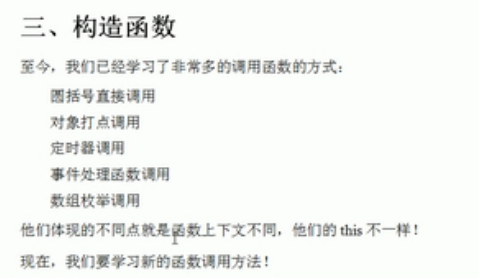
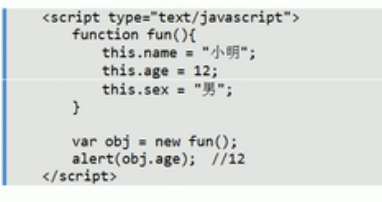
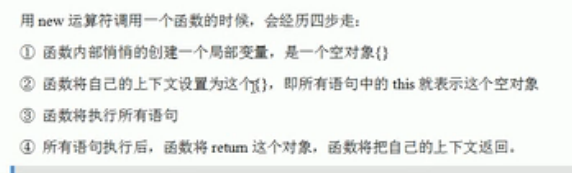
11 构造函数

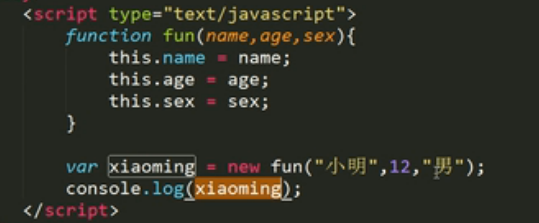


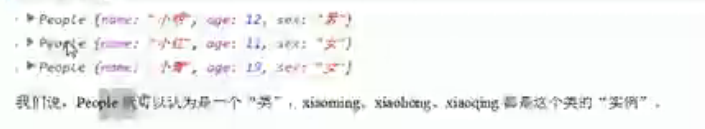


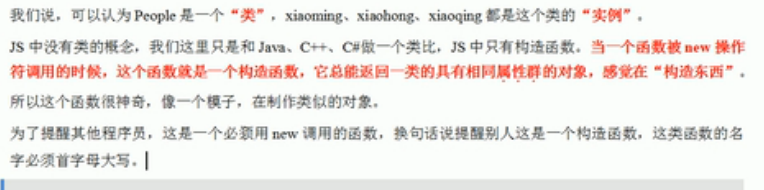


使用new运算符调用一个函数时，会经历四步走：

1. 函数内部悄悄的创建一个局部变量，是一个空对象{}
2. 函数将自己的上下文设置为整个对象，即所有语句中的this就表示这个对象
3. 函数执行所有语句
4. 所有语句执行完毕后，函数将return这个对象，函数将把自己的上下文返回

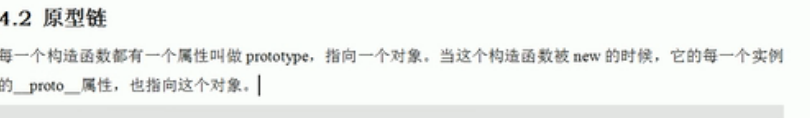


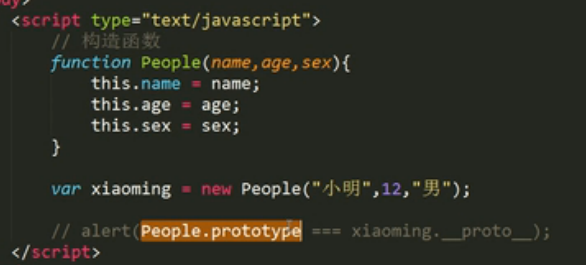


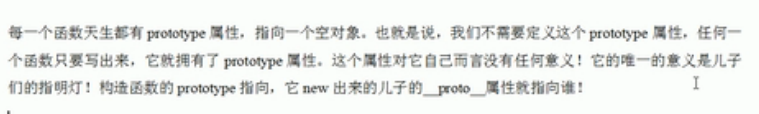


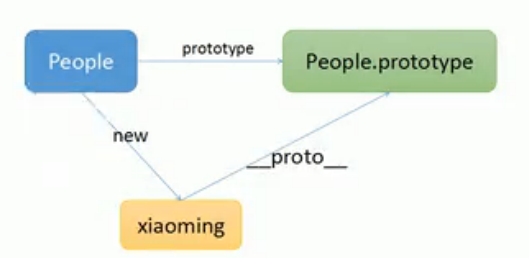


12 原型链



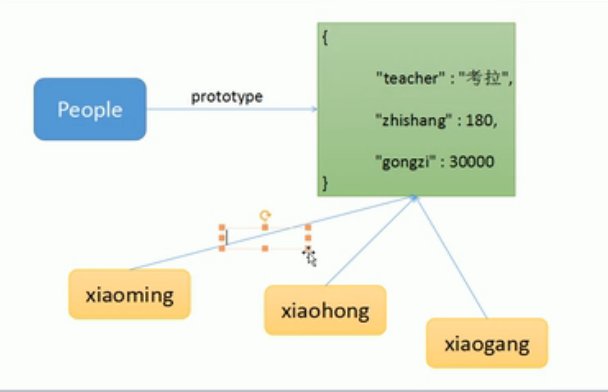


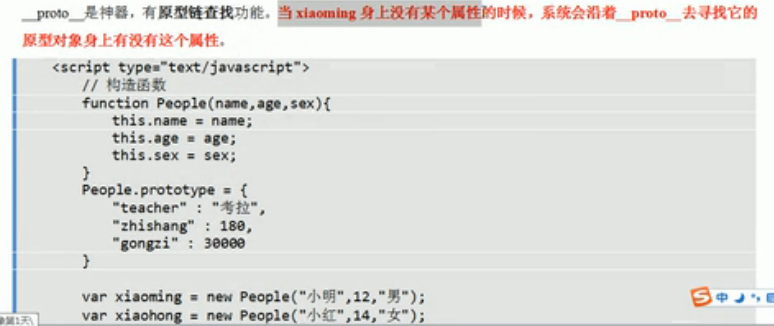


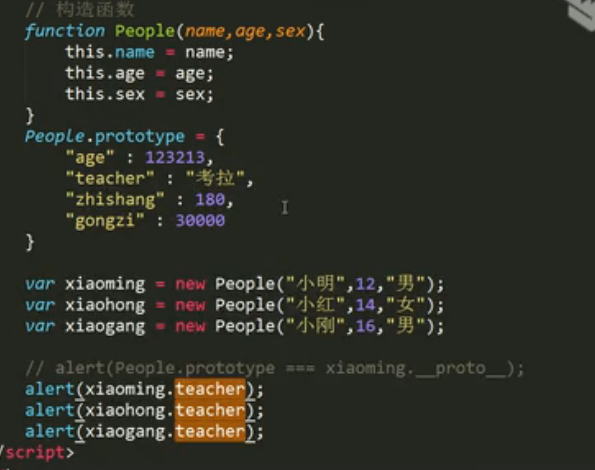


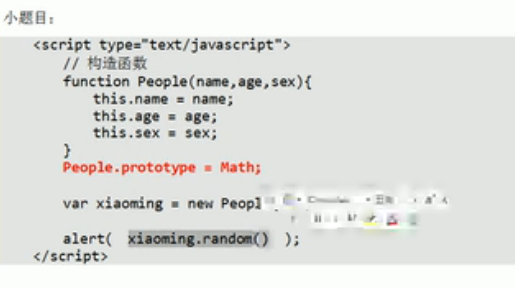


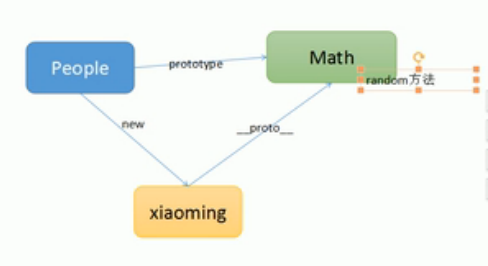










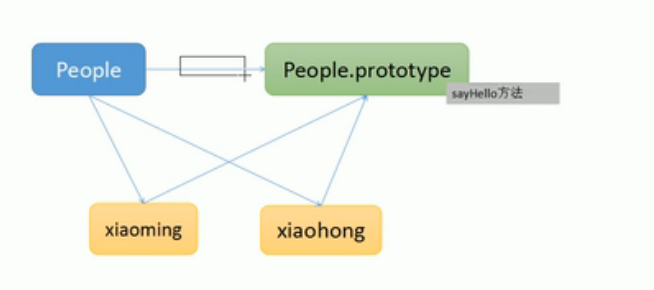




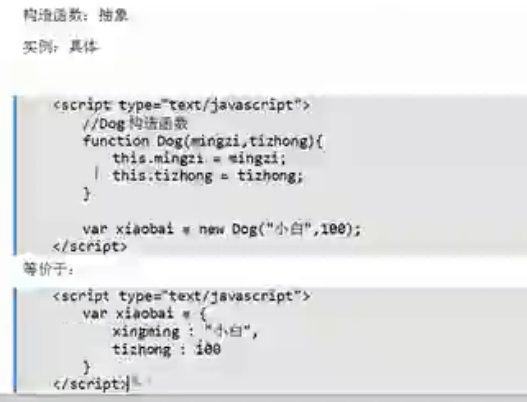


从而能够使用这个方法。





13 构造函数和实例的关系

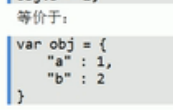


14 内置构造函数1







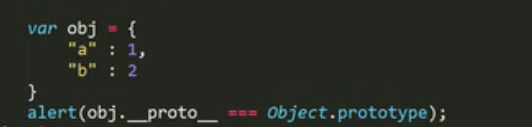


也就是说字面量语法底层也是调用了new Object

console.log(new Object);

console.log({});

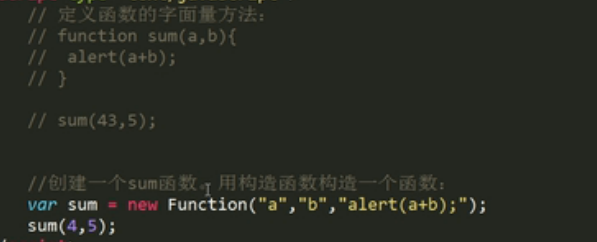






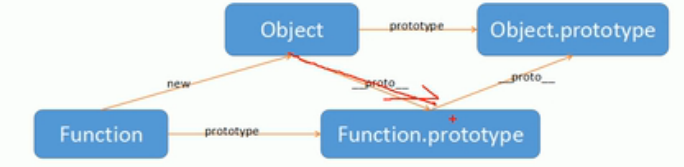


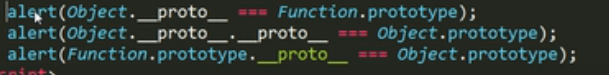




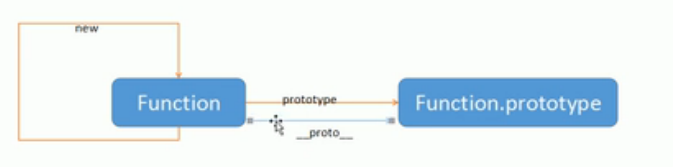




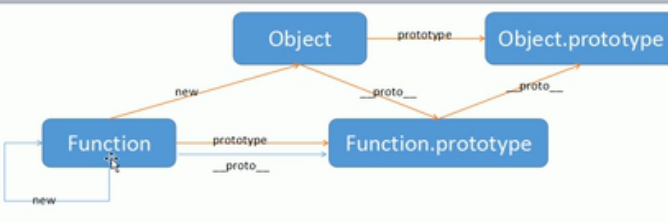




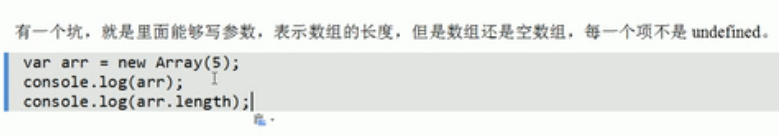
Object是Function的实例所以Object.\_\_proto\_\_指向Function.prototype，而Function.prototype是Object的实例，所以Function.prototype.\_\_proto\_\_指向Object.prototype;Function是Function的实例。



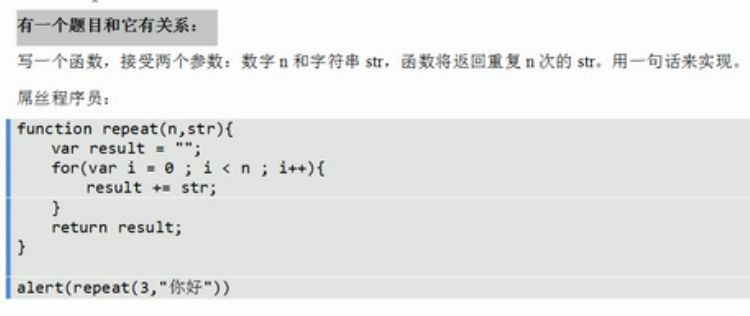
console.log(Function.prototype===(new Function).\_\_proto\_\_);//true

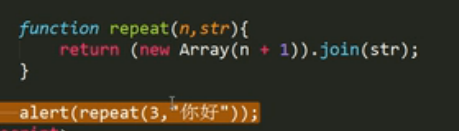


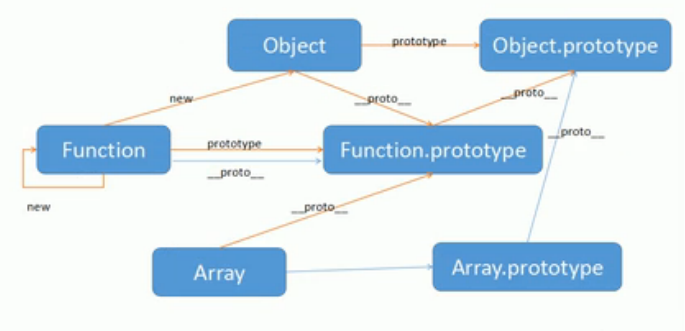


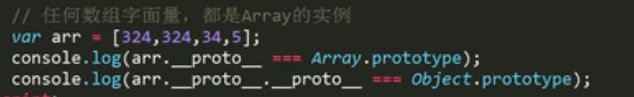




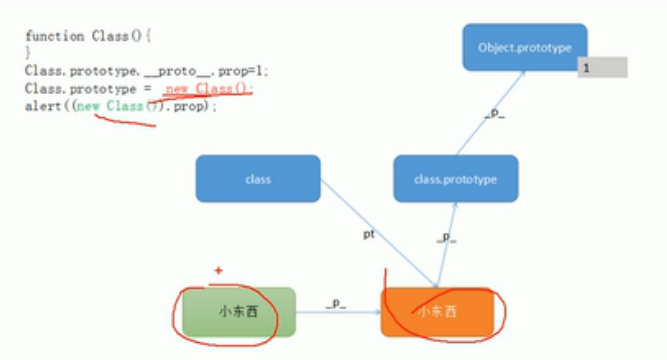




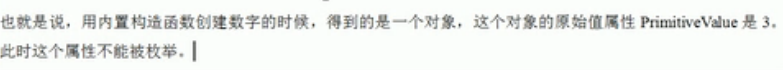


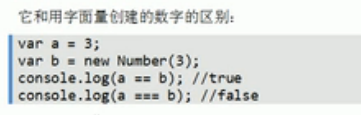






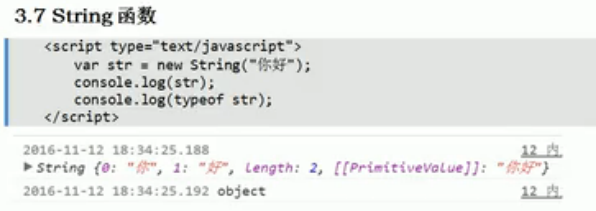




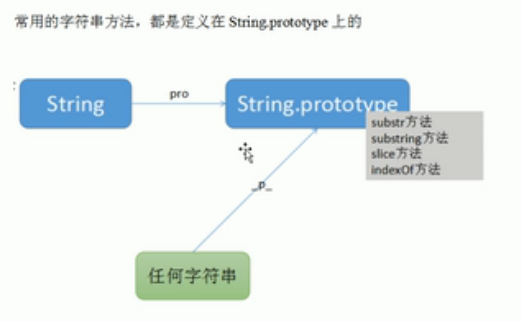




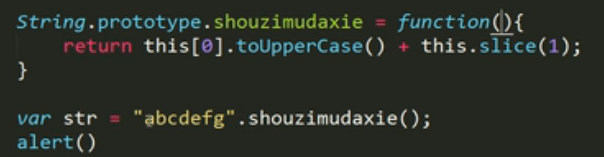




在js上，字面量的字符串和new String()建立的字符串是完全一样的，这不同于java，所以在js中基本不使用new String来建立字符串。



在原型上增加方法，拓展内置原型或者对象的方法





//===================20170606