手写 Promise

JS 专精 - 后端方向

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

面试答题方法论

- 顺序

- ✓ 该技术要解决什么问题 - why
- ✓ 该技术是怎么解决它的 - how
- ✓ 该技术有什么优点（对比其他技术）- pros
- ✓ 该技术有什么缺点 - cons
- ✓ 如何解决这些缺点 - more

Promise

要解决什么问题

回调地狱

```
fs.readdir(source, function (err, files) {
  if (err) {
    console.log('Error finding files: ' + err)  }
  else {
    files.forEach(function (filename, fileIndex){
      console.log(filename)
      gm(source + filename).size(function (err, values) {
        if (err) {
          console.log('Error identifying file size: ' + err)
        } else {
          console.log(filename + ' : ' + values)
          aspect = (values.width / values.height)
          widths.forEach(function (width, widthIndex) {
            height = Math.round(width / aspect)
            console.log('resizing ' + filename + 'to ' + height + 'x' + height)
            this.resize(width, height)
              .write(dest + 'w' + width + '_' + filename, function(err) {
                if (err) console.log('Error writing file: ' + err)
            })
          }.bind(this))
        }
      })
    })
  }
})
```

回调地狱真的是个问题吗

有没有可能是这个程序员水平不行

回调不地狱

```
fs.readdir(source, (err, files)=> {
  travalFiles = () => {
    if(err){ return console.log('Error: 找不到目录 ' + err)  }
    files.forEach(gmFile)
  }
  gmFile = (filename) => {
    console.log(filename)
    gm(source + filename).size(afterGetSize)
  }
  afterGetSize = (err, values) => {
    if (err) return console.log('无法读取文件尺寸: ' + err)
    console.log(filename + ' : ' + values)
    aspect = (values.width / values.height)
    widths.forEach((width, widthIndex) => resize(width, aspect))
  }
  resize = (width, aspect) => {
    height = Math.round(width / aspect)
    console.log('将' + filename + '的尺寸变为 ' + width + 'x' + height)
    this.resize(width, height)
      .write(dest + 'w' + width + '_' + filename, (err) =>
         err && console.log('Error writing file: ' + err)
    )
  }
  travalFiles(err, files)
})
```

回调没有问题

出现地狱的水平问题

但是水平差的人就是多

好吧，妥协吧

如果面试官问你

Promise 解决了什么问题

你还是回答「回调地狱」吧

Promise 有什么优点

怎么答

Promise 两个优点

- 减少缩进
  - ✓ 把「函数里的函数」变成「then 下面的 then」（链式）

```
f1(xxx, function f2(a){
  f3(yyy, function f4(b){
    // f4 是函数里的函数
    f5(a+b, function f6(){})
  })
})
```

```
f1(xxx)
  .then(f2) // f2 里面调用 f3
  .then(f4) // f4 里面调用 f5
  .then(f6)
提问：f5 怎么得到 a 和 b
答：f2 的输出作为 f4 的输入
```

- 消灭 if (err)
  - ✓ 错误处理单独放到一个函数里
  - ✓ 如果不处理，就一直等到往后抛

```
f1(xxx)
  .then(f2, error1)
  .then(f4, error2)
  .then(f6, error3)
  .then(null, errorAll)
// 最后一句可以写成 .catch
```

用户怎么用 Promise

- before

```
function 摇色子(fn){
  setTimeout(()=>{
    const result = Math.floor(Math.random()*6+1)
    fn(result) // 等价于 fn.call(null, result)
  },3000)
}
摇色子(n=>console.log(`摇到了${n}`))
```

- after

```
function 摇色子(){
  // new Promise 接受一个函数，返回一个 Promise 实例
  return new Promise((resolve, reject)=>{
    setTimeout(()=>{
      const result = Math.floor(Math.random()*6+1)
       resolve(result)
    },3000)
  })
}

摇色子().then(n=>console.log(`摇到了${n}`))
```

Promise 的完整 API 是什么

- Promise 是一个类
  - ✓ JS 里类是特殊的函数
  - ✓ 类属性：length（可忽略）
  - ✓ 类方法：all / allSettled / race / reject / resolve
  - ✓ 对象属性：then（重要） / finally / catch
  - ✓ 对象内部属性：state = pending / fulfilled / rejected

API 的规则是什么

- Promise/A+ 规格文档
  - ✓ JS 的 Promise 的公开标准
  - ✓ 有中文翻译，但是我不保证其准确性
- 开始写代码
  - ✓ 按照文档写测试用例
  - ✓ 先让用例失败
  - ✓ 然后让用例通过
  - ✓ 直到把文档的所有情况都考虑清楚

使用 chai

- 更牛X的测试方案
  - ✓ 我们之前的手写用例太原始了，不利于定位错误
- 步骤
  - ✓ yarn global add ts-node mocha 全局安装
  - ✓ 创建目录 promise-demo（名字随意）
  - ✓ yarn init -y 或者 npm init -y
  - ✓ yarn add chai mocha --dev
  - ✓ yarn add @types/chai @types/mocha --dev
  - ✓ 创建 test/index.ts
  - ✓ mocha -r ts-node/register test/index.ts 运行测试

yarn test

- 步骤

✓ 在 package.json 里面加 test 命令，内容为 mocha -r ts-node/register test/**/*.ts
✓ yarn test 此时报错，缺少 ts-node 和 typescript
✓ yarn add ts-node typescript --dev
✓ yarn test 不报错

使用 sinon 测试函数

- 安装

- yarn add sinon sinon-chai --dev
- yarn add @types/sinon @types/sinon-chai --dev