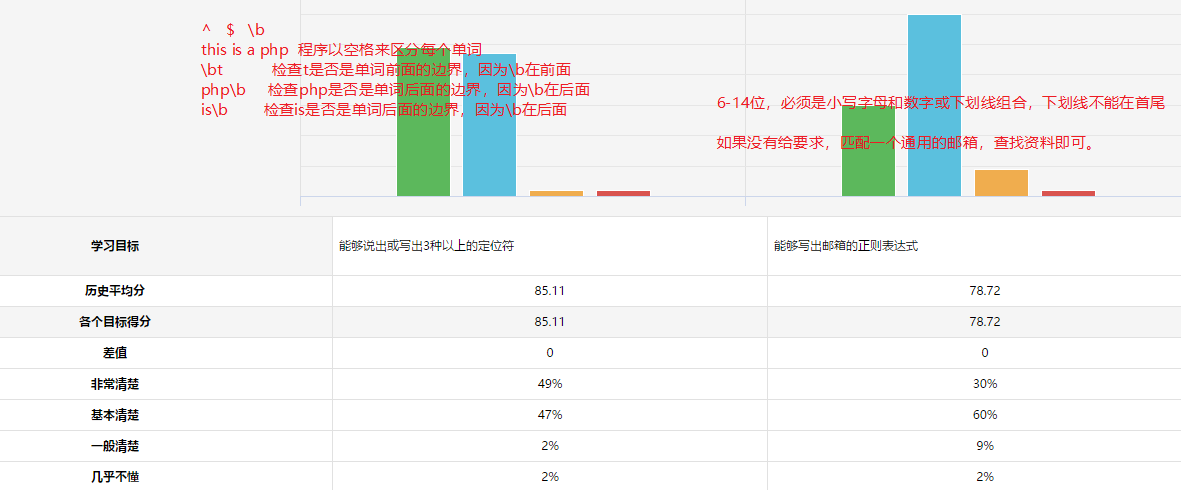
# 一、回顾



# 二、今日目标

1、理解什么是面向对象（编程）

2、掌握定义对象的多种方式，并知道各种方式的优缺点

3、掌握什么是原型对象（难点）

4、理解原型链的概念（或原型链的查找方式）

5、掌握什么是回调函数

6、了解arguments对象的作用

7、理解能够把函数当做对象使用（意思是函数也可以使用对象的点语法去调用属于函数的成员）

# 三、面向对象编程

1、什么是对象

ECMAScript-262 把对象定义为：**无序属性的集合，其属性可以包含基本值、对象或者函数**。

var obj = {

name:'张三',

age:20,

say:function(){},

sanwei:['100cm', '90cm', '105cm'],

abc:另外一个对象

};

我喜欢大眼睛、长头发、大长腿、会洗衣服、会做饭、会生孩子的女孩，比如有孙莉、冰冰….

上面一句话就描述了类和对象的关系，大眼睛、大长腿、长头发是类的属性；会洗衣服、会做饭是类的方法；后面的符合条件的两个女孩就是类的实例对象。

2、面向对象编程

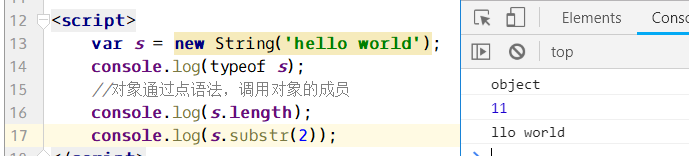
面向对象编程就是基于对象的编程。面向对象编程简称OOP（Object-Oritened Programming）为软件开发人员敞开了一扇大门，它使得代码的编写更加简洁、高效、可读性和维护性增强。它实现了软件工程的三大目标：（代码）重用性、（功能）扩展性和（操作）灵活性，它的实现是依赖于面向对象的三大特性：封装、继承、多态。在实际开发中 使用面向对象编程 可以实现系统化、模块化和结构化的设计 它是每位软件开发员不可或缺的一项技能。

# 四、定义对象

1、new 内置函数

之前学习过的String对象、Date对象、Array对象、RegExp对象。使用这些对象的时候，可以new这些函数。然后将得到的返回值当做对象来使用。比如使用字符串对象：

var s = new String('hello world'); // 通过new内置的函数，得到对象。

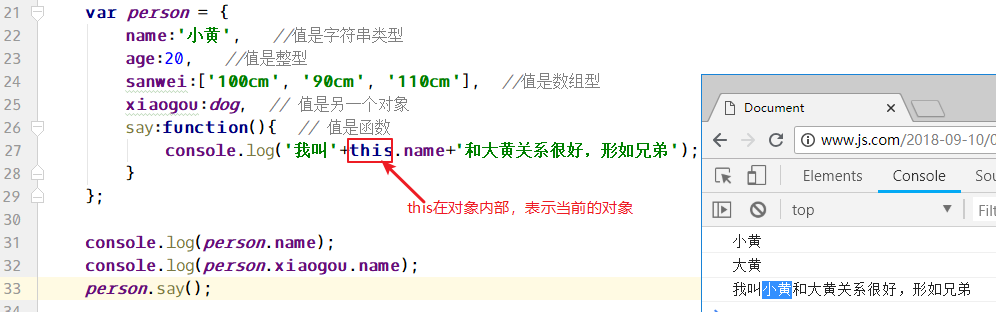


2、直接量语法

直接量语法定义的对象，值可以是任何的数据类型：



直接量语法定义的对象中，this表示当前的对象：

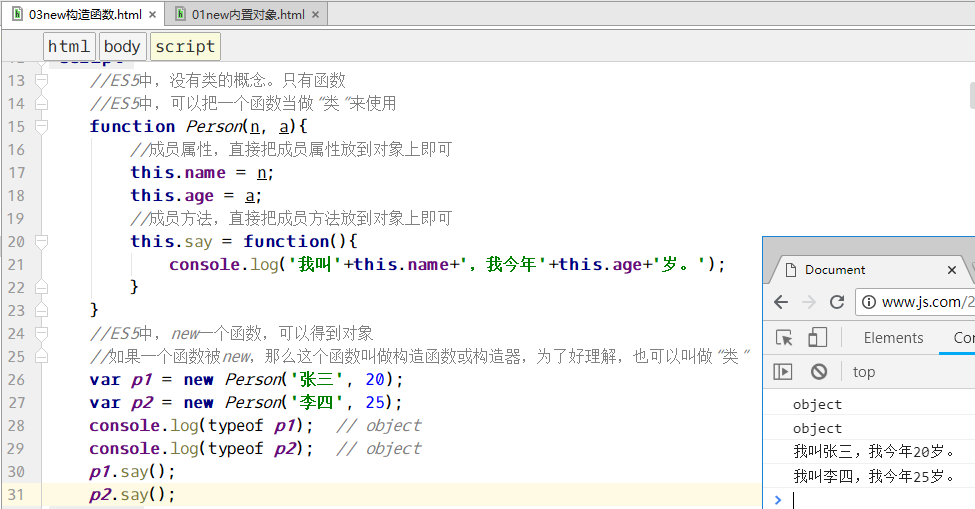


3、new 构造函数方式

ES5中没有类的概念，只有构造函数或构造器。要想得到对象，只能new一个构造函数。

**什么是构造函数？什么是普通函数**？

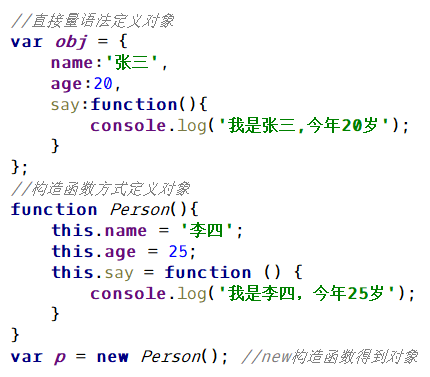
定义函数的时候，正常按照函数的语法来定义即可。如果这个函数正常使用，那么还是一个函数，如果一个函数被new了，那么这个函数就可以叫做构造函数。



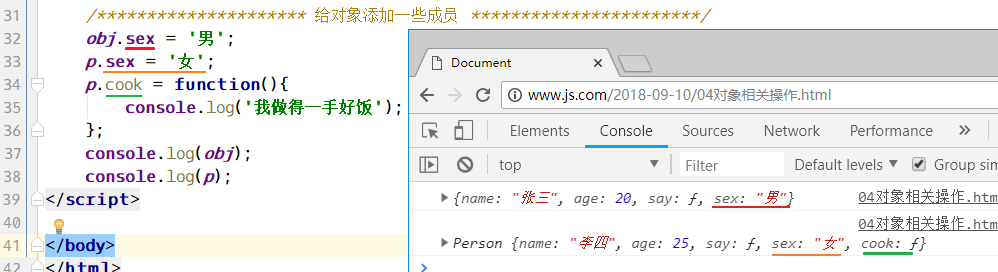
# 五、对象相关操作

1、添加成员

先定义两个对象，一个用直接量语法，另一个用构造函数方式。



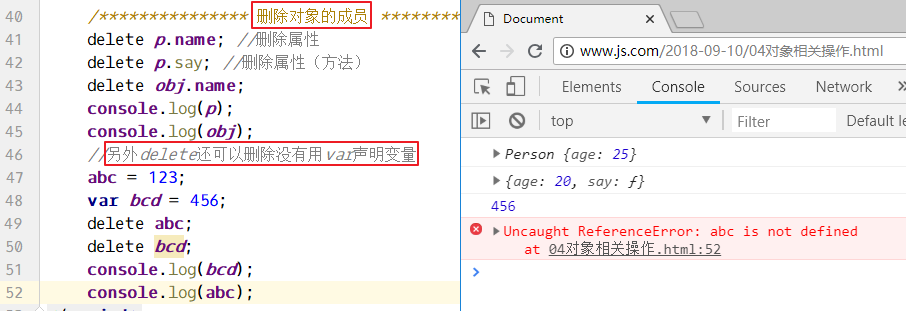
得到对象之后，可以为对象添加一些成员（属性和方法）：



2、删除成员

使用delete关键字来删除对象的成员（属性、方法）；

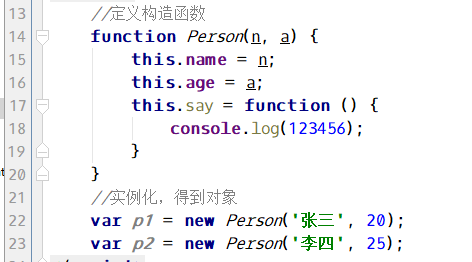
另外，delete也可以删除没有用var声明的变量。



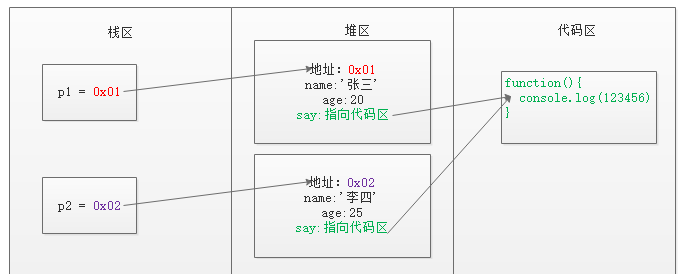
# 六、对象在内存中的存在形式

**对象在传值上，是引用传递**。在使用对象的时候，实际上都是使用的对象的地址。

代码：根据构造函数得到两个对象：

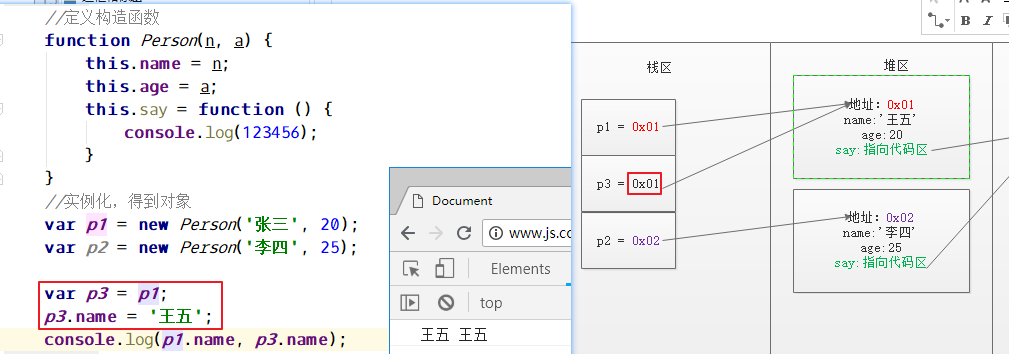


得到的两个对象在内存中的形式：

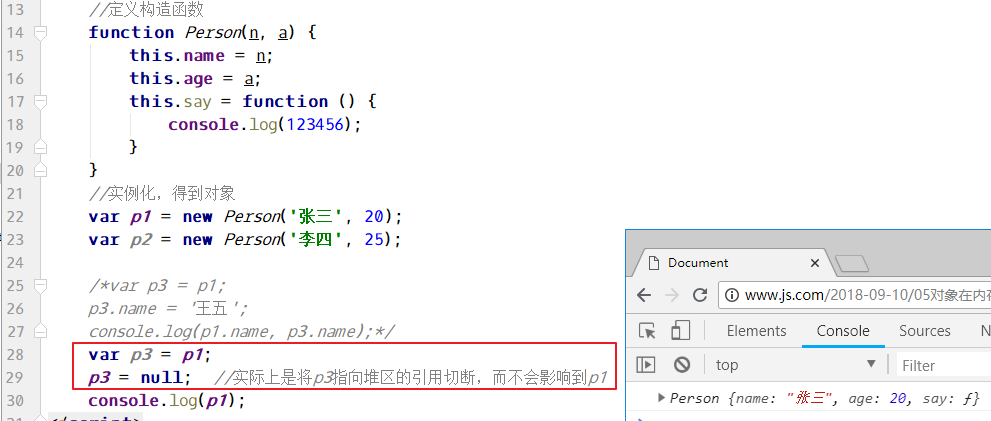


**在实际使用对象的时候，实际上都是使用的对象的地址**。

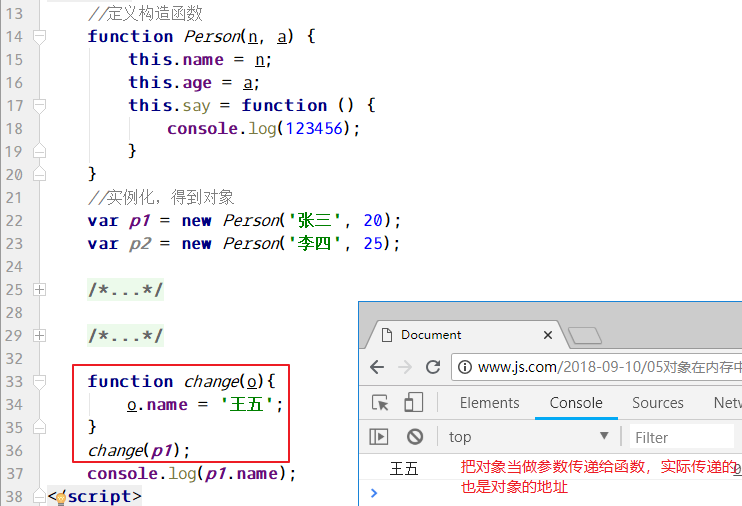
小例子一：



小例子二：



小例子三：



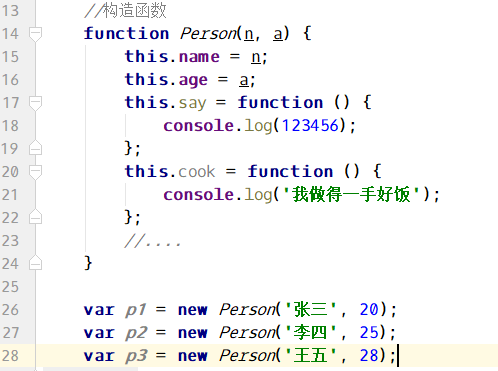
# 七、原型对象（关键）

1、原型对象

1、没有原型对象的情况

在实例化得到一个对象的时候，会为这个对象分配一个原型对象。

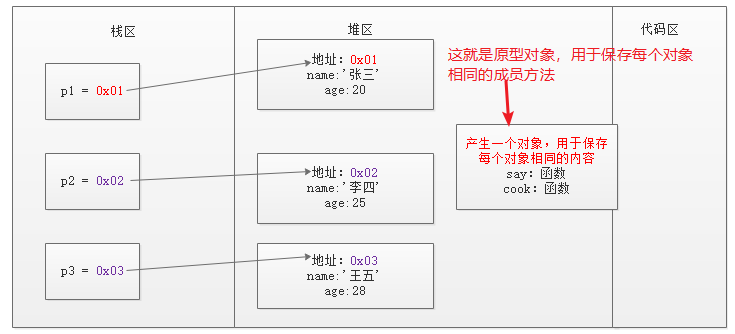
代码：一个构造函数，实例化得到三个对象。



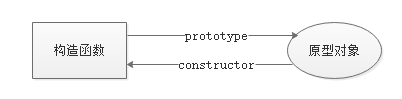
在内存中，会分别为每个对象开辟新的空间。发现每个对象中的say和cook都一样，这样的话，会占用大量的内存。解决办法就是使用原型对象。

2、原型对象

**在实例化得到对象的时候，系统会为构造器创建一个对象，该对象会保存构造器的每个实例对象的相同内容，这个对象就是原型对象**。



原型对象不能单独存在，肯定要和构造函数产生关系才行。原型对象和构造函数的关系如下：





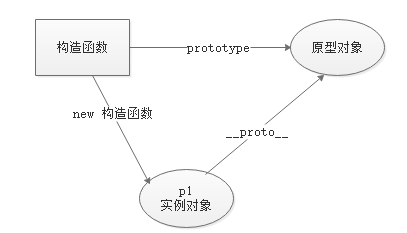
3、使用构造函数和原型对象共同来定义“类”

有了原型对象，再定义构造函数的时候，就可以将每个对象独有的内容放到构造函数中，将每个对象相同的内容都放到原型对象上，具体看下面的代码：

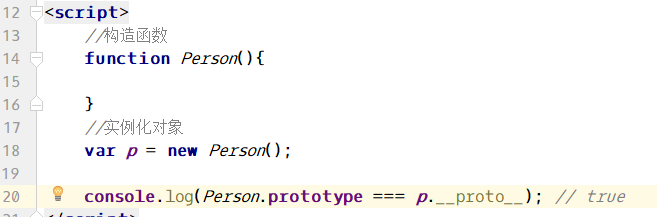


**这样做的目的是既能区分开每个实例对象，又能节省内存**。

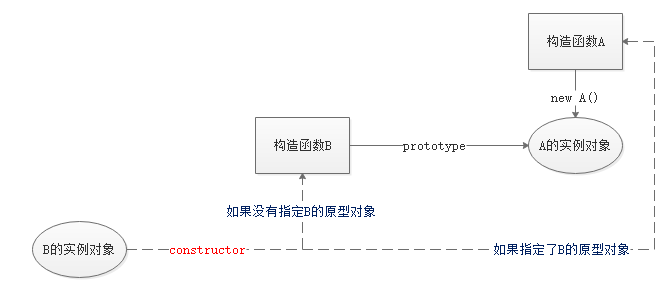
4、原型对象、实例对象、构造函数的关系

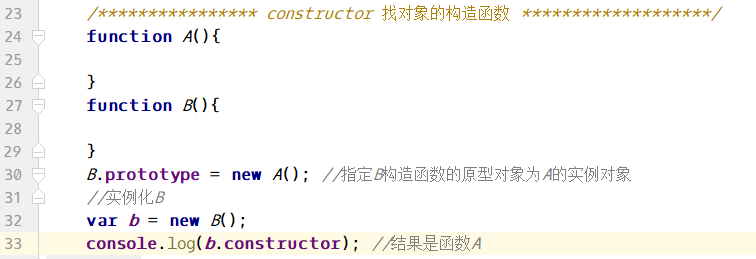


测试：

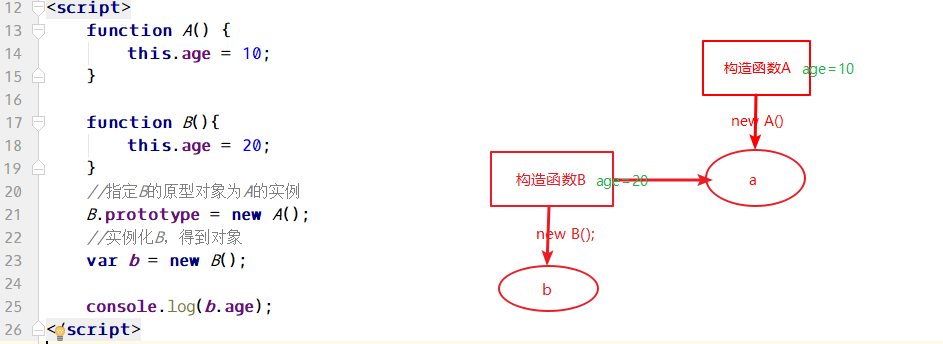


**constructor：真实作用是找对象（实例对象、原型对象）的构造函数的**。





2、原型链



**找一个对象的成员时：**

**优先从对象自身查找；**

**然后从对象的构造函数中查找；**

**然后从构造函数的原型对象上查找；**

**然后从原型对象的构造函数中查找**

….

**这种查找的方式就是原型链**。

一个问题：这样一直向上查找，最顶层是什么呢？

最顶层是内置的Object对象。Object是所有对象的默认的原型对象。

3、应用--扩展内置对象

比如在使用内置的String对象的时候：

var s = new String('hello world');

s.length;

s.substr(1,3);

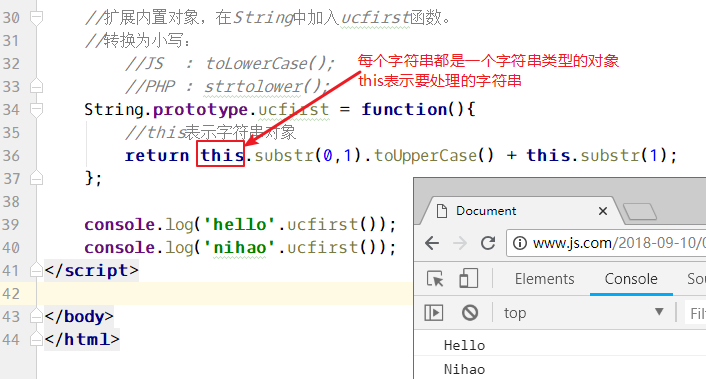
s.indexOf('h');

实例化String之后，就可以调用String对象中的各个方法了。

下面模拟内置的String构造函数的写法：



扩展内置对象，为String加入ucfirst函数（功能是将字符串的首字母大写）：



4、应用--为DOM对象添加方法

为所有的dom对象添加一个css方法，因为所有对象的最顶层的原型是Object，所以在Object的原型对象中添加一个css方法即可。



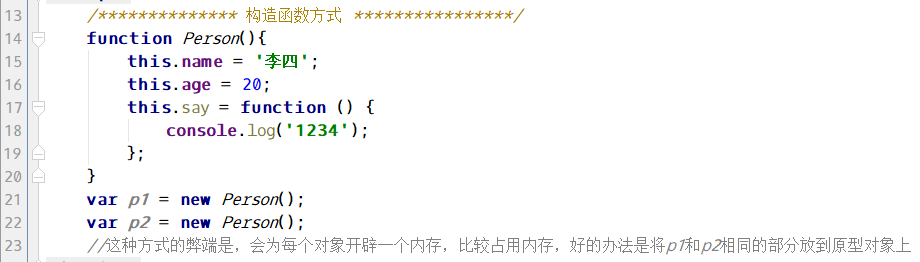
有因为this表示调用css的每个对象，如果css方法中返回this将会形成链式的调用方式。



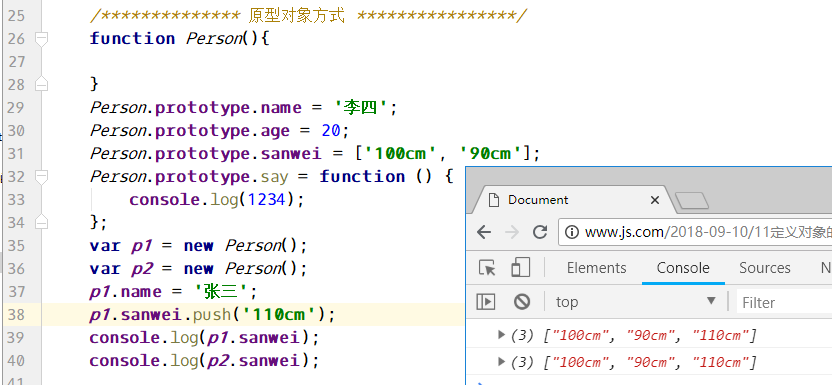
# 八、定义对象进阶

1、构造函数方式

定义一个函数，然后实例化即可。



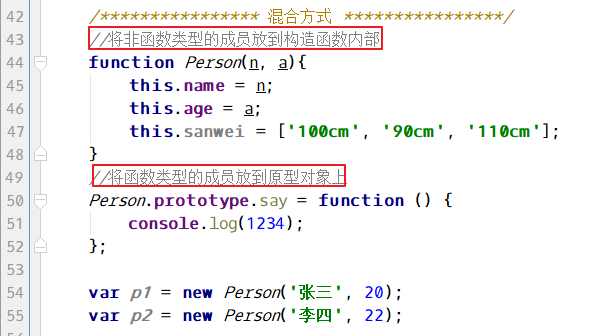
2、原型对象方式



**这种方式真正的弊端是，如果原型对象中有一个引用类型的值，则修改其中一个实例对象，另一个也会修改**。

3、混合方式定义

**思路是：将函数类型的成员绑定到原型对象上，将其他值绑定到构造函数内部**。



**实际开发中，推荐使用这个方式**。

4、动态混合方式

上面的混合方式，从功能和性能上已经没有问题了，但是从结构上看起来不像PHP中的类，所以希望将所有的成员属性和方法的代码都放到一个大括号中，所以才会出现下面的动态混合方式。



# 九、函数进阶

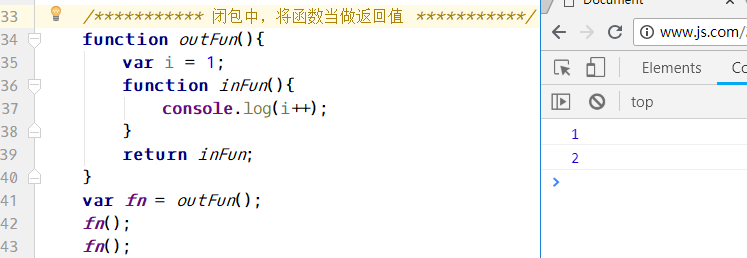
1、函数也是值

JavaScript 语言将函数看作一种 **值**，与其它值（数值、字符串、布尔值等等）地位相同。凡是可以使用值的地方，就能使用函数。比如，可以把**函数赋值给变量和对象的属性**，也可以当作**参数传入其他函数**，或者**作为函数的结果返回**。函数只是一个可以执行的值，此外并无特殊之处。

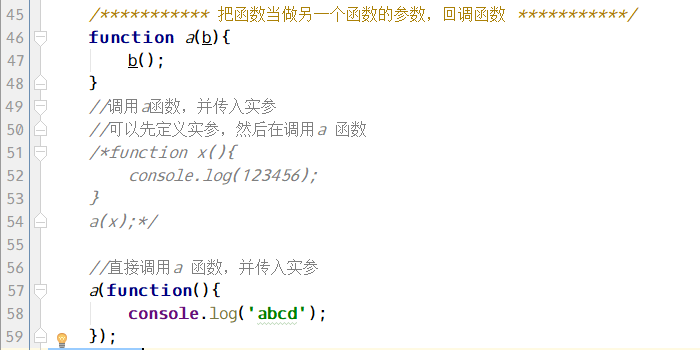
由于函数与其他数据类型地位平等，所以在 JavaScript 语言中又称函数为 **第一等公民**。



除此以外，还可以把函数当做返回值。



也可以把函数当做另外一个函数的参数，如果一个函数当做另一个函数的参数了，那么把函数类型的参数叫做**回调函数**。

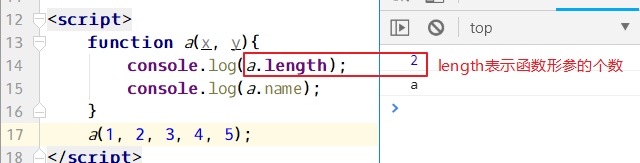


2、函数也是对象

JS中处处皆对象，函数也不例外。

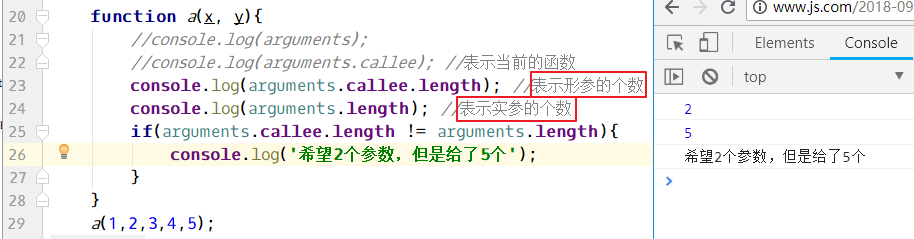
那么如何理解函数也是对象呢？这里主要要表达的是**函数也可以使用对象那样的点语法**。比如前面学习的Person**.**prototype;

比如在函数内部，可以使用“函数.length”来表示函数的形参个数，可以使用“函数.name”来获取当前的函数名。

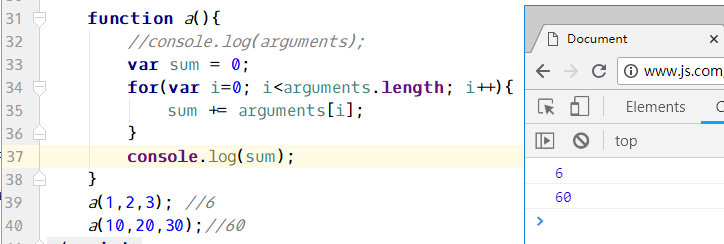


3、arguments对象

arguments对象，存在于函数的内部，它能够表达函数的实际参数（实参），除此以外，arguments对象还有一个属性callee，它表示函数的名字，arguments的length属性表示实参的个数。



小例子：计算函数参数的和：



4、call、apply、bind函数

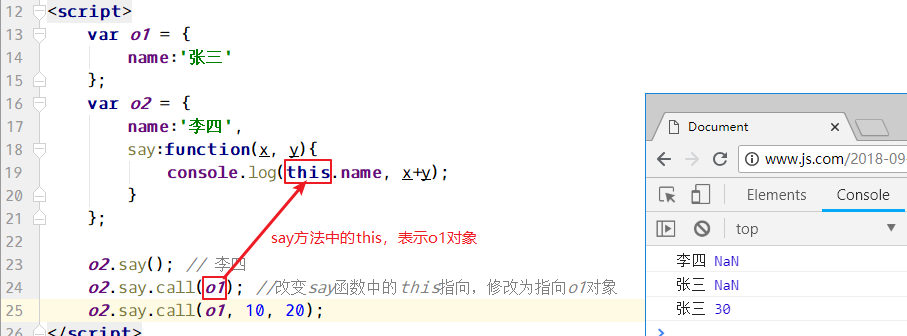
这三个函数都可以改变函数内部this的指向。

函数**.**call(需要指向的对象，参数1， 参数2…); 调用之后，会直接执行函数

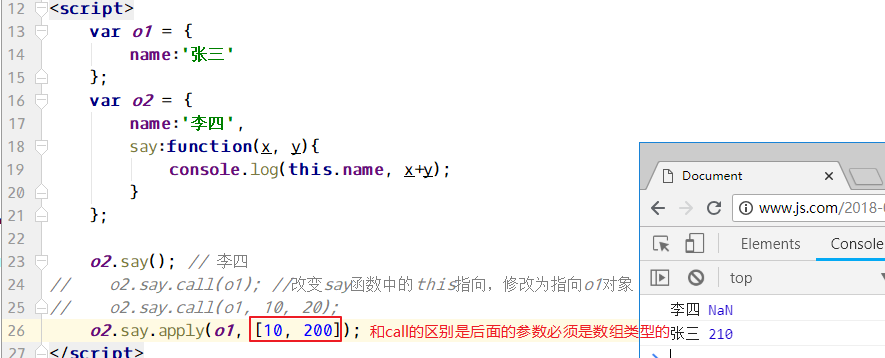
函数**.**apply(需要指向的对象，[参数1， 参数2…]); 调用之后，会直接执行函数

函数**.**bind(需要指向的对象，参数1， 参数2…); 调用之后，不会执行函数。

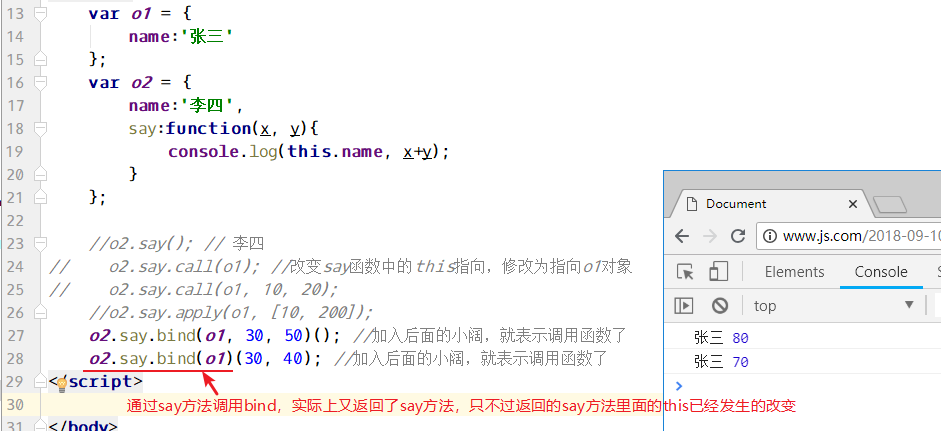
call方法演示：



apply方法演示：



bind函数演示：



小例子，查找数组中的最大值：

