目录

1：回调函数的嵌套 1

2：ajax的依赖调用 1

3：Promise介绍 2

4：promise改造ajax依赖调用 11

应用： 11

5：ajax的同源策略 12

6：jsonp跨域 12

7：服务器端跨域 13

8：get和post提交 15

9：兼容问题 16

应用： 16

综合应用： 17

## 1：回调函数的嵌套

## 2：ajax的依赖调用

在使用ajax异步调用的时候，可能碰到同时调用多个ajax的情况，而且多个ajax之间还存在依赖关系。

举个例子：验证用户名和密码之后才能获取用户的信息。

## 3：Promise介绍

新推出的Promise函数,是一个构造函数,它实际上是对回调函数的一种封装 对异步编程的一种改进。

**Promise对象有以下两个特点：**

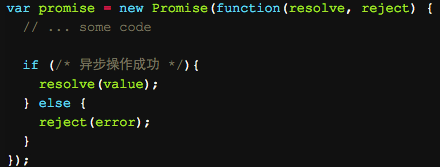
1：Promise对象代表一个异步操作，有三种状态：Pending（进行中）、Resolved（已完成，又称Fulfilled）和Rejected（已失败）。只有异步操作的结果，可以决定当前是哪一种状态，任何其他操作都无法改变这个状态。这也是Promise这个名字的由来，它的英语意思就是“承诺”，表示其他手段无法改变。

2：一旦状态改变，就不会再变，任何时候都可以得到这个结果。Promise对象的状态改变，只有两种可能：从Pending变为Resolved和从Pending变为Rejected。只要这两种情况发生，状态就凝固了，不会再变了，会一直保持这个结果。就算改变已经发生了，你再对Promise对象添加回调函数，也会立即得到这个结果。这与事件（Event）完全不同，事件的特点是，如果你错过了它，再去监听，是得不到结果的。

有了Promise对象，就可以将异步操作以同步操作的流程表达出来，避免了层层嵌套的回调函数。此外，Promise对象提供统一的接口，使得控制异步操作更加容易。

Promise也有一些缺点。首先，无法取消Promise，一旦新建它就会立即执行，无法中途取消。其次，如果不设置回调函数，Promise内部抛出的错误，不会反应到外部。第三，当处于Pending状态时，无法得知目前进展到哪一个阶段（刚刚开始还是即将完成）。

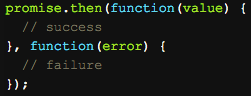
**基本用法：**



Promise构造函数接受一个函数作为参数，该函数的两个参数分别是resolve和reject。它们是两个函数，由JavaScript引擎提供，不用自己部署。

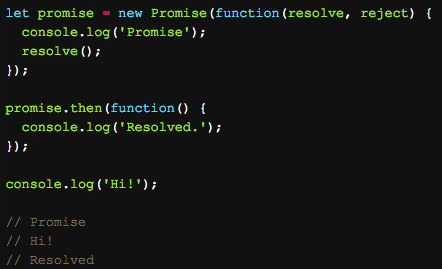
~~resolve函数的作用是，将Promise对象的状态从“未完成”变为“成功”（即从Pending变为Resolved），在异步操作成功时调用，并将异步操作的结果，作为参数传递出去；reject函数的作用是，将Promise对象的状态从“未完成”变为“失败”（即从Pending变为Rejected），在异步操作失败时调用，并将异步操作报出的错误，作为参数传递出去。~~

Promise实例生成以后，可以用then方法分别指定Resolved状态和Reject状态的回调函数。



then方法可以接受两个回调函数作为参数。第一个回调函数是Promise对象的状态变为Resolved时调用，第二个回调函数是Promise对象的状态变为Reject时调用。其中，第二个函数是可选的，不一定要提供。这两个函数都接受Promise对象传出的值作为参数。

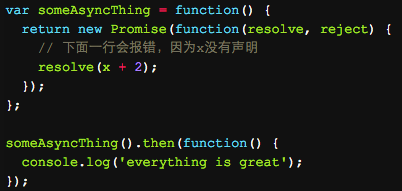
Promise新建后就会立即执行。



上面代码中，Promise新建后立即执行，所以首先输出的是“Promise”。然后，then方法指定的回调函数，将在当前脚本所有同步任务执行完才会执行，所以“Resolved”最后输出。

**catch方法：**

catch方法是.then(null, rejection)的别名，用于指定发生错误时的回调函数。Promise对象的错误具有“冒泡”性质，会一直向后传递，直到被捕获为止。也就是说，错误总是会被下一个catch语句捕获。一般来说，不要在then方法里面定义Reject状态的回调函数（即then的第二个参数），总是使用catch方法。如果没有使用catch方法指定错误处理的回调函数，Promise对象抛出的错误不会传递到外层代码，即不会有任何反应。



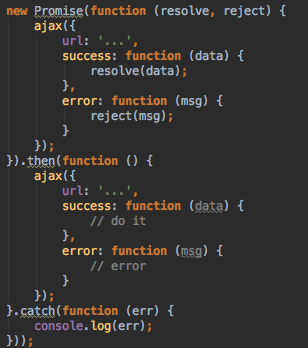
上面代码中，someAsyncThing函数产生的Promise对象会报错，但是由于没有指定catch方法，这个错误不会被捕获，也不会传递到外层代码，导致运行后没有任何输出。

下面我们看看它能解决哪些以前难以解决的问题:

**问题: 当有两个或以上的ajax请求需要按顺序执行,该如何编写?**

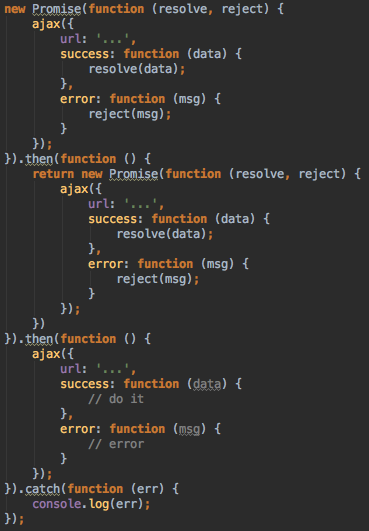


而使用promise的写法如下:



也就是代码被改进成了new Promise(ajax1).then(ajax2).catch() 使用了链式调用代替了嵌套结构。resolve() 和 reject() 则意味着成功回调和失败回调的执行。

如果有三个或更多的ajax请求呢?



代码结构大概是这样的: new Promise(ajax1).then(return new Promise(ajax2)).then(ajax3); 如果有第四个请求要顺序执行,则:

new Promise(ajax1).then(return new Promise(ajax2)).then(return new Promise(ajax3)).then(ajax4);

从这里仔细观察,可以看出,只有被Promise封装过了,回调才能保证顺序。 也就是Promise(意为承诺)设计的初衷。

但前一个方法必须在它最后执行resolve(),后一个方法才可以开始。 如果执行了reject(),则进入catch()方法。

其实这里不能单纯的理解为 resolve就是success,reject就是error 现就职于阿里的大名鼎鼎的阮一峰老师,喜欢管它叫状态机,这是非常恰当的叫法。

**Promise.all()：**

Promise.all方法用于将多个Promise实例，包装成一个新的Promise实例。

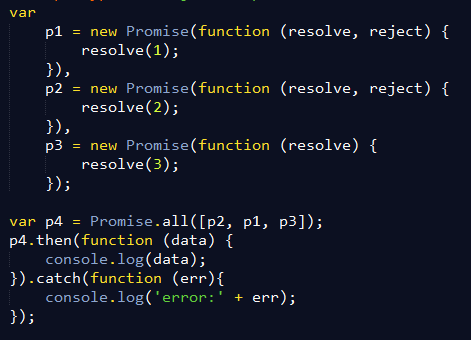


上面代码中，Promise.all方法接受一个数组作为参数，p1、p2、p3都是Promise对象的实例。

p的状态由p1、p2、p3决定，分成两种情况。

（1）只有p1、p2、p3的状态都变成fulfilled，p的状态才会变成fulfilled，此时p1、p2、p3的返回值组成一个数组，传递给p的回调函数。

（2）只要p1、p2、p3之中有一个被rejected，p的状态就变成rejected，此时第一个被reject的实例的返回值，会传递给p的回调函数。

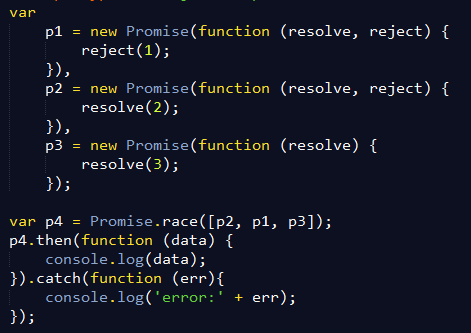


**Promise.race()：**

Promise.race方法同样是将多个Promise实例，包装成一个新的Promise实例。



上面代码中，只要p1、p2、p3之中有一个实例率先改变状态，p的状态就跟着改变。那个率先改变的 Promise 实例的返回值，就传递给p的回调函数。



## 4：promise改造ajax依赖调用



## 应用：

1：掌握ajax依赖调用。

2：掌握回调函数的应用。

3：掌握promise规范。

## 5：ajax的同源策略

同源策略(same origin policy)是一种约定，是由netscape公司引入浏览器的，他是浏览器最核心最基本的安全功能。现在所有支持JS的浏览器都会使用这个策略。

所谓同源：是指协议、域名、端口都相同。

目的：保证用户信息的安全，防止恶意的网站窃取数据。

限制范围：

1：cookie，localStorage和IndexDB无法读取。

2：DOM无法获得。

3：AJAX请求不能发送。

## 6：jsonp跨域

JSONP = JSON WITH PADDING。一种数据调用的方式。（协议）

AJAX无论请求什么页面，因为同源策略的限制，只要是跨域就一律不准。不过我们发现凡是拥有src这个属性的标签都拥有跨域的能力，如img，iframe，script。那我们就可以把数据装载到JS文件内，供客户端调用和进一步处理。

步骤：

1：定义一个回调函数。

2：创建script标签，指定src地址，并添加到页面中。

注：src必须跟着一个参数callback来指定回调函数名称。

## 7：服务器端跨域

CORS是一个W3C标准，全称是"跨域资源共享"（Cross-origin resource sharing）。它允许浏览器向跨源服务器，发出XMLHttpRequest请求，从而克服了AJAX只能同源使用的限制。

CORS需要浏览器和服务器同时支持。目前，所有浏览器都支持该功能，IE浏览器不能低于IE10。整个CORS通信过程，都是浏览器自动完成，不需要用户参与。对于开发者来说，CORS通信与同源的AJAX通信没有差别，代码完全一样。浏览器一旦发现AJAX请求跨源，就会自动添加一些附加的头信息，有时还会多出一次附加的请求，但用户不会有感觉。因此，实现CORS通信的关键是服务器。只要服务器实现了CORS接口，就可以跨源通信。

浏览器发起CORS请求，都会在请求头信息中增加一个Origin字段。该字段用来说明本次请求来自哪个源（协议 + 域名 + 端口）。服务器根据这个值，来决定是否同意这次请求。

**实现方式：**

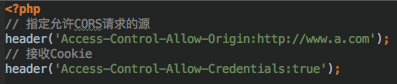
在服务器端指定Access-Control-Allow-Origin头信息，该字段是必须的。它的值要么是请求时Origin字段的值，要么是一个\*，表示接受任意域名的请求。如果Origin指定的值不在许可范围内，服务器返回一个正常的HTTP响应。浏览器如果发现响应头信息中没有包含Access-Control-Allow-Origin字段就知道出错了，从而抛出一个错误，被XMLHttpRequest的onerror回调函数捕获。





**向服务器发送Cookie：**

在服务器端指定Access-Control-Allow-Credentials头信息，该字段可选。它的值是一个布尔值，表示是否允许发送Cookie。默认情况下，Cookie不包括在CORS请求之中。设为true，即表示服务器明确许可，Cookie可以包含在请求中，一起发给服务器。这个值也只能设为true，如果服务器不要浏览器发送Cookie，删除该字段即可。



另一方面：开发者必须在AJAX中打开withCredentials字段。



这里需要注意的是，如果要发送cookie，Access-Control-Allow-Origin的值不能是\*，必须指定明确的，与请求网页一致的域名。同时cookie依然遵循同源策略，请求源与被请求源必须拥有同一个主域名，否则cookie还是不会被发送。

case 1：www.a.com访问bbs.a.com，并且cookie的domain设置为.a.com，那么cookie才会随着CORS发送给后台。

case 2：www.a.com访问www.b.com，这种情况下，cookie是不会随着CORS发送给后台。



## 8：get和post提交

1：功能

get多用于从服务器上获取数据，post多用于向服务器发送数据。

2：数据提交

GET请求的数据会附在URL之后（就是把数据放置在HTTP协议头中），以?分割URL和传输数据，参数之间以&相连。POST把提交的数据则放置在是HTTP包的包体中。

3：数据大小

GET请求传送的数据量比较小，达能大于2KB，这个受浏览器的限制。而POST请求传送的数据量比较大，一般默认为不受限制。其实大小受服务器的限制。

4：安全性

GET安全性非常低，POST的安全性较高。

## 9：兼容问题

**与JSONP比较：**

CORS与JSONP使用目的相同，但是比JSONP更强大。JSONP只支持GET请求，CORS支持所有类型的HTTP请求。JSONP的优势在于支持老式浏览器，以及可以向不支持CORS的网站请求数据。

## 应用：

1：天气预报

阿里云数据 天气预报接口 需要注册一个帐号。

2：请求淘宝搜索下拉提示。



## 综合应用：

