# 异步编程

## Promise

回调地狱 :callback hella



异步读取文件不是按代码顺序执行，文件越小越先执行

但是想要文件按顺序读取无论多大，就可以使用回调地狱，嵌套

fs.readFile('./a.txt','utf8',function(err,data){

    if(err){

        throw err

    }

    console.log(data)

    fs.readFile('./b.txt','utf8',function(err,data){

        if(err){

            throw err

        }

        console.log(data)

        fs.readFile('./c.txt','utf8',function(err,data){

            if(err){

                throw err

            }

            console.log(data)

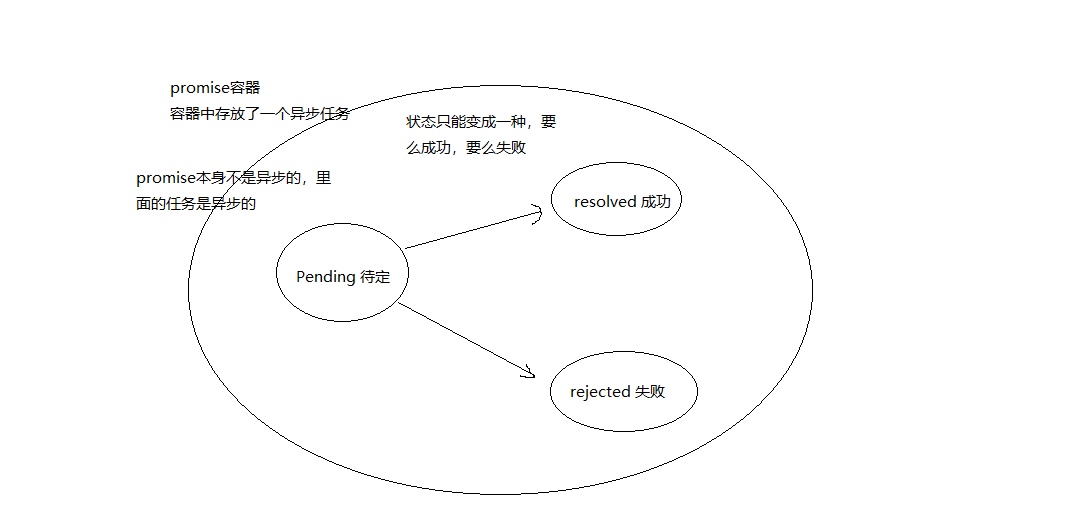
        })

    })

})

但是代码重复太多，而且看着很丑，所有为了解决回调地狱嵌套的问题，EcmaScript6中就新增了一个API：promise 是一个构造函数

Promise容器 存放了一个异步任务



Promise基本语法

//创建promise容器

//1.给别人一个承诺promise

//promise本身不是异步，但是里面的任务是异步的

// console.log('1')

//resolve:成功 reject；失败

var p1 = new Promise(function(resolve,reject){

    // console.log('2')

    //异步任务 读取文件

    fs.readFile('./data/a.txt','utf8',function(err,data){

        if(err){

            //把容器的pending状态改为reject

            //这里面调用的方法是then方法传递的funtion

            reject(err)

        }

        // console.log('3')

        // console.log(data);

        //把容器的pending状态改为reject

        //这里面调用的方法是then方法传递的funtion

        resolve(data)

    })

})

// console.log('4')

var p2 = new Promise(function(resolve,reject){

    fs.readFile('./data/b.txt','utf8',function(err,data){

        if(err){

            reject(err)

        }

        resolve(data)

    })

})

var p3 = new Promise(function(resolve,reject){

    fs.readFile('./data/c.txt','utf8',function(err,data){

        if(err){

            reject(err)

        }

        resolve(data)

    })

})

//当p1成功，然后（then）做指定的事情

// resolve(data) = function(data){}

// reject(err) = function(err){}

p1

    .then(function(data){//resolve

        console.log(data)

        //当return的是123，接收的是123

        //  return的是hello，接收的是hell

        //  没有return，接收的是undefind

        //当return的是一个promise对象时，后面的then方法中的第一个参数会作为p2的返回值

        return p2

    },function(err){//reject

        console.log(err);

    })

    .then(function(data){

        console.log(data)

        return p3

    })

    .then(function(data){

        console.log(data)

    })

Promise封装

//封装promise

function preadFile(path){

    return new Promise(function(resolve,reject){

        fs.readFile(path,'utf8',function(err,data){

                if(err){

                    reject(err)

                }

                resolve(data)

        })

    })

}

preadFile('./data/a.txt')

    .then(function(data){

        console.log(data)

        return preadFile('./data/b.txt')

    })

    .then(function(data){

        console.log(data)

        return preadFile('./data/c.txt')

    })

    .then(function(data){

        console.log(data)

    })