学习笔记

异步编程

只支持定义回调函数表明异步完成。串联多个回调函数需要深度嵌套回调函数（回调地狱）

```
function double(value){
    setTimeout(()=>setTimeout(console.log,0,value*2),1000);
}
double(3)//6
```

setTimeout定义了一个指定时间之后会被执行的调度函数。

如果要将setTimeout返回值传给其他地方，就给一步提供一个回调。

```
function double(value,callback){
    setTimeout(()=>callback(value*2),1000)
}
double(3,(x)=>console.log(`I get ${x}`))
```

在函数闭包中回调

期约

期约状态机

把期约实例传给console.log时，控制台输出表明实例处于待定（pending）状态。期约一共有三个状态

1. 待定（pending)
2. 兑现（fulfilled有时也叫解决“resolved”）
3. 拒绝（rejected)

待定状态可以转为兑现和拒绝，落定后不可逆，作用是可以抽象表示一个异步错做的状态。期约状态是私有的，只能内部操作，在七月执行函数完成。执行函数主要初始化期约状态和控制状态最终转换。状态转换函数为reject()和resolve()

```
let p1 = new Primise((resolve,reject)=> resolve());
setTimeout(console.log,0,p1)//Promise <resolved>
let p2 = new Promise((resolve,reject)=>reject());
setTimeout(console.log,0,p2);//Promise <rejectd>
```

在初始化期约时就已经改变了每个期约的状态，执行器函数是同步的，因为执行器是期约的初始化程序。

```
new Promise(()=> setTimeout(console.log,0,'executor'))
setTimeout(console.log,0,'promisee init')
//executor
//promisee init

let p = new Promise((resolve, reject)=>setTimeout(resolve,1000))
setTimeout(console.log,0,p)  //Promise <pending>
```

状态改变后再继续修改会静默失败。

为防止期约卡在待定状态，可以设置退出功能，比如setTimeout十秒后无论如何都会拒绝。

```
let p = new Promise((resolve,reject) =>{
    setTimeout(reject,10000);   //10s后执行reject
});
setTimeout(console.log,0,p)     //Promise <pending>
setTimeout(console.log,11000,p) //11s后再检查状态
```

期约一开始并非一定是待定状态，也可以是实例化一个解决期约Promise.resolve()。

期约的实例方法

实现外部同步代码和内部异步代码之间的桥梁。

1. Thenable接口

   then()方法实现了Thenable接口

2. Promise.prototype.then()

   then()接受最多的是onResolved和onRejected处理程序

   ```
   function onResolved(id){
       setTimeout(console.log,0,id,'resolved')
   }
   function onRejected(id){
       setTimeout(console.log,0,id,'rejected')
   }
   ```

leetcode406实现功能：

根据二维数组的一维数组的第二个元素升序排序

```
    for(let i = 0; i < people.length - 1; i++){
        if(people[i][0] > people[i + 1][0]){
            for(j = start; j < i + 2; j++){
                for(k = start; k < i + 1 - j; k++){
                    if(people[k][1] > people[k + 1][1]){
                        let temp
                        temp=people[k + 1][1]
                        people[k + 1][1]=people[k][1]
                        people[k][1]=temp
                    }
                }
            }
            start = i+1;
        }
    }
```

在实现了数组元素向指定位置移动：

```
for(let i = 1; i < people.length; i++){
        if(people[i][0] != people[i - 1][0]){
            if(i > people[i][1]){
                people.splice(people[i][1],0,people[i]);
                people.splice(i + 1,1)
            }
            else{
                people.splice(people[i][1] + 1,0,people[i]);
                people.splice(i,1)
            }
        }
}
```

明日计划

1. 看js项目视频
2. 把leetocde406调试出来