

# Node实战

---

## 1. 课程目标

1. 通过命令行，输入关键词、数量，实现下载百度搜索来的图片；

## 2. 课程大纲

- node爬虫&CLI 实战；

## 3. Node爬虫

### 3.1. 什么是爬虫？

爬虫就是一个探测机器，自动模拟人的行为去各个网站，点点按钮，查查数据，或者把看到的信息背回来。就像一只虫子在一幢楼里不知疲倦地爬来爬去。

- 网络引擎：更新自己的网站内容，以及对其他网站的索引，是良性的；
- APP：抢票软件，1s 成千上万次，对12306服务器压力过大，是恶性的；
- 个人：使用爬虫，获取网站的内容，建议只用作学习等用途；

### 3.2. 爬虫是否可以肆无忌惮地爬取所有内容么？

不是的，爬取访问的网络，会消耗网站的流量、带宽或者其他服务器资源；

Q：那小型网站如何避免被爬虫爬取消耗带宽？

robots.txt：是一种存放于网站根目录下的ASCII编码的文本文件，它通常告诉爬虫此网站中的哪些内容是不应被爬的，或者哪些搜索引擎是被允许爬取的；

这个协议也不是一个规范，而只是约定俗成的，有些搜索引擎会遵守这一规范，有些则不然。通常搜索引擎会识别这个元数据，不索引这个页面；

```
1  // 允许所有机器人
2  User-agent: *
3  Disallow:
4
5  User-agent: *
6  Allow: /
7
8  // 允许特定的机器人
9  User-agent: name_spider (用其他爬虫的名称代替)
10 Allow:
11
12 // 拦截所有爬虫
13 User-agent: *
14 Disallow: /
15
16 // 禁止所有机器人访问特定的目录
17 User-agent: *
18 Disallow: /cgi-bin/
19 Disallow: /images/
20 Disallow: /tmp/
21 Disallow: /private/
22
23 // 禁止所有机器人访问特定文件
24 User-agent: *
25 Disallow: /*.php$
26 Disallow: /*.js$
27 Disallow: /*.inc$
28 Disallow: /*.css$
```

或者其他形式：

```
1  <meta name="robots" content="noindex,nofollow" />
```

### 3.3. 如何实现一个爬虫应用？

1. 明确要爬取的网站、页面；
2. 分析网站的数据及DOM；

### 3. 确定技术选型

#### a. 模拟浏览器端请求：

i. [request](#)：已经不维护了，不建议使用；

✅ [superagent](#)：是在node层或服务端实现代理请求的模块，支持全量的ajax methods，可以设置headers等；

#### b. 解析DOM：

✅ [cheerio](#)：api类似jQuery

i. [jsDOM](#)：可以解析dom文本

#### c. 模拟用户行为操作：

i. [puppeteer](#)：相当于在node端启动一个浏览器，用来模拟chrome浏览器的各种运行，常用来定期去巡检页面功能等；

## 4. CLI

Command Line Interface，命令行交互界面，像我们经常使用的[create-react-app](#)，[vue-cli](#)都是我们耳熟能详的cli。

作用：代替人实现重复劳动，提升开发效率

1. 快速生成应用模板，如vue-cli等根据与开发者的一些交互式问答生成应用框架
2. 创建module模板文件，如angular-cli，创建component,module；sequelize-cli 创建与mysql表映射的model等
3. 服务启动，如ng serve
4. eslint，代码校验，如vue,angular，基本都具备此功能
5. 自动化测试 如vue,angular，基本都具备此功能
6. 编译build，如vue,angular，基本都具备此功能

### 4.1. 与npm scripts对比

npm scripts也可以实现开发工作流，通过在package.json 中的scripts对象上配置相关npm 命令，执行相关js来达到相同的目的；但是cli工具与npm scripts相比有什么优势呢？

1. npm scripts是某个具体项目的，只能在该项目内使用，cli可以是全局安装的，多个项目使用；
2. 使用npm scripts 在业务工程里面嵌入工作流，耦合太高；使用cli 可以让业务代码工作流相关代码剥离，业务代码专注业务；

3. cli工具可以不断迭代开发，演进，沉淀；

## 4.2. 如何实现一个CLI?

涉及库：

[commander](#)：为cli提供命令行接入的方案

[inquirer](#)：提供交互的GUI

## 5. 实战

课程上以搜索“柯基”为例子

### 5.1. 初始化项目

1. npm init
2. 安装superagent、cheerio
3. 指定项目运行入口

```
1  npm init
2
3  // 安装基础依赖
4  npm i -S superagent cheerio
5
6  // 修改项目npm scripts
7  start": "node index.js"
8
9  // 运行
10 npm run start
```

JavaScript | 复制代码

### 5.2. superagent、cheerio使用测试

1. 测试获取资源html，方法是否生效

```
1 // index.js
2 const superagent = require('superagent');
3 const cheerio = require('cheerio');
4
5 superagent.get('http://www.baidu.com').end((err, res) => {
6   if (err) {
7     console.err('访问失败, 原因', err);
8     return;
9   }
10
11   console.log(res);
12 });
13
```

## 2. 解析获取到的html

```
1 // index.js
2 const superagent = require('superagent');
3 const cheerio = require('cheerio');
4
5 superagent.get('http://www.baidu.com').end((err, res) => {
6   if (err) {
7     console.err('访问失败, 原因: ', err);
8     return;
9   }
10
11   const htmlText = res.text;
12   const $ = cheerio.load(htmlText);
13
14   $('meta').each((index, ele) => {
15     console.log(`${index}: ${$(ele).attr('content')}`);
16   });
17 });
18
```

## 5.3. 解析获取到的html，抓取百度图片

搜索：柯基，在DOC栏下查看

完整的url:

```
https://image.baidu.com/search/index?
tn=baiduimage&ps=1&ct=201326592&lm=-1&cl=2&nc=1&ie=utf-
8&dyTabStr=MCwzLDEsNSw0LDYsNywyLDgsOQ%3D%3D&word=%E6%9F%AF%E5%9F%BA
```

有用的url:

```
https://image.baidu.com/search/index?tn=baiduimage&ie=utf-
8&word=%E6%9F%AF%E5%9F%BA
```

- tn: 百度图片
- ie: 编码格式
- word: encode(keyword)

查询到相关资源为: objURL, 存在于JSON

抓取数据源: 查看网页源代码里, 会看到是资源类型都是放到JSON里的, 通过JS将图片渲染进页面中, 没法通过cheerio, 解析dom

获objURL后发现, 图片都是经过转义的, 格式为:

```
https://\gimg2.baidu.com\image_search\src=http%3A%2F%2Fimg11.51tietu.net%2Fpic%2F2
019112217%2F3fhqw1z53c33fhqw1z53c3.jpg&refer=http%3A%2F%2Fimg11.51tietu.net&app=200
2&size=f9999,10000&q=a80&n=0&g=0n&fmt=auto?
sec=1659948766&t=9de4acba8fd9610a9f8bee2374bd42e3
```



JavaScript | 复制代码

```
1  "objURL": "", "XX": "XXX"
```

Q: 如何获取objURL?

使用正则表达式

```
1  /"objURL": "(.*?)", /
```

将图片获取处理方法放到另一个文件img.handler.js中

- keyword要encode
- 访问连接为http，避免https导致的各种安全性问题

```
1  // img.handle.js
2  const superagent = require('superagent');
3  const cheerio = require('cheerio');
4
5  const word = '柯基';
6
7  superagent
8    .get(
9      `http://image.baidu.com/search/index?tn=baiduimage&ie=utf-
10      8&word=${encodeURIComponent(
11        word
12      )}`
13    )
14    .end((err, res) => {
15      if (err) {
16        console.error('访问失败, 原因: ', err);
17        return;
18      }
19      const htmlText = res.text;
20      const $ = cheerio.load(htmlText);
21
22      console.log(htmlText);
23    });
24
```

Q: 为什么会走进百度的安全验证?

1. 单位时间内大量访问百度图片

✓ 没有携带header

```
1  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
2  Accept-Encoding: gzip, deflate, br
3  Accept-Language: zh-CN,zh;q=0.9
4  Cache-Control: max-age=0
5  Connection: keep-alive
6  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
   (KHTML, like Gecko) Chrome/103.0.0.0 Safari/537.36
7  sec-ch-ua: ".Not/A)Brand";v="99", "Google Chrome";v="103",
   "Chromium";v="103"
```

注意：尽量保证所带的header跟浏览器保持一致，尤其是Accept、UA



```
1  const superagent = require('superagent');
2  const cheerio = require('cheerio');
3
4  const word = '柯基';
5
6  const header = {
7    Accept:
8      'text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/w
9      ebp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9',
10     'Accept-Encoding': 'gzip, deflate, br',
11     'Accept-Language': 'zh-CN,zh;q=0.9',
12     'Cache-Control': 'max-age=0',
13     Connection: 'keep-alive',
14     'User-Agent':
15       'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
16       like Gecko) Chrome/103.0.0.0 Safari/537.36',
17     'sec-ch-ua': '".Not/A)Brand";v="99", "Google Chrome";v="103",
18     "Chromium";v="103"',
19   };
20
21  superagent
22    .get(
23      `http://image.baidu.com/search/index?tn=baiduimage&ie=utf-
24      8&word=${encodeURIComponent(
25        word
26      )}`
27    )
28    .set('Accept', header['Accept'])
29    .set('Accept-Encoding', header['Accept-Encoding'])
30    .set('Accept-Language', header['Accept-Language'])
31    .set('Cache-Control', header['Cache-Control'])
32    .set('Connection', header['Connection'])
33    .set('User-Agent', header['User-Agent'])
34    .end((err, res) => {
35      if (err) {
36        console.err('访问失败, 原因: ', err);
37        return;
38      }
39
40      const htmlText = res.text;
41      const $ = cheerio.load(htmlText);
42
43      console.log(htmlText);
44    });
```

## 5.4. 获取图片链接列表

通过正则获取objURL的数组

▼ JavaScript | 复制代码

```
1  const htmlText = res.text;
2  const imageMatches = htmlText.match(/"objURL": "(.*?)" /g);
```

再通过正则只获取链接

```
1  superagent
2    .get(
3      `http://image.baidu.com/search/index?tn=baiduimage&ie=utf-
      8&word=${encodeURIComponent(
4        word
5      )}`
6    )
7    .set('Accept', header['Accept'])
8    .set('Accept-Encoding', header['Accept-Encoding'])
9    .set('Accept-Language', header['Accept-Language'])
10   .set('Cache-Control', header['Cache-Control'])
11   .set('Connection', header['Connection'])
12   .set('User-Agent', header['User-Agent'])
13   .end((err, res) => {
14     if (err) {
15       console.err('访问失败, 原因: ', err);
16       return;
17     }
18
19     const htmlText = res.text;
20     const imageMatches = htmlText.match(/"objURL": "(.*)" /g);
21
22     //
23     ""objURL": "https://gimg2.baidu.com/image_search/src=http%3A%2F%2Fimg9.51t
24     ietu.net%2Fpic%2F2019-
25     091304%2Fv451j2jqusqv451j2jqusq.jpg&refer=http%3A%2F%2Fimg9.51tietu.net&a
26     pp=2002&size=f9999,10000&q=a80&n=0&g=0n&fmt=auto?
27     sec=1659950528&t=c082067e049cfa937a180c3e3eae8ec1", '
28
29     const imageUrlList = imageMatches.map(item => {
30       const imageUrl = item.match(/: "(.*)" /g);
31       return RegExp.$1;
32     });
33
34     console.log(imageUrlList);
35   });
```

## 5.5. 获取图片的标题

根据页面source获取图片名称，为：fromPageTitle

```
1  superagent
2    .get(
3      `http://image.baidu.com/search/index?tn=baiduimage&ie=utf-
      8&word=${encodeURIComponent(
4        word
5      )}`
6    )
7    .set('Accept', header['Accept'])
8    .set('Accept-Encoding', header['Accept-Encoding'])
9    .set('Accept-Language', header['Accept-Language'])
10   .set('Cache-Control', header['Cache-Control'])
11   .set('Connection', header['Connection'])
12   .set('User-Agent', header['User-Agent'])
13   .end((err, res) => {
14     if (err) {
15       console.err('访问失败, 原因: ', err);
16       return;
17     }
18
19     const htmlText = res.text;
20     const imageMatches = htmlText.match(/"objURL": "(.*)" /g);
21
22     //
23     ""objURL": "https://gimg2.baidu.com/image_search/src=http%3A%2F%2Fimg9.51t
24     ietu.net%2Fpic%2F2019-
25     091304%2Fv451j2jqusqv451j2jqusq.jpg&refer=http%3A%2F%2Fimg9.51tietu.net&a
26     pp=2002&size=f9999,10000&q=a80&n=0&g=0n&fmt=auto?
27     sec=1659950528&t=c082067e049cfa937a180c3e3eae8ec1", '
28
29     const imageUrlList = imageMatches.map(item => {
30       const imageUrl = item.match(/: "(.*)" /g);
31       return RegExp.$1;
32     });
33
34     const titleMatches = htmlText.match(/"fromPageTitle": "(.*)" /g);
35
36     const titleList = titleMatches.map(item => {
37       const title = item.match(/: "(.*)" /g);
38       return RegExp.$1;
39     });
40
41     console.log(imageUrlList, titleList);
42   });
43 }
```

无法直接使用，需要对title进行处理，且需要封装正则

JavaScript | 复制代码

```
1
2
3 ▾ function getValueListByReg(str, key) {
4     const reg = new RegExp(`"${key}":"(.*)"`, 'g');
5     const matchResult = str.match(reg);
6
7 ▾     const resultList = matchResult.map(item => {
8         const result = item.match(/:"(.*)"/g);
9         return RegExp.$1;
10    });
11
12     return resultList;
13 }
```

## 5.7. 创建目录，存储图片

```
1 function mkImageDir(pathname) {
2   const fullPath = path.resolve(__dirname, pathname);
3
4   if (fs.existsSync(fullPath)) {
5     console.log(`${pathname}已存在, 跳过此步骤`);
6     return;
7   }
8
9   fs.mkdirSync(fullPath);
10  console.log(`目录创建成功! 目录为: ${pathname}`);
11 }
12
13 superagent
14   .get(
15     `http://image.baidu.com/search/index?tn=baiduimage&ie=utf-
16     8&word=${encodeURIComponent(
17       word
18     )}`
19   )
20   .set('Accept', header['Accept'])
21   .set('Accept-Encoding', header['Accept-Encoding'])
22   .set('Accept-Language', header['Accept-Language'])
23   .set('Cache-Control', header['Cache-Control'])
24   .set('Connection', header['Connection'])
25   .set('User-Agent', header['User-Agent'])
26   .end((err, res) => {
27     if (err) {
28       console.err('访问失败, 原因: ', err);
29       return;
30     }
31
32     const htmlText = res.text;
33
34     const imageUrlList = getValueListByReg(htmlText, 'objURL');
35     const titleList = getValueListByReg(htmlText,
36       'fromPageTitle').map(item =>
37       item.replace('<strong>', '').replace('<\\//strong>', ''))
38       );
39     mkImageDir('images');
40   });
```

## 5.8. 下载图片到本地

```
1 function downloadImage(url, name, index) {
2   const fullPath = path.join(__dirname, 'images', `${index +
3     1}.${name.replace('?', '')}.png`);
4   if (fs.existsSync(fullPath)) {
5     console.log(`已存在, ${fullPath}`);
6     return;
7   }
8
9   superagent.get(url).end((err, res) => {
10     if (err) {
11       console.log(err, `获取链接出错, 内容为: ${res}`);
12       return;
13     }
14
15     if (JSON.stringify(res.body) === '{}') {
16       console.log(`第${index + 1}图片内容为空`);
17       return;
18     }
19
20     fs.writeFile(fullPath, res.body, 'binary', err => {
21       if (err) {
22         console.log(`第${index + 1}张图片下载失败: ${err}`);
23         return;
24       }
25       console.log(`第${index + 1}张图片下载成功: ${url}`);
26     });
27   });
28 }
29
30 superagent
31   .get(
32     `http://image.baidu.com/search/index?tn=baiduimage&ie=utf-
33     8&word=${encodeURIComponent(
34       word
35     )}`
36   )
37   .set('Accept', header['Accept'])
38   .set('Accept-Encoding', header['Accept-Encoding'])
39   .set('Accept-Language', header['Accept-Language'])
40   .set('Cache-Control', header['Cache-Control'])
41   .set('Connection', header['Connection'])
42   .set('User-Agent', header['User-Agent'])
43   .end(async (err, res) => {
44     if (err) {
```



```
44     console.err('访问失败, 原因: ', err);
45     return;
46 }
47
48 const htmlText = res.text;
49
50 const imageUrlList = getValueListByReg(htmlText, 'objURL');
51 const titleList = getValueListByReg(htmlText,
'fromPageTitle').map(item =>
52     item.replace('<strong>', '').replace('<\\//strong>', '')
53 );
54
55 await mkImageDir('images');
56
57 ▼ imageUrlList.forEach((url, index) => {
58     downloadImage(url, titleList[index], index);
59 });
60 });
61
```

## 5.9. 加进度条

安装 cli-progress

创建文件、下载图片及转为promise, 改为链式调用

```
1  const cliProgress = require('cli-progress');
2
3  const bar = new cliProgress.SingleBar(
4  {
5    clearOnComplete: false,
6  },
7    cliProgress.Presets.shades_classic
8  );
9
10 superagent
11   .get(
12     `http://image.baidu.com/search/index?tn=baiduimage&ie=utf-
13     8&word=${encodeURIComponent(
14       word
15     )}`
16   )
17   .set('Accept', header['Accept'])
18   .set('Accept-Encoding', header['Accept-Encoding'])
19   .set('Accept-Language', header['Accept-Language'])
20   .set('Cache-Control', header['Cache-Control'])
21   .set('Connection', header['Connection'])
22   .set('User-Agent', header['User-Agent'])
23   .end(async (err, res) => {
24     if (err) {
25       console.err('访问失败, 原因: ', err);
26       return;
27     }
28
29     const htmlText = res.text;
30
31     const imageUrlList = getValueListByReg(htmlText, 'objURL');
32     const titleList = getValueListByReg(htmlText,
33       'fromPageTitle').map(item =>
34         item.replace('<strong>', '').replace('<\\strong>', '')
35       );
36
37     try {
38       await mkImageDir('images');
39       bar.start(imageUrlList.length, 0);
40
41       imageUrlList.forEach((url, index) => {
42         downloadImage(url, titleList[index], index)
43           .then(() => {
44             finished++;
45             bar.update(finished);
46           })
47       });
48     } catch (err) {
49       console.err('下载失败, 原因: ', err);
50     }
51   })
52 }
```

```
44         })
45     }
46     .then(() => {
47         if (finished === imageUrlList.length) {
48             bar.stop();
49             console.log('恭喜, 图片下载完成! ');
50         }
51     });
52 } catch (e) {
53     console.log(e);
54 }
55 });
56
```

## 5.10. 删除文件

避免每次判断有文件存在时退出执行

```
1 ▾ function mkImageDir(pathname) {
2 ▾   return new Promise((resolve, reject) => {
3     const fullPath = path.resolve(__dirname, pathname);
4
5 ▾     if (fs.existsSync(fullPath)) {
6       removeDir(pathname);
7     }
8
9     fs.mkdirSync(fullPath);
10    console.log(`目录创建成功! 目录为: ${pathname}`);
11    return resolve();
12  });
13 }
14
15 ▾ function removeDir(pathname) {
16   const fullPath = path.resolve(__dirname, pathname);
17   console.log(`${pathname}目录已存在, 准备执行删除`);
18
19 ▾   fs.rmdirSync(fullPath, {
20     force: true,
21     recursive: true,
22   });
23   console.log(`目录${pathname}已删除! `);
24
25   // const process = require('child_process');
26   // process.execSync(`rm -rf ${fullPath}`);
27 }
28
29
```

## 5.11. 使用CLI输入关键词

在img.handle.js中, 将superagent export出去

```
1  #!/usr/bin/env node
2
3  const inquirer = require('inquirer');
4  const commander = require('commander');
5
6  const { runImg } = require('./img.handler.js');
7
8  const question = [
9    {
10     type: 'checkbox',
11     name: 'channels',
12     message: '请选择想要搜索的渠道',
13     choices: [
14       {
15         name: '百度图片',
16         value: 'images',
17       },
18       {
19         name: '百度视频',
20         value: 'videos',
21       },
22     ],
23   },
24   {
25     type: 'input',
26     name: 'keyword',
27     message: '请输入想要搜索的关键词',
28   },
29 ];
30
31 inquirer.prompt(question).then(result => {
32   const { keyword, channels } = result;
33
34   for (let channel of channels) {
35     switch (channel) {
36       case 'images':
37         runImg(keyword);
38         break;
39     }
40   }
41 });
42
```

## 5.12. 设置指定图片张数

继续查看百度图片，翻页

完整请求：

```
https://image.baidu.com/search/acjson?
tn=resultjson_com&logid=10828563366224423546&ipn=rj&ct=201326592&is=&fp=result&fr=&wor
d=%E6%9F%AF%E5%9F%BA&queryWord=%E6%9F%AF%E5%9F%BA&cl=2&lm=-1&ie=utf-
8&oe=utf-
8&adpicid=&st=&z=&ic=&hd=&latest=&copyright=&s=&se=&tab=&width=&height=&face=&istype
=&qc=&nc=1&expermode=&nojc=&isAsync=&pn=30&rn=30&gsm=1e&1657388413826=
```

处理后请求：

```
https://image.baidu.com/search/acjson?
tn=resultjson_com&logid=10828563366224423546&word=%E6%9F%AF%E5%9F%BA&queryWord
=%E6%9F%AF%E5%9F%BA&ie=utf-8&oe=utf-8&pn=60&rn=30&1657387001036=
```

是JSON结构，取middleURL

```
1  // index.js
2  {
3    type: 'number',
4    name: 'counts',
5    message: '请输入要下载的图片张数（最小30张） ',
6  },
7
8  inquirer.prompt(question).then(result => {
9    const { keyword, channels, counts } = result;
10
11    for (let channel of channels) {
12      switch (channel) {
13        case 'images':
14          runImg(keyword, counts);
15          break;
16      }
17    }
18  });
```

JavaScript | 复制代码

```
1  const superagent = require('superagent');
2  const cheerio = require('cheerio');
3  const path = require('path');
4  const fs = require('fs');
5  const cliProgress = require('cli-progress');
6
7  const bar = new cliProgress.SingleBar(
8    {
9      clearOnComplete: false,
10    },
11    cliProgress.Presets.shades_classic
12  );
13
14  let total = 0;
15  let finished = 0;
16
17  const header = {
18    Accept:
19      'text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9',
20    'Accept-Encoding': 'gzip, deflate, br',
21    Accept2: 'text/plain, /*; q=0.01',
22    'Accept-Language': 'zh-CN,zh;q=0.9',
23    'Cache-Control': 'max-age=0',
24    Connection: 'keep-alive',
25    'User-Agent':
26      'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/103.0.0.0 Safari/537.36',
27    'sec-ch-ua': '".Not/A)Brand";v="99", "Google Chrome";v="103", "Chromium";v="103"',
28  };
29
30  function getValueListByReg(str, key) {
31    const reg = new RegExp(`"${key}": "(.*?)"`, 'g');
32    const matchResult = str.match(reg);
33
34    const resultList = matchResult.map(item => {
35      const result = item.match(/:"(.*?)" /g);
36      return RegExp.$1;
37    });
38
39    return resultList;
40  }
41
42  function mkImageDir(pathname) {
```

```

42 ▼ return new Promise((resolve, reject) => {
43     const fullPath = path.resolve(__dirname, pathname);
44
45 ▼     if (fs.existsSync(fullPath)) {
46         removeDir(pathname);
47     }
48
49     fs.mkdirSync(fullPath);
50     console.log(`目录创建成功! 目录为: ${pathname}`);
51     return resolve();
52 });
53 }
54
55 ▼ function removeDir(pathname) {
56     const fullPath = path.resolve(__dirname, pathname);
57     console.log(`${pathname}目录已存在, 准备执行删除`);
58
59 ▼     fs.rmdirSync(fullPath, {
60         force: true,
61         recursive: true,
62     });
63     console.log(`目录${pathname}已删除!`);
64
65     // const process = require('child_process');
66     // process.execSync(`rm -rf ${fullPath}`);
67 }
68
69 ▼ function downloadImage(url, name, index) {
70 ▼     return new Promise((resolve, reject) => {
71         const fullPath = path.join(
72             __dirname,
73             'images',
74             `${index + 1}.${name.replace('?', '').replace('|', '')}.png`
75         );
76
77 ▼         if (fs.existsSync(fullPath)) {
78             return reject(`图片已存在, ${fullPath}`);
79         }
80
81 ▼         superagent.get(url).end((err, res) => {
82 ▼             if (err) {
83                 return reject(err, `获取链接出错, 内容为: ${res}`);
84             }
85
86 ▼             if (JSON.stringify(res.body) === '{}') {
87                 return resolve(`第${index + 1}图片内容为空`);
88             }
89

```



```

90     fs.writeFile(fullPath, res.body, 'binary', err => {
91         if (err) {
92             return reject(`第${index + 1}张图片下载失败: ${err}`);
93         }
94         return resolve(`第${index + 1}张图片下载成功: ${url}`);
95     });
96 });
97 });
98 }
99
100 function request(url, acceptKey = 'Accept') {
101     return new Promise((resolve, reject) => {
102         superagent
103             .get(url)
104             .set('Accept', header[acceptKey])
105             .set('Accept-Encoding', header['Accept-Encoding'])
106             .set('Accept-Language', header['Accept-Language'])
107             .set('Cache-Control', header['Cache-Control'])
108             .set('Connection', header['Connection'])
109             .set('User-Agent', header['User-Agent'])
110             .end(async (err, res) => {
111                 if (err) {
112                     reject('访问失败, 原因: ', err);
113                     return;
114                 }
115
116                 resolve(res);
117             });
118         });
119     }
120
121     async function getImageByPage(start, total, word) {
122         let allImages = [];
123
124         while (start < total) {
125             const size = Math.min(60, total - start);
126             const res = await request(
127                 `https://image.baidu.com/search/acjson?
128                 tn=resultjson_com&word=${encodeURIComponent(
129                     word
130                 )}&queryWord=${encodeURIComponent(
131                     word
132                 )}&ie=utf-8&oe=utf-8&pn=${start}&rm=${size}&${Date.now()}=`,
133                 'Accept2'
134             );
135
136             allImages = allImages.concat(JSON.parse(res.text).data);
137             start += size;

```

```

137     }
138
139     return allImages;
140 }
141
142 function runImg(keyword, counts) {
143     request(
144         `http://image.baidu.com/search/index?tn=baiduimage&ie=utf-
145         8&word=${encodeURIComponent(
146             keyword
147         )}`
148     ).then(async res => {
149         const htmlText = res.text;
150
151         const imageUrlList = getValueListByReg(htmlText, 'objURL');
152         const titleList = getValueListByReg(htmlText,
153 'fromPageTitle').map(item =>
154     item.replace('<strong>', '').replace('<\\</strong>', ''
155 );
156
157     let allImageUrls = imageUrlList.map((imageUrl, index) => {
158         return {
159             imageUrl,
160             title: titleList[index],
161         };
162     });
163
164     const firstPageCount = allImageUrls.length;
165
166     if (counts > firstPageCount) {
167         const restImgUrls = await getImageByPage(firstPageCount, counts,
168 keyword);
169         const formatImgUrls = restImgUrls
170             .filter(item => item.middleURL)
171             .map(item => {
172                 return {
173                     imageUrl: item.middleURL,
174                     title: item.fromPageTitle.replace('<strong>',
175 '').replace('</strong>', ''),
176                 };
177             });
178
179         allImageUrls = allImageUrls.concat(formatImgUrls);
180     }
181
182     total = allImageUrls.length;
183
184     try {

```

```

181     await mkImageDir('images');
182     bar.start(total, 0);
183
184     allImageUrls.forEach((item, index) => {
185         downloadImage(item.imageUrl, allImageUrls[index].title, index)
186             .then(() => {
187                 finished++;
188                 bar.update(finished);
189             })
190             .then(() => {
191                 if (finished === total) {
192                     bar.stop();
193                     console.log('恭喜, 图片下载完成! ');
194                 }
195             });
196     });
197     } catch (e) {
198         console.log(e);
199     }
200 });
201 }
202
203 module.exports = {
204     runImg,
205 };
206

```