看这一篇就够了！浅谈ES6的Promise对象

次人君在野原之森网络工作室 关注
0.496 2017.01.14 22:56:27 字数 1,361 阅读 75,281

相信凡是写过javascript的童鞋也一定都写过回调方法（callback），简单说回调方法就是将一个方法func2作为参数传入另一个方法func1中，当func1执行到某一步或者满足某种条件的时候才执行传入的参数func2，例如下面的代码段

```
// 当参数a大于10且参数func2是一个方法时 执行func2
function func1(a, func2) {
    if (a > 10 && typeof func2 == 'function') {
        func2()
    }
}

func1(11, function() {
    console.log('this is a callback')
})
```

一般来说我们会碰到的回调嵌套都不会很多，一般就一到两级，但是某些情况下，回调嵌套很多时，代码就会非常繁琐，会给我们的编程带来很多的麻烦，这种情况俗称——回调地狱。极端情况如下图：

badcallback.jpeg

由此，Promise的概念就由社区提出并实现，作用与回调方法几乎一致，都是在某种情况下执行预先设定好的方法，但是使用它却能够让代码变得更简洁清晰

什么是Promise

Promise是异步编程的一种解决方案，它有三种状态，分别是**pending-进行中**、**resolved-已完成**、**rejected-已失败**

当Promise的状态又pending转变为resolved或rejected时，会执行相应的方法，并且状态一旦改变，就无法再次改变状态，这也是它名字promise-承诺的由来

ES6之前的Promise

在ES6中，Promise终于成为了原生对象，可以直接使用。但是在这之前，小伙伴们想要使用Promise，一般会借助于第三方库，或者当你知道其中的原理以后，也可以手动实现一个简易的Promise

当然，为了防止不可预知的bug，在生产项目中最好还是不要使用原生的或者自己编写的Promise（目前为止并不是所有浏览器都能很好的兼容ES6），而是使用已经较为成熟的有大量小伙伴使用的第三方Promise库，下面就为小伙伴推荐一个—— Bluebird

Promise的基本用法

声明一个Promise对象

```
// 方法1
let promise = new Promise ( (resolve, reject) => {
    if ( success ) {
        resolve(a) // pending --> resolved 参数将传递给对应的回调方法
    } else {
        reject(err) // pending --> rejectd
    }
} )

// 方法2
function promise () {
    return new Promise ( function (resolve, reject) {
        if ( success ) {
            resolve(a)
        } else {
            reject(err)
        }
    } )
}
```

注意：实例化的Promise对象会立即执行

Promise.prototype.then() VS Promise.prototype.catch()

.then()方法使Promise原型链上的方法，它包含两个参数方法，分别是已成功resolved的回调和已失败rejected的回调

```
promise.then(
    () => { console.log('this is success callback') },
    () => { console.log('this is fail callback') }
)
```

次人君在野原之... 关注
总资产6 (约0.52元)

前端如何导出二进制流文件
阅读 4,479
在陪伴父母这件事情上，我们都该向马天宇学习！
阅读 246

推荐阅读
代码生成器
阅读 4,185
js:Promise的简单理解
阅读 10,453
VUE判断当前设备是PC还是移动端
阅读 6,141
前端面试之简历篇
阅读 11,014
安装包--save与 --save-dev的区别
阅读 203

.catch()的作用是捕获Promise的错误，与then()的rejected回调作用几乎一致。但是由于Promise的抛错具有冒泡性质，能够不断传递，这样就能够在下一个catch()中统一处理这些错误。同时catch()也能够捕获then()中抛出的错误，所以建议不要使用then()的rejected回调，而是统一使用catch()来处理错误

```
promise.then(
    () => { console.log('this is success callback') }
).catch(
    (err) => { console.log(err) }
)
```

同样，catch()中也可以抛出错误，由于抛出的错误会在下一个catch中被捕获处理，因此可以再添加catch()

使用rejects()方法改变状态和抛出错误 throw new Error() 的作用是相同的

当状态已经改变为resolved后，即使抛出错误，也不会触发then()的错误回调或者catch()方法

then() 和 catch() 都会返回一个新的Promise对象，可以链式调用

```
promise.then(
    () => { console.log('this is success callback') }
).catch(
    (err) => { console.log(err) }
).then(
    ...
).catch(
    ...
)
```

Promise实例的异步方法和then()中返回promise有什么区别?

```
// p1异步方法中返回p2
let p1 = new Promise ( (resolve, reject) => {
    resolve(p2)
} )
let p2 = new Promise ( ... )

// then()中返回promise
let p3 = new Promise ( (resolve, reject) => {
    resolve()
} )
let p4 = new Promise ( ... )
p3.then(
    () => return p4
)
```

p1异步方法中返回p2

p1的状态取决于p2，如果p2为pending，p1将等待p2状态的改变，p2的状态一旦改变，p1将会立即执行自己对应的回调，即then()中的方法针对的依然是p1

then()中返回promise

由于then()本身就会返回一个新的promise，所以后一个then()针对的永远是一个新的promise，但是像上面代码中我们自己手动返回p4，那么我们就可以在返回的promise中再次通过 resolve() 和 reject() 来改变状态

Promise的其他api

Promise.resolve() / Promise.reject()

用来包装一个现有对象，将其转变为Promise对象，但Promise.resolve()会根据参数情况返回不同的Promise：

参数是Promise：原样返回
参数带有then方法：转换为Promise后立即执行then方法
参数不带then方法、不是对象或没有参数：返回resolved状态的Promise

Promise.reject()会直接返回rejected状态的Promise

Promise.all()

参数为Promise对象数组，如果有不是Promise的对象，将会先通过上面的Promise.resolve()方法转换

```
var promise = Promise.all( [p1, p2, p3] )
promise.then(
    ...
).catch(
    ...
)
```

当p1、p2、p3的状态都变成resolved时，promise才会变成resolved，并调用then()的已完成回调，但只要有一个变成rejected状态，promise就会立刻变成rejected状态

Promise.race()

```
var promise = Promise.race( [p1, p2, p3] )
promise.then(
    ...
).catch(
    ...
)
```

"竞速"方法，参数与Promise.all()相同，不同的是，参数中的p1、p2、p3只要有一个改变状态，promise就会立刻变成相同的状态并执行对于的回调

Promise.done() / Promise. finally()

Promise.done() 的用法类似 .then() ，可以提供resolved和rejected方法，也可以不提供任何参数，它的主要作用是在回调链的尾端捕捉前面没有被 .catch() 捕捉到的错误

Promise .finally() 接受一个方法作为参数，这个方法不管promise最终的状态是怎样，都一定会被

执行

88人点赞> 前端周记 ···

"小礼物走一走，来简书关注我"

赞赏支持

还没有人赞赏，支持一下

次人君在野原之森网络工作室 次人君就是说疯狂的话、做疯狂的事的人。野...
总资产6 (约0.52元) 共写了7.1W字 获得472个赞 共278个粉丝
关注

写下你的评论...

全部评论 13 只看作者 按时间倒序 按时间正序

佛系怪兽君
8楼 2019.05.08 21:55
佛系怪兽君带着孩子最爱的沙拉脆面肉丸#丘比沙拉酱#
来支持你啦,简友加油！欢迎到我这里来看看！！点个小赞，好运不断，来个关注，青春常驻，打个小赏，工资疯涨。。。
赞 回复

师传志
7楼 2018.11.17 16:51
作者您好，文中有句话没有理解，“同样，catch()中也可以抛出错误，由于抛出的错误会在下一个catch中被捕获处理，因此可以再添加catch()”，这样的话不是要一直写下一个catch方法去捕获上一个catch的错误吗？请问这个错误是咋处理的？
赞 回复

小陈陈酱
6楼 2018.08.09 15:24
简单明了！
有例可循！
赞
赞 回复

程序员中的文科生
5楼 2018.07.30 12:06
cChannels({ commit,state }) {
return new Promise((resolve, reject) => {
axios.get('/api/news/channel?appkey=ca05a06b9221f5d1')
.then(res => {
commit('muChannels', res.data.result)
})
resolve()
})
},
朋友，最后这一步resolve具作用是什么啊？
赞 回复

次人君在野原之森网络工作室 作者
2018.07.30 22:24
触发成功回调，写错位置了吧，写在这边就是不等请求完成立即resolve
回复

程序员中的文科生
2018.07.31 10:24
@次人君在野原之森网络工作室 resolve触发成功回调，具体是回调什么函数啊，如果传参数，那个函数接收啊
回复

添加新评论

八戒活的真自在哈
4楼 2018.02.26 22:19
简洁明了，点赞！
赞 回复

迷_书
3楼 2017.10.23 18:14
那个，楼主，race应该是种类吧 - 0-，感觉翻译成竞速有点，，，，
1 回复

迷_书
2017.10.23 18:15
不过写的很棒，很简洁，阮大大写的太详细了，看起来脑仁疼
回复

添加新评论

omglhw
2楼 2017.09.21 19:19
es6的promise好像没有finally等方法
1 回复

迷_书
2017.10.23 18:14

我看过的阮一峰写的是有的
http://es6.ruanyifeng.com/#docs/promise

回复

次人君在野原之森网络工作室 作者
2017.10.23 18:31

@迷_书 是哒，也是学习了阮大大写的以后总结的，英语不好只能按照常见释义写啦😠请原谅这些瑕疵，有机会一定恶补英语！

回复

Jianshu9527
2018.08.07 10:36

ES8已经引入标准了

回复

添加新评论

被以下专题收入，发现更多相似内容

Web前端之路 | 前端开发 | 首页投稿（暂停... | 让前端飞 | 前端开发那些事 | es 6 | 前端知识梳理 | 展开更多

推荐阅读

更多精彩内容 >

JavaScript(ES6) - Promise对象

Promise的含义:　　Promise是异步编程的一种解决方案，比传统的解决方案——回调函数和事件——更合理和...

呼呼哥　阅读 1,164　评论 0　赞 15

13、Promise 对象

00、前言Promise 是异步编程的一种解决方案，比传统的解决方案——回调函数和事件——更合理和更强大。它由社区...

夜幕小草　阅读 985　评论 0　赞 12

ES6之Promise深入学习笔记

Promiese 简单说就是一个容器，里面保存着某个未来才会结束的事件（通常是一个异步操作）的结果，语法上说，Pr...

雨飞飞雨　阅读 1,675　评论 2　赞 18

ECMAScript Promise对象

Promise的含义 Promise，简单说就是一个容器，里面保存着某个未来才会结束的事件（通常是一个异步操作）...

冰豹　阅读 289　评论 1　赞 7

王莹：幸福是做你有兴趣也有能力做的事

岁数渐长，才真正明白，一个人做事只有兴趣和能力相结合，才是幸福，不会别扭。一直都有个发财梦，梦想能挣很多钱。后来发...

Herplus王莹　阅读 75　评论 0　赞 0