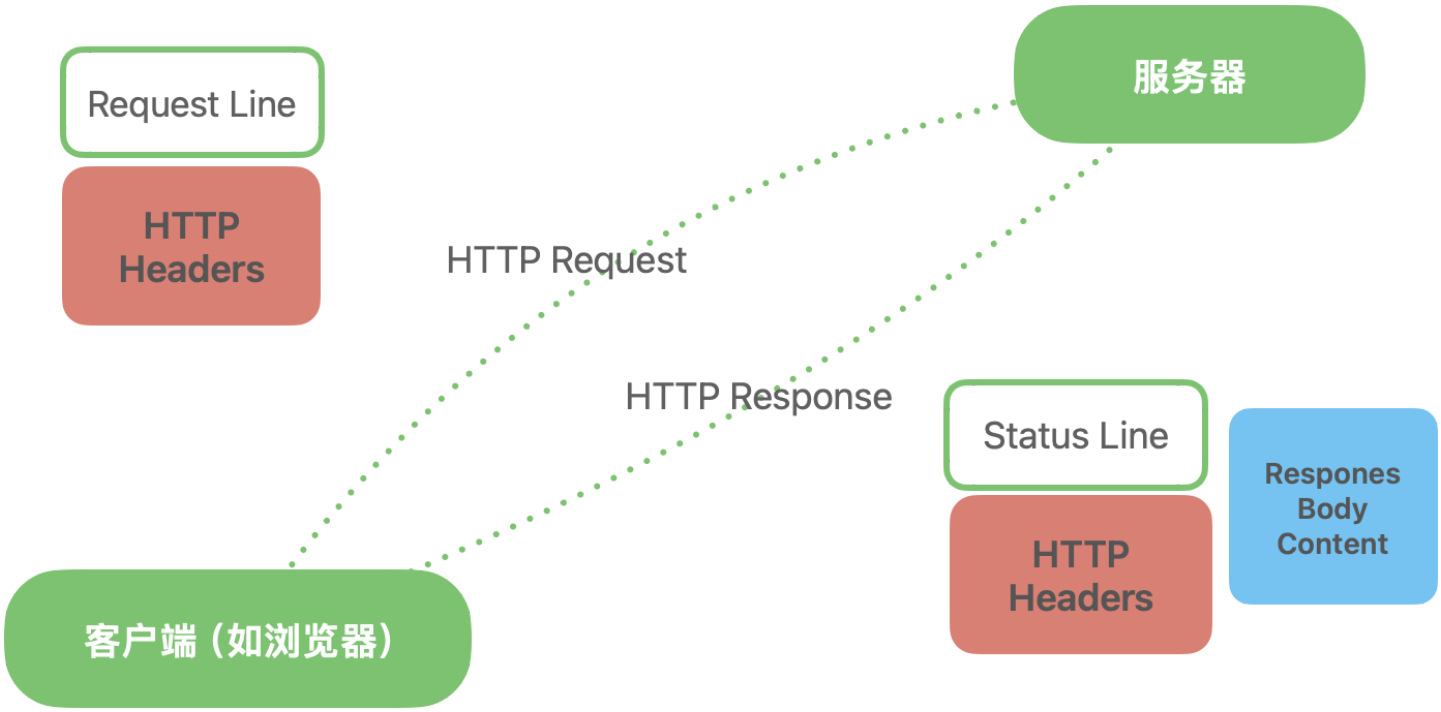
Node.js结合Puppeteer处理有状态页面

> 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

有了上面对于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：

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

```
// 通过 request-promise 来爬取网页
rp(url).then(function(html) {
  // 利用cheerio 来分析网页内容，拿到所有文章的描述，
  const
$ = cheerio.load(html);
  // todo: 注意文章是异步获取的，所以只能拿到第一页的数据
  const articles = $
('.content-main')
  let totalViews = 0
  let totalLikes = 0
  let totalArticles = articles.length
  // 遍历册子节点，分别统计它的购买人数，和销售额总和
  articles.each((index, element) => {
    console.log($(element).text());
    const article =
$(this )
    const view = article.find('.view').find('span').text()
    const like = article.find('.like').find('span').text()
    totalViews += Number(view)
    totalLikes += Number(like)
  })
  // console.log({articles})
  // 最后打印出来
  console.log(
  `共发表 $
{totalArticles} 篇文章
,
共
${totalViews} 次浏览`,
  `约 $
{totalLikes} 点赞`
  )
})
```

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：

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

```
// 把之前安装到 node_modules 下的 puppeteer 模块加载进来
const puppeteer = require('puppeteer')
// 在 getHomePage 函数里面，定制一系列任务，让他们顺序执行
async function getHomePage (link) {
// 启动一个 Chrome 引擎实例，加上 await 会一直等待它启动完成
// 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
// const browser = await puppeteer.launch({headless: false})
const browser = await puppeteer.launch()
// 启动成功后，打开一个新页面
const page = await browser.newPage()
// 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
await page.goto(link)
// 设置网页视窗的宽高
await page.setViewport({width: 1080, height: 750})
// 告诉 puppeteer 开始截图，直到截图完成，存储图片到当前目录
await page.screenshot({path: Date.now() + '.png'})
// 最后关闭浏览器，销毁所有变量
await browser.close()
return 'done!'
}
// 调用这个异步函数 getHomePage，传入待截图网站，任务开始执行
getHomePage('https://juejin.im/books').then(v => {})
```

会得到这样的一个截图：

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：

```
const cheerio = require('cheerio')
const puppeteer = require('puppeteer');
const url = 'https://juejin.cn/user/3705833332160473/posts'
async function autoScroll(page) {
  await page.evaluate(async () => {
    await new Promise((resolve, reject) => {
      let totalHeight = 0;
      const distance = 100; // 每次滚动的距离
      const scrollTime = 1000
      const timer = setInterval(() => {
        const scrollHeight = document.body.scrollHeight;
        window.scrollBy(0, distance);
        totalHeight += distance;
        if (totalHeight >= scrollHeight) {
          clearInterval(timer);
          resolve();
        }
      }, scrollTime); // 每次滚动的间隔时间
```

```
    });
  });
}
(async () => {
  const browser = await puppeteer.launch({
    headless: false,
    executablePath: '/Applications/Chromium.app/Contents/MacOS/Chromium',
  });
  const page = await browser.newPage();
  await page.goto(url, { waitUntil: 'networkidle2' })
  // 自动滚动页面，加载所有内容
  await autoScroll(page);
  const html = await page.content()
  const
$ = cheerio.load(html)
  const articles = $
('.content-main')
  let totalViews = 0
  let totalLikes = 0
  let totalArticles = articles.length
  // 遍历文章节点，分别统计它的浏览量和点赞量
  articles.each(( index, element ) => {
    const article =
$(element)
    const view = article.find('.view').find('span').text()
    const like = article.find('.like').find('span').text()
    // console.log({like});
    totalViews += Number(view)
    totalLikes += Number(like) | 0
    if(index === articles.length -1 ){
      console.log($
(element).text())
    }
  })
  // 最后打印出来
  console.log(

共发表 ${totalArticles} 篇文章
,

共 ${totalViews} 次浏览
,

约 ${totalLikes} 点赞
  )
  // await browser.close();
})();
```

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！