NodeJS第三方模块

在建立nodejs的项目之前，它与我们之前建立HTML的网页项目是不一样的，它需要先初始化。所以在开始讲课之前，我有一个求：每一个nodejs的项目最好是一个单独的文件（因为nodejs是以文件夹为项目依据）

nodejs项目的初始化

nodejs在建立之前一定要先初始化，我们可以通过下面的命令来进行初始化

```
$ npm init
```

softeem · 杨标

> 以后要是在笔记里面或文档里面看到以`$`这个东西开始的，代表的是`dos`命令，不是代码

当我们在控制台输入上面的命令以后，根据提示信息输入内容，最后它会帮我们在当前的文件夹（项目）下面生成一个`package.json`的文件，这个文件里面就记录了我们当前这个项目的基本信息，如下

```
{
  "name": "013004",
  "version": "1.0.0",
  "description": "标哥的授课",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [
    "标哥，授课，抓包"
  ],
  "author": "标哥",
  "license": "ISC"
}
```

softeem · 杨标

> 1. name代表项目的名称，这个里面不要有中文，也不能使用驼峰命名
> 2. version代表当前项目的版本信息
> 3. description代表当前项目的描述信息
> 4. main代表当前项目的入口文件，也就是这个项目从什么地方开始启动
> 5. scripts代表当前项目可以执行哪些脚本，后期我们可以添加一些测试命令或启动命令
> 6. keywords代表关键字，这个项目主要的关键字是哪些
> 7. author代表当前项目的作者
> 8. license代表当前项目的版权信息

这个文件除了记录上在的这些基本信息以后，后期还可以记录我们项目里面的其它信息。所以我们在开始nodejs的项目之前，一定要先初始化项目，也就是生成上面的这个文件

如果想使用默认的配置来生成一个`package.json`我们可以直接使用下面的命令

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

softeem · 杨标

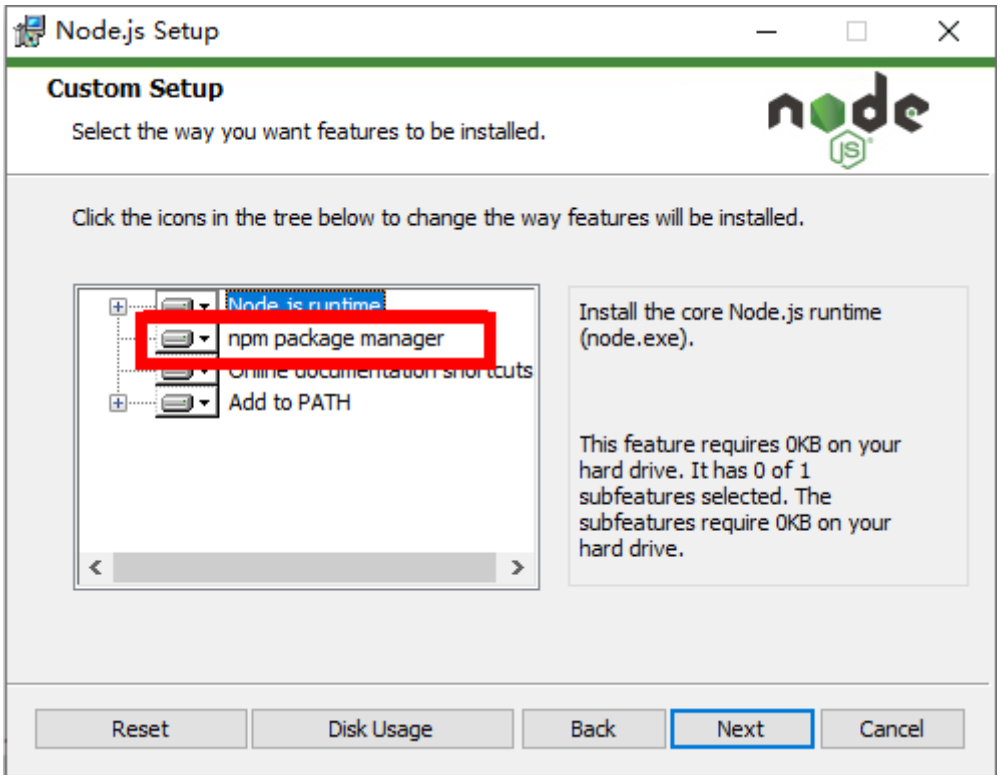
我们在后面添加了一个`--yes`这代表直接使用默认的值，不要询问

第三方模块的使用

我们现在已经初始化项目了，关键是我们怎么样使用第三方模块呢？

我们之前已经了解过怎么使用`nodejs`平台自带的模块，如`os/path/fs`这些都是系统内置的模块，那么第三方模块到底在哪里呢？

一般情况下，当我们在安装`nodejs`的环境的时候，默认会自带一个`npm`的管理工具，如下



这个`npm`全称叫**包管理管理工具**，可以理解为

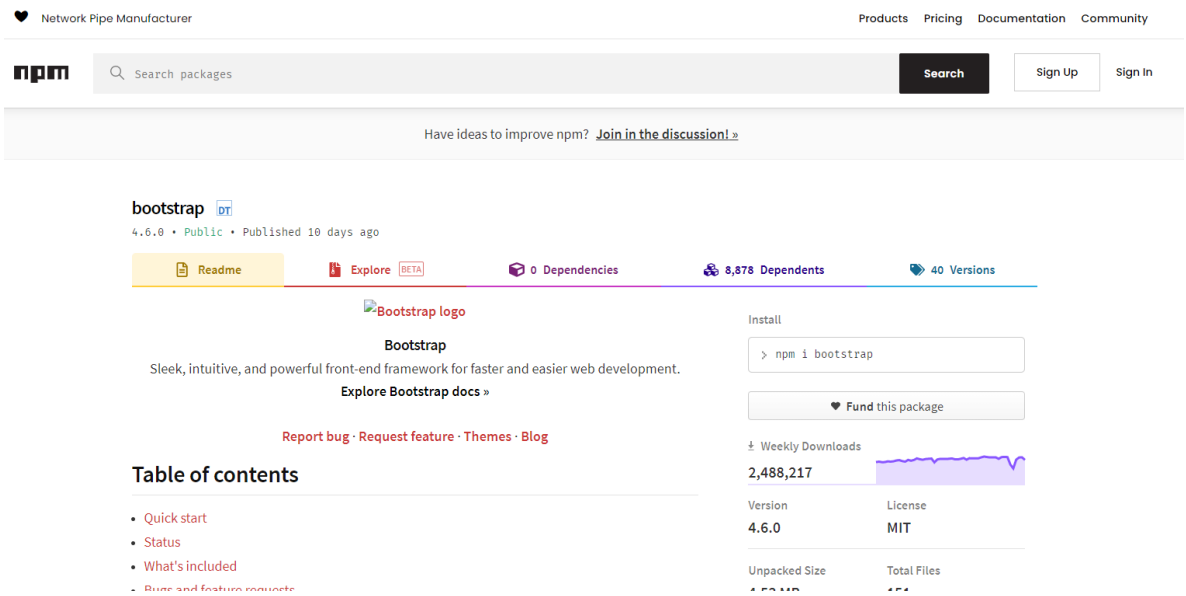
- `network package manager`：网络包管理工具
- `node module package package`：node平台的模块管理工具

这个npm主要就是用于管理nodejs平台下面的包的，也可以管理你自己的包，它自己有一系列的命令，例如我们可以使用它从网络上面下载一个包，也可以删除一包，也可以更新一个包等

关键一点就是这些包从哪里下载下来的

[npmjs官网：https://www.npmjs.com/](https://www.npmjs.com/)

我们首先可以从这个网站上面去搜索一下有没有自己需要的包的信息，如我们现在去搜索`bootstrap`，结果如下图



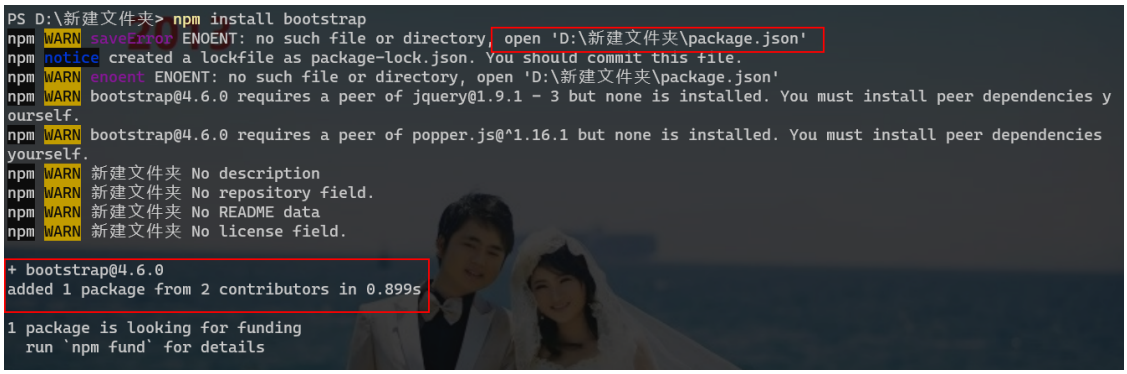
有了这个包的信息以后，我们就可以直接使用我们的`npm`去下载了

1. `npm info 包名`查看某些包的信息，如果这个包是存在的，则会显示这个包的信息，如果不存在，则没有信息

这个时候我们已经在控制台上面看到了这个包的相关信息，这个信息与我们之前在官方网站上面看到的信息是一样的

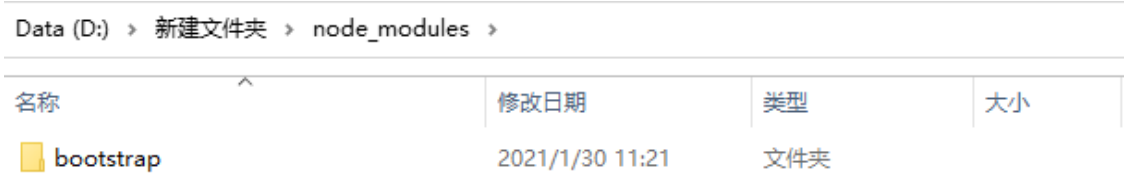
2. `npm install 包名`当我们去查询到这个包的信息以后，我们就可以使用这条命令，把这个包给下载下来

当我们执行完这个命令以后，它会有这么一个下载进度条，告诉我们下载多少，下载完成以后就是下面的这个界面



在红色线条标记的地方有2个，第一个是代表当前的文件夹下面没有`package.json`这个文件，它给了我们一个警告信息。第二个红线代表我们这个包已经下载好了，并且告诉我们版本是`4.6.0`

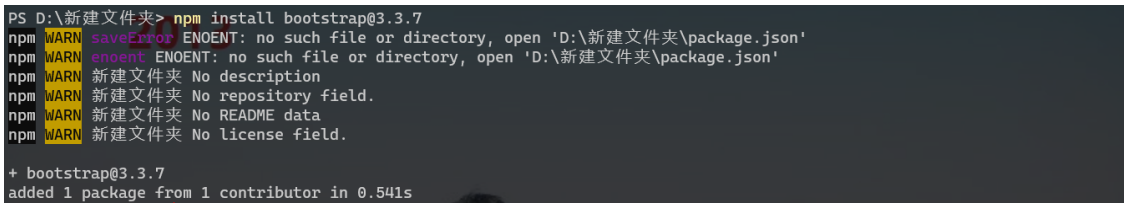
如果下载成功以后，我们会在当前的目录下面出现一个`node_modules`，这个目录用于存放**通过npm从网络上面下载的第三方包**



这个时候我们在这个目录下面就看到了bootstrap这个目录了，说明我们之前的文件已经下载下来了

默认情况之下它下载的是最新版本，如果我们要下指定的版本，则要使用后面的命令

3. `npm install 包名@版本`这个命令和上面的命令是一样的，只是在后面添加了一个版本号

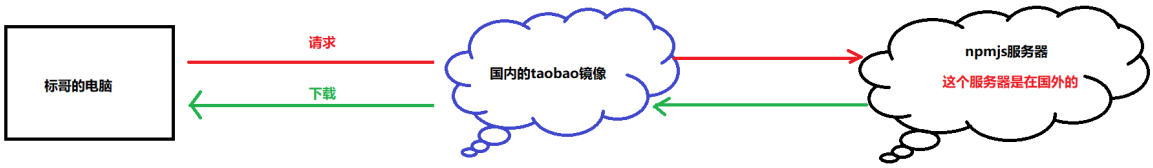


上面的命令就是我们去下载`bootstrap 3.3.7`这个版本

设置npmjs的国内镜像

当我们去通过npmjs去下载第三方模块的时候，这些都是从网络上面再下载，但是这些包都是在国外，我们每次都从国外去下载非常慢

这个时候国内有一家公司干了一件非常好的事情，它自己备份了这个npmjs的服务器，然后放到国内，这样我们只用访问国内的网站就行了，这个公司就是taobao



经过了这么一次中转以后，我们每次访问taobao国内镜像就可以了，这样百度会非常快，所以我们分别在控制台执行下面的命令就可以了

```
$ 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/
```


当执行完上面的命令以后，我们就把地址换成了国内的镜像地址了【上面的命令只用执行一次就可以了，后在就不用执行了，它会一直生效】

当执行完毕以后，我们可以通过npm config list --json来查看是否成功了

```
PS C:\Users\YangBiao> npm config list --json
{
  "json": true,
  "user-agent": "npm/6.14.8 node/v12.16.1 win32 x64",
  "metrics-registry": "https://registry.npm.taobao.org/",
  "scope": "",
  "registry": "https://registry.npm.taobao.org/",
  "disturl": "https://npm.taobao.org/dist",
  "electron_mirror": "https://npm.taobao.org/mirrors/electron/",
  "sass_binary_site": "https://npm.taobao.org/mirrors/node-sass/",
  "phantomjs_cdnurl": "https://npm.taobao.org/mirrors/phantomjs/",
```

当我们把所有的准备工作都完成以后，我们就可以放心大胆的使用第三方模块（包）

使用第三包模块完成数据抓取

在nodejs平台下面如果要完成抓包，我们需要借用两个第三方的模块

1. axios

   这是一个非常流行的模块，无论后期是在浏览器里面还是在nodejs平台下面，这个包都可以使用，它是专门用于做http/https数据请求的

2. cheerio

> 这个模块要当于jQuery一样，可以把一个HTML文档加载进来，然后使用之前所学习的jQuery的语法去里面获取我所需要的信息

首先，我们在这里先加载第三方的模块

```
$ npm install cheerio --save
```


> 后面加的这个`--save`的用意就是将本次的安装信息记录在`package.json`里面


```
"dependencies": {
    "cheerio": "^1.0.0-rc.5"
}
```

接下来我们就可以根据步骤来完成


```
/**
 * 抓包
 */

//第一步：导入我们刚刚下载的包
const cheerio = require("cheerio");
const path = require("path");
const fs = require("fs");

//第二步：先读取1.txt里面的内容，这里面放的是一个html的字符串文本，是我们直接从网站上面copy下来的
let p1 = path.join(__dirname, "./1.txt");
let htmlStr = fs.readFileSync(p1, { encoding: "utf8" });

//第三步：调用cheerio加载这个html的字符中
let $ = cheerio.load(htmlStr);
//这个时候，我们就可以像使用jQuery一样去使用这个cherrio

//第四步：我们现在想在这些字符串当中获取需要的信息
/**
 * movieName电影的名子
 * movieDesc电影的描述信息
 * movieProgress电影更新的进度
 * movieLink电影的链接地址
 * moviePicLink电影图片的地址
 */

//第四步：开始提取我们所需要的数据
let result = [];
$(".videolist_container_inner .g-col").each((index, ele) => {
    let movieName = $(ele).find(".categorypack_title.categorypack_short_title a").text();
    let movieDesc = $(ele).find(".categorypack_subtitle").text();
    let movieProgress = $(ele).find(".categorypack_p_rb>span").text();
    let movieLink = $(ele).find(".categorypack_pack_cover>a").attr("href");
    let moviePicLink = $(ele).find(".categorypack_pack_cover>a>img").attr("src");
    result.push({ movieName, movieDesc, movieProgress, movieLink, moviePicLink });
});

```

```
//第五步：将上面的数组转化成json以后，再写入文件当中去
let jsonStr = JSON.stringify(result);
let resultPath = path.join(__dirname, "./result.txt");
fs.writeFileSync(resultPath, jsonStr, { encoding: "utf8" });
```

但是我们现在并不满足这种情况，因为我们这个`txt`的文件是个固定的，是我们直接从网上copy下来的，这样做不好

思路：我们能不能够主动的去访问下面链接，然后让服务器返回内容给我们

https://www.youku.com/category/show/c_97.html?spm=a2ha1.14919748_WEBTV_JINGXUAN.drawer3.1

答案是可以的，我们以前在浏览器里面，我们可以使用`ajax`去发送请求，但是`ajax`受浏览器的限制不能跨域（因为浏览器要遵守同源策略），但是现在我们不是在浏览器里面，我们是在nodejs的平台上面，这个平台不存在跨域，所以我们可以在nodejs的平台去发送一个`http`的请求，只不过我们在`nodejs`平台发请求的时候，我们使用的是一个第三方的模块叫`axios`

目前与`axios`模块相类似的还有很多模块，如`flyio`或`superagent`等，这些都可以！只是说目前大众使用的都是`axios`这个模块，所以我们今天在讲课的时候也会使用这个模块

首先我们先安装`axios`这个模块

```
$ npm install axios --save
```


接下来使用`axios`来请求数据


```
/**
 * axios模块的使用
 * 可以使用这个模块去发送http或https请求
 */
const axios = require("axios").default;
const cheerio = require("cheerio");
const path = require("path");
const fs = require("fs");


async function getData() {
    //第一步：先准备要请求的url
    let url = "https://www.youku.com/category/show/c_96.html?spm=a2hcb.12701310.app.5~5!2~5!2~5~5~DL~DD~A!2&theme=dark"
    //现在们要向这个地址上面发送请求，所以我们要使用到 axios

    //第二步：开始发送请求，要注意axios是基于Promise存在的
    //之前上面的这个url地址是通过浏览器的地址栏在请求，而通过浏览器地址栏的请求，我们都是get请求
    try {
        let p = await axios.get(url);   //这个时候返回的是一个Promise ， await等待的只能是成功
        // 通过axios请求回来的数据是在data这个属性里同的
        let htmlStr = p.data;           //准备好html字符串
        //第三步：现在可以开始分析爱个html字符串了
        let $ = cheerio.load(htmlStr);
        let result = [];
        $(".videolist_container_inner .g-col").each((index, ele) => {
```

```
            let movieName = $(ele).find(".categorypack_title.categorypack_short_title a").text();
            let movieDesc = $(ele).find(".categorypack_subtitle").text();
            let movieProgress = $(ele).find(".categorypack_p_rb>span").text();
            let movieLink = $(ele).find(".categorypack_pack_cover>a").attr("href");
            let moviePicLink = $(ele).find(".categorypack_pack_cover>a>img").attr("src");
            result.push({ movieName, movieDesc, movieProgress, movieLink, moviePicLink });
        });
        let jsonStr = JSON.stringify(result);
        let resultPath = path.join(__dirname, "./result.txt");
        fs.writeFileSync(resultPath, jsonStr, { encoding: "utf8" });
        console.log("抓取数据成功");
    } catch (error) {
        console.log("你请求失败了");
        console.log(error);
    }
}
getData();
```

这个时候我们打印的`p.data`就是请求回来的html字符串了，再结合刚刚的cherrio来分析这个字符串不就可以了吗?

当我们去抓取第三方的数据以后，我们会看到里面有一个图片的路径，我们能不能在抓包的时候直接把这个图片下载下来呢?


```
/**
 * 抓youku电影的图片
 */
//第一步：先导入模块
const path = require("path");
const fs = require("fs");
const axios = require("axios").default;
const cheerio = require("cheerio");

const getData = async () => {
    try {
        let result = [];
        //第二步：准备要抓取的url
        let url = "https://www.youku.com/category/show/c_100.html?spm=a2ha1.14919748_WEBCOMIC_JINGXUAN.drawer3.1";
        //第三步：请求数据
        let p = await axios.get(url);
        //第四步：使用cheerio加载结果
        let $ = cheerio.load(p.data);
        //第五步：分析数据
        $(".g-col").each((index, ele) => {
            let movieName = $(ele).find(".categorypack_title.categorypack_short_title>a").text();
            let movieLink = $(ele).find(".categorypack_title.categorypack_short_title>a").attr("href");
            let movieDesc = $(ele).find(".categorypack_subtitle").text();
            let movieProgress = $(ele).find(".categorypack_p_rb>span").text();
```

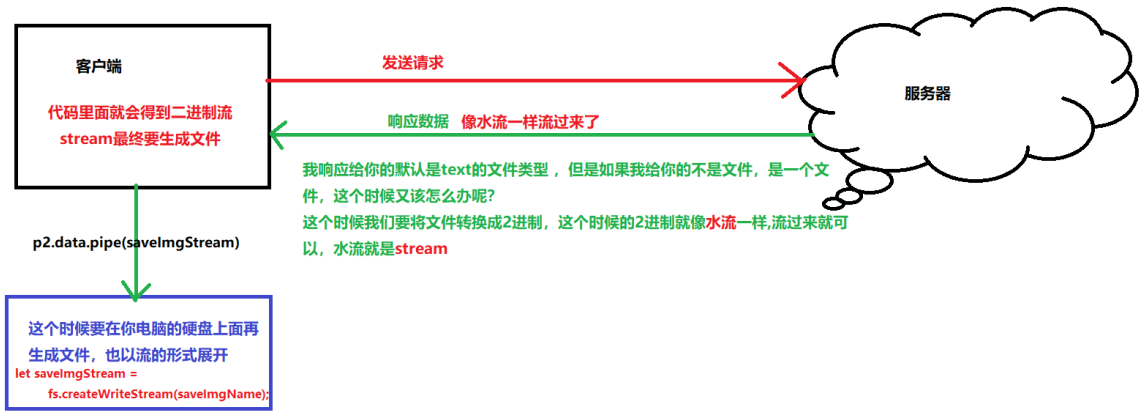
```
            let imgLink = $(ele).find(".categorypack_pack_cover>a>img").attr("src");
            result.push({ movieName, movieLink, movieDesc, movieProgress, imgLink });
        });
        fs.writeFileSync(path.join(__dirname, "./data/result.txt"), JSON.stringify(result), { encoding: "utf8" });
        console.log("数据定入完成");
        //现在数据已经抓完了，我们要去下载图片了，仍然使用axios
        for (let { imgLink, movieName } of result) {
            //现在请求的是图片的地址
            //我们之前通过axios去请求的时候，它是请求以后返回的是文字
            //但是我们现在请求以后返回的就不是文字的，而是一个图片，怎么办呢,这个时候我们要更改它的响应数据类型
            let newURL = `https:${imgLink}`;
            let p2 = await axios.get(newURL, {
                responseType: "stream"
            });
            //这个时候的p2.data就是一个返回过来的数据流
            //第一步：保存图片的位置与名称
            let saveImgName = path.join(__dirname, "./movieImgs", `${movieName}.jpg`);
            //第二步：以文件流的形式表示这个文件
            let saveImgStream = fs.createWriteStream(saveImgName);
            //第三步：将2个流进行对接
            p2.data.pipe(saveImgStream);
            console.log("图片下载成功");
        }

    } catch (error) {
        console.log("数据请求失败");
        console.log(error);
    }
}

getData();
```

上面的代码当中，我们在爬图片的时候我们还是使用了`axios`这个包，它使用nodejs平台发送http请求以后，将图片转换成`stream`以后再返回过来，然后我们又在本地创建了一个文件流`fs.createWriteStream()`,将这2个流进行了对接，这个时候图片就下载下来了

关于流的概念，可以通过下面这张图来进行理解



使用第三方模块去发送邮件

在我们使用nodejs开发项目的时候经常会涉及到注册用户信息这个环节，在这个环节下面，用户需要填入自己的邮箱，为了验证这个邮箱的真实性，我们一般会向这个邮箱发送一个注册验证码，让用户打开自己的邮箱去接收

这个时候我们就需要使用代码来发送邮件。

在NodeJS的平台是可以实现这个功能的，并且是有第三方模块来实现的，这个模块我们叫`nodemailer`

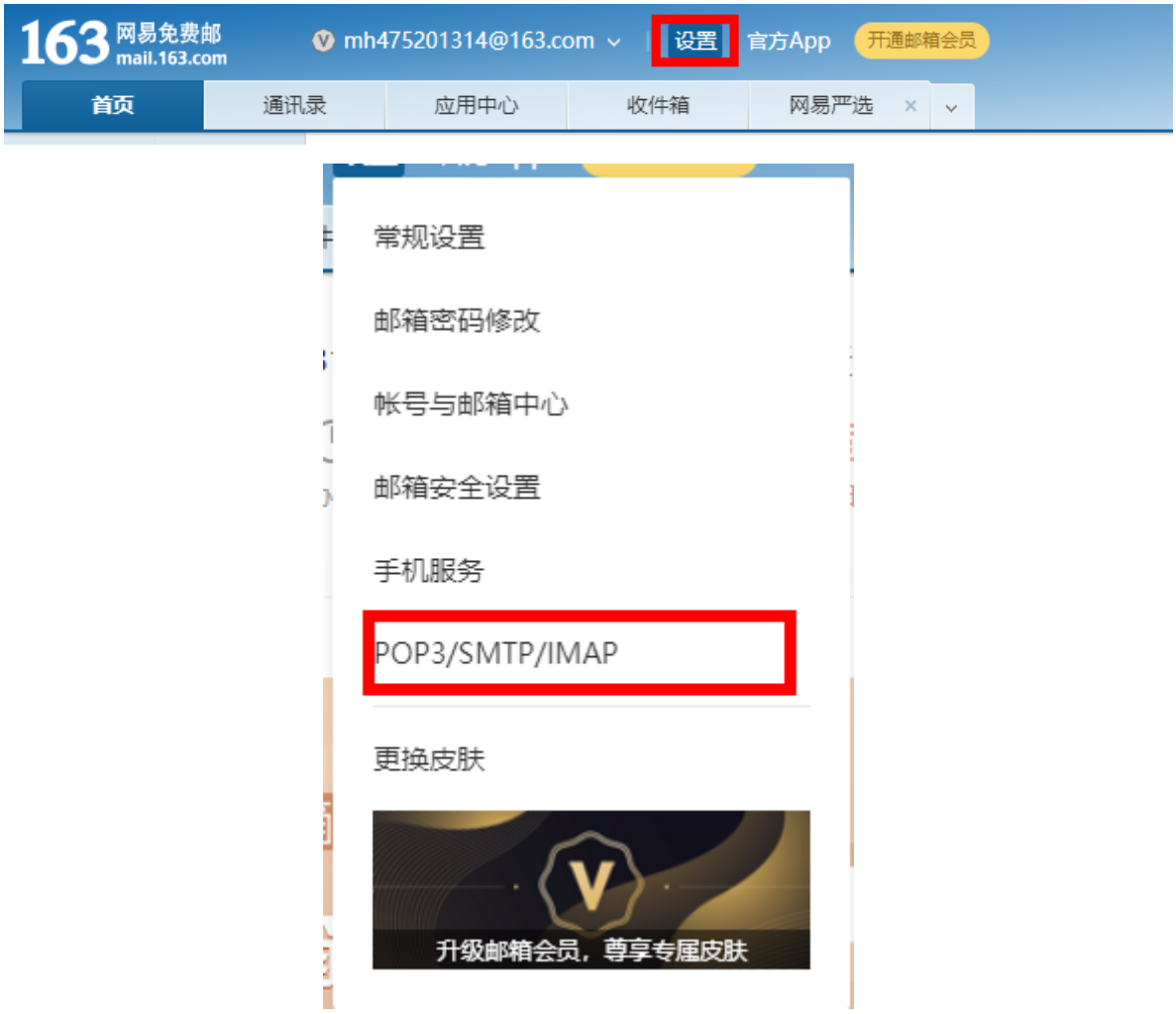
第一步：安装所需要的包

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

softeem · 杨标

第二步：准备发送邮箱的账号，本次案例里面我们使用的是网易163的邮箱

在这一步里面，我们需要使用到邮件的相关知识！发送邮件的时候我们是要借用于163的服务器去发送邮件



在上图中的pop3/smtp/imap这些都是用于发送邮件的，其中`pop3`则是用于接收邮件，`smtp`则是用于发送邮件

打开页面以后，我们会看到上面的信息。有了上面的信息以后，我们就可以进行相关的设置了，默认情况下，这个pop3与smtp服务是关闭的，我们要打开

当我们开启smtp服务以后，这个时候就会出现一个授权的密码与授权状态，如下图

同学们在这里要记住自己授权密码，不要告诉别人

具体的网易163的设置信息可以参考下面的网址

https://help.mail.163.com/faqDetail.do?code=d7a5dc8471cd0c0e8b4b8f4f8e49998b374173cfe9171305fa1ce630d7f67ac2cda80145a1742516

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

当我们设置好了以后就可以开始我们的代码了


```
/**
 * nodejs去发送邮件
 */
//第一步：导入我们所需要使用到的包
const mailer = require("nodemailer");
//第二步：准备我们所需要使用到的邮箱信息数据
//我们们列在使用163邮箱来发送
```

```
8
9  const sendFirstMail = async () => {
10     //第三步：创建一个发送邮件的对象，用于后期我们发送邮件使用
11     let mailTransport = mailer.createTransport({
12         host: "smtp.163.com",          //你要使用哪个服务器去发送邮件
13         port: 465,                      //发送邮件的端口号
14         secure: true,                  //这个邮箱是否有账号密码，true代表的是有账号
   与密码
15         auth: {
16             user: "mh475201314@163.com",         //邮箱账号
17             pass: "RJUNSTDKWDMYWRLB"
18         }
19     });
20     //第四步：创建邮箱内容
21     //它返回的是一个Promise,所以我可以使用await等一下，但是只能等到成功的结果
22     try {
23         let result = await mailTransport.sendMail({
24             from: "mh475201314@163.com",
25             to: [
26                 "1845728120@qq.com",
27                 "2844560455@qq.com",
28                 "zoudanyi@qq.com",
29                 "1515966293@qq.com",
30                 "1448733574@qq.com",
31                 "1183134100@qq.com",
32                 "905528548@qq.com",
33                 "961022195@qq.com",
34                 "1731208592@qq.com",
35                 "2717355901@qq.com",
36                 "1924345978@qq.com",
37                 "410959611@qq.com",
38                 "2482417247@qq.com",
39                 "770966267@qq.com",
40                 "2845466613@qq.com",
41                 "1178794090@qq.com",
42                 "2670820178@qq.com",
43                 "1284664932@qq.com",
44                 "2691580383@qq.com",
45                 "729153711@qq.com",
46                 "1641399087@qq.com",
47                 "1576897781@qq.com",
48                 "528491526@qq.com",
49                 "2856179262@qq.com",
50                 "lovesnsfi@163.com"
51             ],
52             subject: "这是标哥给各位同学们的新年问候",
53             text: `
54             各们同学：
55                 新年好！
56                 值此辞旧迎新之际，我仅带表我个人向每们同学表示新年祝福，希望各们同学们能
   够在新的一年里面，团结奋进，永往直前，
57             不畏艰难，勇攀高峰，再创佳绩！
58                 同时，在寒假里面，也希望各位同学们不要忘记老师对同学们的淳淳教诲，寒假期
   间按时完成家庭作业，做到长大一岁，成熟一岁！
59                 顺祝各位同学样的家人身体健康，万事如意，新快快乐！
60             ------------------------------------------------------------------
   ------------------------------------------
```

```
            收到本邮件以后，可以凭验证码${parseInt(Math.random() * 9999)}来标哥家领取礼品一份，惊喜多多，先到先得！
            ----------------------------------------------------------------------------------------


             你们亲爱的标哥哥

             ${new Date().toLocaleString()}

            `
        });
        console.log("邮件发送成功");
        console.log(result);
    } catch (error) {
        console.log("邮件发送失败");
        console.log(error);
    }
}
//调用上面的箭头函数，发送第一封邮件
sendFirstMail();
```

当我们在发送了邮件以后，我位学可以向邮件当中去添加我们的附件，如下所示


```
/**
 * 邮件发送，附件
 */

const mailer = require("nodemailer");
const path = require("path");           //处理与路径相关的模块
const fs = require("fs");

const sendHomeWork = async () => {
    //先准备邮件的附件呢
    //读data这个文件夹,生成附件的文件格式
    try {
        let p1 = fs.readdirSync(path.join(__dirname, "./data"));
        let attactmentFiles = p1.map(item => {
            return {
                filename: item,
                path: path.join(__dirname, `./data/${item}`)
            }
        });
        //准备发送邮件的对象
        let mailTransport = mailer.createTransport({
            host: "smtp.163.com",          //你要使用哪个服务器去发送邮件
            port: 465,                      //发送邮件的端口号
            secure: true,                  //这个邮箱是否有账号密码，true代表的是有账号与密码
            auth: {
                user: "mh475201314@163.com",        //邮箱账号
                pass: "RJUNSTDKWDMYWRLB"
            }
        });
        let result = await mailTransport.sendMail({
            from: "mh475201314@163.com",
            to: ["1845728120@qq.com",
```

```
                "2844560455@qq.com",
                "zoudanyi@qq.com",
                "1515966293@qq.com",
                "1448733574@qq.com",
                "1183134100@qq.com",
                "905528548@qq.com",
                "961022195@qq.com",
                "1731208592@qq.com",
                "2717355901@qq.com",
                "1924345978@qq.com",
                "410959611@qq.com",
                "2482417247@qq.com",
                "770966267@qq.com",
                "2845466613@qq.com",
                "1178794090@qq.com",
                "2670820178@qq.com",
                "1284664932@qq.com",
                "2691580383@qq.com",
                "729153711@qq.com",
                "1641399087@qq.com",
                "1576897781@qq.com",
                "528491526@qq.com",
                "2856179262@qq.com",
                "lovesnsfi@163.com"],
            subject: "标哥给大家的新礼物-----寒假作业",
            //之前的时候，我们可以使用text属性，现在我们可以使用html的属性
            text: "标哥的寒假大礼包----请愉快的接收吧",
            html: "<div style='color:red;font-size:'36px''>标哥的寒假大礼包---
-请愉快的接收吧</div>",
            attachments: attactmentFiles
        });
        console.log("邮件发送成功");
        console.log(result);
    } catch (error) {
        console.log("邮件发送失败");
        console.log(error);
    }
}
sendHomework();
```

在附件里面，主要使用`attachments`，它的格式是一个数组，格式如下


```
attachments: [
    {
        filename: "2018上.jpg",
        path: path.join(__dirname, "./data/2018上.jpg")
    }, {
        filename: "第二套寒假作业.md",
        path: path.join(__dirname, "./data/第二套寒假作业.md")
    }
    , {
        filename: "第三套寒假作业.md",
        path: path.join(__dirname, "./data/第三套寒假作业.md")
    }
]
```

我们在上面的代码直接调用了`fs`模块去读取某一个文件夹，然后把这个文件夹的内容读出来以后生成了上面的格式