1. 如何从ftp上下载html和js文件?

答: 右键->链接另存为

2. 闭包的外层函数向外抛出内层函数共有几种方式?

答: 3种:

1. return function(){ ... }

2. 全局变量=function(){ ... }

3. return [ function, ...]

return { 方法: function(){ ... } }

3. 为什么lilei.intr()可以调用到原型对象中的intr()函数?

答: 因为原型对象是lilei的父级对象。程序中规定，调用函数时，先在对象本地查找。如果对象本地没找到，就自动延\_ \_proto\_ \_向父对象找。

4. 到底怎样创建一种类型?

答: 2步:

1. 先定义构造函数:

function Student(sname,sage){

this.sname=sname;

this.sage=sage;

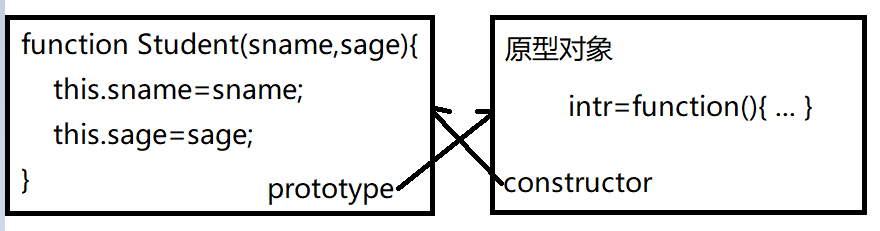
}

2. 为构造函数的原型对象强行添加共有方法:

Student.prototype.intr=function(){

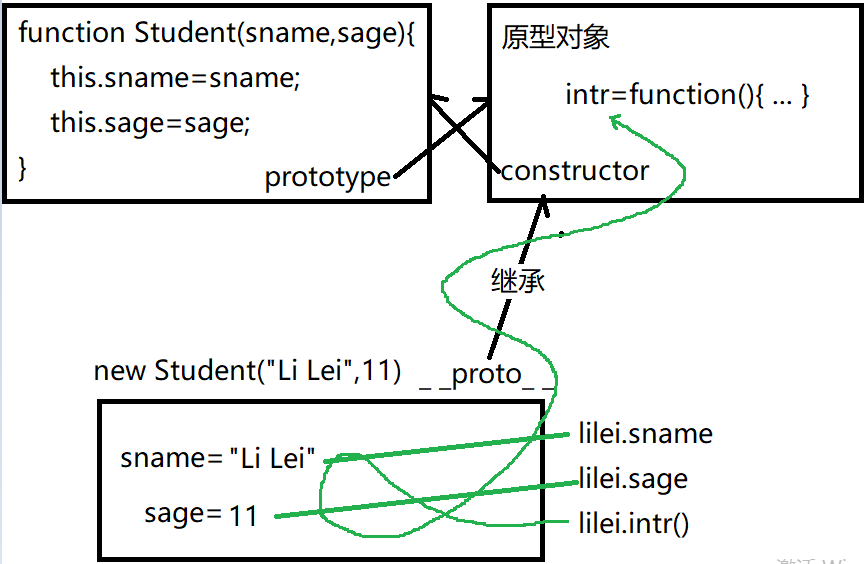
console.log(`我是 ${this.sname}, 我今年 ${this.sage} 岁`)

}



创建类型后，如何用类型，创建子对象:

var lilei=new Student("Li Lei",11)



5. 都指向原型对象，为什么一个属性叫prototype，一个属性叫\_ \_proto\_ \_?

答: prototype和\_ \_proto\_ \_指的是同一个对象，都是构造函数的原型对象。

和现实一样，以不同的角色去访问原型对象时，必须用与自己身份相符的称呼。必须有所区分。

比如: 妈妈称呼爸爸，就叫老公

孩子称呼爸爸，就应该叫爸爸！

6. 原型对象中的this->?

答: 判断this一定不好看定义在哪里，保存在哪里

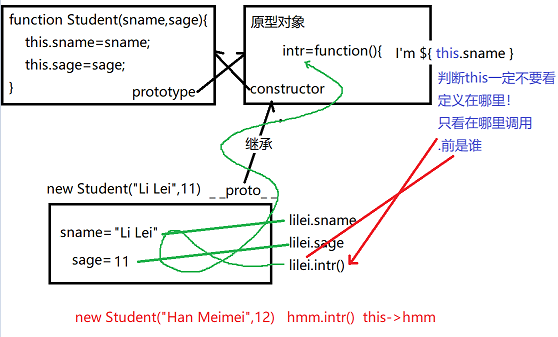
只看在哪里调用！看.前是谁

Student.prototype.intr=function(){ ... this.sname ... }

虽然存在原型对象中，但是不是在原型对象中被调用的

而是在稍后，由子对象调用: lilei.intr() hmm.intr();

所以，原型对象中的方法中的this，指将来调用该函数的.前的子对象。



7. 一共讲了几种this的情况?

答: 4种:

1. obj.fun() -> this->obj

2. fun() 或(function(){ ... })() this->window

3. new fun() -> this->新对象

4. Student.prototype.intr=function(){ ... this.sname ...}

this-> 将来调用intr()的.前的子对象

8. function Student()是一个函数，为什么能.prototype属性？

答：js中一切都是对象，函数也是对象。只要是对象，都可以用.访问属性。都可以添加属性。其实，函数和数组，和lilei一样都是存东西的对象。只不过，函数中，存的是代码段，而数组和lilei中存的是数据而已。

9. 何时使用prototype，何时使用\_ \_proto\_ \_?

答: 1. 向原型对象中添加共有属性和方法时，必须通过构造函数妈妈找到原型对象爸爸才能，向其中添加共有属性和方法。

类型名.prototype.共有属性=值

2. 如果通过儿子，查看自己爹里的内容时，可以通过子对象.\_ \_proto\_ \_获得儿子的爹。

其实，向原型对象中添加共有属性，也可以通过子对象.\_ \_proto\_ \_.共有属性=值，强行添加，但是通常，向原型对象中添加共有成员时，还没有子对象呢。所以，很少用子对象.\_ \_proto\_ \_去操作父对象。

10. 如果孩子擅自修改了共有属性，导致在自己身上添加了同名自有属性，那么，如果调用原型对象中的方法时，方法中刚好用到这个共有属性的话，方法使用的是共有的还是自有的属性？

答：方法使用的是自有的同名属性。因为，访问属性使用的是this.属性名。 this查找属性时，都是按先自有，再共有的顺序查找的。

11. 何时使用console.log(), 何时使用console.dir()

答: 需要详细了解对象在内存中的存储结构和内容时，就用console.dir()

如果只是一般的输出对象的普通内容，就用console.log()。

有时console.log会自动调用对象的toString但是不是总是。

12. 为什么new Student()生出的lilei和new Array()生出的数组，以及new Date()生出的日期对象，内部存储结构不一样？

答：子对象的存储结构具体是什么样，其实是由构造函数妈妈决定的。构造函数妈妈肚子里规定了哪些属性，将来孩子身上就会长出哪些属性。

比如，function Student(){}妈妈中规定，所有学生孩子都有sname属性和sage属性两个属性，不是三个属性，也不是四个属性，属性名不叫name，也不叫age。这些都是构造函数妈妈肚子里规定的。

同理，别人家的构造函数妈妈肚子里也规定了别人家的孩子的属性结构和类型。不同的构造函数妈妈肚子里规定的数据结构，属性名，属性个数，都是不一样。所以，用不同构造函数妈妈生出的孩子，长的都不一样。

但是，因为Array()和Date()，是浏览器内置的，使用浏览器底层代码实现的。所以，我们看不到源代码。只能想象。

13. 构造函数和原型对象之间的关系是什么意思，为什么构造函数和原型对象之间要有关系?

答: 构造函数通过prototype属性指向自己的老公——原型对象

原型对象通过constructor属性指向自己的老婆——构造函数

因为开发时，很可能需要随时找到构造函数和原型对象

如何找原型对象？原型对象本身没有名字，不能直接找到。要找原型对象，只能通过先找到构造函数，再通过构造函数的prototype属性，才能找到隐藏的原型对象

万一不知道构造函数的情况下，想找到一个孩子的妈妈:

先找到孩子，再找到还的爸爸: 孩子.\_\_proto\_\_

通过孩子的爸爸，才能找到孩子的妈妈：

孩子.\_\_proto\_\_.constructor

14. 将Array.prototype.sum=function(){ ... } 换成

arr1.\_\_proto\_\_.sum=function(){ ... } 为什么报错？

答: 这个时候有arr1吗？通常添加原型对象方法时，还没有孩子呢！

15. 上午讲的子对象强行修改父对象中共有属性，导致自动在子对象添加自有属性，是不是多态？

答: 不是。多态，专指函数在不同情况表现出不同的状态。和属性无关。

16. 匿名函数自调前为什么用+，!，~，都能代替()呢？

比如: +function(){ ... }()

!function(){ ... }()

~function(){ ... }()

都能匿名函数自调？

答：+ ! 和~都是运算符，运算符后跟表达式时，js引擎会先试图执行运算符后的表达式，获得结果，然后在将结果和之前的运算符组合，进行下一步计算。所以，当匿名函数前跟+ ! 或跟~时，其实是在诱导js引擎去执行后边的函数表达式，就相当于调用了。

17. Array.prototype.sum=function(){ ... }中, Array是爷爷？还是爸爸？还是类型？

答：看图：Array是构造函数妈妈： function Array(){ ... }  
 Array.prototype是原型对象爸爸，是Array的老公。

类型：是指由妈妈+爸爸组成的一个“家庭”称为一个类型。反过来说，每一个类型都是由构造函数妈妈和原型对象爸爸组成。

18. arguments是不是多态？

答：js中实现重载，要借助于arguments。而重载属于多态的一种情况。所以重载这种现象是多态。而arguments本身只是实现多态所借助的一个对象。

19. var a=10;  
 function fun (){

console.log(a)

}

fun();

这里函数内可以使用外部的a是不是继承？

答：不是。一个函数执行时，可用的作用域对象，都保存在一个scopes数组（好友列表）中。查找变量时，是按顺序遍历scopes数组中的每个作用域对象，查找变量的。属于遍历。

而继承是通过\_ \_proto\_ \_属性，逐级查找父级对象中的属性。而不是变量。继承必须先有继承关系，才能通过\_ \_proto\_ \_查找父对象的属性。

20. 为什么Array.prototype.sum=function(){}, 放在上边的<script>就能用。而Student.prototype.toString=function(){} 放在上边的<script>就不能用？

答：执行顺序问题！

Array类型是内置类型，你不用创建function Array(){}，内存中已经有了。所以你可以直接使用Array.prototype.sum=function(){}。

而Student类型是你自己定义的，你不定义，内存中，就没有。所以，Student.prototype，只能在创建function Student(){}之后才能使用。

21. Cannot set property 'sname' of undefined ?

答: 不是sname写错了，而是sname .前的那个值，本应该是一个对象，但是因为各种原因，没拿到，是undefined。而undefined加任何后缀都会报错！

解决: 看sname前的变量是谁！往前找，哪里给这个变量赋的值。