JavaScript was initially developed as a browser-only language but has now evolved to be everywhere, even on hardware like microcontrollers and wearables.

For the purposes of SEO automation, there are two main environments where you can automate SEO tasks with JavaScript:

* A browser (front-end).
* Directly on a computer/laptop (back-end).
* **SEO Automation with Your Browser**
* One of the main advantages that separate JavaScript from other scripting languages is that browsers can execute JavaScript. This means the only thing you need to get started with JavaScript automation is a browser.
* **Automation Using the Browser’s Console**
* The easiest way to get started is using JavaScript directly in your browser’s console.
* There are some easy and fun automations you can do. For example, you can make any website editable by typing “*document.body.contentEditable = true*” in your console.
* This could be useful for mocking up new content or headings on the page to show to your clients or other stakeholders in your company without the need for image editing software.
* But let’s step it up a bit more.
* **The Lesser-Known Bookmarklets**
* Since a browser’s console can run JavaScript, you can create custom functions that perform specific actions like extracting information from a page.
* However, creating functions on the spot can be a bit tedious and time-consuming. Therefore, Bookmarklets are a simpler way to save your own custom functions without the need for browser plugins.
* Bookmarklets are small code snippets saved as browser bookmarks that run functions directly from the browser tab you are on.

For example, Dominic Woodman created a bookmarklet [here](https://www.domwoodman.com/posts/search-console-crawl-data-bookmarklet/) that allows users to extract crawl stats data from the old Google Search Console user interface and download it to a CSV.

It might sound a bit daunting, but you can learn how to create your own Bookmarklets following the steps in [this great resource on GitHub](https://gist.github.com/caseywatts/c0cec1f89ccdb8b469b1).

**Snippets, a User-Friendly Version of Bookmarklets**

If you use Chrome, there is an even simpler solution using [Snippets](https://developer.chrome.com/docs/devtools/javascript/snippets/). With this, you can create and save the same type of functions in a much more user-friendly way.

For example, I’ve created a small Snippet that checks how many “crawlable” links are on-page and download the list to a CSV file. You can [download the code from GitHub here](https://github.com/jlhernando/chrome-snippets/blob/master/link-counter.js).

While these are usually small tasks that are “nice to have” you’d probably want to do more heavy-lifting tasks that can help with your SEO workload in a more significant way.

Therefore, it’s much better to use JavaScript directly on your laptop (or a server in the Cloud) using Node.js.

### SEO Automation in the Back-End with Node.js

Node.js is software that lets you run JavaScript code on your laptop without the need for a browser.

There are some differences between running JavaScript in your browser and JavaScript on your laptop (or a server in the Cloud) but we’ll skip these for now as this is just a short intro to the topic.

In order to run scripts with Node.js, you must have it installed on your laptop. I have written a short blog post where I go step by step on [how you can install Node](https://jlhernando.com/blog/how-to-install-node-for-seo/) as well as a few additional setup tips to make it easier to get you started.

Although your imagination is the limit when it comes to automation, I’ve narrowed it down to a few areas that I see SEO professionals come back to when using Node.js.

I will include scripts that are ready to run so you can get started in no time.

**Extracting Data From APIs**

Collecting information from different sources to provide insights and recommend actions is one of the most common jobs in SEO.

Node.js makes this incredibly simple with different options, but my preferred go-to module is [Axios](https://www.npmjs.com/package/axios" \t "_blank).

// Create an index.js file inside a folder & paste the code below

// Import axios module

const axios = require('axios');

// Custom function to extract data from PageSpeed API

const getApiData = async (url) => {

const endpoint = 'https://www.googleapis.com/pagespeedonline/v5/runPagespeed';

const key = 'YOUR-GOOGLE-API-KEY' // Edit with your own key;

const apiResponse = await axios(`${endpoint}?url=${url}&key=${key}`); // Create HTTP call

console.log(apiResponse.data); // Log data

return apiResponse.data;

};

// Call your custom function

getApiData('https://www.searchenginejournal.com/');

To start interacting with APIs, you need a module that’s able to handle HTTP requests (HTTP client) and an endpoint (a URL to extract information).

In some cases, you might also need an API key, but this is not always necessary.

For a taste of how Node.js interacts with APIs, check out [this script](https://github.com/jlhernando/google-pagespeed-bulk) I published that uses Google’s PageSpeed API to extract Core Web Vitals data and other lab metrics in bulk.

**Scraping Websites**

Whether you would like to monitor your own website, keep an eye on your competitors or simply extract information from platforms that don’t offer an API, scraping is an incredibly useful tool for SEO.

Since JavaScript interacts well with the DOM, there are many advantages to using Node.js for scraping.

The most common module I’ve used for scraping is Cheerio, which has a very similar syntax to jQuery, in combination with an HTTP client like Axios.

// Import modules

const cheerio = require('cheerio');

const axios = require('axios');

// Custom function to extract title

const getTitle = async (url) => {

const response = await axios(url); // Make request to desired URL

const $ = cheerio.load(response.data); // Load it with cheerio.js

const title = $('title').text(); // Extract title

console.log(title); // Log title

return title;

};

// Call custom function

getTitle('https://www.searchenginejournal.com/');

If you need the rendered version of a website, popular modules like [Puppeteer](https://developers.google.com/web/tools/puppeteer) or [Playwright](https://playwright.dev/) can launch a headless instance of an actual browser like Chrome or Firefox and perform actions or extract information from the DOM.

[Chris Johnson’s](https://twitter.com/defaced) Layout Shift Generator is a great example of how to use Puppeteer for SEO. You can find [more info here](https://defaced.dev/tools/layout-shift-gif-generator/) or [download the script here](https://github.com/workeffortwaste/layout-shift-gif).

There are also other options like JSDOM that emulate what a browser does without the need for a browser. You can play around with a JSDOM-based script using this [Node.js SEO Scraper](https://github.com/NachoSEO/seo-scraper) built by [Nacho Mascort](https://twitter.com/NachoMascort).

**Processing CSV and JSON Files**

Most of the time, the data extracted from APIs comes as JSON objects, and JavaScript is perfect for dealing with those.

However, as SEOs, we normally deal with data in spreadsheets.

Node.js can easily handle both formats using built-in modules like the [File System module](https://nodejs.org/api/fs.html) or more simplified versions like [csvtojson](https://www.npmjs.com/package/csvtojson" \t "_blank)or [json2csv](https://www.npmjs.com/package/json2csv).

Whether you want to read data from a CSV and transform it into JSON for processing, or you’ve already manipulated the data and you need to output to CSV, Node.js has your back.

// Import Modules

const csv = require('csvtojson');

const { parse } = require('json2csv');

const { writeFileSync } = require('fs');

// Custom function to read URLs and convert it to JSON

const readCsvExportJson = async () => {

const json = await csv().fromFile('yourfile.csv');

console.log(json); // Log conversion JSON

const converted = parse(json);

console.log(converted); // Log conversion to CSV

writeFileSync('new-csv.csv', converted);

};

readCsvExportJson();

**Create Cloud Functions to Run Serverless Tasks**

This is a more advanced case, but it’s incredibly helpful for technical SEO.

Cloud computing providers like Amazon AWS, Google Cloud Platform, or Azure make it incredibly simple (and cheap) to set up “instances of servers” that run custom scripts created for specific purposes without the need for you to configure that server.

A useful example would be to schedule a function that extracts data from the Google Search Console API automatically at the end of every day and stores the data into a BigQuery database.

There are a few moving parts in this case, but the possibilities are truly endless

Just to show you an example of how to create a cloud function, check out this [episode of Agency Automators](https://www.youtube.com/watch?v=Ec866XolPvU) where [Dave Sottimano](https://twitter.com/dsottimano) creates his own Google Trends API using Google Cloud Functions.

### A Potential Third Avenue, Apps Script

For me, it made more sense to start with the less-opinionated approach.

But Apps Script may offer a less intimidating way to learn to code because you can use it in apps like Google Sheets which are the bread and butter of technical SEO.

There are really useful projects that can give you a sense of what you can do with Apps Script.

For example, [Hannah Butler](https://twitter.com/HannahRampton)’s [Search Console explorer](https://www.hannahrampton.co.uk/search-console-explorer-sheet-free-google-sheet/) or [Noah Lerner](https://twitter.com/noahlearner)’s [Google My Business Postmatic](https://workspace.google.com/marketplace/app/%F0%9F%A4%96_postamatic/460920737213) for local SEO.

If you are interested in learning Apps Script for SEO, I would recommend [Dave Sottimano](https://twitter.com/dsottimano)’s [Introduction to Google Apps Scripts](https://opensourceseo.org/introduction-google-apps-scripts/). He also gave this [awesome presentation](https://www.youtube.com/watch?v=voCyYmcdGP0) at Tech SEO Boost, which explains many ways to use Apps Script for SEO.