属性描述符：描述属性的属性，元属性

1.原型和原型链

对象的__ptoto__指向它构造函数的 prototype（原型对象）

原型对象：它不是一个空的 Object 对象，但是它的__proto__永远指向 Object 的 prototype

原型链：隐式原型链

原型链的头：

Object.prototype.__proto__===>null

Function.__proto__ === Function.prototype

Object.__proto__ === Function.prototype

Function.prototype.__proto__ === Object.prototype

Object.prototype.__proto__=== null

获取对应属性的描述符

Object.getOwnPropertyDescriptor(obj,"name");

第一个参数：对应的对象
第二个参数：对应对象的属性

```
var obj={ name:"snw"};
Object.defineProperty(obj,"age",{
    value:18,
    writable:true,
    configurable:true,
    emumerable:true
});
console.log(obj); //Object {name: "snw", age: 18}
console.log(obj.age);//18
```

属性描述符的默认值都是 false

writable(决定是否可以修改属性的值**)**是否可改

当 writable 为 false 时,对应的属性的值是无法修改的。
在默认情况下：

继续修改的话会**静默失败**

在严格模式下：
会报错

configurable(来决定属性是否可以配置)是否可删除可

当 configurable 为 false 时,对应的属性是无法重新定义的，无法删除的。但该对象是还可以继续添加属性的。
不管是默认情况还是严格模式底下
进行重新定义和删除都会报错
注意一个小小的例外：
当 configurable 为 false 时，writable 可以进行重新定义，

但只满足由 writable：true ==> writable：false

enumerable

控制属性的课枚举权限

可枚举：是否可以出现在对象的 for in 循环中

```
for(item in damu){
if(damu.hasOwnProperty(item)){
    console.log(item); }    }
```

可枚举：可以出现在对象属性的遍历中（for in 循环）
（不会遍历原型链）obj.propertyIsEnumerable("a")
（不会遍历原型链）Object.keys(obj)
（不会遍历原型链）Object.getOwnPropertyNames(obj)

对象不变性

对象常量属性

将属性的 writable 和 configurable 设置为 false
由于属性描述符是对属性的管理

禁止对象扩展（Object.preventExtensions(obj);）

所以想禁止对象扩展不能使用属性描述符来控制，而是需要调用

Object.preventExtensions(obj);

参数：要求禁止扩展的对象
默认情况下
    为对象添加新的属性会静默失败
严格模式底下
    报错
注意：禁止对象扩展只是禁止了对象去扩展属性，
而之前的属性是可以进行重新定义或者删除的，属性值也是可以修改的
在禁止对象扩展（Object.preventExtensions(obj);的基础上把现有属性的 configurable 都调整为 false

密封对象（Object.seal(obj)）

调用 **Object.seal(obj)**密封一个对象
密封之后禁止了对象去扩展属性，原有的属性不可以进行重新定义或者删除，但属性值是可以修改的
在密封对象（Object.seal(obj))的基础上把现有属性的 writable 都调整为 false
调用 Object.freeze(obj)密封一个对象
eg：

```
var xfz={
            age:48
        };

        Object.seal(xfz);
        delete xfz.age;
        xfz.wife="fbb";
        xfz.age=18;
        console.log(xfz)
```

冻结对象

冻结之后禁止了对象去扩展属性，原有的属性不可以进行重新定义或者删除，属性值不可以进行修改

冻结对象：

```
var xfz={
            age:48
        };

        Object.freeze(xfz);
        delete xfz.age;
        xfz.wife="fbb";
        xfz.age=18;
        console.log(xfz)
```

深度冻结

```
var xfz={
    wifes:{
        wife1:"fj",
        wife2:"tg",
        wife3:"bhj",
        wife4:"qhf"
            }
        };
```

属性描述符

用来修饰属性的属性(元属性) 5 个

数据描述符

具有 writable 和 value 属性描述符的属性 我们称之为数据描述符

访问描述符

具有 set 和 get 属性描述符的属性 我们称之为访问描述符

访问描述符可以使我们在对属性进行取赋值操作时预先变规定逻辑

访问描述符可以使我们的属性变的更加安全!!!! 封装

set（）/get（）

```
var obj={};
        Object.defineProperty(obj,"age",{
            set(val){
              if(val<0){
                val=0;
                }else if(val>120){
                val =120;
                }
                this.__age__ = val; },
            get(){
                return this.__age__;
            }
        })
        obj.age=100000000;
        console.log(obj.age);//120
        console.log(obj.__age__);//120
```

writable 为 true 的数据描述符

属性的查找 设置

查找：

1.在对象中查找是否具有相同名称的属性，如果找到，就会返回这个属性的值。

2.如果没有找到，则遍历原型链

3.无论·如何都没找到，返回 undefined

设置：不管什么情况，设置操作只会影响对象的直接属性

属性的设置

[[put]]：

1.如果属性直接存在于对象中 不在原型链上

找到直接存在于对象中的属性

-数据描述符(没有 setter/getter)

直接修改对象中的属性(注意 writbale 的值)

-访问描述符

直接调用 set 方法

2.如果属性不直接存在于对象中也不在原型链上

在对象的直接属性中添加一个属性（数据描述符）

value:"a"

writable:true

configurable:true

enumerable:true

3.如果属性不直接存在于对象中 在原型链上

①.该属性是数据描述符(没有 setter/getter)

-writbale 为 true

直接在对象中添加一个属性，我们称之为屏蔽属性

-writbale 为 false

报错，不会生成屏蔽属性

②.该属性是访问描述符

调用 set，不会生成屏蔽属性

4.如果属性直接存在于对象中 也在原型链上

找到直接存在于对象中的属性

-数据描述符(没有 setter/getter)

直接修改对象中的属性(注意 writbale 的值)

-访问描述符

直接调用 set 方法

obj.age=18;

1.如果属性直接存在于对象中 不在原型链上

找到直接存在于对象中的属性

-数据描述符(没有 setter/getter)

直接修改对象中的属性的值(注意 writbale 的值)

-访问描述符

直接调用 set 方法

4.如果属性直接存在于对象中 也在原型链上

找到直接存在于对象中的属性

-数据描述符(没有 setter/getter)

直接修改对象中的属性(注意 writbale 的值)

-访问描述符

直接调用 set 方法

```
Object.prototype.age=48;
    var obj ={
        age:"芳龄 18"
    }
    obj.age=18;
    console.log(obj);
    console.log(Object.prototype);
*/


/*
 2.如果属性不直接存在于对象中也不在原型链上
    在对象的直接属性中添加一个属性（数据描述符）

        value:"a"
        writable:true
        configurable:true
        enumerable:true

    var obj={};
    obj.age=18;
    console.log(obj);
    console.log(Object.prototype);

console.log(Object.getOwnPropertyDescriptor(obj,"age"));
    */

    /*3.如果属性不直接存在于对象中　在原型链上
        ①.该属性是数据描述符(没有 setter/getter)
            -writable 为 true
                直接在对象中添加一个属性，我们称之为屏蔽属性

            -writable 为 false
            报错，不会生成屏蔽属性

        ②.该属性是访问描述符
            调用 set，不会生成屏蔽属性
    */
```