# How to Perform Web-Scraping using Node.js

Ankit Jain
Dec 25, 2018 · 5 min read

Web Scraping

In this post, we'll learn how to use Node.js and it's packages to perform a quick and effective web-scraping for single-page applications. This can help us gather and use valuable data which isn't always available via APIs. Let's dive in.

## Tip: Share and reuse JS Modules with bit.dev

Use **Bit** to encapsulate modules/components with all their dependencies and setup. Share them in Bit's cloud, collaborate with your team and use them anywhere.

## What is web scraping?

Web scraping is a technique used to extract data from websites using a script. Web scraping is the way to automate the laborious work of copying data from various websites.

Web Scraping is generally performed in the cases when the desirable websites don't expose the API for fetching the data. Some common web scraping scenarios are:

1. Scraping **emails** from various websites for sales leads.
2. Scraping **news** headlines from news websites.
3. Scraping **product's data** from E-Commerce websites.

Why do we need web scraping when e-commerce websites expose the API (Product Advertising APIs) for fetching/collecting product's data?

E-Commerce websites only expose some of their product's data to be fetched through APIs therefore, web scraping is the more effective way to collect the maximum product's data.

Product comparison sites generally do web scraping. Even Google Search Engine does crawling and scraping to index the search results.

## What will we need?

Getting started with web scraping is easy and it 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 for web-scraping. If you're not familiar with Node, check out this article **"The only NodeJs introduction you'll ever need"**.

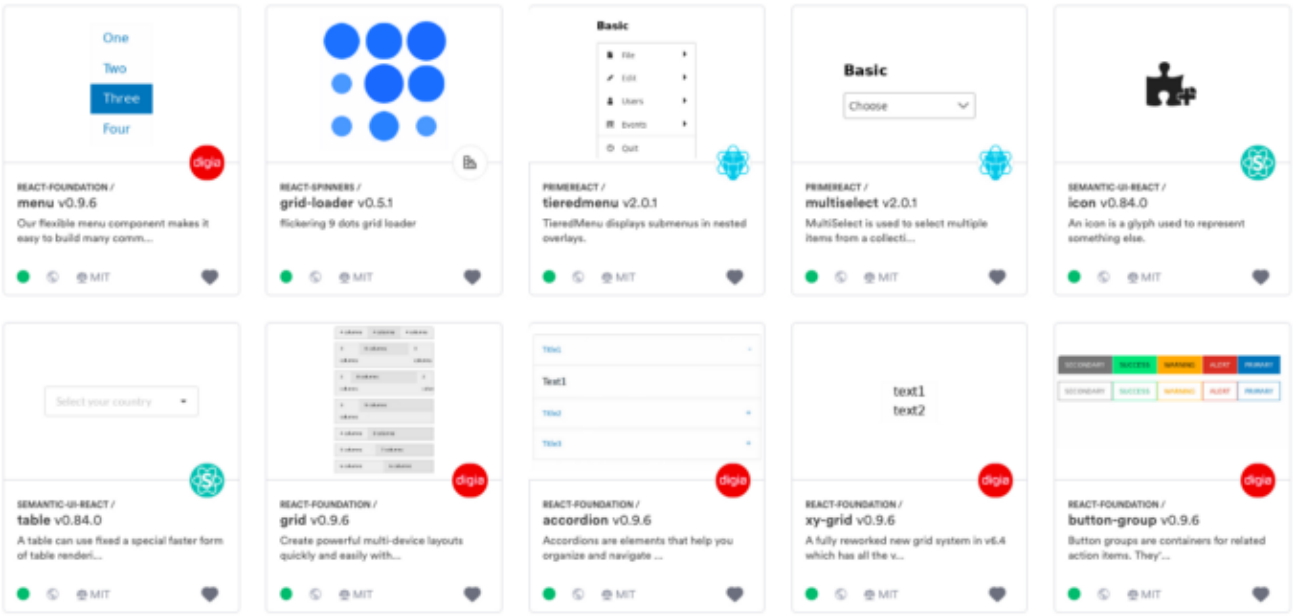
We will also use two open-source npm modules:

- **axios** — Promise based HTTP client for the browser and node.js.
- **cheerio** — jQuery for Node.js. Cheerio makes it easy to select, edit, and view DOM elements.

You can learn more about comparing popular HTTP request libraries here.

. . .

**Tip**: Don't duplicate common code. Use tools like **Bit** to organize, share and discover components across apps- to build faster. Take a look.




**Component Discovery and Collaboration · Bit**

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

bit.dev

. . .

## Setup

Our setup is pretty simple. We create a new folder and run this command inside that folder to create a **package.json** file. Let's cook the recipe to make our food delicious.

```
npm init -y
```

Before we start cooking, let's collect the ingredients for our recipe. Add **Axios** and **Cheerio** from npm as our dependencies.

```
npm install axios cheerio
```

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

```
const axios = require('axios');
const cheerio = require('cheerio');
```

## Make the Request

We are done with collecting the ingredients for our food, let's start with the cooking. We are scraping data from the **HackerNews** website for which we need to make an HTTP request to get the website's content. That's where axios come into action.

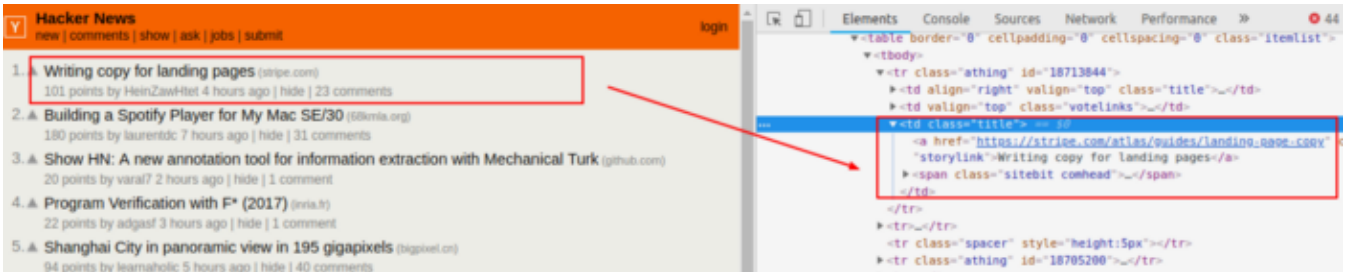
HTTP request using axios

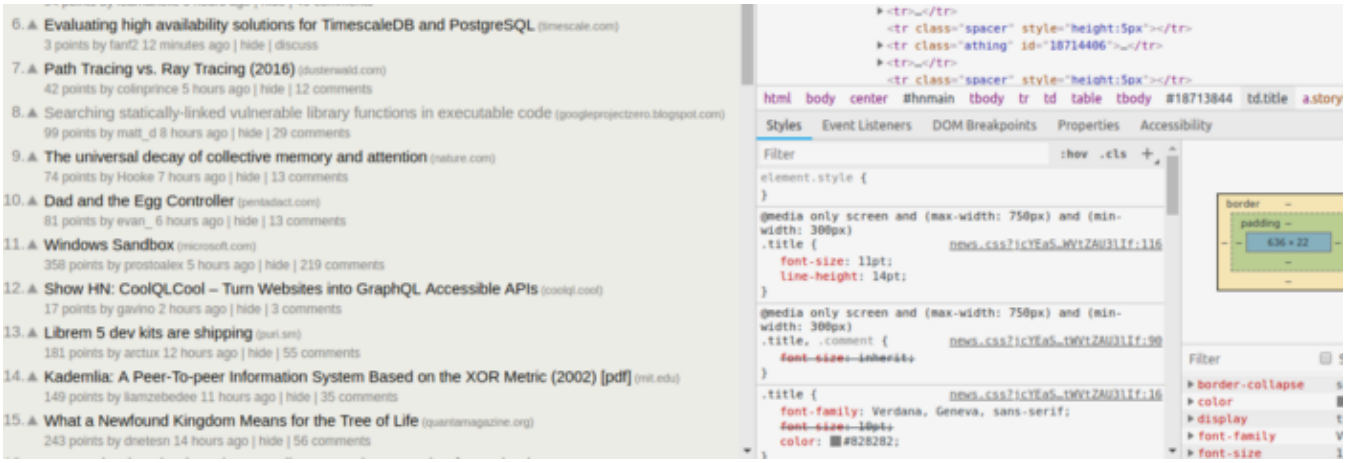
Our response will look like this —

```
<html op="news">
<head>
<meta name="referrer" content="origin">
<meta name="viewport" content="width=device-width, initial-
scale=1.0">
<link rel="stylesheet" type="text/css" href="news.css?
oq5SsJ3ZDmp6sivPZMMb">
<link rel="shortcut icon" href="favicon.ico">
<link rel="alternate" type="application/rss+xml" title="RSS"
href="rss">
<title>Hacker News</title>
</head>
<body>
<center>
<table id="hnmain" border="0" cellpadding="0" cellspacing="0"
width="85%" bgcolor="#f6f6ef">
.
.
.
</body>
<script type='text/javascript' src='hn.js?oq5SsJ3ZDmp6sivPZMMb'>
</script>
</html>
```

We are getting similar HTML content which we get while making a request from Chrome or any browser. Now we need some help of **Chrome Developer Tools** to search through the HTML of a web page and select the required data. You can learn more about the Chrome DevTools from here.

We want to scrape the News heading and its associated links. You can view the HTML of the webpage by right-clicking anywhere on the webpage and selecting “Inspect”.





Chrome DevTools to inspect the HTML

## Parsing HTML with Cheerio.js

Cheerio is the jQuery for Node.js, we use selectors to select tags of an HTML document. The selector syntax was borrowed from jQuery. Using Chrome DevTools, we need to find selector for news headlines and its link. Let's add some spices to our food.

```
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')
    });
  });
  console.log(data);
}

getData(response.data)
```

Parsing HTML with Cheerio.js

First, we need to load in the HTML. This step in jQuery is implicit since jQuery operates on the one, baked-in DOM. With Cheerio, we need to pass in the HTML document. After loading the HTML, we iterate all the occurrences of the table row <tr> to scrape each and every news on the page.

The Output will look like —

```
[
  {
    title: 'Malaysia seeks $7.5B in reparations from Goldman Sachs
(reuters.com)',
    link: 'https://www.reuters.com/article/us-malaysia-politics-
1mdb-goldman/malaysia-seeks-7-5-billion-in-reparations-from-goldman-
sachs-ft-idUSKCN1OK0GU'
  },
  {
    title: 'The World Through the Eyes of the US (pudding.cool)',
    link: 'https://pudding.cool/2018/12/countries/'
  },
  .
  .
  .
]
```

Now we have an array of JavaScript Object containing the title and links of the news from the HackerNews website. In this way, we can scrape the data from various large number of websites. So, our food is prepared and looks delicious too.

## Conclusion

In this article, we first understood what is web scraping and how we can use it for automating various operations of collecting data from various websites.

Many websites are using **Single Page Application** (SPA) architecture to generate content dynamically on their websites using JavaScript. We can get the response from the initial HTTP request and can't execute the javascript for rendering dynamic content using axios and other similar npm packages like request. Hence, we can only scrape data from static websites.

Don't miss out on part 2: "how to scrape data from dynamic websites".

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

Scrape dynamic websites using nightmare and cheerio

blog.bitsrc.io

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

. . .

## Learn more

**6 JavaScript User Authentication Libraries for 2019**

"Build me a user-authentication in two weeks!" — Useful ways to get the job done, quick and effective.

blog.bitsrc.io

**The Most In-Demand JavaScript Frameworks for Developers in 2019**

2018 "State of JS" report is here- which frameworks will rule 2019?

blog.bitsrc.io

**SOLID Principles every Developer Should Know**

A short yet detailed introduction to understanding SOLID design principles.

blog.bitsrc.io

JavaScript Nodejs Web Development Programming Web Scraping