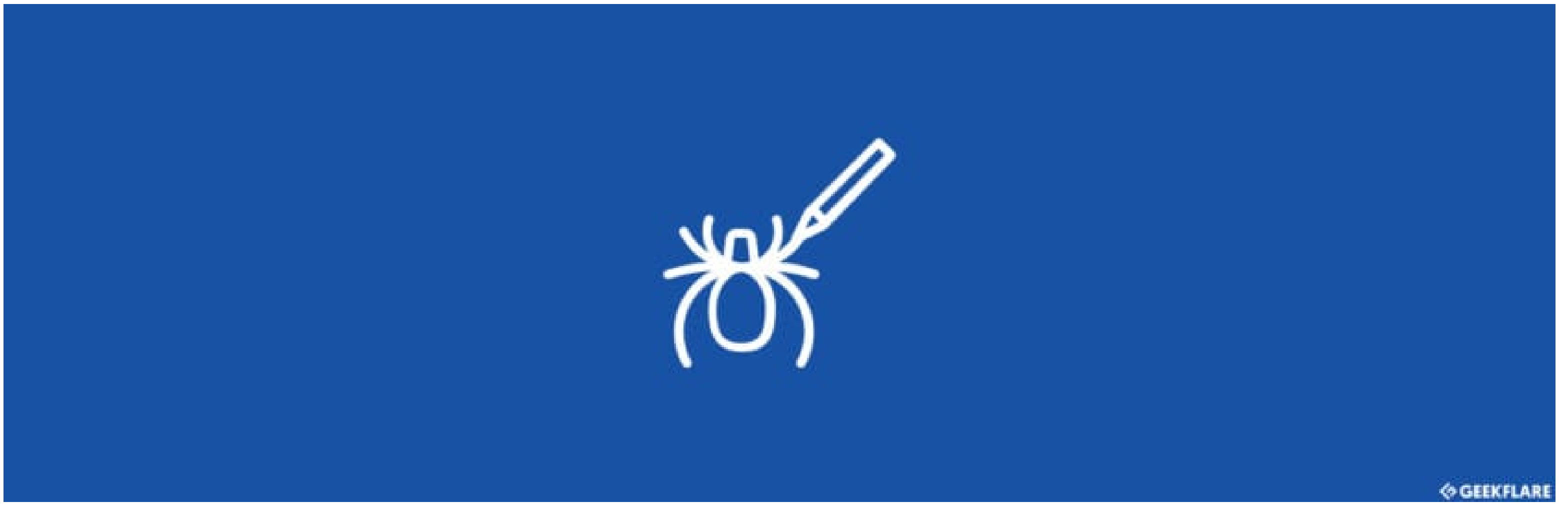
By Hafeezul Kareem Shaik in Development | Last updated: April 4, 2021

Share on:

Getting Started with Web Scraping in JavaScript



Web scraping is one of the most interesting things in the coding world.

What is web scraping?

Why is it even exist?

Let's find out the answers.

What is Web Scraping?

Web scraping is an automated task to extract data from websites.

There are many applications of web scraping. Extracting the prices of products and comparing them with different e-Commerce platforms. Getting a daily quote from the web. Building your own search engine like Google, Yahoo, etc.., The list goes on.

The program which extracts the data from websites is called a **web scraper**. You are going to learn to write web scrapers in JavaScript.

There are mainly two parts to web scraping.

- Getting the data using request libraries and a headless browser.
- Parsing the data to extract the exact information that we want from the data.

Without further ado let's get started.

Project Setup

I assume you have Node installed, if not check out the NodeJS installation guide.

We are going to use the packages `node-fetch` and `cheerio` for web scraping in JavaScript. Let's set up the project with the npm to work with a third-party package.

Let's quickly see the steps to complete our setup.

- Create a directory called `web_scraping` and navigate to it.
- Run the command `npm init` to initialize the project.
- Answer all the questions based on your preference.
- Now, install the packages using the command

```
npm install node-fetch cheerio
```

Copy

Let's see the glimpses of the installed packages.

node-fetch

The package `node-fetch` brings the `window.fetch` to the node js environment. It helps to make the HTTP requests and get the raw data.

cheerio

The package cheerio is used to parse and extract the information that is necessary from the raw data.

Two packages `node-fetch` and `cheerio` are good enough for web scraping in JavaScript. We are not going to see every method that the packages are providing. We will see the flow of web scraping and

Scraping Cricket World Cup List

Here in this section, we are going to do actual web scraping.

What are we extracting?

By the title of the section, I think you would easily guess it. Yeah, whatever you are thinking is correct. Let's extract all cricket world cup winners and runner-ups till now.

- Create a file called `extract_cricket_world_cups_list.js` in the project.
- We will be using the Wikipedia Cricket World Cup page to get the desired information.
- First, get the raw data using the `node-fetch` package.
- Below code gets the raw data of the above Wikipedia page.

```
const fetch = require("node-fetch");

// function to get the raw data
const getRawData = (URL) => {
   return fetch(URL)
      .then((response) => response.text())
      .then((data) => {
         return data;
      });
};

// URL for data
const URL = "https://en.wikipedia.org/wiki/Cricket_World_Cup";

// start of the program
const getCricketWorldCupsList = async () => {
   const cricketWorldCupRawData = await getRawData(URL);
   console.log(cricketWorldCupRawData);
};

// invoking the main function
getCricketWorldCupsList();
```

Copy

Extracting data that involves HTML tags with cheerio is a cakewalk. Before getting into the actual data, let's see some sample data parsing using `cheerio` .

- Parse the HTML data using `cheerio.load` the method.

```
const parsedSampleData = cheerio.load(
      `<div id="container"><p id="title">I'm title</p></div>`
   );
```

Copy

- We have parsed the above HTML code. How to extract the `p` tag content from it? It's the same as the selectors in JavaScript DOM manipulation.

```
console.log(parsedSampleData("#title").text());
```

You can select the tags as you want. You can check out different methods from the cheerio official website.

- Now, it's time to extract the world cup list. To extract the information, we need to know the HTML tags that information lies on the page. Go to the cricket world cup Wikipedia page and inspect the page to get HTML tags information.

Here is the complete code.

```
const fetch = require("node-fetch");
const cheerio = require("cheerio");

// function to get the raw data
const getRawData = (URL) => {
   return fetch(URL)
      .then((response) => response.text())
      .then((data) => {
         return data;
      });
};

// URL for data
const URL = "https://en.wikipedia.org/wiki/Cricket_World_Cup";

// start of the program
const getCricketWorldCupsList = async () => {
```

```
// parsing the data
const parsedCricketWorldCupData = cheerio.load(cricketWorldCupRawData);

// extracting the table data
const worldCupsDataTable = parsedCricketWorldCupData("table.wikitable")[0]
   .children[1].children;

console.log("Year --- Winner --- Runner");
worldCupsDataTable.forEach((row) => {
   // extracting `td` tags
   if (row.name === "tr") {
      let year = null,
         winner = null,
         runner = null;

      const columns = row.children.filter((column) => column.name === "td");

      // extracting year
      const yearColumn = columns[0];
      if (yearColumn) {
         year = yearColumn.children[0];
         if (year) {
            year = year.children[0].data;
         }
      }

      // extracting winner
      const winnerColumn = columns[3];
      if (winnerColumn) {
         winner = winnerColumn.children[1];
         if (winner) {
            winner = winner.children[0].data;
         }
      }

      // extracting runner
      const runnerColumn = columns[5];
      if (runnerColumn) {
         runner = runnerColumn.children[1];
         if (runner) {
            runner = runner.children[0].data;
         }
         if (year && winner && runner) {
            console.log(`${year} --- ${winner} --- ${runner}`);
         }
      }
   });
};

// invoking the main function
getCricketWorldCupsList();
```

Copy

And, here is the scraped data.

```
Year --- Winner --- Runner
1975 --- West Indies --- Australia
1979 --- West Indies --- England
1983 --- India --- West Indies
1987 --- Australia --- England
1992 --- Pakistan --- England
1996 --- Sri Lanka --- Australia
1999 --- Australia --- Pakistan
2003 --- Australia --- India
2007 --- Australia --- Sri Lanka
2011 --- India --- Sri Lanka
2015 --- Australia --- New Zealand
2019 --- England --- New Zealand
```

Copy

Cool 😎, is int' it?

Scraping Template

Getting the raw data from the URL is common in every web scraping project. The only part that changes is extracting the data as per the requirement. You can try the below code as a template.

```
const fetch = require("node-fetch");
const cheerio = require("cheerio");
const fs = require("fs");
// function to get the raw data
const getRawData = (URL) => {
        .then((data) => {
            return data;
        });
};
// URL for data
const URL = "https://example.com/";
// start of the program
const scrapeData = async () => {
    const rawData = await getRawData(URL);
    // parsing the data
    const parsedData = cheerio.load(rawData);
    console.log(parsedData);
    // write code to extract the data
    // here
    // ...
    // ...
};
// invoking the main function
scrapeData();
```

Copy

Conclusion

You have learned how to scrape a webpage. Now, it's your turn to practice coding.

I would also suggest checking out popular web scraping frameworks to explore and cloud-based web-scraping solutions.

Happy Coding 🙂

Tagged as
JavaScript

Thanks to our Sponsors

More great readings on Development

8 Best Email Testing Tools to Use for your Mailer Campaigns

By Tanish Chowdhary on February 1, 2023

Email marketing isn’t dead. Even with the growing popularity of social media platforms for communication, emails are still one of the best methods of reaching out to people.

7 Best IDE for Linux to Develop Complex Software with Ease

By Murtuza Surti on February 1, 2023

IDEs play a vital role in increasing developers’ productivity by offering a consolidated environment to write computer code. Check out the best IDE for Linux.

Ultimate Guide to Security Information and Event Management

By John Walter on February 1, 2023

Meta Description: (SIEM) Security information and event management is a way for organizations to detect threats before they shatter their business.

11 Best Flashcard Apps to Help You Learn Faster [2023]

By Shalabh Garg on February 1, 2023

Flashcard applications can make your learning process faster, easier, and efficient. Check these powerful flashcard apps to enhance your learning experience.

How to Add Social Proof on Site to Increase Conversions?

By Adnan Rehan on February 1, 2023

Adding social proof can massively increase your conversions.

Top 13 Content Marketing Tools for Growth and Engagement

By Ruby Goyal on February 1, 2023

Digital marketing experts don’t lie when they say: Content is the king. Content is the driving force in this new age of data and information.

Power Your Business

Some of the tools and services to help your business grow.