Promise：一种更优的异步编程统一方案

Promise：有三种状态pending（等待），fulfilled（成功），rejecte（失败）。当成功时会调用：onFulfilled函数，失败时调用onRejected函数。Promise的状态只是三者中的一个，可以从pending转为fulfilled或者rejected，但是一旦转为fulfilled或者rejected则不可更改为其他状态。

**1、Promise的基本使用：**

Promise中有一个then与catch方法，当成功时会执行then 方法，在then方法中可以通过return将本次的数据返回，可以在下一个then方法获取到进行下一步的数据处理，也就形成了promise的链式调用，而失败时会执行catch方法。

1. const promise = new Promise((resolve,reject) => {
2. resolve('success')
3. reject('failure')
4. })
5. promise
6. .then(resolve => {
7. console.log(resolve)
8. return '传递的数据：' + resolve
9. })
10. .then(resolve => {
11. console.log(resolve)
12. })
13. .catch(reject => {
14. console.log(reject);
15. })

输出结果：

1. success //第7行打印结果
2. 传递的数据：success //第11行打印结果

附：链式调用解决了回调地域的问题，这也就是promise相对于Ajax的优势，关于链式调用就是then方法的重复调用，这里比较简单，就不举例说明了

### 2、Promise的数据请求应用：

写一个请求数据的ajax函数，运用XMLHttpRequest进行数据请求，请求的地址为本地文件：/api/posts.json，代码注释已标明如下：

1. function ajax(url) {
2. return new Promise((resolve,reject) => {
3. let xhr = new XMLHttpRequest()
4. // 初始化 HTTP 请求参数
5. xhr.open('get',url)
6. // 接收到的数据类型视为 JSON 解析
7. xhr.responseType = 'json'
8. // 请求成功完成时调用的函数。
9. xhr.onload = () => {
10. if(this.status === 200){
11. // response: 响应的数据
12. resolve(this.response)
13. }else{
14. // statusText: HTTP 服务器返回的完整的响应状态文本
15. reject(new Error(this.statusText))
16. }
17. }
18. // 发送请求
19. xhr.send()
20. })
21. }
22. // 假设路径中有/api/posts.json该目录
23. ajax('/api/posts.json').then(function(res){
24. console.log(res);
25. },function(error){
26. console.log(error)
27. })

附：关于XMLHttpRequest参考文档：

<https://developer.mozilla.org/zh-CN/docs/Web/API/XMLHttpRequest>

### 3、Promise的all方法：

all方法可实现promise的并行执行，即可同时执行多条数据请求，但是只有所有请求成功才会返回成功，如果有一个请求失败则会返回失败，如下例：

1. ajax('/api/users.json'),
2. ajax('/api/urls.json')
3. ])
4. .then(function(resolve) {
5. console.log(resolve);
6. })
7. .catch(function(error){
8. console.log(error)
9. })

### 4、Promise的race方法

race方法只会等待第一个结束的任务返回结果，常用于处理请求超时的任务处理：

1. const url = ajax('/api/urls.json')
2. const timeout = new Promise((reaolve,reject) => {
3. setTimeout(() => {
4. reject(new Error('timeout'))
5. },500)
6. })
7. Promise.race([
8. url,
9. timeout
10. ])
11. .then(function(resolve) {
12. console.log(resolve);
13. })
14. .catch(function(error){
15. console.log(error)
16. })