# Promise

1 使用Promise

一个 [Promise](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global_Objects/Promise) 就是一个对象，它代表了一个异步操作的最终完成或者失败

本质上，Promise 是一个绑定了回调的对象，而不是将回调传进函数内部。

Promise说得通俗一点就是一种写代码的方式，并且是用来写JavaScript编程中的异步代码的

代码:

let p = new Promise((resolve, reject) => { // 做一些事情 // 然后在某些条件下resolve，或者reject

if (/\* 条件随便写^\_^ \*/) { resolve() }

else { reject() }

})

p.then(() => {

// 如果p的状态被resolve了，就进入这里

}, () => {

// 如果p的状态被reject

})

1. 构造实例

构造函数接收一个函数作为参数

调用构造函数得到实例p的同时,作为参数的函数会立即执行

参数函数 { 接收两个 resolve & reject}

在参数函数调用的时候,如果在其内部调用resolve,会将p的状态fulfilled,或者调用reject,会将p的状态变成reject

1. 调用then()方法

 调用.then可以为实例p注册两种状态回调函数

 当实例p的状态为fulfilled，会触发第一个函数执行

 当实例p的状态为rejected，则触发第二个函数执行 拒绝

总结: 构造promise实例,然后调用 then.的编写代码方式,就是promise

基本模式:

1 将一部队向转换成promise对象

2 对象有三种状态

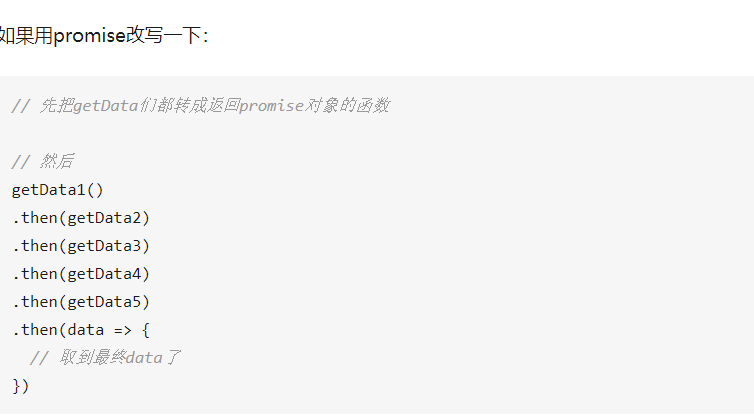
3通过.then注册状态的回调

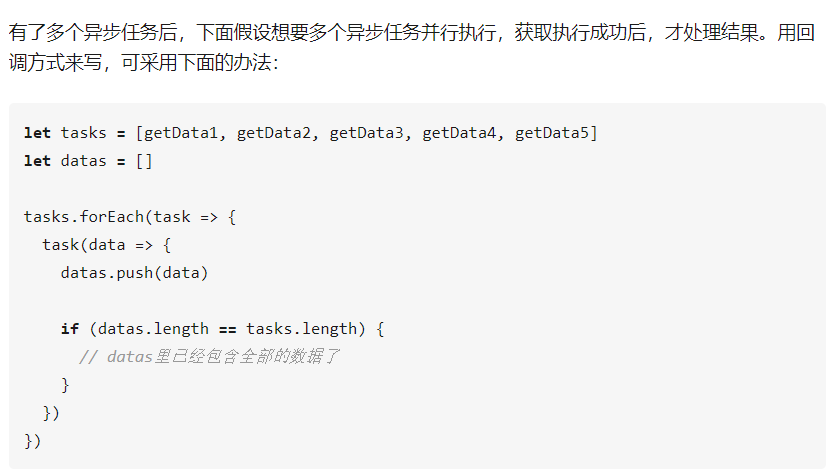
4已完成的状态能触发回调

回调地狱



使用promise改下:







promise实例有三种状态:

pending (待定)

fulfilled (已执行)

rejected (拒绝)

**resolve和reject**

**Promise是ES6新增加的类: 目的是为了管理异步操作的**

调用resolve和reject能将分别将promise实例的状态变成fulfilled和rejected，只有状态变成已完成（即fulfilled和rejected之一），才能触发状态的回调

1. new Promise() 创建类的一个实例;每一个实例都可以管理一个异步操作

->必须传递一个回调函数进去,(回调函数中管理你的异步操作);不传递会报错

->回调 函数中会有两个参数

resolve:异步操作成功做的事情 (代指成功后的事件队列 =>成功后要做的所有事情都存放到成功这个事件队列中)

reject: 异步操作失败做的事情 (代表失败后的事件队列)

new Promise的时候立即把回调函数执行了 promise是同步的

1. 基于Promise.prototype.the方法 [catch/finally 两个方法],向成功队列和失败队列加入需要处理的事情

let promisel = new Promise( () => {

$ajax( {

url: ‘json/data.json’,

success(result){

resolve(result);

},

error() {

reject(msg);

}) ;

2 promisel.then (

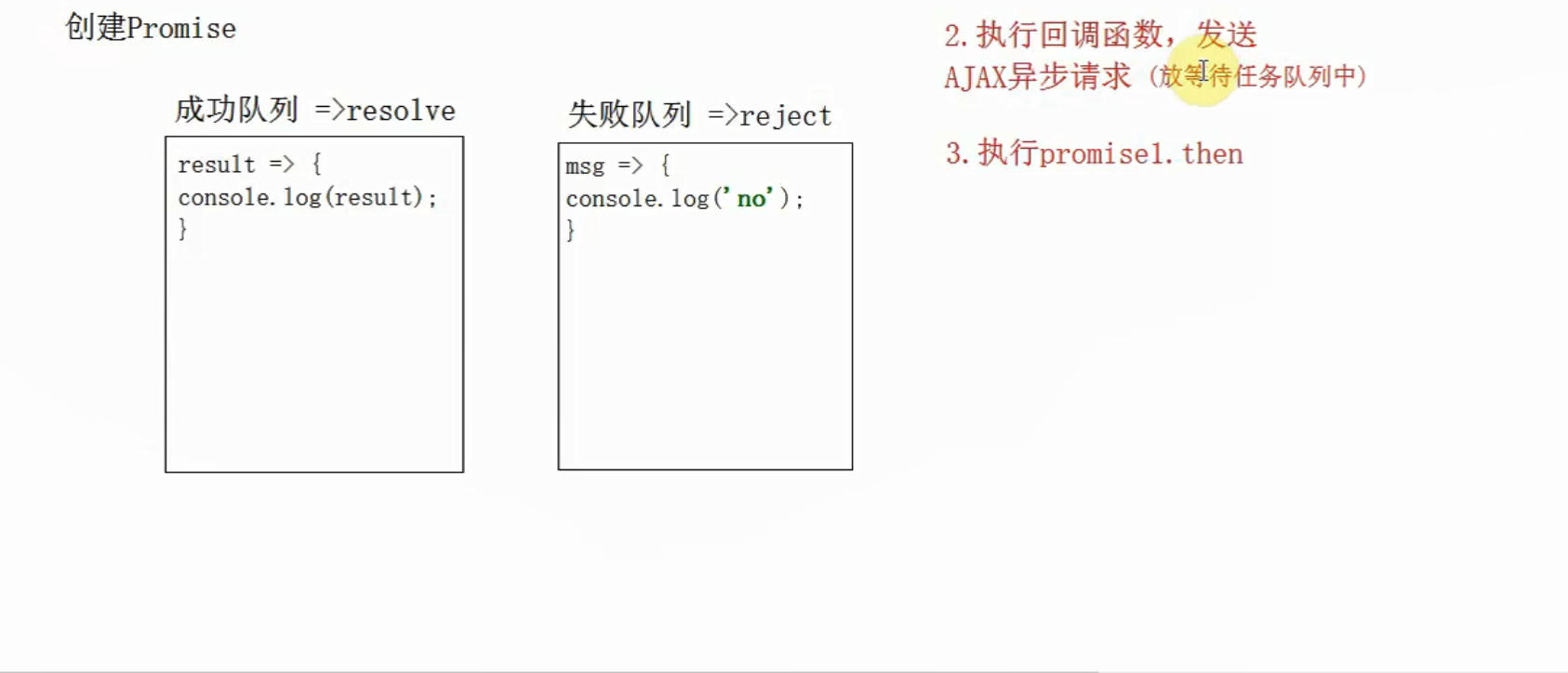
result =>{

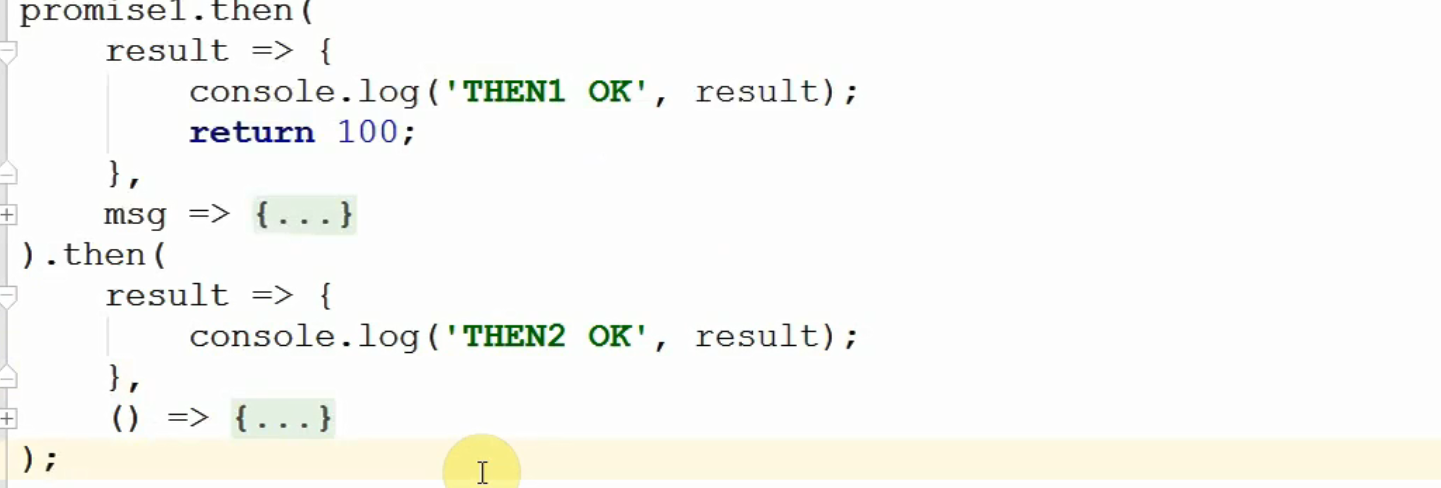
console.log( result)

},

msg => { }

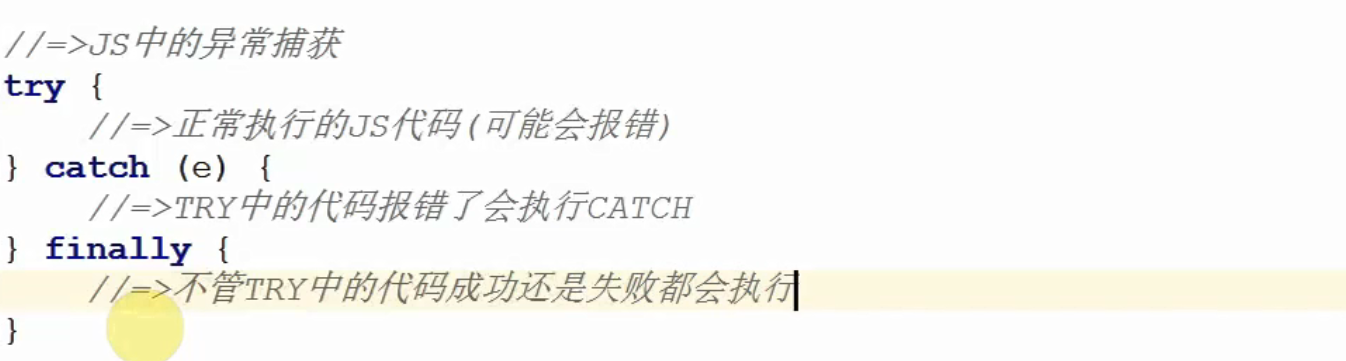
)





第一个catch

1. 异步请求失败会执行它
2. 第一个then方法失败也会执行它



真实项目中:

Promise 管控一个实例

let queryA = queryA() => {

return new Promise ( resolve => {

$.ajax ( {

url: ‘’,

success: resolve

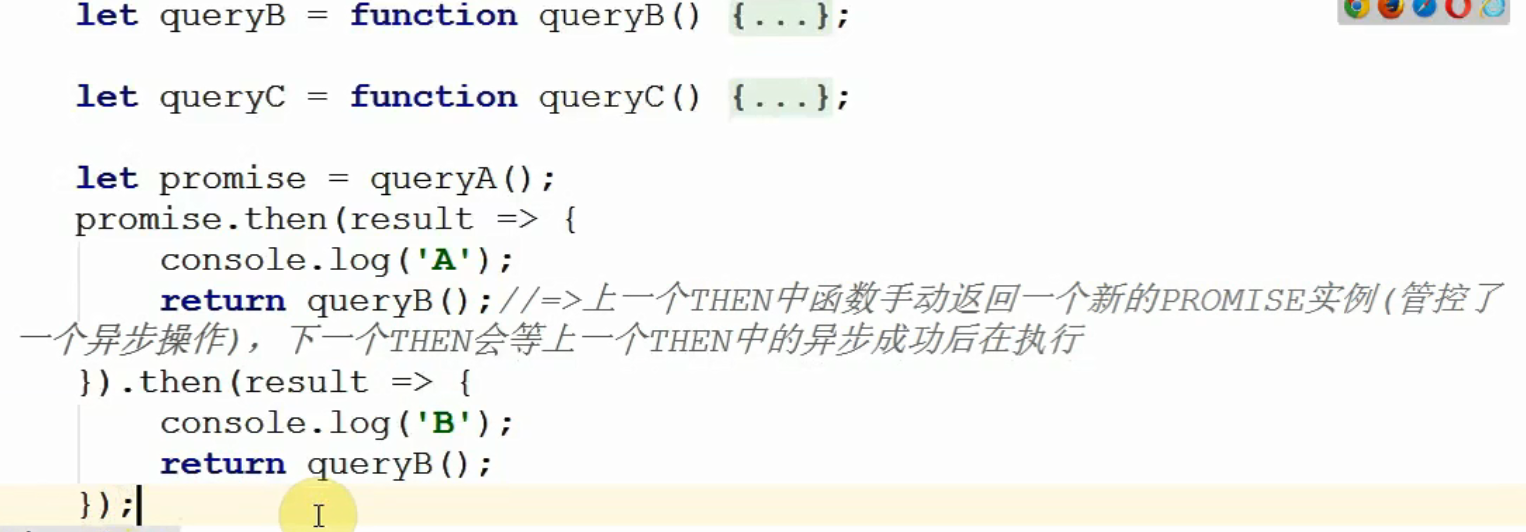
});

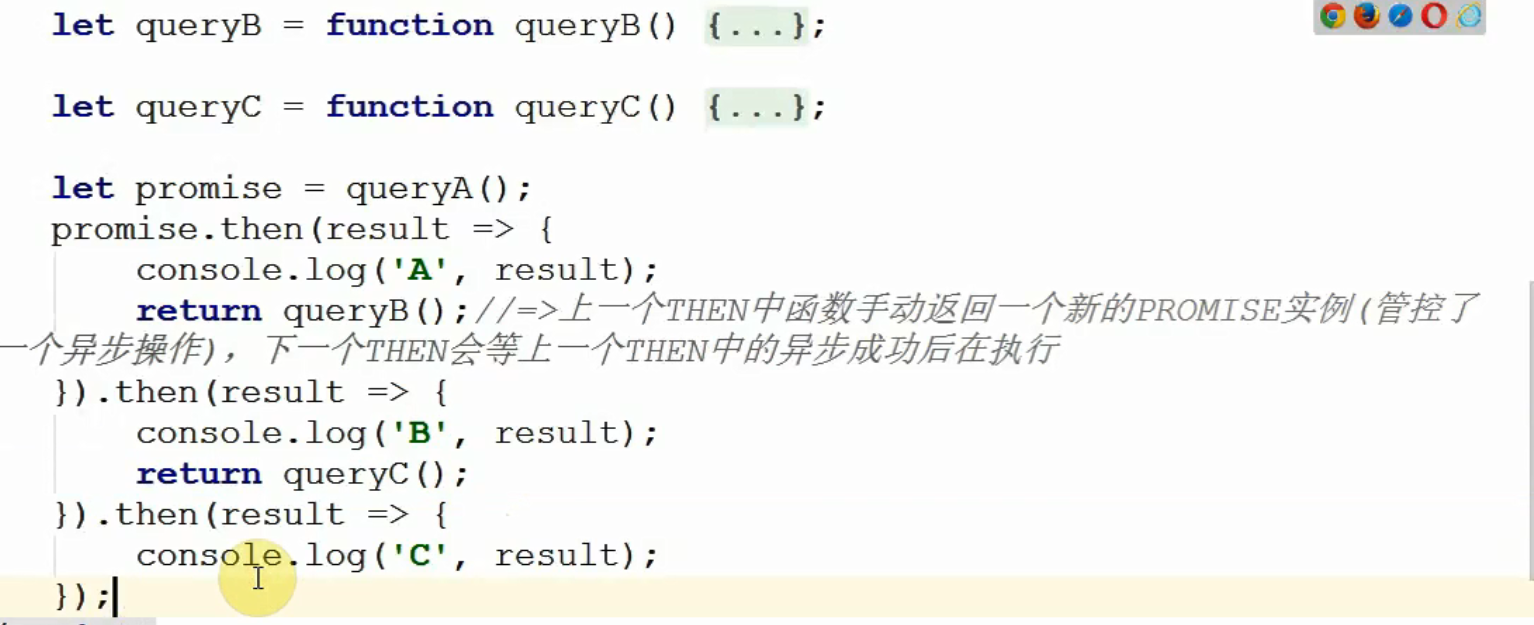
});

}

把每一个ajax请求封装成一个一个方法,每一个方法都返回promise实例





]

**Promise API**

promise的内容分为构造函数、实例方法和静态方法

* 1个构造函数： new Promise
* 2个实例方法：.then 和 .catch
* 4个静态方法：Promise.all、Promise.race、Promise.resolve和Promise.reject

解释说明:

* new Promise能将一个异步过程转化成promise对象。先有了promise对象，然后才有promise编程方式。
* .then用于为promise对象的状态注册回调函数。它会返回一个promise对象，所以可以进行链式调用，也就是.then后面可以继续.then。在注册的状态回调函数中，可以通过return语句改变.then返回的promise对象的状态，以及向后面.then注册的状态回调传递数据；也可以不使用return语句，那样默认就是将返回的promise对象resolve。
* .catch用于注册rejected状态的回调函数，同时该回调也是程序出错的回调，即如果前面的程序运行过程中出错，也会进入执行该回调函数。同.then一样，也会返回新的promise对象。
* 调用Promise.resolve会返回一个状态为fulfilled状态的promise对象，参数会作为数据传递给后面的状态回调函数
* Promise.reject与Promise.resolve同理，区别在于返回的promise对象状态为rejected

面试题: