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；

```
// 允许所有机器人
User-agent: *
Disallow:

User-agent: *
Allow:/

// 允许特定的机器人
User-agent: name_spider（用其他爬虫的名称代替）
Allow:

// 拦截所有爬虫
User-agent: *
Disallow: /

// 禁止所有机器人访问特定的目录
User-agent: *
Disallow: /cgi-bin/
Disallow: /images/
Disallow: /tmp/
Disallow: /private/

// 禁止所有机器人访问特定文件
User-agent: *
Disallow: /*.php$
Disallow: /*.js$
Disallow: /*.inc$
Disallow: /*.css$
```

或者其他形式：

```
<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

JavaScript 复制代码

```
npm init

// 安装基础依赖
npm i -S superagent cheerio

// 修改项目npm scripts
start": "node index.js"

// 运行
npm run start
```

5.2. superagent、cheerio使用测试

1. 测试获取资源html，方法是否生效

JavaScript 复制代码

```
// index.js
const superagent = require('superagent');
const cheerio = require('cheerio');

superagent.get('http://www.baidu.com').end((err, res) => {
  if (err) {
    console.err('访问失败，原因', err);
    return;
  }

  console.log(res);
});

```

2. 解析获取到的html

JavaScript 复制代码

```
// index.js
const superagent = require('superagent');
const cheerio = require('cheerio');

superagent.get('http://www.baidu.com').end((err, res) => {
  if (err) {
    console.err('访问失败，原因: ', err);
    return;
  }

  const htmlText = res.text;
  const $ = cheerio.load(htmlText);

  $('meta').each((index, ele) => {
    console.log(`${index}: ${$(ele).attr('content')}`);
  });
});

```

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%2F2019112217%2F3fhqw1z53c33fhqw1z53c3.jpg&refer=http%3A%2F%2Fimg11.51tietu.net&app=2002&size=f9999,10000&q=a80&n=0&g=0n&fmt=auto?sec=1659948766&t=9de4acba8fd9610a9f8bee2374bd42e3

JavaScript 复制代码

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

Q：如何获取objURL?

使用正则表达式

```
/"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

```
// img.handle.js
const superagent = require('superagent');
const cheerio = require('cheerio');

const word = '柯基';

superagent
  .get(
    `http://image.baidu.com/search/index?tn=baiduimage&ie=utf-8&word=${encodeURIComponent(
      word
    )}`
  )
  .end((err, res) => {
    if (err) {
      console.err('访问失败，原因: ', err);
      return;
    }

    const htmlText = res.text;
    const $ = cheerio.load(htmlText);

    console.log(htmlText);
  });

```

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

JavaScript | 复制代码

```
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
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9
Cache-Control: max-age=0
Connection: keep-alive
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
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

JavaScript | 复制代码

```javascript
const superagent = require('superagent');
const cheerio = require('cheerio');

const word = '柯基';

const header = {
  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',
  'Accept-Encoding': 'gzip, deflate, br',
  'Accept-Language': 'zh-CN,zh;q=0.9',
  'Cache-Control': 'max-age=0',
  Connection: 'keep-alive',
  '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',
  'sec-ch-ua': '".Not/A)Brand";v="99", "Google Chrome";v="103", "Chromium";v="103"',
};

superagent
  .get(
    `http://image.baidu.com/search/index?tn=baiduimage&ie=utf-8&word=${encodeURIComponent(
      word
    )}`
  )
  .set('Accept', header['Accept'])
  .set('Accept-Encoding', header['Accept-Encoding'])
  .set('Accept-Language', header['Accept-Language'])
  .set('Cache-Control', header['Cache-Control'])
  .set('Connection', header['Connection'])
  .set('User-Agent', header['User-Agent'])
  .end((err, res) => {
    if (err) {
      console.err('访问失败，原因: ', err);
      return;
    }

    const htmlText = res.text;
    const $ = cheerio.load(htmlText);

    console.log(htmlText);
  });

```

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

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

再通过正则只获取链接

JavaScript | 复制代码

```
superagent
  .get(
    `http://image.baidu.com/search/index?tn=baiduimage&ie=utf-8&word=${encodeURIComponent(
      word
    )}`
  )
  .set('Accept', header['Accept'])
  .set('Accept-Encoding', header['Accept-Encoding'])
  .set('Accept-Language', header['Accept-Language'])
  .set('Cache-Control', header['Cache-Control'])
  .set('Connection', header['Connection'])
  .set('User-Agent', header['User-Agent'])
  .end((err, res) => {
    if (err) {
      console.err('访问失败，原因: ', err);
      return;
    }

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

    //
'"objURL":"https://gimg2.baidu.com/image_search/src=http%3A%2F%2Fimg9.51tietu.net%2Fpic%2F2019-091304%2Fv451j2jqusqv451j2jqusq.jpg&refer=http%3A%2F%2Fimg9.51tietu.net&app=2002&size=f9999,10000&q=a80&n=0&g=0n&fmt=auto?sec=1659950528&t=c082067e049cfa937a180c3e3eae8ec1",',

    const imageUrlList = imageMatches.map(item => {
      const imageUrl = item.match(/:"(.*?)"/g);
      return RegExp.$1;
    });

    console.log(imageUrlList);
  });
```

5.5. 获取图片的标题

根据页面source获取图片名称，为：fromPageTitle

JavaScript | 复制代码

```
superagent
  .get(
    `http://image.baidu.com/search/index?tn=baiduimage&ie=utf-8&word=${encodeURIComponent(
      word
    )}`
  )
  .set('Accept', header['Accept'])
  .set('Accept-Encoding', header['Accept-Encoding'])
  .set('Accept-Language', header['Accept-Language'])
  .set('Cache-Control', header['Cache-Control'])
  .set('Connection', header['Connection'])
  .set('User-Agent', header['User-Agent'])
  .end((err, res) => {
    if (err) {
      console.err('访问失败，原因: ', err);
      return;
    }

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

    //
'"objURL":"https://gimg2.baidu.com/image_search/src=http%3A%2F%2Fimg9.51tietu.net%2Fpic%2F2019-091304%2Fv451j2jqusqv451j2jqusq.jpg&refer=http%3A%2F%2Fimg9.51tietu.net&app=2002&size=f9999,10000&q=a80&n=0&g=0n&fmt=auto?sec=1659950528&t=c082067e049cfa937a180c3e3eae8ec1",',

    const imageUrlList = imageMatches.map(item => {
      const imageUrl = item.match(/:"(.*?)"/g);
      return RegExp.$1;
    });

    const titleMatches = htmlText.match(/"fromPageTitle":"(.*?)",/g);

    const titleList = titleMatches.map(item => {
      const title = item.match(/:"(.*?)"/g);
      return RegExp.$1;
    });

    console.log(imageUrlList, titleList);
  });

```

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

```
function getValueListByReg(str, key) {
  const reg = new RegExp(`"${key}":"(.*?)"`, 'g');
  const matchResult = str.match(reg);

  const resultList = matchResult.map(item => {
    const result = item.match(/:"(.*?)"/g);
    return RegExp.$1;
  });

  return resultList;
}
```

5.7. 创建目录，存储图片

JavaScript | 复制代码

```javascript
function mkImageDir(pathname) {
  const fullPath = path.resolve(__dirname, pathname);

  if (fs.existsSync(fullPath)) {
    console.log(`${pathname}已存在，跳过此步骤`);
    return;
  }

  fs.mkdirSync(fullPath);
  console.log(`目录创建成功！目录为：${pathname}`);
}

superagent
  .get(
    `http://image.baidu.com/search/index?tn=baiduimage&ie=utf-8&word=${encodeURIComponent(
      word
    )}`
  )
  .set('Accept', header['Accept'])
  .set('Accept-Encoding', header['Accept-Encoding'])
  .set('Accept-Language', header['Accept-Language'])
  .set('Cache-Control', header['Cache-Control'])
  .set('Connection', header['Connection'])
  .set('User-Agent', header['User-Agent'])
  .end((err, res) => {
    if (err) {
      console.err('访问失败，原因: ', err);
      return;
    }

    const htmlText = res.text;

    const imageUrlList = getValueListByReg(htmlText, 'objURL');
    const titleList = getValueListByReg(htmlText, 'fromPageTitle').map(item =>
      item.replace('<strong>', '').replace('<\\/strong>', '')
    );

    mkImageDir('images');
  });

```

5.8. 下载图片到本地

JavaScript | 复制代码

```javascript
function downloadImage(url, name, index) {
  const fullPath = path.join(__dirname, 'images', `${index + 1}.${name.replace('?', '')}.png`);

  if (fs.existsSync(fullPath)) {
    console.log(`已存在，${fullPath}`);
    return;
  }

  superagent.get(url).end((err, res) => {
    if (err) {
      console.log(err, `获取链接出错，内容为：${res}`);
      return;
    }

    if (JSON.stringify(res.body) === '{}') {
      console.log(`第${index + 1}图片内容为空`);
      return;
    }

    fs.writeFile(fullPath, res.body, 'binary', err => {
      if (err) {
        console.log(`第${index + 1}张图片下载失败：${err}`);
        return;
      }
      console.log(`第${index + 1}张图片下载成功：${url}`);
    });
  });
}

superagent
  .get(
    `http://image.baidu.com/search/index?tn=baiduimage&ie=utf-8&word=${encodeURIComponent(
      word
    )}`
  )
  .set('Accept', header['Accept'])
  .set('Accept-Encoding', header['Accept-Encoding'])
  .set('Accept-Language', header['Accept-Language'])
  .set('Cache-Control', header['Cache-Control'])
  .set('Connection', header['Connection'])
  .set('User-Agent', header['User-Agent'])
  .end(async (err, res) => {
    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

JavaScript | 复制代码

```
const cliProgress = require('cli-progress');

const bar = new cliProgress.SingleBar(
  {
    clearOnComplete: false,
  },
  cliProgress.Presets.shades_classic
);

superagent
  .get(
    `http://image.baidu.com/search/index?tn=baiduimage&ie=utf-8&word=${encodeURIComponent(
      word
    )}`
  )
  .set('Accept', header['Accept'])
  .set('Accept-Encoding', header['Accept-Encoding'])
  .set('Accept-Language', header['Accept-Language'])
  .set('Cache-Control', header['Cache-Control'])
  .set('Connection', header['Connection'])
  .set('User-Agent', header['User-Agent'])
  .end(async (err, res) => {
    if (err) {
      console.err('访问失败，原因: ', err);
      return;
    }

    const htmlText = res.text;

    const imageUrlList = getValueListByReg(htmlText, 'objURL');
    const titleList = getValueListByReg(htmlText, 'fromPageTitle').map(item =>
      item.replace('<strong>', '').replace('<\\/strong>', '')
    );

    try {
      await mkImageDir('images');
      bar.start(imageUrlList.length, 0);

      imageUrlList.forEach((url, index) => {
        downloadImage(url, titleList[index], index)
          .then(() => {
            finished++;
            bar.update(finished);
```

```
            })
            .then(() => {
              if (finished === imageUrlList.length) {
                bar.stop();
                console.log('恭喜，图片下载完成！');
              }
            });
        });
      } catch (e) {
        console.log(e);
      }
    });

```

5.10. 删除文件

避免每次判断有文件存在时退出执行

JavaScript | 复制代码

```javascript
function mkImageDir(pathname) {
  return new Promise((resolve, reject) => {
    const fullPath = path.resolve(__dirname, pathname);

    if (fs.existsSync(fullPath)) {
      removeDir(pathname);
    }

    fs.mkdirSync(fullPath);
    console.log(`目录创建成功！目录为：${pathname}`);
    return resolve();
  });
}

function removeDir(pathname) {
  const fullPath = path.resolve(__dirname, pathname);
  console.log(`${pathname}目录已存在，准备执行删除`);

  fs.rmdirSync(fullPath, {
    force: true,
    recursive: true,
  });
  console.log(`目录${pathname}已删除！`);

  // const process = require('child_process');
  // process.execSync(`rm -rf ${fullPath}`);
}


```

5.11. 使用CLI输入关键词

在img.handle.js中，将superagent export出去

JavaScript | 复制代码

```javascript
#!/usr/bin/env node

const inquirer = require('inquirer');
const commander = require('commander');

const { runImg } = require('./img.handler.js');

const question = [
  {
    type: 'checkbox',
    name: 'channels',
    message: '请选择想要搜索的渠道',
    choices: [
      {
        name: '百度图片',
        value: 'images',
      },
      {
        name: '百度视频',
        value: 'videos',
      },
    ],
  },
  {
    type: 'input',
    name: 'keyword',
    message: '请输入想要搜索的关键词',
  },
];

inquirer.prompt(question).then(result => {
  const { keyword, channels } = result;

  for (let channel of channels) {
    switch (channel) {
      case 'images':
        runImg(keyword);
        break;
    }
  }
});

```

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=&word=%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

JavaScript 复制代码

```
// index.js
{
  type: 'number',
    name: 'counts',
      message: '请输入要下载的图片张数（最小30张）',
},

inquirer.prompt(question).then(result => {
  const { keyword, channels, counts } = result;

  for (let channel of channels) {
    switch (channel) {
      case 'images':
        runImg(keyword, counts);
        break;
    }
  }
});
```

JavaScript | 复制代码

```javascript
const superagent = require('superagent');
const cheerio = require('cheerio');
const path = require('path');
const fs = require('fs');
const cliProgress = require('cli-progress');

const bar = new cliProgress.SingleBar(
  {
    clearOnComplete: false,
  },
  cliProgress.Presets.shades_classic
);

let total = 0;
let finished = 0;

const header = {
  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',
  'Accept-Encoding': 'gzip, deflate, br',
  Accept2: 'text/plain, */*; q=0.01',
  'Accept-Language': 'zh-CN,zh;q=0.9',
  'Cache-Control': 'max-age=0',
  Connection: 'keep-alive',
  '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',
  'sec-ch-ua': '".Not/A)Brand";v="99", "Google Chrome";v="103", "Chromium";v="103"',
};

function getValueListByReg(str, key) {
  const reg = new RegExp(`"${key}":"(.*?)"`, 'g');
  const matchResult = str.match(reg);

  const resultList = matchResult.map(item => {
    const result = item.match(/:"(.*?)"/g);
    return RegExp.$1;
  });

  return resultList;
}

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
```

```
      fs.writeFile(fullPath, res.body, 'binary', err => {
        if (err) {
          return reject(`第${index + 1}张图片下载失败: ${err}`);
        }
        return resolve(`第${index + 1}张图片下载成功: ${url}`);
      });
    });
  });
}

function request(url, acceptKey = 'Accept') {
  return new Promise((resolve, reject) => {
    superagent
      .get(url)
      .set('Accept', header[acceptKey])
      .set('Accept-Encoding', header['Accept-Encoding'])
      .set('Accept-Language', header['Accept-Language'])
      .set('Cache-Control', header['Cache-Control'])
      .set('Connection', header['Connection'])
      .set('User-Agent', header['User-Agent'])
      .end(async (err, res) => {
        if (err) {
          reject('访问失败，原因: ', err);
          return;
        }

        resolve(res);
      });
  });
}

async function getImageByPage(start, total, word) {
  let allImages = [];

  while (start < total) {
    const size = Math.min(60, total - start);
    const res = await request(
      `https://image.baidu.com/search/acjson?
tn=resultjson_com&word=${encodeURIComponent(
        word
      )}&queryWord=${encodeURIComponent(
        word
      )}&ie=utf-8&oe=utf-8&pn=${start}&rm=${size}&${Date.now()}=`,
      'Accept2'
    );

    allImages = allImages.concat(JSON.parse(res.text).data);
    start += size;
```

```
  }

  return allImages;
}

function runImg(keyword, counts) {
  request(
    `http://image.baidu.com/search/index?tn=baiduimage&ie=utf-8&word=${encodeURIComponent(
      keyword
    )}`
  ).then(async res => {
    const htmlText = res.text;

    const imageUrlList = getValueListByReg(htmlText, 'objURL');
    const titleList = getValueListByReg(htmlText, 'fromPageTitle').map(item =>
      item.replace('<strong>', '').replace('<\\/strong>', '')
    );

    let allImageUrls = imageUrlList.map((imageUrl, index) => {
      return {
        imageUrl,
        title: titleList[index],
      };
    });

    const firstPageCount = allImageUrls.length;

    if (counts > firstPageCount) {
      const restImgUrls = await getImageByPage(firstPageCount, counts, keyword);
      const formatImgUrls = restImgUrls
        .filter(item => item.middleURL)
        .map(item => {
          return {
            imageUrl: item.middleURL,
            title: item.fromPageTitle.replace('<strong>', '').replace('</strong>', ''),
          };
        });

      allImageUrls = allImageUrls.concat(formatImgUrls);
    }

    total = allImageUrls.length;

    try {
```

```
        await mkImageDir('images');
        bar.start(total, 0);

        allImageUrls.forEach((item, index) => {
          downloadImage(item.imageUrl, allImageUrls[index].title, index)
            .then(() => {
              finished++;
              bar.update(finished);
            })
            .then(() => {
              if (finished === total) {
                bar.stop();
                console.log('恭喜，图片下载完成！');
              }
            });
        });
      } catch (e) {
        console.log(e);
      }
    });
  }

  module.exports = {
    runImg,
  };

```