非构造函数的继承

object()方法

```
var Chinese = {nation:'中国'};
var Doctor = {career:'医生'};

   function object(o) {
       function F() {}
       F.prototype = o;
       return new F();
    }

var Doctor = object(Chinese);
Doctor.career = '医生';
console.log(Doctor);
```

浅拷贝

```
var Chinese = {nation:'中国'};
var Doctor = {career:'医生'};

   function extendCopy(p) {
       var c = {};
       for (var i in p) {
           c[i] = p[i];
       }
       c.uber = p;
       return c;
   }

var Doctor = extendCopy(Chinese);
Doctor.career = '医生';
console.log(Doctor);
```

constructor

属性返回对创建此对象的数组函数的引用;

```
function employee(name,job,born)
{
this.name=name;
this.job=job;
```

```
this.born=born;
}

var bill=new employee("Bill Gates","Engineer",1985);

console.log(bill);
console.log(bill.constructor);

//

var test=new Array();
console.log(test.constructor);
```

深拷贝

```
var Chinese = {nation:'中国',birthPlaces: ['北京','上海','香港']};
var Doctor = {career:'医生'};



   function deepCopy(p, c) {
       var c = c || {};
       for (var i in p) {
           if (typeof p[i] === 'object') {
               c[i] = (p[i].constructor === Array) ? [] : {}; //三元运算符?:
               deepCopy(p[i], c[i]);
           } else {
                c[i] = p[i];
           }
       }
       return c;
   }



var Doctor = deepCopy(Chinese);
Doctor.career = '医生';

Doctor.birthPlaces=['北京','上海','香港','厦门'];
console.log(Doctor.birthPlaces); //北京, 上海, 香港, 厦门
console.log(Chinese.birthPlaces); //北京, 上海, 香港
console.log(Doctor);

////////////////////////////////////////

JSON.stringify(obj1) )
function deepCopy(oldObj){
```

```
        var newObj=oldObj;
        if (oldObj&&typeof oldObj==="object") {
            newObj=    Object.prototype.toString.call(oldObj)==="[object Array]
"? []:{};
            for(var i in oldObj){
                newObj[i]=deepCopy(oldObj[i]);
            }
        }
        return newObj;
    }

function deepCopy(oldObj){
        var newObj=JSON.stringify(oldObj);
        return JSON.parse(newObj);
    }
```