How to Perform Web-Scraping using Node.js- Part 2

Ankit Jain
Jan 23, 2019 · 5 min read

Web Scraping

In the **previous post**, we learned how to scrape static data using Node.js. In previous years, we have seen **exponential growth in javascript** whether we talk about libraries, plugins or frameworks. We have moved to **Single Page Application**, you can know more about SPA in this blog post — How Single-Page Applications Work.

Now, websites are more dynamic in nature i.e content is rendered through javascript. For an instance, let's make a request to any SPA website like I have taken this vue admin template website and disable javascript using Chrome DevTools, this is the response I got —

```
<html lang="tr">

<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width,initial-
scale=1">
  <link href="https://fonts.googleapis.com/icon?
family=Material+Icons" rel="stylesheet">
  <link
href="https://use.fontawesome.com/releases/v5.6.3/css/all.css"
rel="stylesheet">
  <link rel="shortcut icon" type="image/png" href="static/logo.png">
  <title>Vue Admin Template</title>
  <link href="static/css/app.48932a774ea0a6c077e59d0ff4b2bfab.css"
rel="stylesheet">
</head>

<body data-gr-c-s-loaded="true" cz-shortcut-listen="true">
  <div id="app"></div>
  <script type="text/javascript"
src="static/js/manifest.2ae2e69a05c33dfc65f8.js">
  </script>
  <script type="text/javascript"
src="static/js/vendor.ef9a9a02a332d6b5e705.js">
  </script>
  <script type="text/javascript"
src="static/js/app.3ec8db3ca107f46c3715.js">
  </script>

</body>

</html>
```

So, the actual content that we need to scrape will be rendered within the **div#app** element through javascript so methods that we used to scrape data in the previous post fails as we need something that can run the javascript files similar to our browsers.

We will use **web automation tools** and libraries like **Selenium**, **Cypress**, **Nightmare**, **Puppeteer**, **x-ray**, Headless browsers like **phantomjs** etc.

Why these tools and libraries? These tools and libraries are generally used by Software testers in tech industries for software testing. All of these opens a browser instance in which our website can run just like other browsers and hence executes javascript files which render the dynamic content of the website.

. . .

Easily organize, discover and reuse JS components and APIs — to build faster. Give **Bit** a try, it's open source and free.

Component Discovery and Collaboration · Bit

Bit is where developers share components and collaborate to build amazing software together. Discover components shared...

bit.dev

. . .

What will we need?

As I explained in my **previous post** that web scraping is divided into two simple parts —

1. Fetching data by making an HTTP request
2. Extracting important data by parsing the HTML DOM

We will be using Node.js and browser automation library:

- **Nightmare** — is a high-level browser automation library from Segment.
- **cheerio** — jQuery for Node.js. Cheerio makes it easy to select, edit, and view DOM elements.

Setup

Our setup is pretty simple. We create a new folder and run this command inside that folder to create a **package.json** file. And add **Nightmare** and **Cheerio** from npm as our dependencies.

```
npm init -y
npm install nightmare cheerio --unsafe-perm=true
```

Now, require them in our `index.js` file

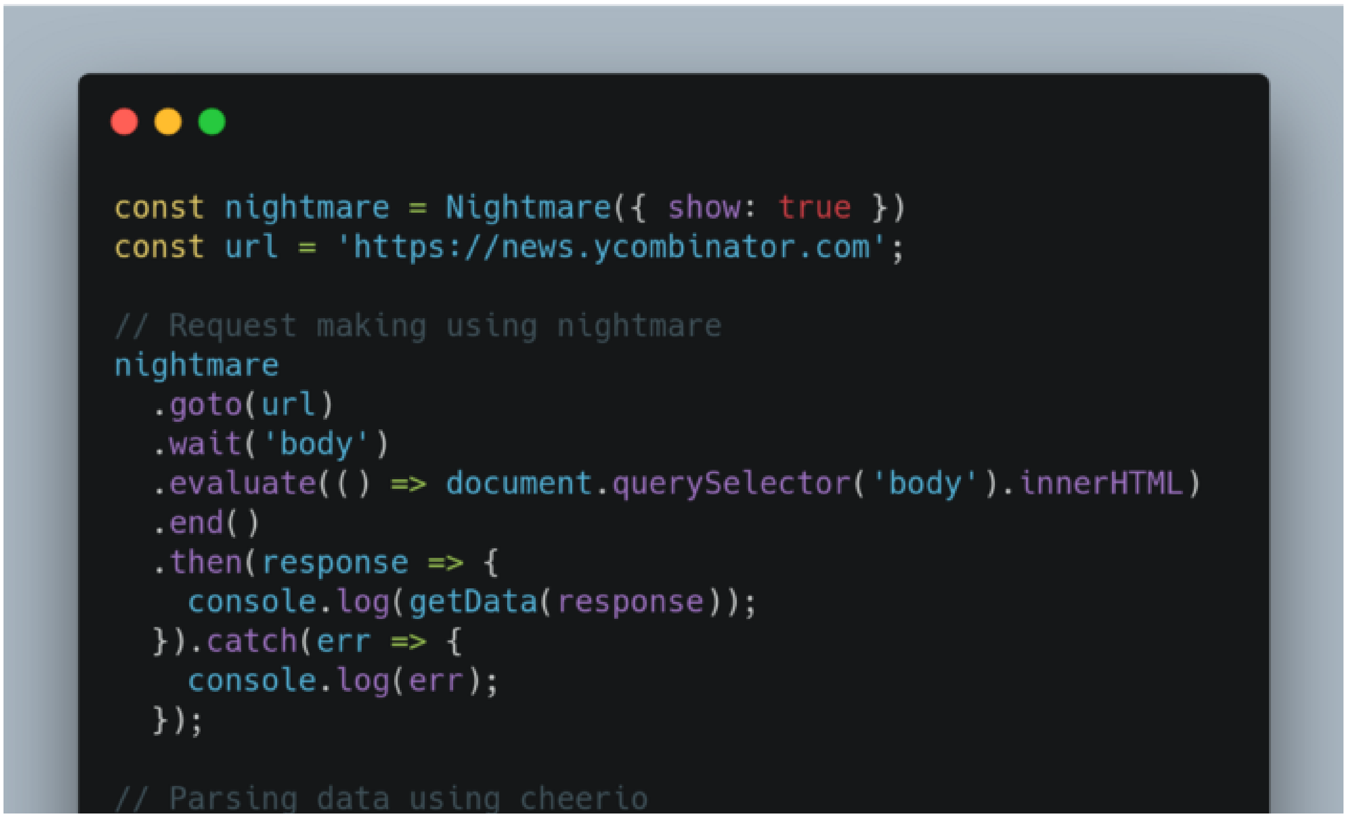
```
const Nightmare = require('nightmare');
const cheerio = require('cheerio');
```

Nightmare is similar to axios or any other request making library but what makes it odd is that it uses **Electron** under the cover, which is similar to **PhantomJS** but roughly **twice as fast** and more modern. Nightmare uses Javascript for handing/manipulating DOM through `evaluate` function which is complex to implement. So, we will use Cheerio for handling DOM content by fetching innerHTML through `evaluate` function and pass the content (innerHTML) to Cheerio which is easy, fast, and flexible to implement. Moreover, Nightmare also supports proxies, promises and async-await.

Scrape the static content

We have already scrapped the static content in the previous post, but before proceeding before, let's do the same with Nightmare so we can have a wider vision and clear understanding of nightmare.

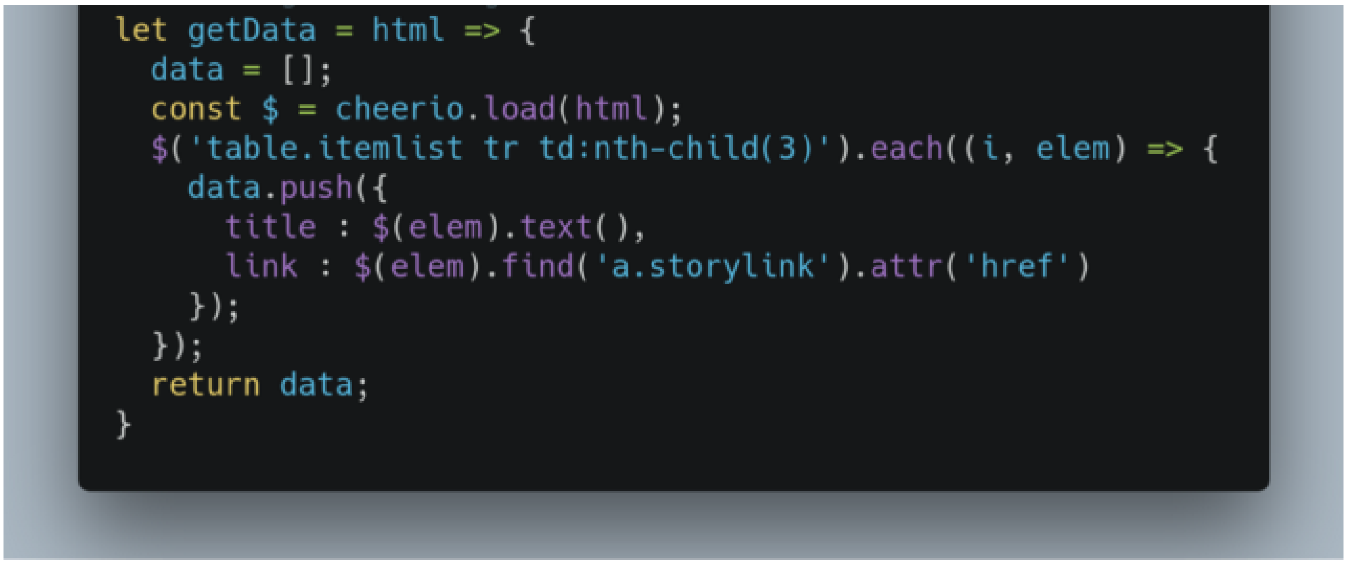
We are scraping data from the **HackerNews** website for which we need to make an HTTP request to get the website's content and parse the data using cheerio.



```
const nightmare = Nightmare({ show: true })
const url = 'https://news.ycombinator.com';

// Request making using nightmare
nightmare
  .goto(url)
  .wait('body')
  .evaluate(() => document.querySelector('body').innerHTML)
  .end()
  .then(response => {
    console.log(getData(response));
  }).catch(err => {
    console.log(err);
  });

// Parsing data using cheerio
```



```
let getData = html => {
  data = [];
  const $ = cheerio.load(html);
  $('table.itemlist tr td:nth-child(3)').each((i, elem) => {
    data.push({
      title : $(elem).text(),
      link : $(elem).find('a.storylink').attr('href')
    });
  });
  return data;
}
```

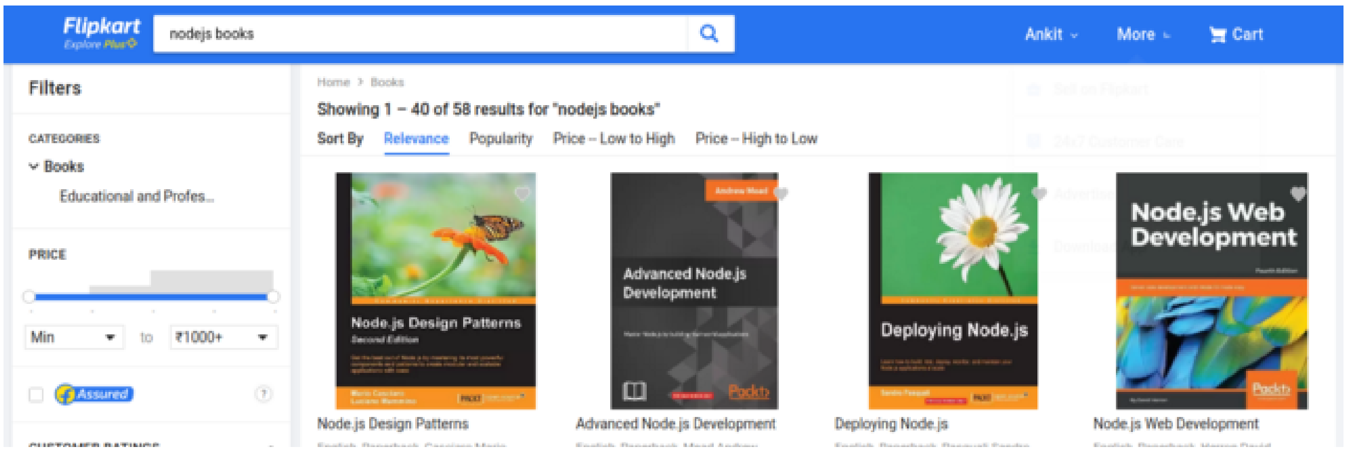
Scraping static website using nightmare

At the first line, we initialize the **nightmare** and set the **show** property **true** so we can monitor what the **browser** is doing on execution. We made a request to url through `goto` function and wait for the DOM to be rendered through `wait` function else it executes the next followup steps without rendering of the content completely. Once the body is loaded completely, we fetch the innerHTML using `evaluate` function and return the data. At last, we close the browser by calling `end` function.

You can use try/catch blocks but if you are using promises, `.then()` must be called after `.end()` to run the `.end()` task.

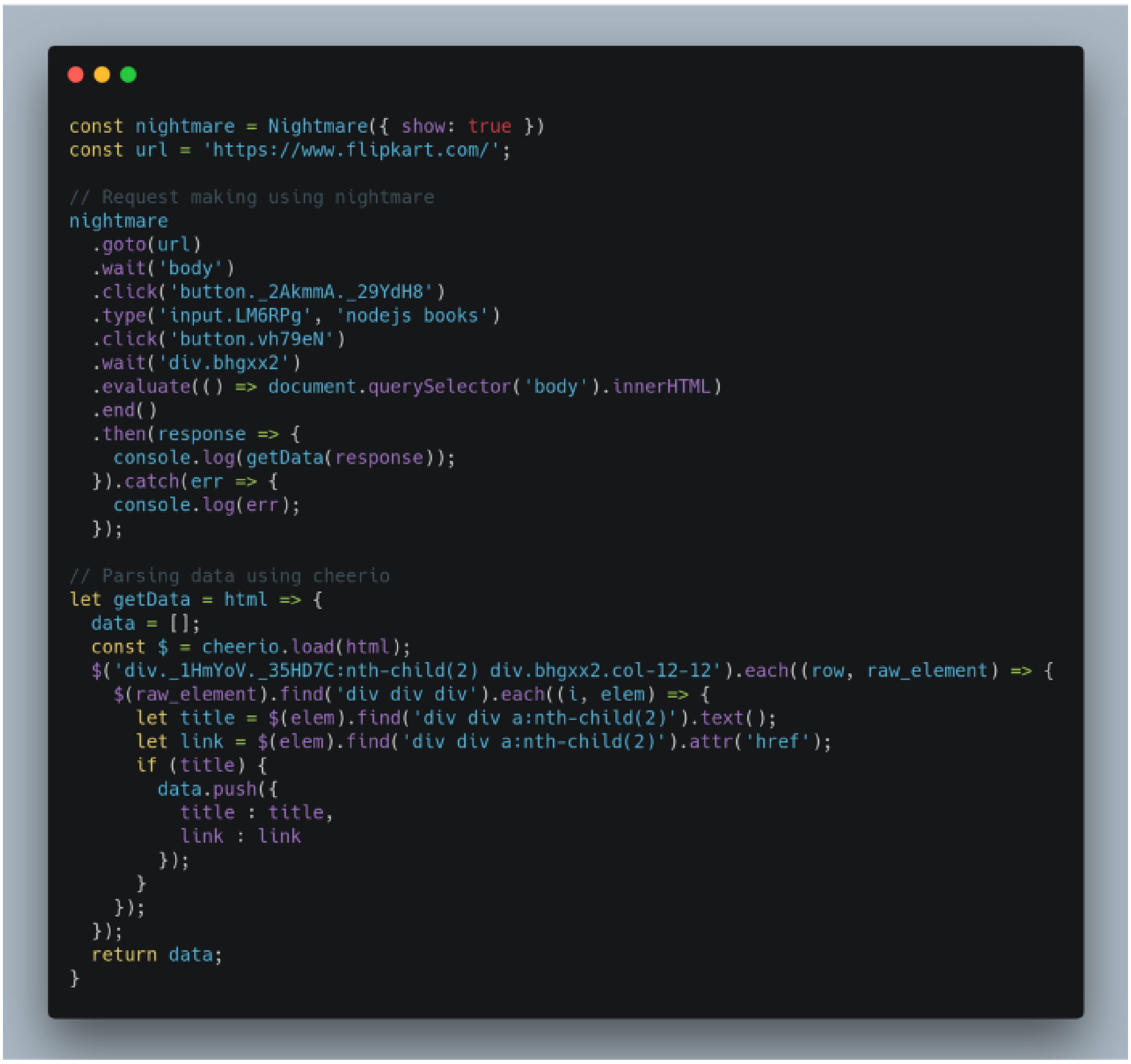
Scrape the dynamic content

Now, we are familiar with nightmare and how it works. So, let’s try to scrape content from any website which uses javascript to render data. Apart from it, nightmare also interact with websites like we can make clicks, fill the forms, so let’s do this. We will scrape the products listed on the flipkart.com website but only those which appears in the search for **nodejs books**



nodejs books

For interacting with the webpage, we need to use `click` and `type` function. Let's write some code.



```
const nightmare = Nightmare({ show: true })
const url = 'https://www.flipkart.com/';

// Request making using nightmare
nightmare
  .goto(url)
  .wait('body')
  .click('button._2AkmmA._29YdH8')
  .type('input.LM6RPg', 'nodejs books')
  .click('button.vh79eN')
  .wait('div.bhgxx2')
  .evaluate(() => document.querySelector('body').innerHTML)
  .end()
  .then(response => {
    console.log(getData(response));
  }).catch(err => {
    console.log(err);
  });

// Parsing data using cheerio
let getData = html => {
  data = [];
  const $ = cheerio.load(html);
  $('div._1HmYoV._35HD7C:nth-child(2) div.bhgxx2.col-12-12').each((row, raw_element) => {
    $(raw_element).find('div div div').each((i, elem) => {
      let title = $(elem).find('div div a:nth-child(2)').text();
      let link = $(elem).find('div div a:nth-child(2)').attr('href');
      if (title) {
        data.push({
          title : title,
          link : link
        });
      }
    });
  });
  return data;
}
```

Scrape dynamic website using nightmare

In the first part, we are making a request to the Flipkart website and enter **nodejs books** in the search bar by selecting the appropriate HTML selector using `type` function. Selectors can be found by inspecting the HTML using Chrome DevTools. After that, we will click on the search button using `click` function. On clicking, It loads the requested content similar to any other browsers and we will fetch the innerHTML for scraping content as described in the second part using cheerio.

The Output will look like —

```
[
  {
    title: 'Node.js Design Patterns',
    link: '/node-js-design-patterns/p/itmfbg2fhurfy97n?
pid=9781785885587&lid=LSTBOK9781785885587OWNSQH&marketplace=FLIPKART
&srno=s_1_1&otracker=search&fm=SEARCH&iid=1b09be55-0fad-44c8-a8ad-
e38656f3f1b0.9781785885587.SEARCH&ppt=Homepage&ppn=Homepage&ssid=dsi
scm68uo0000001547989841622&qH=ecbb3518fc5ee1c0'
  },
  {
    title: 'Advanced Node.js Development',
    link: '/advanced-node-js-development/p/itmf4eg8asapzfeq?
pid=9781788393935&lid=LSTBOK9781788393935UKCC4O&marketplace=FLIPKART
&srno=s_1_2&otracker=search&fm=SEARCH&iid=b7ec6a1b-4e79-4117-9618-
085b894a11dd.9781788393935.SEARCH&ppt=Homepage&ppn=Homepage&ssid=dsi
scm68uo0000001547989841622&qH=ecbb3518fc5ee1c0'
  },
  {
    title: 'Deploying Node.js',
    link: '/deploying-node-js/p/itmehyfmeqdbxwg5?
pid=9781783981403&lid=LSTBOK9781783981403BYQFE4&marketplace=FLIPKART
&srno=s_1_3&otracker=search&fm=SEARCH&iid=7824aa08-f590-4b8d-96ec-
4df08cc1a4bb.9781783981403.SEARCH&ppt=Homepage&ppn=Homepage&ssid=dsi
scm68uo0000001547989841622&qH=ecbb3518fc5ee1c0'
  },
  .


.
]
```

Now we have an array of JavaScript Object containing the title and links of the products(books) from the Flipkart website. In this way, we can scrape the data from any dynamic websites.

Conclusion

In this article, we first understood what is dynamic websites and how we can scrape data using **nightmare** and **cheerio** regardless of the type of website. The code is available in this Github Repo. Feel free to contribute and open issues.

Feel free to comment and ask me anything. You can follow me on Twitter and Medium. Thanks for reading! 👍

. . .

Learn more

How to perform web-scraping using Node.js

We will use Axios and Cheerio for web scraping!

blog.bitsrc.io

5 Tools for Faster Development in React

5 tools to speed the development of your React application, focusing on components.

blog.bitsrc.io

11 React UI Component Libraries you Should Know in 2019

11 React component libraries with great components for building your next app’s UI interface in 2019.

blog.bitsrc.io

Component Discovery and Collaboration · Bit

Bit is where developers share components and collaborate to build amazing software together. Discover components shared…

bit.dev

JavaScript Nodejs Web Development Web Scraping Programming

About Help Legal