Js高级

1）面向对象特征：

封装，继承，抽象

2）创建对象的方式：

字面量创建：var obj = {};obj.name=”你好”

构造函数：var obj = new object();

工厂制造函数：

3）自定义构造函数的特点：

1首字母大写；2通过new关键字调用，不用写return返回值

3可以返回一个实例对象；4通过构造函数创建出来的对象自带一个constructor属性，里面存的是创建这个对象的构造函数，可以改变。

每一个实例对象都具有一个 constructor 属性，指向创建该实例的构造函数

如果重置了 prototype这个原型对象 记得修正 constructor 的指向

1. 静态成员：直接用构造函数添加的成员，eg: Student.sex=”女”

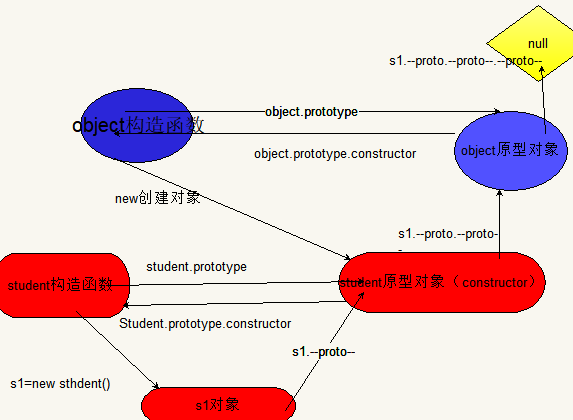
动态成员：跟对象相关的成员：eg: this.name=”abc”

1. 构造函数的原型对象（为了解决多个对象会存储多个sayhi方法）：student.prototype:本身也是一个属性，一般把公用的属性和方法放在原型属性上，只要是对象就有

--proto--原型属性和constructor--指向原来构造函数

对象中的原型：s1.--proto--,指向构造函数中的原型属性

原型图：



1. 自调用函数：有一个封闭空间可以保护私有属性外部访问，有一个封闭的作用域，外部访问不到。需要给最高级对象window设置一个属性，手动暴露出去

同一个页面出现多个自调用函数用符号隔开，否则会报错

优化自调用函数优化，在每一个自调用函数用“；”隔开

1. This的指向：

This指向是由函数的调用方式去决定的

计时器/普通函数：window

事件执行函数：事件触发者

借用构造函数：指向当前实例/对象

1. 函数的一些方法（函数fn是function的对象）---也有方法和属性

bind:方法，新建一个方法，他的第一个参数改变函数this的指向，函数也是对象。返回新函数需手动调用（）

Call方法：1是改变this指向，直接执行去调用call的这个原函数。2第一个参数是要指向的对象，后面参数是对应函数的参数，4返回值就是原函数的返回值，

apply方法：1只有两个参数，第一个设置要指向的对象，改变this指向，第二个是参数数组。Apply没有返回新函数，本身没有返回值，返回值就是原函数的返回值。立即调用

1. ；（function（w.u）{}()) (window,undefind)

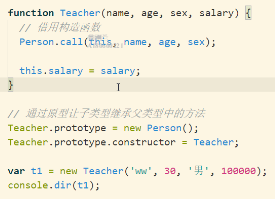
window方便代码压缩少写字母

Undefind处理兼容性问题，防止使用老版浏览器的人写的代码把undefind重新赋值

1. 函数组合继承【借用构造函数继承和原型继承】：让子类型student继承父亲person的方法（方法只在父级原型中才有），如果直接把父级原型给子集原型赋值后不合理，因为复杂数据类型函数赋值的是地址，两个地址还是指向同一个空间，其中一个里面内容改变i新增一个方法也会影响另一个父亲也会增加，为了避免这种弊端（如果给子集原型加上特有的方法后也会影响到person的方法，他俩指向同一个空间）。所以这样有弊端，需要new一个person实例对象p1，赋值给student原型。这个时候把studengt实例对象化，st1对象调用sayhi方法先去studengt自身找，再去他的原型找，原型找不到(此时student原型也是一个person构造函数产生的对象p1-)-再去p1对象原型找（对象不会新增加一个功能属性，没有这样写的避免了这种弊端）
2. 如果这个时候给student增加一个新方法，不会影响（对象person不会新增加一个功能属性，没有这样写的避免了这种弊端）

。只会在自身studengt找到就重新赋值，找不打就直接增加，不会影响person让它也增加一个方法

继承：组合继承。借用构造函数继承，原型继承。



Js中继承是构造函数与构造函数之间的继承，包括两点：

1公共方法的继承：

2对象成员的继承：

函数声明和函数表达式区别：

函数声明：function fn (){}需要提升

函数表达式：var m=function (){}提升的只是var m 变量名

This指向：

1，普通函数调用：fn()---this指向window//window.fn

2，方法调用（对象里面的方法属性）：谁调用指向谁：obj{fn:function(){}},obj.fn---方法

3，作为构造函数调用：this指向该构造函数创建的对象---new出来的对象

4，作为事件处理：on click指向出发事件的对象

5，作为定时器的参数：this指向window

大部分情况下，函数内部的this，是由函数调用时候来确定指向的，看谁调用this就指向谁

函数的创建形式：

1,Var fn =new Function(“a”,”b”,”console.log(a+b)”)--作为一个对象

2,function fn(){};

3,var fn = function(){}

函数的其他成员：fn.argument--是一个装函数实参的伪数组

fn.caller:函数的调用者

fn.name:函数名

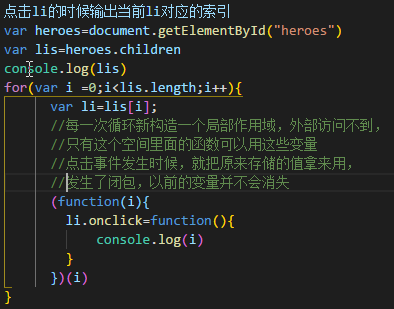
fn.length:行参个数

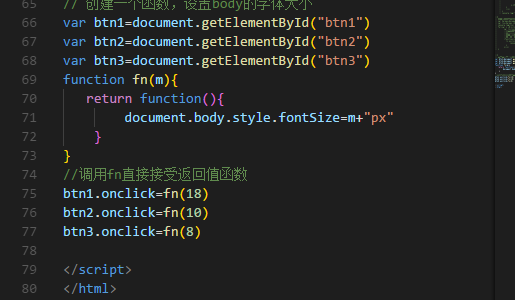
高阶函数有两种

1. 函数作为参数传入另一个函数里面--定时器
2. 函数作为一个返回值的时候---bind

闭包：

闭包就是一个作用域访问另一个局部作用域的变量（属性/方法）



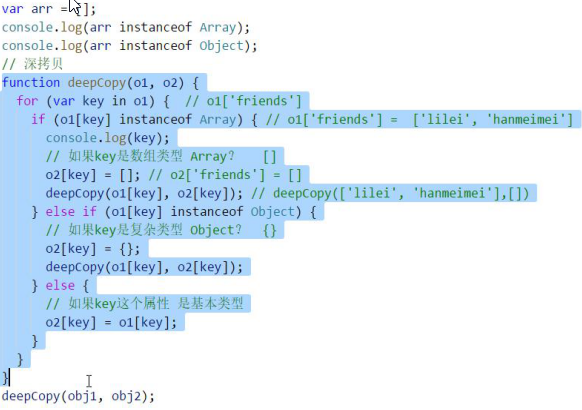


递归：函数自己调用自己，每一步都在等待自己调用的一个结果，一般有一个结束条件，否则会内存溢出



浅拷贝和深拷贝：

浅拷贝：把一个对象o1赋值到另一个对象o2，如果赋值的是o1基本数据类型，可以直接拷贝，如果是复杂数据类型（另一个对象{}或者数组[]）,只会复制这个数据类型的地址，两个地址共同指向同一块空间，{缺陷：如果有一个改变这个里面的数据导致另一个拷贝的对象也会改变}---深拷贝--拷贝的时候需要重新开辟一块空间



正则表达式：用开匹配规律规则的表达式，对字符串操作的一种逻辑公式

应用：匹配/查找/替换 ,Ctrl+f:调用查找框,Ctrl+h：调用替换框

表达式组成：普通字符+特殊字符 验证IP地址 192.168.1.10:(^\d{1,3}\(.\d{1,3}){3}$)

限定符：修饰的是前面的一个字符



内置对象：第一个参数是正则表达式，字符串类型。第二个参数是flag:i忽略大小写。g全局匹配。Eg : var reg = / \d+ /g

两种写法：创建正则表达式的对象，看字符传是否满足要求，返回值是布尔类型fause/true,如下

1)var regularExpression = new RegExp(“ab[a-z]”,”i”);

Var str = “aBd”

Console.log( regularExpression.text(str))

2))var regularExpression=/ab[a-z]/i

Console.log( regularExpression.text(str))

跟正则表达式相关的方法：

1RegExp对象：text( ) 匹配与表达式相符合的字符串 / exec( )：提取

替换功能如下：str . match ( reg )//split//replace

