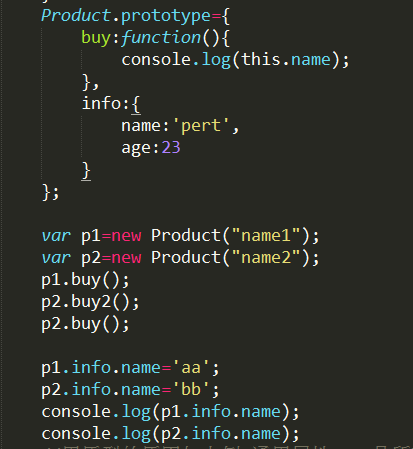


//用原型的原因如上例 通用属性name是所有实例都需要的，并且值可能不同

//但通用属性buy如果所有实例是一样的不会不同，写到原型里，多次实例化的时候只生成一次

//原型对象不管实例化多少次，都只会生成一次，但方法里的会每次实例化都生成一次，不节省内存。

//原型对象中的方法和方法被所有实例共享，修改原型中的属性和方法，就影响到了所有对象中的方法和属性



console.log(p1.name2);//kkk

p1.name='kk22';

console.log(p1.name2);//kkk

//bb/bb info需要是引用类型才行，值类型的不行

var str="string";

console.log(str.length);//前面定义了一个字符串，这里调用 字符串length方法时相当于 var str=new String()初始化构造函数，才有length方法。

var a=10;

function test(){

//a=5;

alert(a);//up is 5 down is undefined;

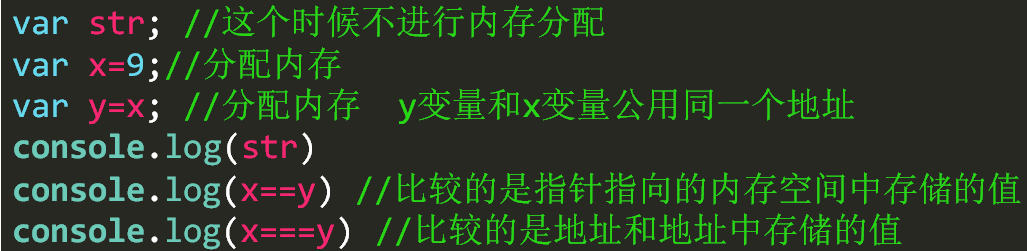
//var a=5;

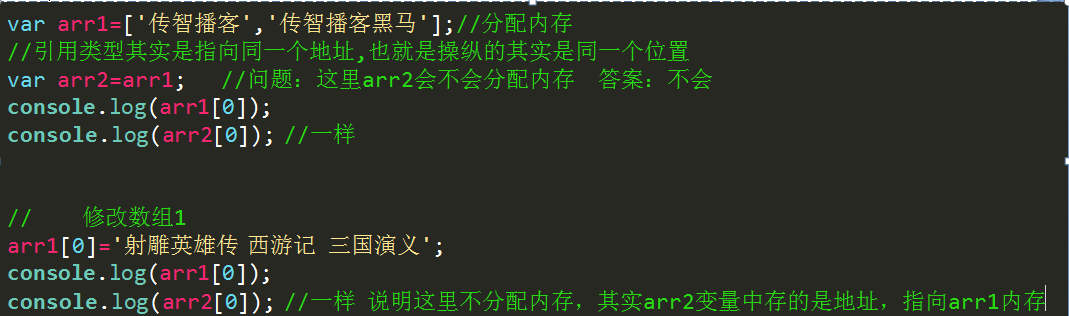
}

test();

值类型赋值，会重新分配内存

引用类型赋值其实赋的是指针







经典面试题 原理如上

var a={"x":1};

var b=a;

a.x=2;

console.log(b.x);//2因为指向同一个内在空间了，引用类型

console.log(a.x);//2因为指向同一个内在空间了，引用类型

a={"x":3}

console.log(b.x);//2新建一个对象要重新分配内存地址空间只对a这个对象来说不影响B

console.log(a.x);//3新建一个对象要重新分配内存地址空间只对a这个对象来说

a.x=4;

console.log(b.x);//2因为各自分配了空间了，值类型

console.log(a.x);//4前面新对象要重新分配内存地址空间时3，这时把3内在位置改值为4

题2

Arguments.length arguments.callee函数参数的个数，函数本身调用，通常用于递归

//2递归方法求积

var fn=function(n){

if(n>1){

return n\*fn(n-1);

}

if(n==1){

return 1;

}

if(n==0){

return 0;

}

}

console.log(fn(4));

//3callee求积

var fn3=function(n){

if(n>1){

return n\*arguments.callee(n-1);

}

if(n==1){

return 1;

}

if(n==0){

return 0;

}

}

console.log(fn3(4));

值类型的内存保存在栈里面，引用类型的保存了堆里面，堆 相当于一块内存集合，。

内存泄露 指一块被分配的内存即不能使用，也不能回收，泄露的内存越来越多，，就一直占用了内存

对象创建方式：

//1object方式

Var a=new Object();

aa.name=’xiaoming’;

aa[“age”]=90

//2工厂模式 不需要了解过程，相当于封装函数，函数输里面包含创建对象。外面不需要创建，交给工厂来做，只需要调用函数

Function createPerson(name){

Var obj=new Object();

Obj.name=name;

Obj.age=19;

Obj.attact=function(){

Alert(‘攻击你’);

}

Return obj;

}

Var person1=createPerson(“张三”);

//3 构造函数 new this function方法

//4原型方法 和构造函数类似，只是通用属性和方法可以放在原型里，节省空间

//5混合模式，构造函数和原型方式，将共享的放在原型中，不共享的放在构造函数里

//6拷贝模式，相当于拷贝对象,json对象，如果要多个json对象拷贝，可以用arguments 函数参数是tar sor1 sor2 sor3;

function extend(tar,sor){

for(var i in sor){

tar[i]=sor[i];

}

return tar;

}

var boy={name:"男生",age:"2"};

var xiaomi=extend({},boy);

var xiaoli=extend({},boy);

xiaoli.name="女生";

console.log(xiaomi.name);//男生

console.log(xiaoli.name);//女生

var boy={name:"男生2",age:"2"};

console.log(xiaomi.name);//男生

boy={name:"男生",age:"2"};

console.log(xiaomi.name);//男生

console.log(xiaoli.name);//女生

JS对象总体分为三种 1.内置对象 String Date Math Number Object Null Boolean Function RegExp 2.BOM对象 Window Document Histroy location cookie session screen 3.自定义对象，就是上面面向对象那些

String有很多内置原型方法比如 string.length

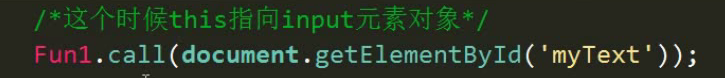
可以自己增加或者修复原始方法 String.prototype.trim=function(str){};

就可以调用 String.trim（str）调用自己给内置对象的String增加和修改trim方法

生成随机10到100的数字Math.floor(Math.random()\*(100-10)+10);

Call a对象要用B对象方法，可以拷贝过来，不过太重复了，可以用call借用一下供爷法则。同时改变了原对象的this指向为自己了。

这时候this输出出来不是fun1的this window对象，而是input对象了



var a1={

getSum:function(a,b){

return a+b;

}

}

var a2={

getJi:function(a,b){

return a\*b;

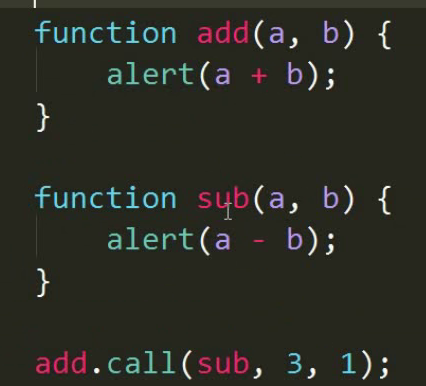
}

}

console.log(a2.getJi(3,4));//调用a2的方法正常调用，要调用a3的加方法用call

console.log(a1.getSum.call(a2,3,4));//a2调用a1的方法，只取方法名，参数在a2后面加上就好了

函数也可以借用，因为函数也是对象

4

伪数组就是包含length属性的json对象。Key都是12345形式，和数组都是数据集合，伪数组每次都要去计算length个数，自己去拼装对象。没有数组好，为什么 要使用伪数组呢？

GetElementsByTagName/ClassName获得的都是伪数组，jquery也是伪数组实现的。比如$(“.class”)其实得到的是json={1:dom1,2:dom2,3:dom3}

伪数组转换成数组方式Array.protytype.slice.call(divs);

Var json={1:”苹果”，2：”香蕉”,3:”菠萝”，length：3}

Var arr=[“苹果”,“香蕉”,”菠萝”];

Aplly功能和call一模一样，只有参数不一样。Call的传参是平铺的。

Appley是把所有数组放在数组里的。console.log(a1.getSum.apply(a2,[3,1]);

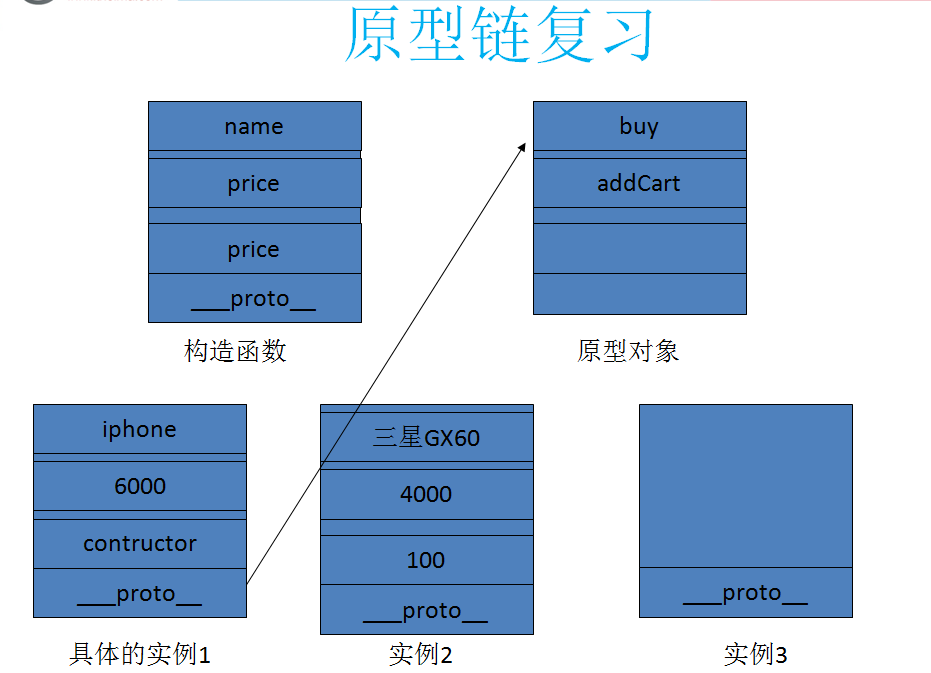
//获得最大值方法

var arr=[2,3,5,33,4];

console.log(Math.max.apply(null,arr));//33

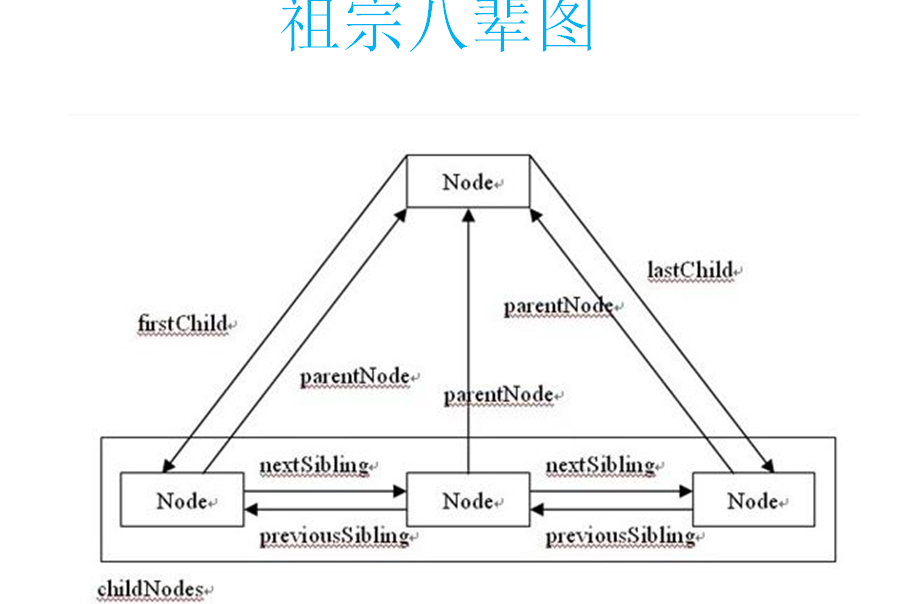
console.log(Math.max.call(null,3,4,44,6));//44

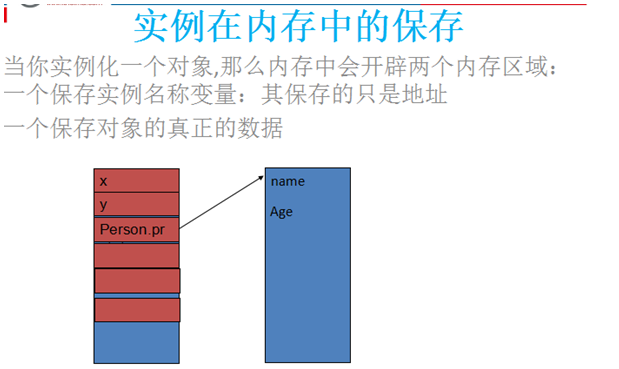
window.location.search可以获得参数内容?id=1&name=33



当我们new 一个实例后，系统自动做了如下事情

* 1. 创建一个空对象 var p = {}
  2. 拷贝构造函数中的方法属性到空对象中
  3. 自动成生一个属性\_proto\_指向类的原型p.\_\_proto\_\_ =  XXXX.prototype





注意ajax格式 dataType,并不一定products.json就一定返回的是json，不行的话查看data类型，或者是字符串string可以用html



错误，1.这里dataType：”json”是不行的，要用html因为json文件并不一定返回的是json 2.json文件不对应，忘记了””,｛｝号不匹配不对应 切记，基本格式要多检查，逗号用成中文了

框架

事件流 捕获和冒泡

冒泡是从最特定的对象到最不特定的对象（body）

捕获是从最不特定的对象到最特定的，与冒泡相反。

委托使用场景

1. 列表，省掉使用循环，用委屈子元素与你元素的关系来处理
2. 动态生成的元素，不能直接使用事件，不在DOM树里，要通过这种方式处理

对象实例化，属性，私有公有属性实例，要理解下

var p=function(name){

this.name=name;//公有属性

name2="ddd";//私有属性

this.show=function(){

console.log(this.name);//kkk

console.log(name2);//ddd

// console.log(name3);//undefined

}

}

p.prototype={

show2:function(){

console.log(this.name);//kkk

console.log(name);//空

// console.log(name3);//undefined

}

}

p.prototype.show3=function(){

console.log(this.name);//kkk

console.log(name);//空

// console.log(name3);//undefined

}

//p.name3="name3";//全会报错undefinded

var p1=new p("KKK");

p1.name3="name3";

p1.show();

p1.show2();

p1.show3();

console.log(p1.name3);//name3如果在实例化以前赋值输出会是undefined,没实例化以前不是对象，不能直接用.方法加属性或者方法

var obj=new Object();

obj.name="name4";

console.log(obj.name);//name4

函数内部变量参数和方法怎么被外部使用？

1. 放在json对象里 然后window.aa=json。外部就可以用aa.name
2. 定义全部变量var o=new Object();在函数内部o.name=name,然后函数外部就可以使用了o.name了

或者闭包(function(w)function{ w.aa=name})(windwo);

3.定义变量，然后函数里用return 返回数据jason格式，就可以使用变量的属性了

/\* (function(w){ w.aa="name";})(window);

console.log(aa);//name\*/

/\* var o=new Object();

(function(){

var name="name1";

o.name=name;

})()

console.log(o.name);\*///nam

(function(){

var name="name1";

var kkk={

name:"anme2",

age:"ange"

};

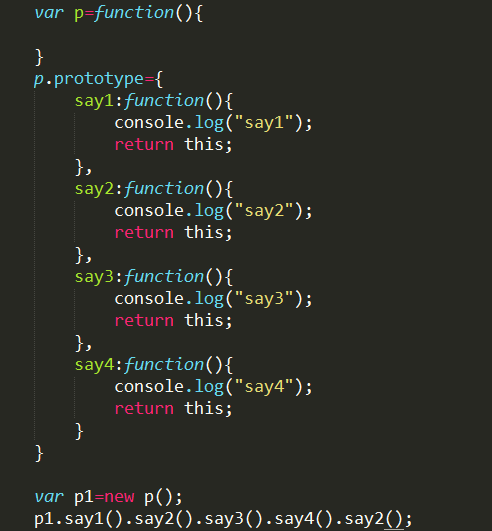
window.json=kkk;

})()

console.log(json.name);

链式访问方法原理，就是在前一个方法里要返回对象this

Jq.add().say().play();在jq的每个方法add say play方法中return this就可以实现链式访问了



以后先写立即函数 （解决污染问题） window.onload=function(){}写在这里面后面的;号不能省略，不然后面的立即函数会报错（）里可以传参，函数内部可以return 就比如XXXonclick=(function(i){})(i);

