学习心得

函数递归

arguments.callee

函数赋值给变量只是给了地址，如果原函数改变，变量也会改变。使用arguments.callee可以避免。他是指向正在执行函数的指针。阶乘可以写成：

```
function fac(num){
    if(num<=1){
        return 1;
    }else{
        return num * arguments.callee(num-1)
    }
}
```

使用arguments.callee写递归，无论通过什么变量调用都可以

闭包

引用了另一个函数作用域中变量的函数，一个函数第一次调用时，首先会创建一个包含arguments和参数的`活动对象`，然后创建全局作用域的的变量，叫全局变量对象。函数执行完毕，局部活动对象被销毁，但是闭包不同，在一个函数内部定义函数，外部函数会把内部函数的活动对象添加到自己的作用域链中，可以访问内部函数的所有变量。所以当内部的函数执行完，它的活动对象也不会被销毁。

立即调用函数

类似于函数声明，因为被包含在括号中，所以被解释为函数表达式，ES5不支持。

私有变量

定义在函数块中的变量是私有的，外部无法访问。

想要访问私有变量可以定义在构造函数中

```
function MyObject(){
    let privateVariable = 10;
    function privateFunction(){
        return false
    }
    this.publicMethod = function(){
        privateVariable++;
        return privateFunction()
    }
}
```

变量和函数只能通过publicMethod访问。

可以使用私有变量和特权方法，隐藏不能修改的数据。

```
function Person(name){
    this.getName =function(){
        return name
    }
    this.setName = function(value){
        name = value
    }
}
let person = new Person("ytt")
console.log(person.getName())//"ytt"
Person.serName("happy")
console.log(person.getName())//"happy"
```

name对每一个Person实例都是独一无二的，因为每次调用都会重新创建变量和方法。为了避免每个实力都会重新创建方法，可以使用静态私有变量。

使用该模式，会将所有实例共用同一个静态变量，只要setName发生改变，都会影响其他实例。

```
(function() {
    let name = '';
    Person = function(value) { name = value; };
    Person.prototype.getName = function() {
    return name;
};
Person.prototype.setName = function(value) {
    name = value;
}; })();

let person1 = new Person('Nicholas');
console.log(person1.getName());
person1.setName('Matt');
console.log(person1.getName());
// 'Nicholas'
// 'Matt'

let person2 = new Person('ytt');
console.log(person1.getName()); console.log(person2.getName());
// 'ytt' // 'ytt'
```

静态私有变量

```
(function(){
    let privateVariable = 10;
    function privateFunction(){
        return false;
    }
    MyObject.prototype.publicMethod = function(){
        privateVariable++;
        return privateFunction()
    }
}
)
```

因为MyObject定义没有用new，不使用关键字的声明的变量会创建在全局作用域中，所以MyObject变成了全局变量，可以外部访问。

模块模式

在单例对象（只有一个实例对象）上实现相同的隔离和封装。

```
let sing = function(){
    let privateVariable = 10;
    function privateFunction(){
        return false
    }
    return {
        publlicProperty:true,
        publicMethod(){
            privateVariable++;
            return privateFunction();
        }
    }
}
```

使用匿名函数范湖对象，随后创建一个匿名函数返回的字面量，包含可以公开访问的属性和方法，匿名函数可以访问同一作用域的私有变量和函数。

明日计划

1. 看第十一章：期约与异步函数
2. 把今天没写完的leetcode406写完。今天的主要卡在对二维数组的拆分冰局部排序的实现上。