async await 精讲

前端精进 - 后端方向：JS 专精

版权声明

联系方式

如果你想要购买本课程
请微信联系 xiedaimala02 或 xiedaimala03

如果你发现有人盗用本课程
请微信联系 xiedaimala02 或 xiedaimala03

参考链接

- ES 6 入门
  - 作者阮一峰
  - 内有全面的 ES 6 介绍

Promise 表示类
promise 表示实例对象

约定

基本用法

const result = await promise

从 Promise 说起

- Promise 封装

```
function 摇色子(){
  return new Promise((resolve, reject)=>{
    setTimeout(()=>{
      resolve(Math.floor(Math.random()*6)+1)
    },3000)
  })
}
```

- 使用 Promise

✓ 摇色子().then(s1, f1).then(s2, f2)

Ma Mi 任务模型

先 Ma 再 Mi

其实一开始没有两个任务队列

为了让 Promise 回调更早执行，强行插入了一个队列

如果没有 Mi 任务，还不如直接用 setTimeout 呢

Promise 的其他API

- Promise.resolve(result)
  - ✓ 制造一个成功（或失败）
- Promise.reject(reason)
  - ✓ 制造一个失败
- Promise.all(数组)
  - ✓ 等待全部成功，或者有一个失败
- Promise.race(数组)
  - ✓ 等待第一个状态改变
- Promise.allSettled(数组)
  - ✓ 等待全部状态改变，目前处于 Stage-4

Promise 的应用场景

- 多次处理一个结果
  - 摇筛子`.then(v => v1).then(v1 => v2)`
- 串行
  - 这里有一个悖论：一旦 promise 出现，那么任务就已经执行了。
  - 所以不是 promise 串行，而是任务串行
  - 解法：把任务放进队列，完成一个再做下一个
- 并行
  - Promise.all([p1,p2]) 不好用
  - Promise.allSettled 又太新
  - 怎么办

Promise 错误处理

- 其实挺好用的
  - ✓ promise.then(s1,f1) 即可
  - ✓ 不行，我想要全局错误处理（以 axios 为例）
- 语法糖
  - ✓ promise.then(s1).catch(f1)
- 错误处理之后
  - ✓ 如果你没有继续抛错，那么错误就不再出现
  - ✓ 如果你继续抛错，那么后续回调就要继续处理错误

前端似乎对 Promise 不满

async / await 替代 Promise 的 6 个理由

async / await 基本用法

- 最常见用法

```
const fn = async ()=>{
  const temp = await makePromise()
  return temp + 1
}
```

- 优点

✓ 完全没有缩进，就像是在写同步代码

封装一个 async 函数

- 代码

```
async function 摇色子(){
  return Math.floor(Math.random()*6)+1
}
```

- ✓ 如果需要 reject，直接 throw Error('xxx')

- 使用

```
async fn(){
  const result = await 摇色子()
}
```

- ✓ 如果需要处理错误，可以 try catch

为什么需要 async

看起来非常多余，await 所在函数就是 async，不是吗?

为了兼容旧代码里普通函数里的await(xxx)

用心良苦

await 错误处理

- 常见

```
let response
try{
  response = await axios.get('/xxx')
}catch(e){
  if(e.response){console.log(e.response.states)}
  throw e
}
console.log(response)
```

- 一个字：丑

await 错误处理

- 正确做法

```
const response =
  await axios.get('/xxx').then(null, errorHandler)
console.log(response)
```

- 细节

- ✓ 可以把 4xx / 5xx 等常见错误用拦截器全局处理
- ✓ await 只用关心成功，失败全部交给 errorHandler
- ✓ errorHandler 也可以放在拦截器里

await 的传染性

- 代码
  - console.log(1)
  - await console.log(2)
  - console.log(3)
- 分析
  - console.log(3) 变成异步任务了
  - Promise 同样有传染性（同步变异步）
  - 谁没有传染性：回调

await 的应用场景

- 多次处理一个结果
  - `const r1 = await makePromise()`
  - `const r2 = handleR1(r1)`
  - `const r3 = handleR2(r2)`
- 串行
  - 天生串行
  - 循环的时候有 bug（JS 又出了一个新东西来弥补）
- 并行
  - await Promise.all([p1,p2,p3]) 就是并行了

问 & 答

交流群提问

问1

- async / await 是 promise 的语法糖，如何用 promise 实现 async / await

✓ 答：有的语法糖好改写，有的语法糖不好改写。这个语法糖就不好改写，因为这是语言层面的改动，而不是 API 层面的。可以举例说明。

问2

- 有些 await 需要等待上一个 await 的结果，有些不用，如何让不用同步的 await 异步进行
  - 答：虽然解法很简单，但好像很多人都不知道。

问3

- 如何实现一个监控，调用异步请求一直等待，监控某个值变了就执行回调
  - 答：不太清楚你的场景，不过 Vue 2 已经做到了这一点。用 Object.defineProperty。也可以用 Proxy 方案。

问3

- 代码题

```
let a = 0;
let test = async () => {
  a = a + await 10;
  console.log(a)
}
test();
console.log(++a);
```

- 答
  - ✓ 倒数第二行有坑

问4

- 代码题

页面有两个按钮A和B，以及一个输入框，A按钮点击后发送一个请求，返回一个字符串A，B也发请求，但返回字符串B，返回后会把字符串赋值给输入框，但是A、B发送的两个请求返回的时间不同,点击两个按钮的顺序也不一定，B要比A先返回，而最终效果要求是输入框字符的顺序是A B。

- 答

- ✓ 有经验的工程师经常遇到
- ✓ 工程师 v.s. 码农

一些题外话

关于 Monad

再见

下节课讲讲 eventloop，我在 2019 前端押题讲过