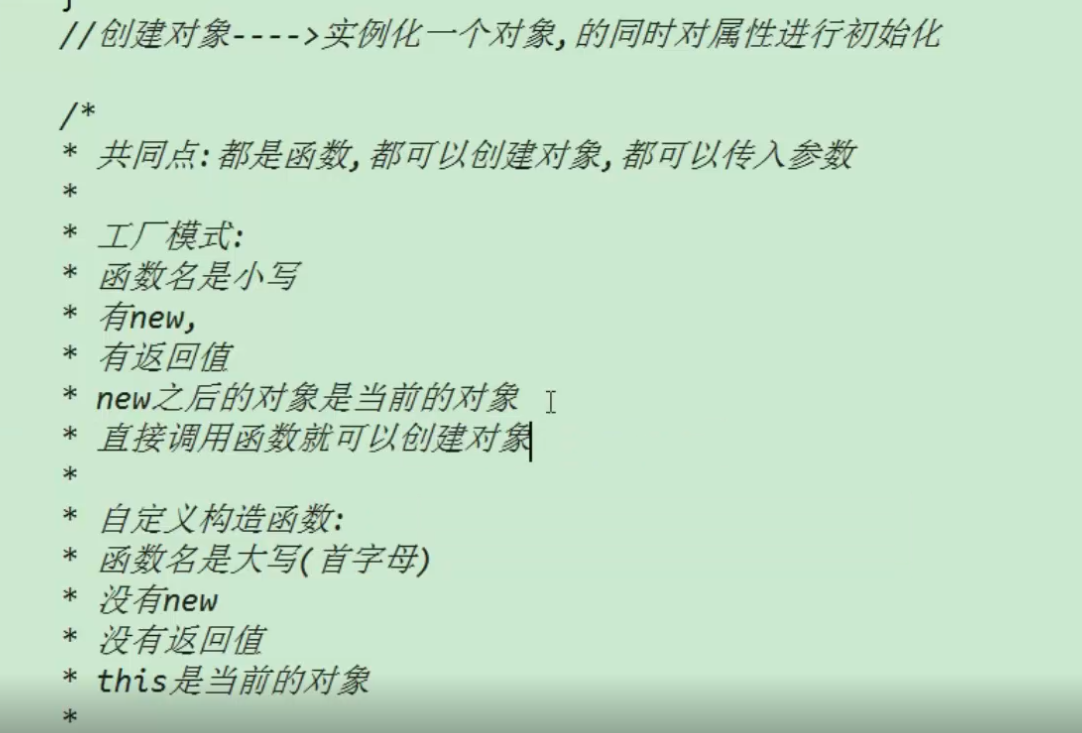
## 工厂式（好像也没啥用）

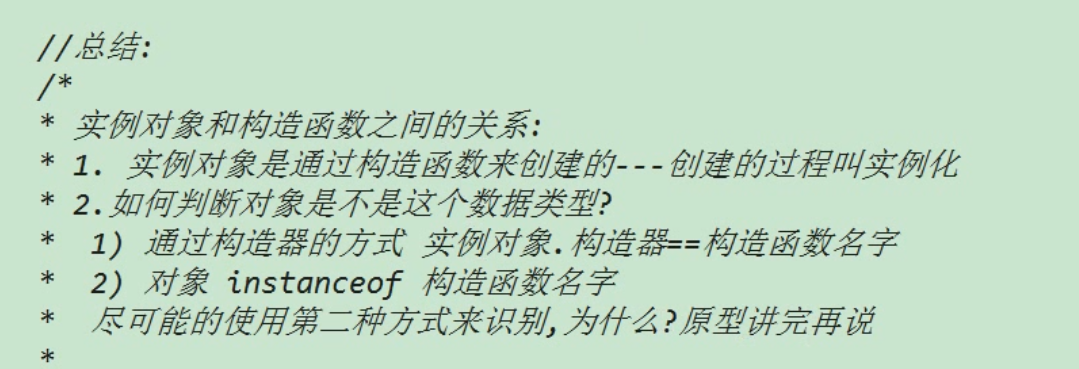
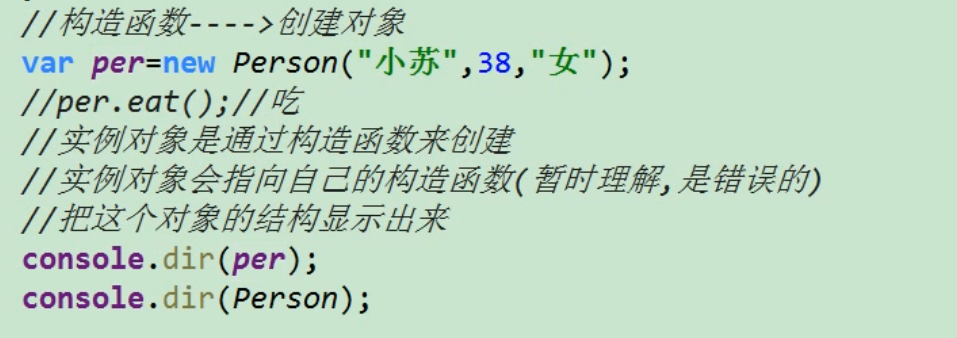
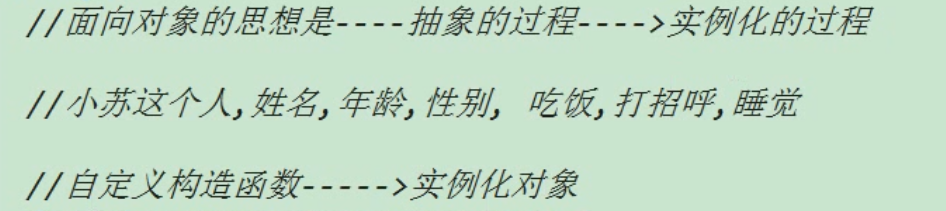


## 自定义构造

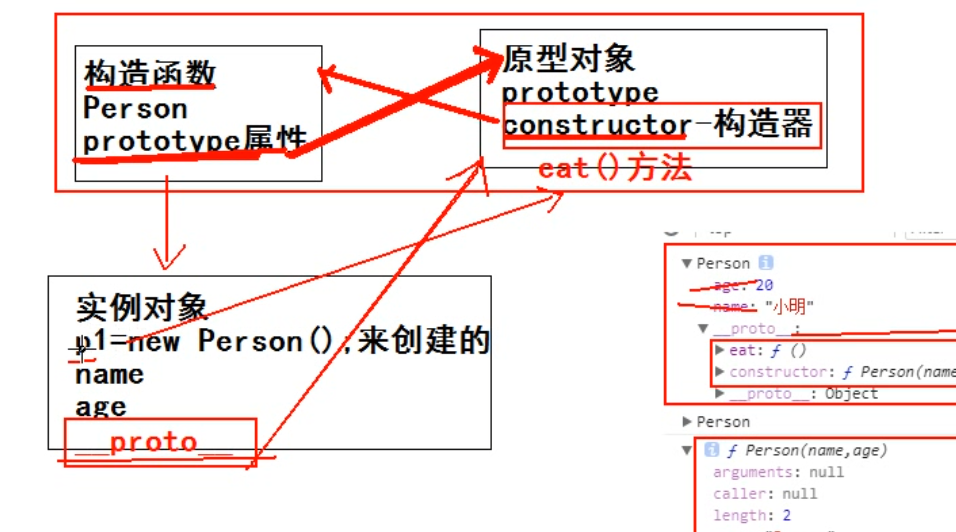


这是生成对象时的过程。

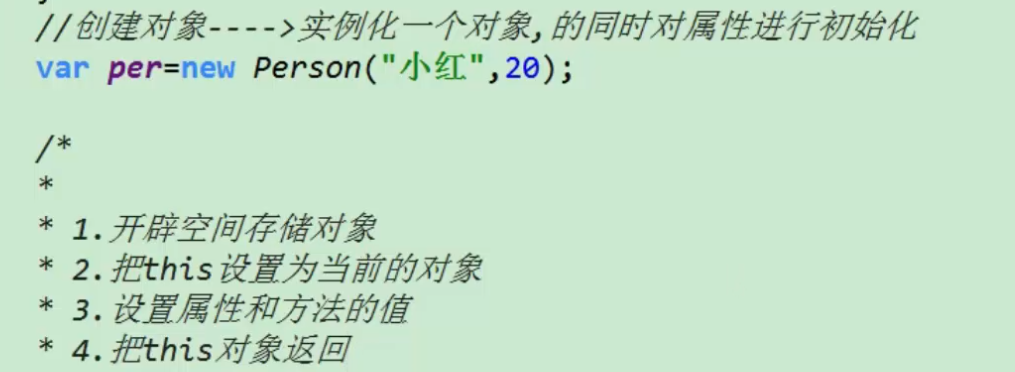




## 原型protetype

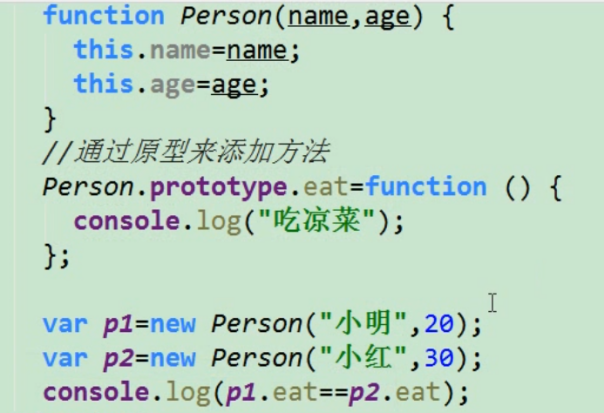


## 一个问题





由于这个可知，每一个对象的声明都会为他的属性和功能开辟地址，不同的属性值是有意义的，但是相同的方法是无意义的，不同对像的方法意义相同但是概念上的不同导致地址占用，所以使用一个构造函数创造多个对象时，重复声明的方法会造成不必要的内存占用。



## 原型作用一

数据共享减少，内存占用：

就是说把方法写到原型对象里，方法就不再是声明函数的，但是还可以通过调用的原型对象的方法（大概是调用地址吧。）

解释：我们都是人，人都有吃饭的能力，小明也是人小明自然能吃饭，数据源形便是人。

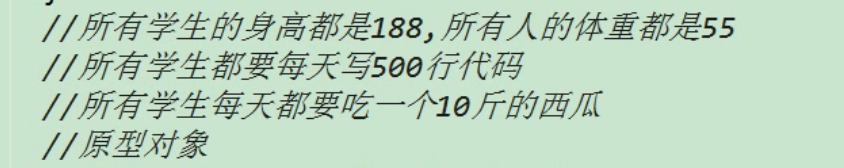
原型对象：是构造函数的一个属性的对象。

原型的作用二是实现继承，而继承的目的也是节省空间。

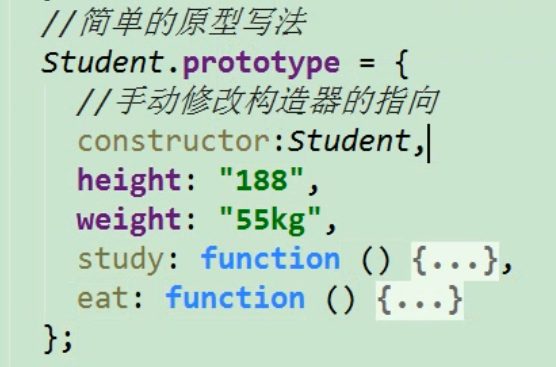


理解便开始了，首先第一个问题，为什么使用that，因为在这个onclick事件里this的当前含义就是button了，我们在实例化对象之前又不知道按钮叫啥，于是事件之前的this是指cs这个对象，于是通过这样的方法就可以改变它的颜色，就是说不包含在方法之中的方法中的this指的就是当前对象。

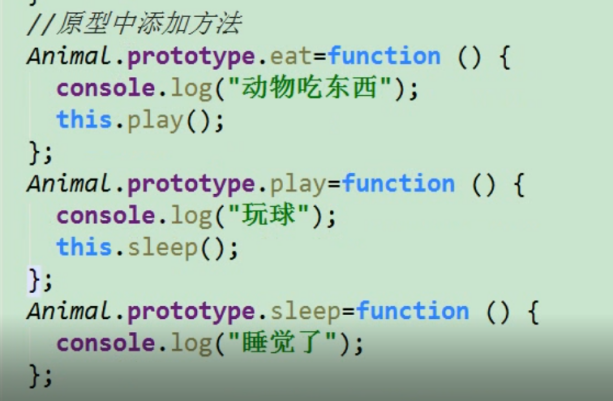
声明对象之后this才是有效的。



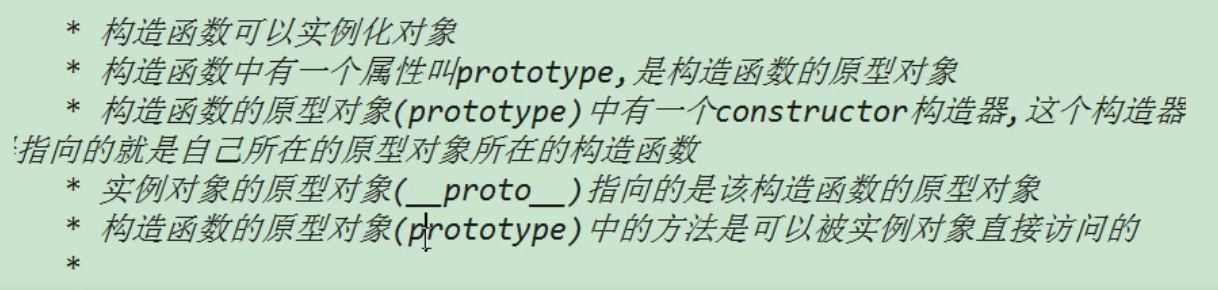
针对于原型对象中的属性可以调用，不是原型对象中的就不需要调用而是创建。



Student.prototype就是一个对象，原型对象，所以可以用字面量的方式来声明。

原型对象是包含在构造函数里的一个对象。

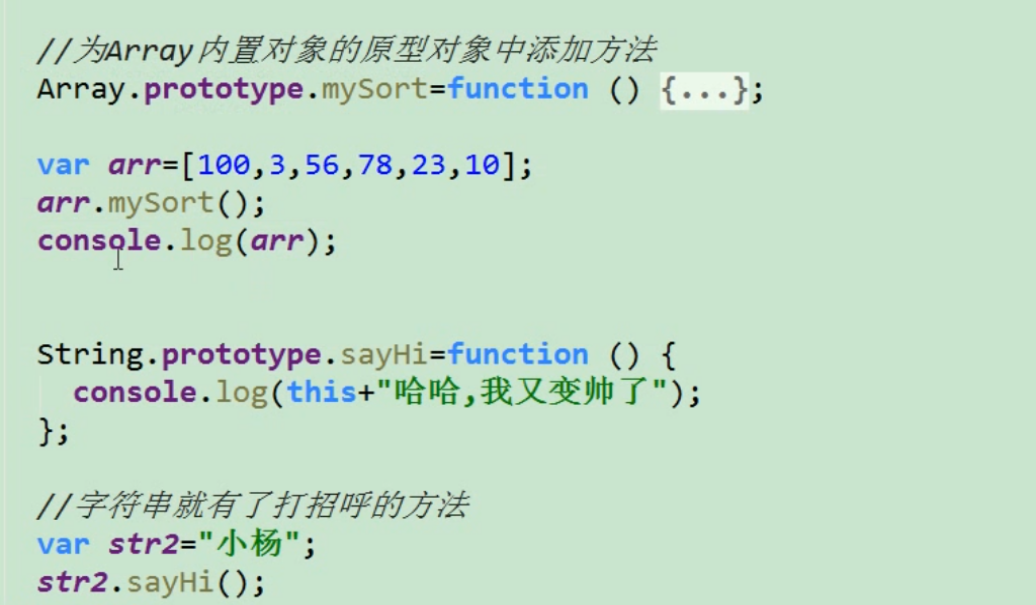
构造函数中的方法可以包含调用，原型对象中的方法也可以包含调用。



由上面的三者关系图可知，实例对象凭空生成两个构造函数中的属性，如果没有这两个属性就会到原型对象中去找，有的话就用不会修改。

就是说现在自己里面找，没有就去，proto所指向的prototype里找。

为内置对象添加方法/



## 局部变量变为全局变量



把随机数对象当成属性写到window对象就i昂当于是全局变量了。

