new、Object、对象屏蔽原型链属性、操作符执行顺序

一、Object：

1、属性描述符【descriptor】：

1-1、分类：

访问器属性描述符：configurable、enumerable、set、get

数值描述符：configurable、enumerable、writable、value

1-2、获取：Object.getOwnPropertyDescriptor(obj,“key”)

1-3、设置及修改：Object.defineProperty(obj,”key”,{descriptor})、Object.defineProperties(obj,{“key”:{descriptor}})

1. 属性值：

2-1：属性值判断：in【原型链】、obj.hasOwnProperty(“key”)

2-2：属性键名获取：for…in【原型链】、Object.getOwnPropertyNames(“obj”)、Object.keys(obj)

1. 原型链：

3-1：原型判断：proto.isPrototypeOf(targetObj)

3-2：原型获取：Object.getPrototypeOf(“obj”)——function的原型为：func.prototype

3-3：原型修改：Object.setPrototypeOf(obj,proto)

二、屏蔽原型链上层值的三种情况——修改值：

1. 如果在[[Prototype]]链上层存在名为foo的普通数据访问属性并且没 有被标记为只读(writable:false)，那就会直接在 myObject 中添加一个名为 foo 的新 属性，它是屏蔽属性。 一般在对象上修改存在对象原型中，不存在对象的属性值，会使用该情况。。

2. 如果在[[Prototype]]链上层存在foo，但是它被标记为只读(writable:false)，那么 无法修改已有属性或者在 myObject 上创建屏蔽属性。如果运行在严格模式下，代码会 抛出一个错误。否则，这条赋值语句会被忽略。总之，不会发生屏蔽。 限于=赋值，Object.defineProperty()赋值不受影响。。

3. 如果在[[Prototype]]链上层存在foo并且它是一个setter，那就一定会 调用这个 setter。foo 不会被添加到(或者说屏蔽于)myObject，也不会重新定义 foo 这 个 setter。

1. this【函数执行上下文对象，函数执行时动态绑定】：

1、默认绑定：函数引用无任何修饰符，global或window。

2、隐式绑定：函数引用有上下文，如函数为obj的属性。

3、显示绑定：修改this指向，.apply(thisObj,argArr)、.call(thisObj,arg1,arg2)、.bind(thisObj,arg1,arg2)【柯里化】。

4、new操作符：将this指向新创建对象。

四、new操作符执行4个步骤：

1、创建新的object；2、对象关联构造函数prototype对象；3、this【函数执行上下文】绑定为新建的object；4、若构造函数没有返回对象，将新建object返回；

五、操作符执行顺序：<https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Operators/Operator_Precedence>

例子：

1. 访问器属性定义：

Object.defineProperty(Car.prototype,"color",{  
 get(){  
 **return this**.newColor  
 },  
 set(newV){  
 console.log(newV);  
 **this**.newColor=newV  
 }  
});