面向对象的理解2的内容

**1. 继承的实现通过原型和原型链**

1.1**为什么存**在原型

对象通过构造函数构造出来，这些对象有一些共同的数据，节省内存空间，方便维护管理，js就帮这些数据内容存在同一个位置，这样对象就可以很方便的去读取。

js把这些数据存在构造函数的prototype的属性下面，构造函数.prototype是一个对象，它可以存储公用的属性和方法。

1.2 **怎么找**原型

为了找这些数据，在new对象的时候，会将构造函数.prototype的属性和方法自动放到对象.\_\_proto\_\_下面。

虽然我们可以通过\_\_proto\_\_调用到构造函数的prototype了，但是js觉得有点复杂了，所以把这个调用进行了简化，当我们去调用一个对象属性或者方法的时候，会按照一种规则进行查找，首先会在对象自身上进行查找，如果找不到，会自动去这个对象\_\_proto\_\_上去查找。

**2. Object对象的方法**

2.1 Object对象的**静态**方法

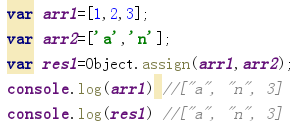
**Object.assign()**——**常量const**的理解

用法：Object.assign(target, ...sources)

作用：对象合并，将源对象合并到目标对象上面，目标对象改变

参数：目标对象，源对象

返回值：合并之后的目标对象



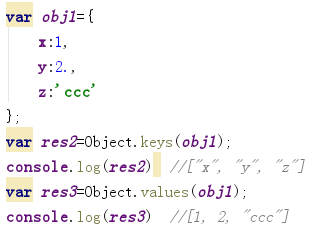
**Object.keys(obj)**

用法：Object.keys(obj)

作用：将对象的key值取出来组成数组

参数：obj

返回值：key组成的数组



**Object.values()**

用法：Object.values(obj)

作用：将对象的value值取出来组成数组

参数：obj

返回值：value组成的数组

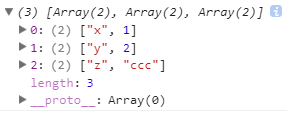
**Object.entries()**

用法：Object.entries(obj)

作用：将对象的key和value值取出来组成数组

参数：obj

返回值：数组



**Object.is()**

用法：Object.is(obj1,obj2)

作用：判断两个对象的值是不是相等

参数：两个对象

返回值：true 、false

**Object.freeze()**

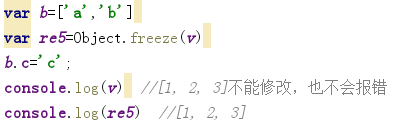
用法：Object.freeze(obj)

作用：将对象的第一层冻结，第一层就不能改变

参数：obj

返回值：改变之后的对象

存在问题：只是浅冻结：冻结了第一层对象，内部对象还是地址的引用，可以自己封装沈冻结



其他知识点：常量const

常量是不能修改的，修改之后会报错，当时常量是数组的话，改变数组，常量数组还是可改变的

clipboard.png

clipboard.png



为了解决这个问题：就出现了Object.freeze()

**Object.definePrototype()**

用法：Object.defineProperty(obj, prop, descriptor)

作用：设置对象的属性

作用：vue——追踪变化。数据双向绑定:数据改变页面自动刷新，用到了Object.definePrototype，将数据转成setter/getter 可监听。数据一变。自动刷新，不用操作DOM，标准浏览器可用，ie9以上可用

参数：obj, prop, descriptor{}：

obj : 要设置属性的对象

prop : 要设置的属性的名称

descriptor : 要设置的属性的描述，特征

描述符：属性的描述，也可称为属性的属性

存在一：

value

configure，可以删除，默认是false

writable，可以修改，默认是false

enumerable，可以枚举，默认是false

存在二：寄存器

get()

set()——存在的问题，坑

解释：当我们去访问一个对象属性的时候，就会自动触发这个属性的get函数，如果我们去设置一个属性，那么就会自动触发这个属性的set函数

存在的问题：

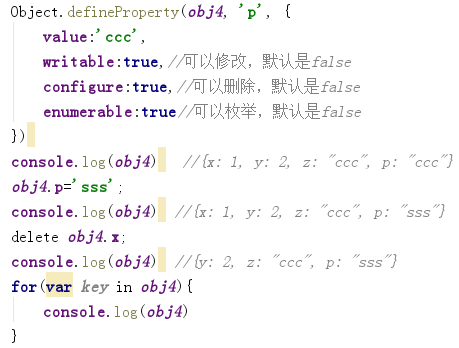
当我们是给数组里面的数组进行设置时，是不会触发set()的，因为，数组里面的数组是对象，是被引用的，即时改变了值，但是引用的地址还是没有改变。

存在一和存在二不能同时存在

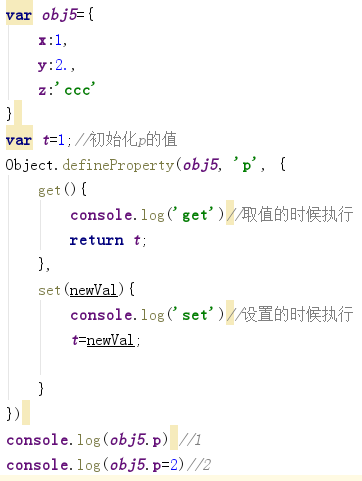
注意：

当一个属性通过 . 或者 [] 来设置的话，那么默认writable、configurable、enumerable值都是true，通过defineProperty来设置默认都是false

一：



二：



2.2 Object对象的动态方法

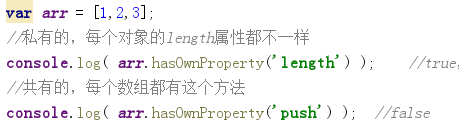
**obj.hasOwnProperty()**

用法：obj.hasOwnProperty(prop)

作用：判断prop这个属性或者方法是不是obj的自有属性或方法

参数：属性或方法

返回值：true、false



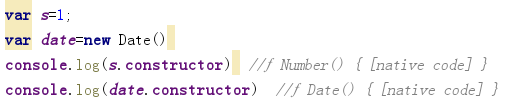
**obj.constructor**

用法：obj.constructor

作用：获取对象的构造函数

参数：无，属性

返回值：构造函数



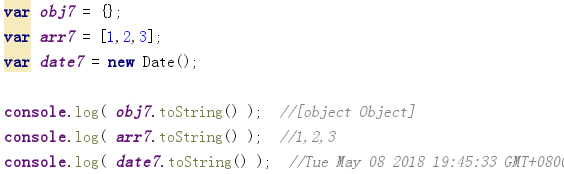
**obj.toString()**

用法：obj.toString()

作用：把obj转成字符串形式，很多的一些其他构造会重写toString方法，所以不同对象toString的内容不一定都是 [object Object]

参数：无

返回值：字符串



3. **indstanceof**运算符的理解

运算符，二元

作用：A是不是B的实例

返回值：true、false

4. 拖拽的例子

父类

子类4

方法一：**es5**

属性：父类的执行，call改变this

prototype：父类prototype的深拷贝

将原型的指向父级：DragLi....prototype=Object.create(Drag.prototype)

方法二：**JSON**

JSON.parse

JSON.stringify

var obj3 = JSON.parse(JSON.stringify(obj1));

方法三：**es6**

**class 子类 extend 父级(){}**

这样就可以引用父级的属性和函数，并且子类的构造函数可以指向父级