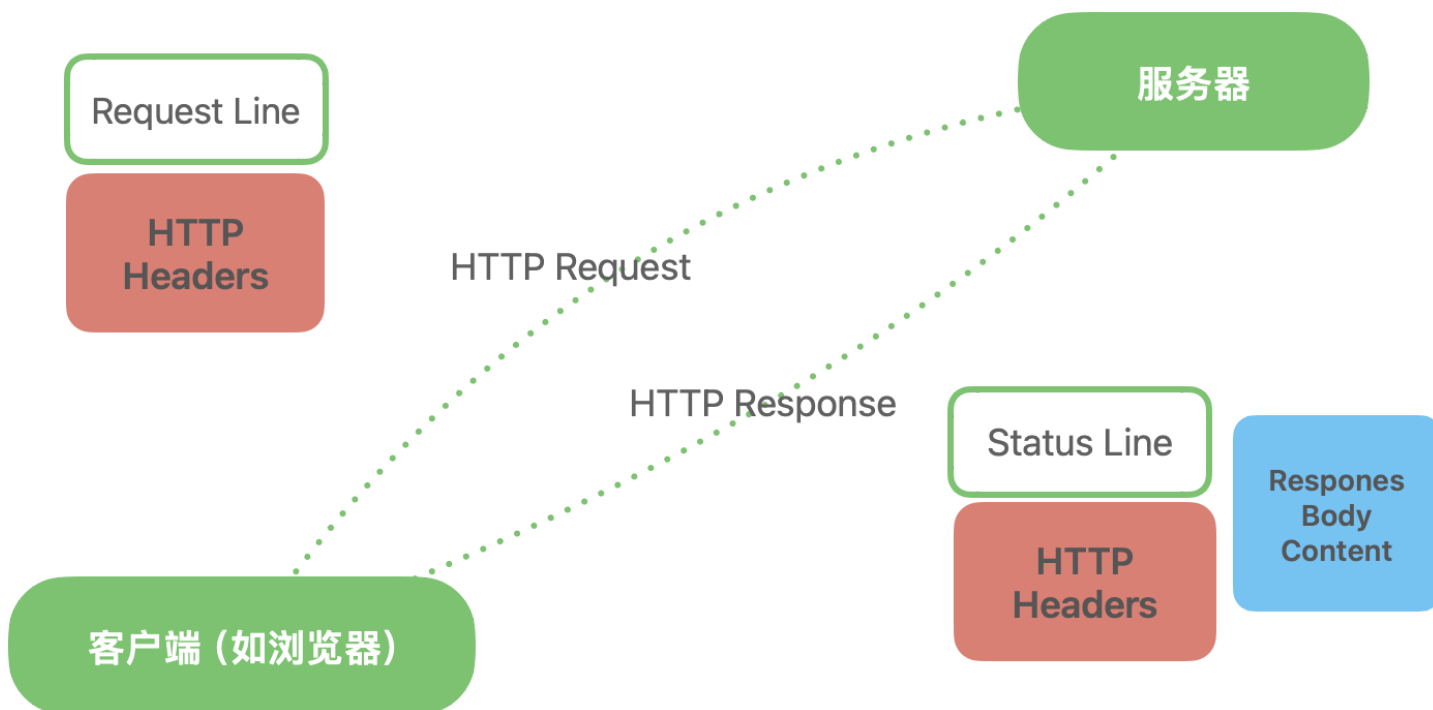


# Node.js结合Puppeteer处理有状态页面

本节目标：[各种手段请求爬取网页内容] 你请求，我应答，网络两头乐开花，Node 之所以成为服务器方案，离不开 HTTP 模块的能力之帆。



## 借助 puppeteer 来爬取有状态或者异步数据页面

有了上面对请求和响应的基本了解，我们知道了一些信心，只要我想要爬取的内容，都可以写一个工具快速拿过来分析，然而有时候会事与愿违，有的目标网站会有异步的内容，甚至会有反爬策略，我们甚至拿不到正确的 HTML 源码，甚至更高级的策略中，我们即便拿到正确的 HTML，却未必能正确解析出来，比如用雪碧图错位来表示数字等等。

这个要具体问题具体分析，我们打开XIAOJUSURVEY的掘金页面，试下爬取这个页面内容，然后统计下一共现在发表了多少文章，有多少浏览量以及共收获多少次点赞。

我们可以写这样一段代码：

```
1 // juejin-book.js
2 // 在 node juejin-book.js 执行代码之前
3 // 先 npm i cheerio request-promise 安装依赖模块
4 const cheerio = require('cheerio')
5 const rp = require('request-promise')
6 const url = 'https://juejin.cn/user/3705833332160473/posts'
```

```

7 // 通过 request-promise 来爬取网页
8 rp(url).then(function(html) {
9   // 利用cheerio 来分析网页内容，拿到所有文章的描述，
10   const
11   $ = cheerio.load(html);
12   // todo: 注意文章是异步获取的，所以只能拿到第一页的数据
13   const articles = $
14   ('.content-main')
15   let totalViews = 0
16   let totalLikes = 0
17   let totalArticles = articles.length
18   // 遍历册子节点，分别统计它的购买人数，和销售额总和
19   articles.each((index, element) => {
20     console.log($(element).text());
21     const article =
22     $(this )
23     const view = article.find('.view').find('span').text()
24     const like = article.find('.like').find('span').text()
25     totalViews += Number(view)
26     totalLikes += Number(like)
27   })
28   // console.log({articles})
29   // 最后打印出来
30   console.log(
31     `共发表 $
32     {totalArticles} 篇文章
33     ,
34     共
35     ${totalViews} 次浏览`,
36     `约 $
37     {totalLikes} 点赞`
38   )
39 })

```

最后打印的结果是：共发表 10 篇文章 共 79 次浏览 约 12 点赞，What? 怎么和掘金主页上显示的数据不一致呢？也许是因为文章数量是分页加载的，一次默认加载10条，这么解释就没毛病了。

针对这种情况下，网页的数据是异步分页加载的，我们是可以通过分析请求头和响应头来模拟一次真实的网页访问，我们也可以通过网页爬取神器 - puppeteer 来获取网页内容，puppeteer 是谷歌开源的，可以通过命令行来启动一个 chrome 实例，从而真实访问网页，并且具备与网页的交互的能力，包括但不限于点击，滚动，截屏等操作。

我们来写一个小例子，来截取下掘金的首页头部，在执行之前，首先安装 puppeteer：

```
1 npm i puppeteer -S
2 puppeteer@1.10.0 install /Users/black/Downloads/node_modules/puppeteer
3 node install.js
4 Downloading Chromium r599821 - 82.9 Mb [ ] 1% 1287.3s
5 # 安装可能会比较耗时，大家可以多尝试几次
```

```
1 // 把之前安装到 node_modules 下的 puppeteer 模块加载进来
2 const puppeteer = require('puppeteer')
3 // 在 getHomePage 函数里面，定制一系列任务，让他们顺序执行
4 async function getHomePage (link) {
5 // 启动一个 Chrome 引擎实例，加上 await 会一直等待它启动完成
6 // 加上 headless: false 会打开一个浏览器，可以眼睁睁看这一切发生，如果是 true 则 静默执行
7 // const browser = await puppeteer.launch({headless: false})
8 const browser = await puppeteer.launch()
9 // 启动成功后，打开一个新页面
10 const page = await browser.newPage()
11 // 新页面里面输入目标网址，跳到这个网页，一直等待页面加载完成
12 await page.goto(link)
13 // 设置网页视窗的宽高
14 await page.setViewport({width: 1080, height: 750})
15 // 告诉 puppeteer 开始截图，直到截图完成，存储图片到当前目录
16 await page.screenshot({path: Date.now() + '.png'})
17 // 最后关闭浏览器，销毁所有变量
18 await browser.close()
19 return 'done!'
20 }
21 // 调用这个异步函数 getHomePage，传入待截图网站，任务开始执行
22 getHomePage('https://juejin.im/books').then(v => {})
```

会得到这样的一个截图：



XIAOJUSURVEY

LV.3 JY.4

滴滴

XIAOJUSURVEY掘金主站：「快速」打造「专属」问卷系统, 让调研「更轻松」

关注

私信

个人成就

文章被点赞 42

文章被阅读 1,759

掘力值 291

获得徽章 0

动态 文章 专栏 沸点 收藏集 关注 作品 赞 18

最新

热门

## MongoDB工程配置入门（二）

MongoDB在Java和Node工程配置入门，是基础和入门的流程，从服务配置到业务模块里的...

20小时前 | 3 | 点赞 | 评论

Mongo... 数据库



## MongoDB快速使用，详细且实用（一）

分享MongoDB的几种安装使用方式，内存安装、远端免费版、本地安装方式，不同的方式用...

20小时前 | 2 | 点赞 | 评论

Mongo... 数据库

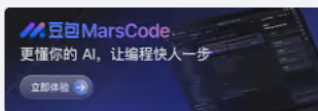


## 工程化实践：工程配置化设计

配置化是很灵活且很常见的使用，那XIAOJUSURVEY里有哪些地方应用到了呢？本篇分享...

1天前 | 6 | 点赞 | 评论

前端工...



收藏集 0

关注标签 0

加入于 2024-03-19



## 编程练习 - 实现XIAOJUSURVEY掘金主页的统计工具

了解 puppeteer 后，我们就可以借助它来获取XIAOJUSURVEY的掘金主页的内容了，代码可以这样写：

```
1 const cheerio = require('cheerio')
2 const puppeteer = require('puppeteer');
3 const url = 'https://juejin.cn/user/3705833332160473/posts'
4 async function autoScroll(page) {
5   await page.evaluate(async () => {
6     await new Promise((resolve, reject) => {
7       let totalHeight = 0;
8       const distance = 100; // 每次滚动的距离
9       const scrollTime = 1000
10      const timer = setInterval(() => {
11        const scrollHeight = document.body.scrollHeight;
12        window.scrollBy(0, distance);
13        totalHeight += distance;
14        if (totalHeight >= scrollHeight) {
15          clearInterval(timer);
16          resolve();
17        }
18      }, scrollTime); // 每次滚动的间隔时间
```

```

19     });
20   });
21 }
22 (async () => {
23   const browser = await puppeteer.launch({
24     headless: false,
25     executablePath: '/Applications/Chromium.app/Contents/MacOS/Chromium',
26   });
27   const page = await browser.newPage();
28   await page.goto(url, { waitUntil: 'networkidle2' })
29   // 自动滚动页面, 加载所有内容
30   await autoScroll(page);
31   const html = await page.content()
32   const
33   $ = cheerio.load(html)
34   const articles = $
35   ('.content-main')
36   let totalViews = 0
37   let totalLikes = 0
38   let totalArticles = articles.length
39   // 遍历文章节点, 分别统计它的浏览量和点赞量
40   articles.each(( index, element ) => {
41     const article =
42     $(element)
43     const view = article.find('.view').find('span').text()
44     const like = article.find('.like').find('span').text()
45     // console.log({like});
46     totalViews += Number(view)
47     totalLikes += Number(like) | 0
48     if(index === articles.length -1 ){
49       console.log($
50 (element).text())
51     }
52   })
53   // 最后打印出来
54   console.log(
55
56   共发表 ${totalArticles} 篇文章
57   ,
58
59   共 ${totalViews} 次浏览
60   ,
61
62   约 ${totalLikes} 点赞
63   )
64   // await browser.close();
65 })();

```

---

共发表 20 篇文章 共 1767 次浏览 约 46 点赞，正好与主页上显示数据一致~ 证明我们爬取程序没问题!