Node 爬虫邮件（定时）发送 — nodelove

- [x]

导语

自从用邮箱注册了很多账号后，便会收到诸如以下类似的邮件,刚开始还以为是一张图片，后来仔细一看不是图片呀，好像还是HTML呀，于是好奇的我，查阅多篇资料后，使用前端知识和Node做一个这样的“邮件网页”。

一、开发工具

1.node下载

https://nodejs.org/zh-cn/

2.vs code下载

https://code.visualstudio.com/

二、主要功能

1.主要说明

- [x] **利用node实现网页爬虫抓取数据**
- [x] **利用模板引擎制作HTML邮件**
- [x] **利用node发送电子邮件【163邮箱、QQ邮箱】**
- [x] **利用node实现定时发送邮件**

2.项目依赖的包（插件）

依赖包名称	功能描述	npm地址
path	路径配置	
superagent	html请求	
cheerio	解析html	
art-template	模板引擎	
nodemailer	发送电子邮件	https://nodemailer.com/about/
node-schedule	定时发送	https://www.npmjs.com/package/node-schedule

三、项目开始

1.初始化项目

```
npm init -y
```

2.安装所有项目依赖包

```
npm install superagent cheerio art-template nodemailer node-schedule
```

```
npm install --save-dev babel-polyfill
```

3.邮件布局参考

注意事项：邮件只允许行内样式。

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, minimum-scale=1, user-scalable=no">
    <meta name="description" content="张成,无风不起浪网，无浪网，无风不起浪@你">
    <meta name="keywords" content="QQmail、163mail">
    <title>node_love邮箱</title>
    <style type="text/css">


</style>
</head>
<body style="margin:0;padding: 0;">
    <div style="width: 100%;margin: 40px auto;font-size: 20px;color: #5f5e5e;text-align: center;">
        <span>今天是我们在一起的第</span>
```

```
        <span style="font-size:24px;color:rgb(221, 73, 73)"  >520</span>
        <span>天</span>
    </div>
    <div style="width: 100%;margin: 40px auto;font-size: 20px;color:
#5f5e5e;text-align: center;">
        <img style="background:#0097e0"
src="https://h5tq.moji.com/tianqi/assets/images/weather/w30.png" alt="晴--天
气图标">
        <!-- <img style="background:#0097e0"
src="http://image.wufazhuce.com/Fu3neP5fitv8cYLulwrjQ9zaXMB5" alt="天气图标">
-->
        <b style="display:block;color:#333;font-size:24px;margin:15px 0;">天
气:太阳</b>
        <span style="display:block;color:#333;font-size:24px;margin:15px
0;">温度：19</span>
        <span style="display:block;color:#333;font-size:24px;margin:15px
0;">温度：19</span>
        <span style="display:block;color:#676767;font-size:20px">提示：近期天
气有雨。</span>
</div>
<div style="text-align:center;margin:35px auto;">
    <span style="display:block;margin-top:55px;color:#676767;font-
size:15px">ONE · 一个</span>
    <!-- <span style="display:block;margin-top:25px;font-size:22px;
color:#9d9d9d; ">2020/3/10</span> -->

    <div style="color: #333; margin: 0 auto;padding: 12.5px;text-align:
center;">
        <p style="font-size: 48px;height: 48px;line-height: 48px;">2020</p>
        <p style=" margin: 0 0 5px;">Mar  10</p>
    </div>
    <img class="fp-one-imagen"
src="http://image.wufazhuce.com/FklQVpjBMzz15-N2MjQkE5LqYtNh"alt="one配图">
    <span style="color:#b0b0b0;font-size:13px;"></span>
    <div style="margin:10px auto;width:85%;color:#5f5e5e;" >生活是苦难的，我又
划着我的断桨出发了。</div>
</div>
<script type="text/javascript"></script>
</body>
</html>
```

4.墨迹天气、中国天气网网址

- http://tianqi.moji.com/
- http://wufazhuce.com/

5.邮件设置

(1)官方配置代码库

https://nodemailer.com/about/

```
"use strict";
```

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

// async..await is not allowed in global scope, must use a wrapper
async function main() {
  // Generate test SMTP service account from ethereal.email
  // Only needed if you don't have a real mail account for testing
  let testAccount = await nodemailer.createTestAccount();

  // create reusable transporter object using the default SMTP transport
  let transporter = nodemailer.createTransport({
    host: "smtp.ethereal.email",
    port: 587,
    secure: false, // true for 465, false for other ports
    auth: {
      user: testAccount.user, // generated ethereal user
      pass: testAccount.pass // generated ethereal password
    }
  });

  // send mail with defined transport object
  let info = await transporter.sendMail({
    from: '"Fred Foo 👻" <foo@example.com>', // sender address
    to: "bar@example.com, baz@example.com", // list of receivers
    subject: "Hello ✓", // Subject line
    text: "Hello world?", // plain text body
    html: "<b>Hello world?</b>" // html body
  });

  console.log("Message sent: %s", info.messageId);
  // Message sent: <b658f8ca-6296-ccf4-8306-87d57a0b4321@example.com>

  // Preview only available when sending through an Ethereal account
  console.log("Preview URL: %s", nodemailer.getTestMessageUrl(info));
  // Preview URL: https://ethereal.email/message/WaQKMgKddxQDoou...
}

main().catch(console.error);
```

(2).我的参考代码

```
//邮箱的发送
const nodemailer = require("nodemailer");

async function sendNodeMail() {
    //html 页面内容
    const html = "<h1>爱的</h1>";
    console.log(html);

    //默认使用smtp传输，创建重用邮箱对象
    let transporter = nodemailer.createTransport({
        host: "smtp.163.com",
        port: 465,
        secure: true,//开启加密协议，需要使用465端口
        auth: {
            user: "13469940053@163.com",//用户名
            pass: "********"//授权密码
```

```
        }
    });
    //设置电子邮件数据
    let mailoptions = {
        from: '"无风不起浪@你" <13469940053@163.com>',//发件人邮箱
        to: "13469940053@163.com",//收件人列表(可以多个发送以,分开)
        subject: "文档html",//邮件名称标题
        text: "正文文本",
        html: html
    };

    transporter.sendMail(mailoptions, (error, info = {}) => {
        if (error) {
            console.log(error);
            sendNodeMail();//再次发送
        }
        console.log("邮件发送成功! ", info.messageId);
        console.log("静静等待下一次发送! ");
    });
}
sendNodeMail();
```

163邮箱设置

QQ邮箱设置

6.node发送邮件

```
var schedule = require('node-schedule');

var j = schedule.scheduleJob('42 * * * *', function(){
  console.log('The answer to life, the universe, and everything!');
});
```

规则参数讲解 *代表通配符

```
*  *  *  *  *  *
┬ ┬ ┬ ┬ ┬ ┬
| | | | | |
| | | | | └ day of week (0 - 7) (0 or 7 is Sun)
| | | | └─── month (1 - 12)
| | | └────── day of month (1 - 31)
| | └─────────── hour (0 - 23)
| └──────────────── minute (0 - 59)
└─────────────────────── second (0 - 59, OPTIONAL)
```

6个占位符从左到右分别代表：秒、分、时、日、月、周几

`*`表示通配符，匹配任意，当秒是`*`时，表示任意秒数都触发，其它类推

下面可以看看以下传入参数分别代表的意思

```
每分钟的第30秒触发： '30 * * * * *'

每小时的1分30秒触发 ：'30 1 * * * *'

每天的凌晨1点1分30秒触发 ：'30 1 1 * * *'

每月的1日1点1分30秒触发 ：'30 1 1 1 * *'

2020年的1月1日1点1分30秒触发 ：'30 1 1 1 2020 *'

每周1的1点1分30秒触发 ：'30 1 1 * * 1'
```

每个参数还可以传入数值范围:

```
const task1 = ()=>{
  //每分钟的1-10秒都会触发，其它通配符依次类推
  schedule.scheduleJob('1-10 * * * * *', ()=>{
    console.log('scheduleCronstyle:'+ new Date());
  })
}

task1()
```

对象文本语法定时器

```
const schedule = require('node-schedule');

function scheduleObjectLiteralSyntax(){

    //dayOfWeek
    //month
    //dayOfMonth
    //hour
    //minute
    //second
```

```
      //每周一的下午16：11分触发，其它组合可以根据我代码中的注释参数名自由组合
    schedule.scheduleJob({hour: 16, minute: 11, dayOfWeek: 1}, function(){
        console.log('scheduleObjectLiteralSyntax:' + new Date());
    });

}

scheduleObjectLiteralSyntax();
```

定时器 定时器对象的cancl()方法即可

```
const schedule = require('node-schedule');

function scheduleCancel(){

    var counter = 1;
    const j = schedule.scheduleJob('* * * * * *', function(){

        console.log('定时器触发次数：' + counter);
        counter++;

    });

    setTimeout(function() {
        console.log('定时器取消')
      // 定时器取消
        j.cancel();
    }, 5000);

}

scheduleCancel();
```

四、完整代码

今天是我们在一起的第 520 天

天气:太阳

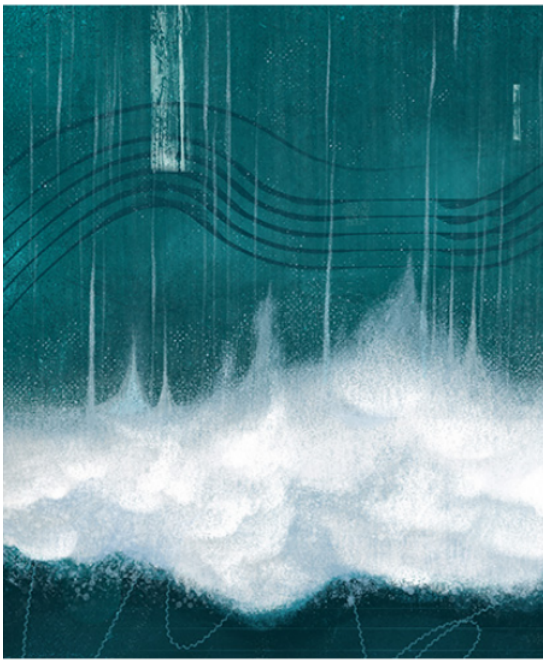
温度：19

提示：近期天气有雨。

无风不起浪@你

2020

Mar 12



生活是苦难的，我又划着我的断桨出发了。

1.认识天数gedayDate函数

```
function getdayDate() {

    return new Promise((resolve, reject) => {
        const today = new Date();

        const meet = new Date("2020-3-9");

        const count = Math.ceil((today - meet) / 1000 / 60 / 60 / 24);
```

```
        //格式化当前日期
        const newyear = today.getFullYear()+ '年';
        const format =  (today.getMonth() + 1) + '月' + today.getDate() + '日';
        //json格式存储数据
        const dayDate = {
            count,
            newyear,
            format
        }
        console.log(dayDate);
        console.log("认识的天数:" + dayDate.count, "认识的时间:" + dayDate.format);
    })
}
getdayDate();
```

返回的数据

```
{
    count: 4,
    newyear: '2020年',
    format: '3月12日'
}
```

2.发起请求获取我的qq链接

```
const QQme = "https://13469940053.github.io/";

function getQQmeDate() {
    //请求获取数据
    request.get(QQme).end((err, res) => {
        if (err) return console.log("数据获取失败,请检查路径是否正确! ");
        // 官网源代码获取
        // console.log(res.text);
        //把字符串解析成html
        const $ = cheerio.load(res.text);
        //温度
        const QQme =$('.social-link a').eq(2).attr('href');

        console.log(QQme);
    });
}
getQQmeDate();
```

返回我的QQ链接

```
tencent://AddContact/?fromId=50&fromSubId=1&subcmd=all&uin=2508723631
```

3.发起请求获取one数据

```
const Oneurl = "http://wufazhuce.com/";

```

```
function getoneDate() {
    //请求获取数据
    request.get(Oneurl).end((err, res) => {
        if (err) return console.log("数据获取失败,请检查路径是否正确！");
        //把字符串解析成html
        const $ = cheerio.load(res.text);
        // console.log(res.text);
        const oneImg = $('.carousel-inner>.item>img, .carousel-inner>.item>a>img').attr('src');
        const oneText = $('.fp-one .fp-one-cita-wrapper .fp-one-cita a').eq(0).text();

        const onedate = {
            oneImg,
            oneText
        }
        console.log(onedate);
    })
}
getoneDate();
```

返回的json数据

```
{
  oneImg: 'http://image.wufazhuce.com/FrOv-josIaX2-nu7z42409DwMijf',
  oneText: '人生需要准备的，不是昂贵的茶，而是喝茶的心情。 '
}
```

4.通过模板引擎替换html数据

```
async function readerTemplate() {
    //日期 数据
    const dayDate = await getdayDate();
    //天气获取数据
    const MojiDate = await getMojiDate();
    //获取one数据
    const oneDate = await getoneDate();
    //获取我的QQ链接
    const QQme = await getQQmeDate();

    // console.log(dayDate+MojiDate+oneDate+QQme);

    //引擎数据替换
    const html = template(path.join(__dirname, "./demo.html"), {
        dayDate,
        MojiDate,
        oneDate,
        QQme
    });
    //检测数据模板中的数据是否替换完成
    console.log(html);
}
 readerTemplate();
```

5.定时发送的时间

```
var schedule = require('node-schedule');
 //6个占位符从左到右分别代表：秒、分、时、日、月、周几   *代表通配符
var j = schedule.scheduleJob('42 * * * * *', function(){
    sendNodeMail();
  console.log('生命、宇宙和一切的答案！发送成功！');
});
```

$$node + html + css + css3 + javascript \tag{1}$$

五、留言（结束）

输入`npm install`安装依赖，再输入`node main.js`，运行脚本，当然你的电脑不可能不休眠，建议你部署到你的云服务器上运行。

目录