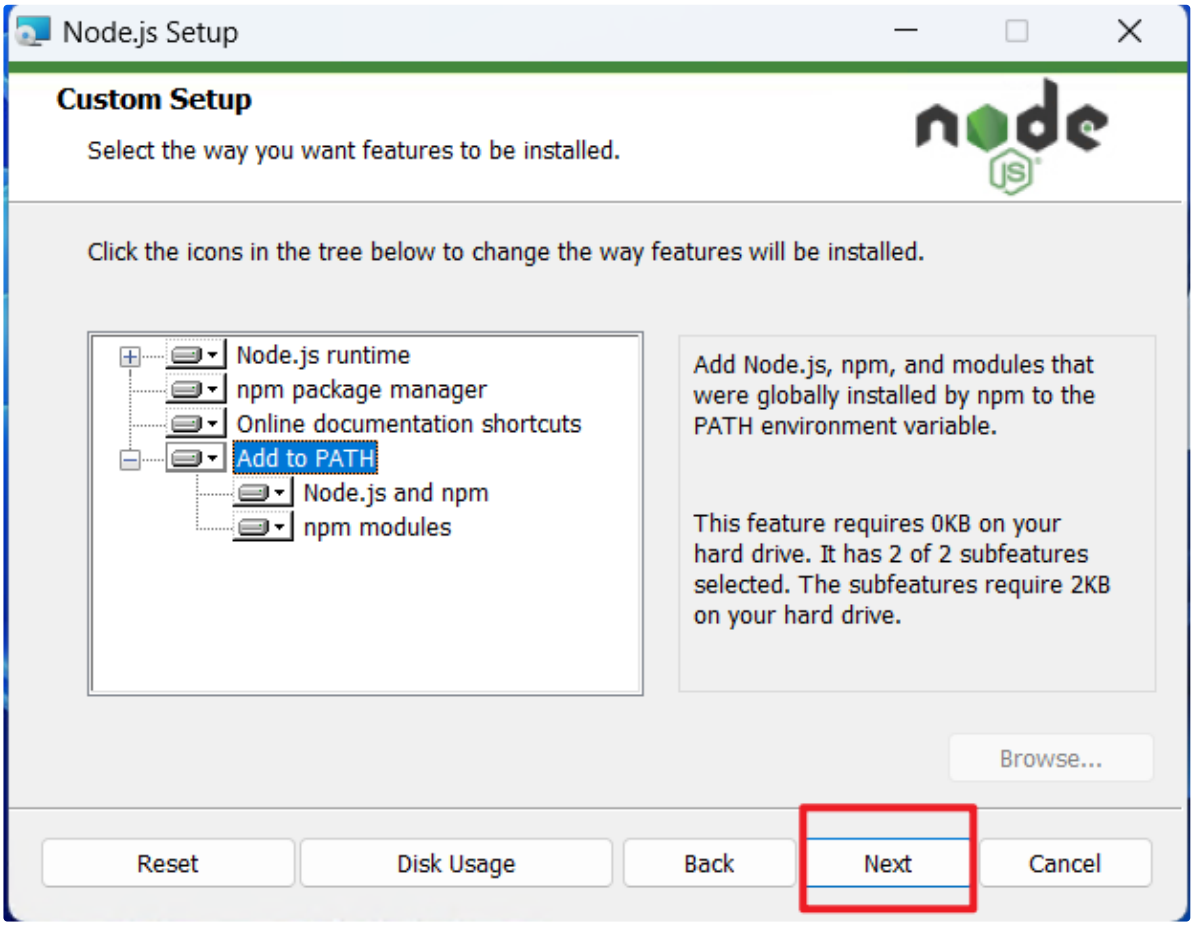
nodejs第三方模块

之前我们所使用的path,fs等模块都是nodejs提供给我们的内置模块 ，可以方便我们直接使用。其实nodejs还有大量的第三模块来给我们使用，如进行数据请求我们使用axios，如果抓包我们要使用cheerio，如连接数据库我们要使用mysql/mysql2，如使用MVC开发我们就要使用express,KOA,nest.js等

第三方模块有一个专门的管理员工具，叫npm，在安装node.js的时候就已经安装了，如下图所示



这个东西是专门用于对第三方的模块进行管理的

1. npm理解为network package manager网络包管理工具
2. npm理解为node.js package manager node.js的包管理工具

npm工具

> nodejs是以文件夹为单位来管理项目的，所以同学们在学习node.js的时候不要在vscode下面同时打开多个项目文件夹

nodejs是以文件夹为单位来创建项目的，也是来管理项目的，那么nodejs是怎么样就认为这个文件夹就是一个项目呢

npm初始化

nodejs如果想以某一个文件夹为单位来管理项目 ，一定要对这个文件夹（项目）进行初始化，初始化以后，nodejs就认这个文件夹就是一个项目了

初始化的命令

```
$ npm init
```

当执行完一系列的操作以后，我们就可以看到在当前的文件夹下面会多出一个 package.json 的文件， 这个文件记录了你初始化的时候一些信息

```
{
  "name": "100706",
  "version": "1.0.0",
  "description": "标哥哥的项目",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [
    "管理系统",
    "标哥",
    "第一个项目"
  ],
  "author": "杨标",
  "license": "ISC"
}
```

> 当一个文件夹的下面有了 package.json 以后，系统不会认为这个文件夹是一个node.js的项目，不再仅仅只是一个文件夹了

package.json介绍

这个文件记录了当前项目的基本信息

- name 代表当前包的名称，也就是nodejs项目的名称
- version 代表当前项目的版本
- description 代表当前项目的描述信息
- main 代表当前项目的入口文件，可以告诉系统当前的项目从哪个文件开始启动
- scripts 脚本配置，后期我们可以通过这个东西来实现项目的快速启动
- keywords 项目的关键字，后期我们把项目发布到网上去以后别人可以通过这个关键字来搜索这个项目
- author 项目的作者
- license 版权

如果我们在初始化项目的时候希望快速的生成 package.json 文件，可以直接使用下面的命令

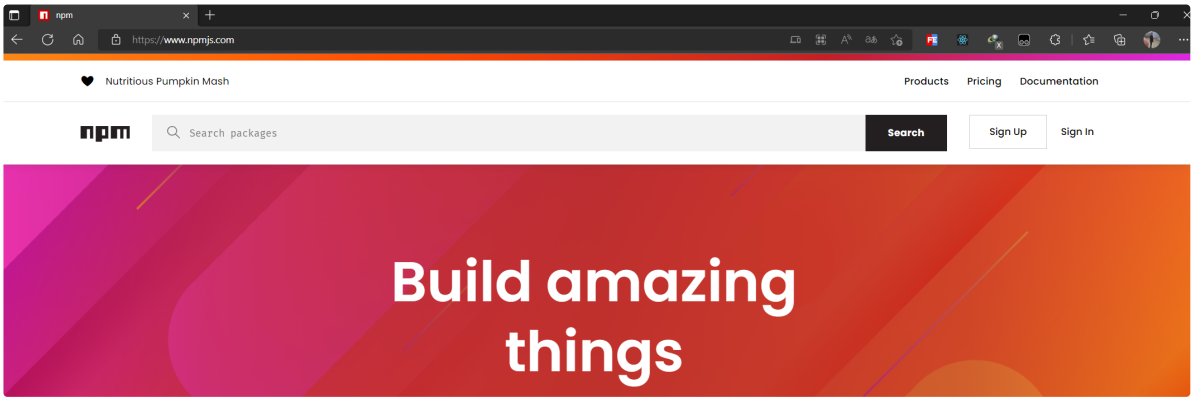
```
$ npm init --yes
```

它会直接以默认的形式帮我们生成 package.json ，后期我们可以再去修改这个文件

npmjs远端仓库

有一个网站，上面保留了我们所有的第三方模块，这个网站就是

npm （npmjs.com）



> 我们可以根据自己的需要求在这个网站找到适合于自己的第三方模块 ，这个东西相当于java里面的maven

npm info查看包的信息

如果我们想从远程服务器npmjs上面下载某一个包，应该先查看一下有没有这个包，并且了解一下这个包的相关信息

```
$ npm info 包名称
```

如我们输入下面的命令就可以查看 jQuery的信息

```
$ npm info jquery
```

```
YangBiao@YB-Huawei D:\杨标的工作文件\班级教学笔记\H2204\1007\code\100706 ❱❱❱ npm info jquery

jquery@3.6.1 | MIT | deps: none | versions: 54
JavaScript library for DOM operations
https://jquery.com

keywords: jquery, javascript, browser, library

dist
.tarball: https://registry.npmmirror.com/jquery/-/jquery-3.6.1.tgz
.shasum: fab0408f8b45fc19f956205773b62b292c147a16
.integrity: sha512-opJeO4nCucVnsjiXOE+/PcCgYw9Gwpvs/a6B1LL/lQhwWwpbVEVYDZ1FokFr8PRc7ghYlrFPuyHuiiDNTQxmcw==
.unpackedSize: 1.3 MB

maintainers:
- openjsfoundation <npm@openjsf.org>
- dmethvin <dave.methvin@gmail.com>
- timmywil <4timmywil@gmail.com>
- mgol <m.goleb@gmail.com>

dist-tags:
beta: 3.6.1    latest: 3.6.1

published a month ago by timmywil <4timmywil@gmail.com>
```

npm install安装包

当我们通过npm info查看了某一个包的信息以后，我们现在想把这个包下载下来，怎么办呢?

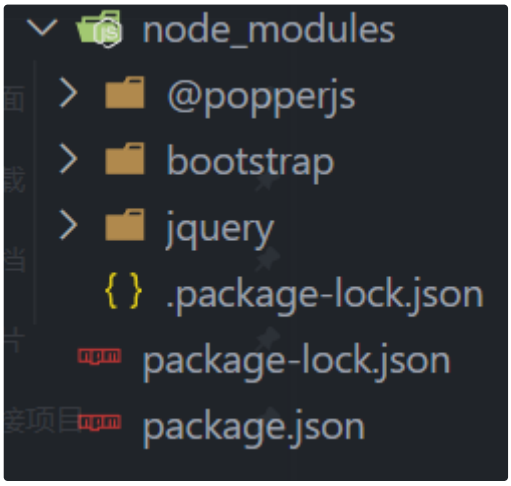
```
$ npm install 包名称
```

如

```
$ npm install jquery
```

上面的代码就下载了jquery的安装包，如果我们还想安装其它的包也一样的

```
$ npm install bootstrap
```



这个时候我们可以的看到，当某一个安装成功以就会在node_modules的目录下面显示出来

每次下载完了以后，它会把安装信息记录在 package.json 里面

--save记录安装信息

在上面我们通过 npm install 安装某一个包的时候，这个包从服务器下载到了本地项目，后期我们的项目会越来越大，第三方的包也会越来越多，会了方便管理这些方，记录你的安装信息，我们一般在安装第三方模块的时候，都要添加一个 --save 的参数，如下所示

```
$ npm install jquery --save
```

在上面的命令里面，我们可以看到在安装的命令后面，我们添加了一个参数 --save ，这代表从服务器安装 jquery 第三模块 ，并且把这次的安装信息记录在 package.json 里面，这样其它的同事看到这个项目以后它就知道这个项目依赖于哪些第三方的包了

> npm的版本如果大于6及以上，则默认可以不添加这个 --save 了，大于6的npm默认会自动携带这个 --save

--save-dev记录安装信息

这一种情况和上一种很像 ，只是它的安装信息会记录在一个devDependencies里面， 这个叫开发依赖（在后面的项目当中讲解开发依赖）

```
$ npm install bootstrap --save-dev
```

这个时候也会记录安装信息，结果如下

```
{
    "name": "100706",
    "version": "1.0.0",
    "description": "标哥哥讲课的项目",
    "main": "index.js",
    "scripts": {
        "test": "echo \"Error: no test specified\" && exit 1"
    },
    "keywords": [
        "标哥",
        "H2204"
    ],
    "author": "杨标",
    "license": "ISC",
    "dependencies": {
        "jquery": "^3.6.1"
    },
    "devDependencies": {
        "bootstrap": "^5.2.2"
    }
}
```

- dependencies代表生产依赖
- devDependencies代表开发依赖

> 开发依赖与依赖的区别?
>
> 举例：标哥去饭店吃饭，点了一个蛋炒饭
>
> 老板为了完成这个蛋炒饭，它需要什么东西?
>
> 1. 米饭【生产依赖】
> 2. 蛋，盐，酱油，油等配料【生产依赖】
> 3. 锅，锅铲，灶等一系列的工具【开发依赖】

为什么要记录安装包的信息

1. 后期的项目为越来越大，第三方包也会越用越多，为了更方全管理自己的包，我们就需要记录这些包的信息
2. 每个项目都会下载第三方的依赖，但是npm会根据当前的系统再自动选择安装包的版本，如苹果的系统就下载苹果的依赖，linux的系统就下载linux的依赖，window10的系统就下载windows10的依赖，还会根据不同的node版本来下载依赖

这个时候就会出现一个问题，我的项目如果发给别人以后，别人要启动，但是它的系统跟不致，这个时候的node_modules里面所存放的第三方依赖包就不通用，会报错，

所以我们在传递项目给别人的时候，都会把node_modules的文件删除，别人打开项目就没有这个依赖，跑不起来

它就需要查看 package.json 的文件，看一下里面是否有记录安装包的信息，如果有，则只需要输入下面的命令就可以了

```
$ npm install
```

这个命令会自动读取当前文件夹下面的 package.json ，再去读取里面记录的依赖包的信息，然后批量下载

npm uninstall卸载包

安装完成一个包以后如果想再把这个包把它卸载掉，那么，我们就可以使用这个命令

```
$ npm uninstall  包名称
```

当卸载了某一个包以后 node_modules 里面就不会再有这个包了，同时 package.json 里面也不会有这个记录信息了

npm install指定包的版本

当我们使用 npm install 去安装一个包的时候 ，它默认情况下是安装的最新的版本，如果我们想使用某一个特殊的版本，我们可以使用下面的命名

```
$ npm install 包名称@版本号
```

如

```
$ npm install bootstrap@3.3.7
```

批量装包

如果假设有多个包需要我们去安装，可以直接批量进行

```
$ npm install 包名称1 包名称2...
```

如

```
$ npm install jquery bootstrap vue
```

npm install的简化命令

npm install 是安装一个包，这也有简化的命令

```
$ npm install 包名
```

简写成

```
    $ npm i
```

npm uninstall简化命令

npm uninstall是卸载一个包，这也有简化的命令

```
$ npm uninstall vue
```

简写

```
$ npm un vue
```

---

npm命令总结

命令	说明
npm init	初始化
npm init --yes	以默认方式初始化
npm info 包名	查看某个包的信息
npm install 包名	安装一个指定的包
npm install 包名 --save	记录到生产依赖
npm install 包名 --save-dev	记录到开发依赖
npm install 包名@版本号	安装指定的版本号
npm uninstall 包名	卸载指定的包
npm install	根据package.json来自动装包
npm i 包名	简化版的安装包的命令
npm un 包名	简化版的卸载包的命名

---

npm国内镜像设置

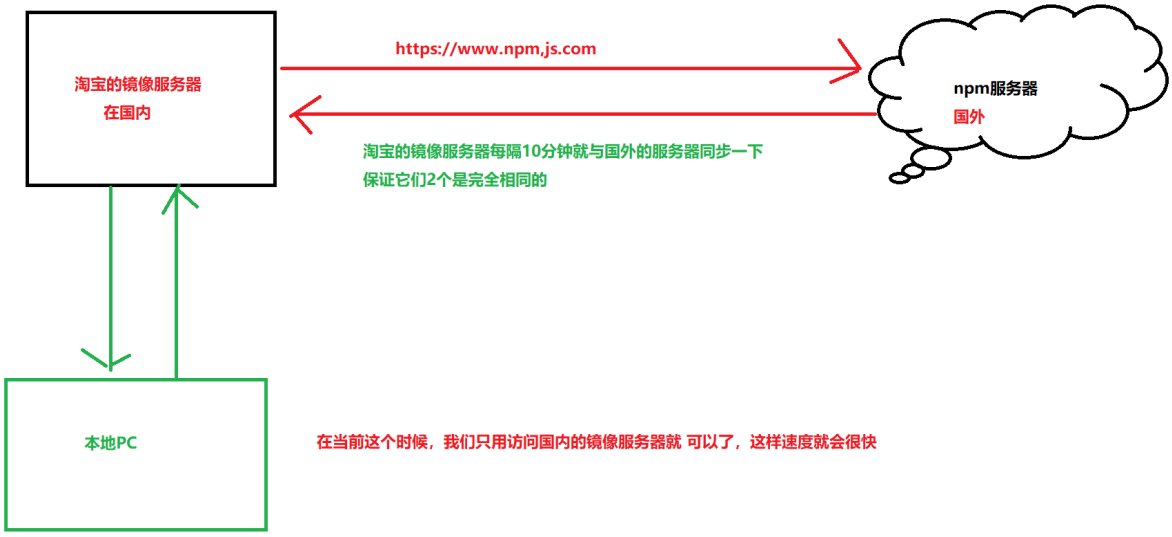
npm的服务器是在国外的，所以我们每次去下载的时候都会访问国外的服务器，这样非常慢

每次下载一个包我们都要从国外的服务器去下载，但是因为一些不可描述的原因，我们访问国外的网站会很慢，特别是后期有些包在github上面的时候更慢。针对这个问题，目前的解决方案有很多，我们大家推荐几种

1. 更改npm的镜像地址
2. 使用国内的包管理工具cnpm

3．使用yarn包管理工具

目前最简单单的就是使用 npm 的镜像地址来完成




```
$ npm config set registry https://registry.npm.taobao.org
$ npm config set disturl https://npm.taobao.org/dist
$ npm config set electron_mirror https://npm.taobao.org/mirrors/electron/
$ npm config set sass_binary_site https://npm.taobao.org/mirrors/node-sass/
$ npm config set phantomjs_cdnurl https://npm.taobao.org/mirrors/phantomjs/
```

在自己的电脑上面，以管理员的身份打开控制台

使用axios+cheerio来完成抓包

如果想完成抓包，我们主要使用的就是下面2个包

1. axios 它是一个跨平台的用于发起http请求的一个包，可以在node.js的平台使用，也可以在浏览器下面使用
2. cheerio 这个包用于分析HTML的网页，提取网页当相关的信息

模块网络请求

想要更好的模块网络请求，我们需要使用一个包，这个包就是 axios

```
const axios = require('axios');

const catchData = async () => {
    console.log("开始请求.....");
    let str = "https://v.qq.com/channel/tv?listpage=1&channel=tv&feature=2";
    // 发起请求，等结果
    try {
        let resp = await axios.get(str,{
            headers:{

 accept:"text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9",
```

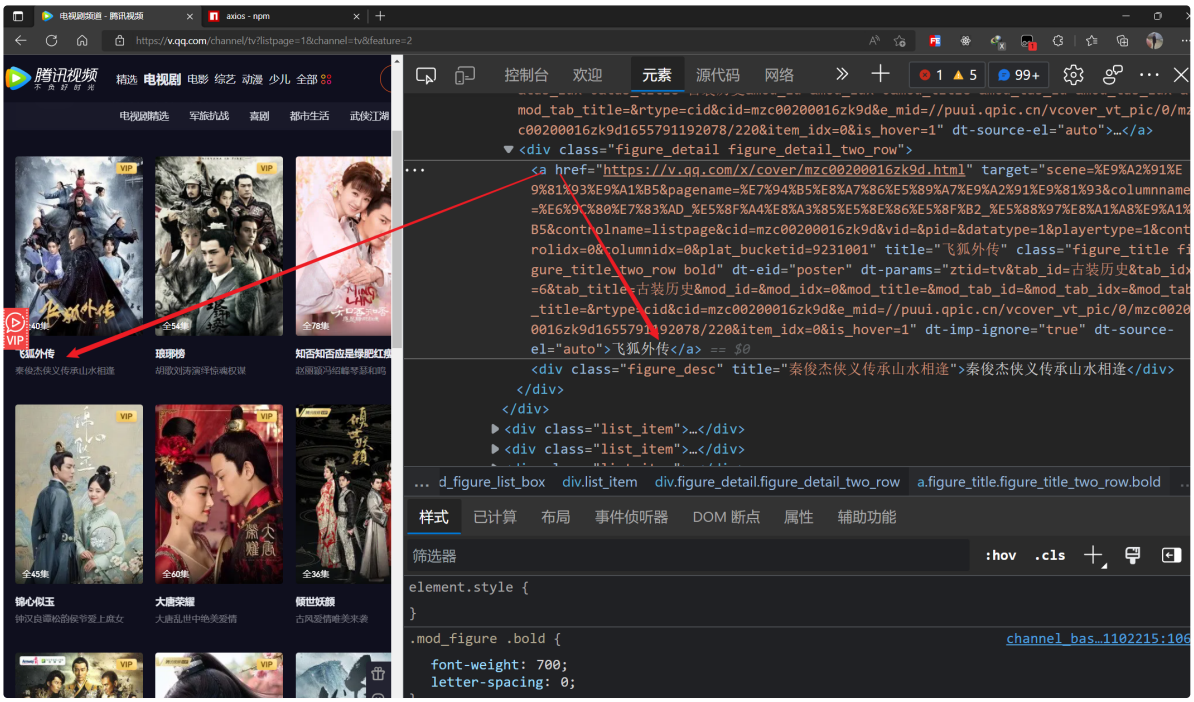
```
            cookie:"RK=0K+NQJHxWO;
ptcz=fb997511491c9ca0a5c7237f76ccc260449f28c806461ac45266c44285bf212b;
fqm_pvqid=671ce23e-3378-4520-918e-539cdde307bd; pac_uid=1_365055754;
iip=0; o_cookie=365055754; tvfe_boss_uuid=f6743d841fc5508a;
ts_uid=2356158233; Qs_lvt_323937=1648900734%2C1648900905%2C1654258098;
Qs_pv_323937=3912163211342407700%2C4490624458211209700%2C31778190449681520
00; video_guid=2ffe4b234d1926ad; compared_guid=582794d913223941;
ts_refer=www.baidu.com/s; tab_experiment_str=8752037#8826908;
ptui_loginuin=365055754@qq.com; pgv_pvid=6857396830; video_platform=2;
pgv_info=ssid=s6013868365; ts_last=v.qq.com/channel/tv; bucket_id=9231001;
ptag=|channel; qv_als=OOoH7obq+/JkA7y9A11665141916yoSxPw==;
ad_play_index=15"
            }
        });
        // 响应的结果在data里面
        console.log(resp.data)
    } catch (error) {
        console.log("报错了");
        console.log(error);
    }
}

catchData();
```

上面的代码就是通过 axios 请求以后得到的结果， resp.data 就是结果，我们把结果打印了一下，发现这个结果就是通过上面的网址返回的网页内容

得到的网页的内容以后，我们就要知道我们要得到哪些数据

分析得到的结果



如果我想得到所有的电影的名称，我应该找到一个 a 标签 ，然后去它下面找名称

为了更好的解析网页上面的内容，我们需要借用一个第三方的包 cheerio

安装

```
npm install cheerio
```

特征

❤ 熟悉的语法： Cheerio 实现了核心 jQuery 的一个子集。Cheerio从jQuery库中删除了所有DOM不一致和浏览器问题，揭示了其真正华丽的API。

ϟ 极快： Cheerio 使用非常简单、一致的 DOM 模型。因此，解析、操作和渲染非常高效。

✪ 令人难以置信的灵活性： Cheerio环绕解析器5解析器，并且可以选择使用@FB55宽容的html解析器2。Cheerio 几乎可以解析任何 HTML 或 XML 文档。

```
/**
 * 抓包
 */

const axios = require('axios');
// cheerio可以把我们抓取的网页像jQuery一样去操作
const cheerio = require('cheerio');
const path = require("path");
const fs = require("fs");


const catchData = async () => {
    console.log("开始请求......");
    let str = "https://v.qq.com/channel/tv?listpage=1&channel=tv&feature=2";
    // 发起请求，等结果
    try {
        let resp = await axios.get(str, {
            headers: {
                accept: "text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9",
                cookie: "RK=0K+NQJHxWO; ptcz=fb997511491c9ca0a5c7237f76ccc260449f28c806461ac45266c44285bf212b; fqm_pvqid=671ce23e-3378-4520-918e-539cdde307bd; pac_uid=1_365055754; iip=0; o_cookie=365055754; tvfe_boss_uuid=f6743d841fc5508a; ts_uid=2356158233; Qs_lvt_323937=1648900734%2C1648900905%2C1654258098; Qs_pv_323937=3912163211342407700%2C4490624458211209700%2C3177819044968152000; video_guid=2ffe4b234d1926ad; compared_guid=582794d913223941; ts_refer=www.baidu.com/s; tab_experiment_str=8752037#8826908; ptui_loginuin=365055754@qq.com; pgv_pvid=6857396830; video_platform=2; pgv_info=ssid=s6013868365; ts_last=v.qq.com/channel/tv; bucket_id=9231001; ptag=|channel; qv_als=OOoH7obq+/JkA7y9A11665141916yoSxPw==; ad_play_index=15"
```

```
            }
        });
        // 响应的结果在data里面
        // console.log(resp.data);
        //开始使用cheerio来分析网页，并加载成$对象
        let $ = cheerio.load(resp.data);
        let result = [];
        $(".list_item").each((index, ele) => {
            let movieName = $(ele).find(".figure_title_two_row").text();
            let movieDesc = $(ele).find(".figure_desc").text();
            let movieCaption = $(ele).find(".figure_caption").text();
            let moviePic = $(ele).find(".figure_pic").attr("src");
            let obj = {
                movieName,
                movieDesc,
                movieCaption,
                moviePic
            }
            result.push(obj);
        });

        let resultJsonStr = JSON.stringify(result);
        fs.writeFileSync(path.join(__dirname, "./data.txt"),
resultJsonStr);
        console.log("数据保存成功");
    } catch (error) {
        console.log("报错了");
        console.log(error);
    }
}

catchData();
```

抓取图片

在上面的代码里面，我们可以看到，我们已经把所有的数据组合成了一个对象 ，现在我们就要把这个图片下载下来，怎么办呢?

当我们得到了上面的信息以后，我们其实就已经完成了50%的功能了，现在我们在文件里面可以看到有一个属性叫 moviePic ，这个属性用于保存了图片地址，所以我们可以再次根据这个地址去抓取图片，把 它下载到本地

上图就是从服务器接收数据以后再转换成我们所需要的流就可以了

```
/**
 * 2022-10-08
 * YangBiao
 **/

const axios = require("axios");    //专门用于模拟请求
const cheerio = require("cheerio");
const path = require("path");
const fs = require("fs");

const catchData = async () => {
    console.log("正在抓数据...");
    //第一步：准备请求的url
    let url = `https://v.qq.com/channel/tv?listpage=1&channel=tv&feature=2`;
    let config = {
        headers: {
            accept: "text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9",
            cookie: "RK=0K+NQJHxWO; ptcz=fb997511491c9ca0a5c7237f76ccc260449f28c806461ac45266c44285bf212b; fqm_pvqid=671ce23e-3378-4520-918e-539cdde307bd; pac_uid=1_365055754; iip=0; o_cookie=365055754; tvfe_boss_uuid=f6743d841fc5508a; ts_uid=2356158233; Qs_lvt_323937=1648900734%2C1648900905%2C1654258098; Qs_pv_323937=3912163211342407700%2C4490624458211209700%2C31778190449681520 00; video_guid=2ffe4b234d1926ad; compared_guid=582794d913223941; ts_refer=www.baidu.com/s; tab_experiment_str=8752037#8826908; ptui_loginuin=365055754@qq.com; pgv_pvid=6857396830; video_platform=2; bucket_id=9231001; ad_play_index=18; pgv_info=ssid=s5465147712; ts_last=v.qq.com/channel/tv"
        }
```

```
    }
    //第二步：开始请求
    let resp = await axios.get(url, config);
    //第三步：加载网页为cheerio，空上就可以像jquery一样去操作选取了
    let $ = cheerio.load(resp.data);
    let result = [];
    $(".list_item").each((index, ele) => {
        //第四步：遍历选中的元素，组合成对象
        let movieName = $(ele).find(".figure_title_two_row").text();
        let movieDesc = $(ele).find(".figure_desc").text();
        let movieCaption = $(ele).find(".figure_caption").text();
        let moviePic = "http:" + $(ele).find(".figure_pic").attr("src");
        let obj = {
            movieName,
            movieDesc,
            movieCaption,
            moviePic
        }
        result.push(obj)
    });
    //我可以把那个数组result遍历一下，拿到里面的moviePic，去模拟请求这个地址，这样这个地址就会返回图片
    for (let {moviePic,movieName} of result) {
        //我们假设要把这个图片放在movieImage的目录下面去
        let movieImagePath = path.join(__dirname, "./movieImage");
        //如果没有这个目录，我们就创建这个目录
        if (!fs.existsSync(movieImagePath)) {
            fs.mkdirSync(movieImagePath);
        }
        //第五步：再次模拟请求，请求图片的地址
        let resp = await axios.get(moviePic, {
            ...config,
            //这个时候 ，请求的结果就会像水流一样
            responseType: "stream"
        });
        //第六步：接收到这个文件流以后，怎么办呢
        let imgPath = path.join(__dirname, "./movieImage", `${movieName}.png`);
        //第七步：将上面的图片地址变成一个可以写入的文件流
        let currentStream = fs.createWriteStream(imgPath);
        //第八步：将两个流对接
        resp.data.pipe(currentStream);
        console.log("图片保存成功");
    }
}
catchData();
```

在上面的代码里面，有两个技术主体

1. 在axios的请求当中 ，我们把响应的数据类型改为了stream
2. 我们使用文件流fs.createStream()来写入文件，这样会非常方便

使用nodemailer发送邮件

在我们以后开发的项目当中我们经常需要使用到邮件验证码，同时也会有些功能需要向用户发送邮件，批量发送等功能 ，这个时候我们就需要通过node.js来编程程序，让程序自动的发送邮件

如果要完成这个功能 我们可以使用第三方的模块叫nodemailer

安装包

```
$ npm install nodemailer --save
```

准备发送邮箱的账号与密码

目前我准备一个163的邮箱，其它的邮箱的设置也是一样的

如果要使用第三方客户端发送邮件，则要开启上面的服务

上面的pop3,smtp,imap就是用于发送邮件和接收邮件，同时还要注意服务器的地址

服务器地址：	POP3服务器: pop.163.com
	SMTP服务器: smtp.163.com
	IMAP服务器: imap.163.com
安全支持：	POP3/SMTP/IMAP服务全部支持SSL连接

- pop3是用于接收邮件的服务器
- smtp是用于发送邮件的服务器

协议类型	协议功能	服务器地址	非SSL端口号	SSL端口号
SMTP	发送邮件	smtp.163.com	25	465
POP	接收邮件	pop.163.com	110	995
IMAP	接收邮件	imap.163.com	143	993

```
/**
 * 2022-10-08
 * YangBiao
 **/

const nodemailer = require("nodemailer");
const path = require("path");

//编写一个函数，用于发送邮件
const sendMail = async () => {
    //第一步：创建一个邮件传输对象
    let passport = nodemailer.createTransport({
        host: "smtp.163.com",
        port: 465,
        auth: {
            user: "mh475201314@163.com",
            //这里填自己的
            pass: "BZVQLUJXHPPQJZQF"
        }
    });
    try {
        //第二步：发送邮件
        let result = await passport.sendMail({
            subject: "高清大图等你来看",
            from: "mh475201314@163.com",
            to: [
                "mh475201314@163.com",
                "173731044@qq.com",
                "2240933562@qq.com",
                "1803334091qq.com",
```

```
                "1635972779@163.com",
                "3256406860@qq.com",
                "1803334091@qq.com",
                "3160656756@qq.com",
                "2434893662@qq.com",
                "xxxiaohanxxx@163.com",
                "1871543651@qq.com",
                "121610681@qq.com",
                "2361229445@qq.com",
                "990715003@qq.com",
                "3089406860@qq.com",
                "1139746253@qq.com",
                "568545793@qq.com",
                "2677573347@qq.com"
            ],
            cc: "lovesnsfi@163.com",
            text: `
            亲爱的各们小伙伴：
                恭喜你！
                你已被本公司抽取为幸运观众，请凭验证码${~~(Math.random() * 10000)}到本公司领导假值19999的大礼一份，过时不候！
                同时为了更好的保证您能够顺利的领导本奖品，请您在领奖之前交纳1000元保证金，恭候您的光临。
            如有疑问，请电联
                18712345678


               XXX骗子公司

               ${new Date().toLocaleString()}
            `,
            //附件
            attachments: [
                {
                    filename: "标哥哥的帅气照片.jpg",
                    path: path.join(__dirname, "./img/2018上.jpg")
                }, {
                    filename: "美女的照片.jpg",
                    path: path.join(__dirname, "./img/w08.jpg")
                }
            ]
        });
        console.log("邮件发送成功");
        console.log(result);
    } catch (e) {
        console.log("邮件发送失败了");
        console.log(e);
    }
}

```

```
77 //调用发送邮件的方法
78 sendMail();
```

node-xlsx的使用

在我们平常的开发与使用当中，我们经常看到下面的场景

如果想实现这个场景，我们就要知道怎么样去实现通过node.js去操作excel

如果要使用nodejs来操作excel我们就要使用node-xlsx,这个包可以帮我们读取excel的文件 ， 也可以把一些数据生成excel

安装

```
1 $  npm install node-xlsx --save
```

读取excel文件

```
1  const xlsx = require("node-xlsx").default;
2  const path = require("path");
3  const fs = require("fs");
4
5
6  //读取 excel文件
7  const readExcel = () => {
8      //第一步：先构建excel文件的路径
9      let excelPath = path.join(__dirname, "./H2204信息.xls");
10     let result = xlsx.parse(excelPath);
11     console.log(result);
12 }
13
14 readExcel();
```

结果如下

```
1  [
2    {
3      name: 'Sheet1',
4      data: [
5        [Array], [Array], [Array], [Array], [Array],
6        [Array], [Array], [Array], [Array], [Array],
7        [Array], [Array], [Array], [Array], [Array],
8        [Array], [Array], [Array], [Array], [Array],
9        [Array], [Array], [Array], [Array], [Array],
10       [Array], [Array], [Array], [Array], [Array],
11       [Array], [Array], [Array], [Array], [Array],
```

```
12        [Array], [Array], [Array], [Array], [Array],
13        [Array], [Array], [Array], [Array], [Array],
14        [Array], [Array], [Array], [Array], [Array],
15        [Array], [Array], [Array], [Array], [Array],
16        [Array], [Array], [Array], [Array], [Array],
17        [Array], [Array], [Array], [Array], [Array]
18      ]
19    },
20    { name: 'Sheet2', data: [ [Array], [Array], [] ] }
21  ]
```

	姓名	性别	年龄	毕业院校	专业
2	何祥宇	男	25	武汉华夏理工学院	软件1196班
3	肖中	男	20	文华学院	机械设计制造及自动化
4	吴燃腾	女	23	湖北工程学院新技术学院	计算机科学与技术
5	贺锐	男	22	武昌理工学院	计科
6	方鹏	男	25	武汉东湖学院	软件工程
7	徐吉成	男	18	湖北工程学院新技术学院	计算机科学与技术
8	冉良超	男	18	武昌工学院	软件工程
9	陈韩家	男	22	武汉华夏理工学院	软件工程
10	刘诗霞	女	23	汉口学院	物联网工程
11	陈祈成	男	19	武昌理工学院	软件工程
12	程志超	男	22	武昌工学院	软件工程
13	徐书畅	男	23	武汉工程科技学院	计算机科学与技术
14	陈怡静	女	18	武汉华夏理工学院	软件工程
15	陈文号	男	20	湖北工程学院新技术学院	计算机科学与技术
16	栗秀峰	男	21	湖北工程学院新技术学院	软件工程
17	王宁娟	女	24	武汉华夏理工学院	软件工程
18	曹方	女	24	湖北工程学院新技术学院	计算机科学与技术
19	赵腾	男	23	武汉工程科技学院	计算机科学与技术
20	万金保	男	20	武汉华夏理工学院	软件工程
21	郝柏林	男	25	湖北工程学院新技术学院	计算机科学与技术
22	马湖圆	女	19	汉江师范学院	旅游英语
23	李康	男	22	武昌理工学院	软件工程
24	洪延军	男	18	武汉华夏理工学院	计算机科学与技术
25	王杜娉	女	20	湖北工程学院新技术学院	计算机科学与技术
26	王泽冰	男	22	武汉东湖学院	软件工程
27	叶俊豪	男	18	湖北师范大学	体育运营与管理
28	韩宏扬	男	24	武汉东湖学院	软件工程
29	吕振阳	男	21	武汉华夏理工学院	软件工程
30	陈润福	男	23	武汉工程大学邮电与信息工程学院	软件工程
31	隗志勇	男	19	武汉工程大学邮电与信息工程学院	软件工程（大数据与云计算方向）
32	陈文	男	23	湖北轻工职业技术学院	计算机网络基础

Sheet1

就绪 辅助功能: 不可用

当我们去打印第一个对象的时候 ， 可以看到下面结果

```
{
  name: 'Sheet1',
  data: [
    [ '姓名', '性别', '年龄', '毕业院校', '专业' ],
    [ '何祥宇', '男', 25, '武汉华夏理工学院', '软件1196班' ],
    [ '肖中', '男', 20, '文华学院', '机械设计制造及自动化' ],
    [ '吴燃腾', '女', 23, '湖北工程学院新技术学院', '计算机科学与技术
    [ '贺锐', '男', 22, '武昌理工学院', '计科' ],
    [ '方鹏', '男', 25, '武汉东湖学院', '软件工程' ],
    [ '徐吉成', '男', 18, '湖北工程学院新技术学院', '计算机科学与技术
    [ '冉良超', '男', 18, '武昌工学院', '软件工程' ],
    [ '陈韩家', '男', 22, '武汉华夏理工学院', '软件工程' ],
    [ '刘诗霞', '女', 23, '汉口学院', '物联网工程' ],
    [ '陈祈成', '男', 19, '武昌理工学院', '软件工程' ],
    [ '程志超', '男', 22, '武昌工学院', '软件工程' ],
    [ '徐书畅', '男', 23, '武汉工程科技学院', '计算机科学与技术' ],
    [ '陈怡静', '女', 18, '武汉华夏理工学院', '软件工程' ],
    [ '陈文昊', '男', 20, '湖北工程学院新技术学院', '计算机科学与技术
```

这个结果就是第一个工作表的结果

现在我们就可以读取里面数据，转换成JS所需要的对象或保存为一个JSON文件

```
/**
 * 2022-10-08
 * YangBiao
 **/
const xlsx = require("node-xlsx").default;
const path = require("path");
const fs = require("fs");


//读取 excel文件
const readExcel = () => {
    //第一步：先构建excel文件的路径
    let excelPath = path.join(__dirname, "./aaa.xlsx");
    //第二步：读取excel工作薄，一个工作薄里面会有多个工作表，我们现在只要第1个工作表
    let [result] = xlsx.parse(excelPath);
    //第一行是属性名
    let keys = result.data.shift();
    //现在开始遍历 ，组合成对象
    let arr = [];
    for(let item of result.data){
        let obj = {};
        for(let i in item){
            obj[keys[i]]=item[i];
        }
```

```
        arr.push(obj);
    }
    //写成JSON文件
    let jsonStr = JSON.stringify(arr);
    //定义一个保存的位置
    let savePath = path.join(__dirname,"./aaa.txt");
    fs.writeFileSync(savePath,jsonStr);
    console.log("保存成功");
}

readExcel();
```

生成excel文件

在上面的操作里面，我们是将一个excel文件读取出来，得到了我们想要的数据 ，那么我们如果将一些数据生成excel呢

```
const xlsx = require("node-xlsx").default;
const path = require("path");
const fs = require("fs");

//要求，现在有一个data.txt的文件 ，将这个文件 生成一个excel的文件
const writeExcel = () => {
    //第一步：先构建需要读取的文件的路径
    let p1 = path.join(__dirname, "./data.txt");
    //第二步：读这个文件,buff【buffer就是内存当中的一些数据】
    let buff = fs.readFileSync(p1);
    let jsonStr = buff.toString();
    //第三步：将json字符串转换成对象
    /** @type {Array}*/
    let arr = JSON.parse(jsonStr);
    //第四步：构建生成excel所需要的数据
    let excelObj = {
        name: "Sheet1",
        data: []
    }
    //第四步：先构建excel的第一行
    let firstRow = Object.keys(arr[0]);
    excelObj.data.push(firstRow);
    //第五步：开始构建第二行以后的东西
    for (let item of arr) {
        excelObj.data.push(Object.values(item));
    }
    //第六步：准备生成的excel的路径
    let excelSavePath = path.join(__dirname, "./data.xlsx");
    //第七步：开始生成,它会生成一个buff【buffer就是内存当中的一些数据】
    let excelBuff = xlsx.build([excelObj]);
    //第八步：将刚刚生成的buff写入到文件里面
    fs.writeFileSync(excelSavePath,excelBuff);
```

```
    console.log("生成成功");
}


writeExcel();
```

> 在上面的代码当中，最重要的一点就是构建要生成excel的数据，然后再将这个数据通过fs.writeFileSync写入到文件就可以了

总结

1. parse()用于读取excel的内容，它会返回一个数组，这个数组就是excel里面的数据
2. build()用于将一些数据生成excel所需要的buffer【buffer就是内存当中的一些数据】，然后我们再通过node.js里面的fs.writeFileSync()写到文件里面
3. 一个excel的工作簿(workbook)里面应该是有多个工作表(sheet)的

综合练习

1. 请抓取华夏学院新闻数据，网址如下

```
http://www.hxut.edu.cn/plus/list.php?tid=69&TotalResult=3682&PageNo=1
```

生成一个excel文件，里面要有新闻的标题newsTitle，要有时间newsTime，新闻的链接newsLink，新闻的作者newsAuthor，新闻的内容newsContent

当excel文件生成了以后，请将它发送到mh475201314@163.com的邮箱，以附件的形式发送，注明姓名是谁