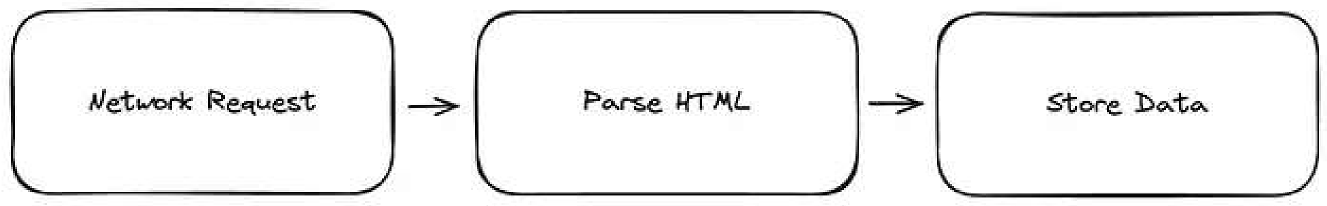
Intro to Web Scraping: Build Your First Scraper in 5 Minutes

A quick guide to help you build a simple web scraper.

Joe Osborne Follow 5 min read · Apr 8, 2024



Web scraping is my favorite area of coding. It's both incredibly frustrating and extremely rewarding. The best thing about scraping is that the possibilities are endless. There's virtually infinite data on the internet, and a lot of it is super easy to collect with a simple scraper.

Getting started is very easy. When I began, I had only written a few lines of code in my entire life, but I had the idea to scrape mortgage interest rates to watch their trend in real time. After a quick Google search and about 20 minutes later, I had built my first web scraper. I want to help others do the same!

I typically use either Python or JavaScript/TypeScript. Both are great options, it's mostly a matter of preference. In this guide I'll go over very simple examples for both.

Prerequisites:

- Python or Node.js installed on your machine depending on your language of choice.
- An IDE — I prefer PyCharm for Python and VSCode for JavaScript/TypeScript.
- If you'd like, you can follow along with my code using the GitHub repo I set up for this guide: https://github.com/thejoeosborne/simple-scraper

That's it! Let's get started.

Step 1

Create a new folder on your machine and a new .js or .py file inside the folder. Let's name them `scraper-python.py` and `scraper-javascript.js`. Open up the folder in your IDE of choice.

We'll quickly download a couple libraries to help us out.

If using Python, open up a terminal and run `pip install requests` and `pip install beautifulsoup4` .

If using JavaScript, run `npm init` and `npm install cheerio` .

For TypeScript specifically, you should run `npm install ts-node` which will allow you to run the scraper later.

Step 2

Choose a website to scrape and make a network request to the website. For this guide, we'll scrape example.com because it's very simple and doesn't have any blocking or authentication. Higher traffic sites like LinkedIn, Indeed, etc are notoriously difficult to scrape due to sophisticated bot detection.

In our code, we'll make a simple `GET` request to example.com. The website will send us back its HTML. In this step, we'll just print the HTML to the console. Later, we'll parse out specific pieces of it.

Python:

```python
# scraper-python.py
# To run this script, paste `python scraper-python.py` in the terminal

import requests
from bs4 import BeautifulSoup


def scrape():

    url = 'https://www.example.com'
    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'html.parser')
    print(soup)



if __name__ == '__main__':
    scrape()
```

JavaScript:

```
/**
 * scraper-javascript.js
 * To run this script, copy and paste `node scraper-javascript.js` in the termin
 */

const cheerio = require('cheerio');



(async () => {
  const url = 'https://www.example.com';
  const response = await fetch(url);

  const $ = cheerio.load(await response.text());
  console.log($.html());

})();
```

Go ahead and run your script by pasting either `python scraper-python.py` or `node scraper-javascript.js` in your terminal. Here's the result you should get from printing the HTML:

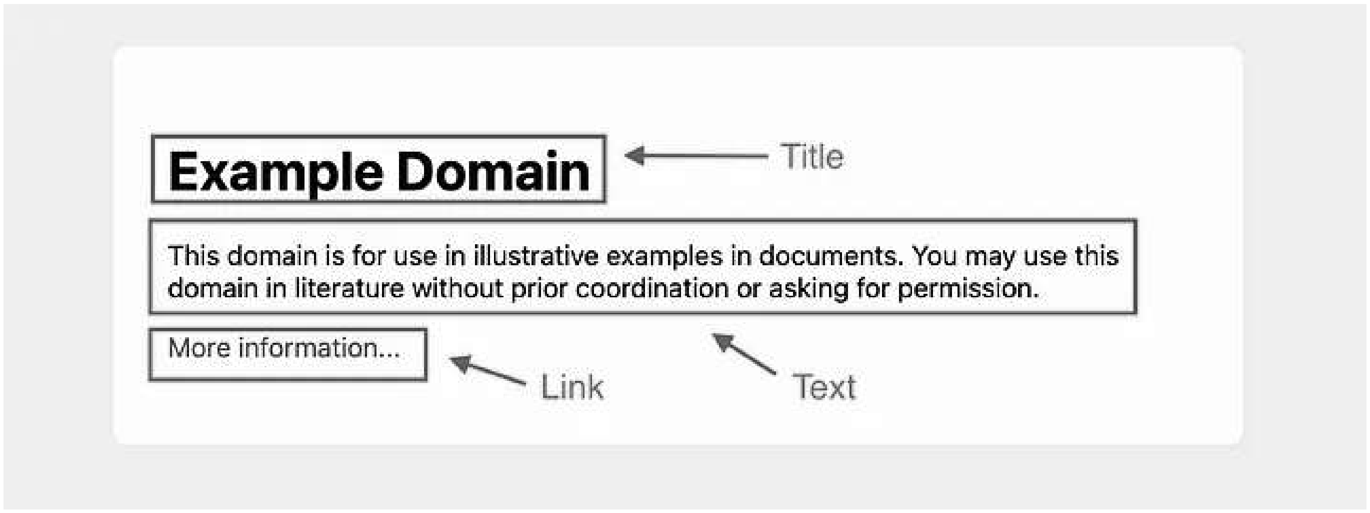
```
<!DOCTYPE html><html><head>
    <title>Example Domain</title>

    <meta charset="utf-8">
    <meta http-equiv="Content-type" content="text/html; charset=utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <style type="text/css">
    body {
        background-color: #f0f0f2;
        margin: 0;
        padding: 0;
        font-family: -apple-system, system-ui, BlinkMacSystemFont, "Segoe UI", "

    }
    div {
        width: 600px;
        margin: 5em auto;
        padding: 2em;
        background-color: #fdfdff;
        border-radius: 0.5em;
        box-shadow: 2px 3px 7px 2px rgba(0,0,0,0.02);
    }
    a:link, a:visited {
        color: #38488f;
        text-decoration: none;
    }
    @media (max-width: 700px) {
        div {
            margin: 0 auto;
            width: auto;
        }
    }
    </style>
</head>

<body>
<div>
    <h1>Example Domain</h1>
    <p>This domain is for use in illustrative examples in documents. You may use
    domain in literature without prior coordination or asking for permission.</p
    <p><a href="https://www.iana.org/domains/example">More information...</a></p
</div>


</body></html>
```

Nice! You made a network request to the website, and got back the HTML. Now, let's parse out specific parts of the page.

Step 3

`BeautifulSoup` and `cheerio` are libraries that help us navigate HTML in code. They allow us to pass in certain paths and patterns to get certain snippets of HTML.

Let’s go ahead and capture 3 things from this page: The title, the text, and the “More information…” link.



Example Domain

This domain is for use in illustrative examples in documents. You may use this domain in literature without prior coordination or asking for permission.

More information...

We’ll use CSS Selectors to find these elements in the HTML. CSS Selectors are notations used to locate HTML elements, and they are very easy to learn. Here’s a cheatsheet you can use. The ones we will use for this guide are very simple, but it’s worth your time to get familiar with more complex ones when scraping real websites. Figuring out the right selector is usually not too hard, and I constantly use Google and ChatGPT to help me come up with good ones.

Let’s capture the title, text, and extract the link from the `<a>` tag. Add these lines to your code.

Python:

```
title = soup.select_one('h1').text
text = soup.select_one('p').text
link = soup.select_one('a').get('href')

print(title)
print(text)
print(link)
```

JavaScript:

```
const title = $('h1').text();
const text = $('p').text();
const link = $('a').attr('href');

console.log(title);
console.log(text);
console.log(link);
```

After adding those lines to your script and running it, the console should print out this text:

```
Example Domain
This domain is for use in illustrative examples in documents. You may use this
    domain in literature without prior coordination or asking for permission.Mor
https://www.iana.org/domains/example
```

Congratulations! You have built a web scraper.

Becoming proficient at web scraping opens up endless possibilities. Especially with the recent advent of AI, mass data collection is more valuable than ever. It's also tons of fun and can be a rewarding hobby!

Resources

- GitHub repo containing the code for this guide: https://github.com/thejoeosborne/simple-scraper
- I do freelance scraping projects, so if you need some data collected feel free to contact me on LinkedIn!
- If you're interested in some more high-level enterprise scraping methods, I wrote an in-depth guide on deploying scrapers at scale.
- If you need premium proxies for difficult to scrape sites, I recommend Browserless.io or Oxylabs.
- For more scraping tutorials, I recommend McKay Johns. He has tons of easy to follow YouTube videos and a couple solid courses.

Web Scraping

Written by Joe Osborne

Follow

164 followers · 0 following

Hi! I'm a software engineer with early stage startup experience. Check out some of my work at https://joeosborne.me :)

Responses (9)

Write a response

What are your thoughts?

Esteban HD
Apr 10, 2024

Nice stuff

7 Reply

Mohit Singla
Oct 24, 2024

Thank you so much for the tutorial.

3 Reply

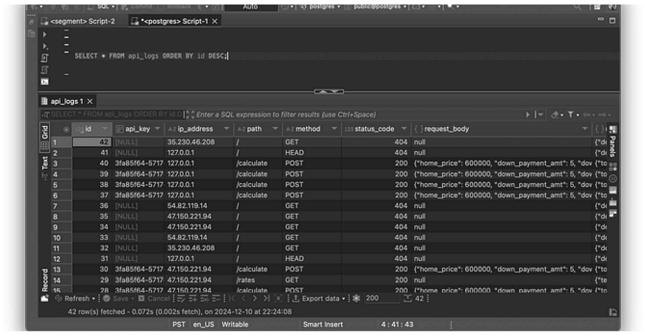
Jack Kirby
Sep 27, 2024

This is fantastic, thank you so much :)

2 Reply

See all responses

More from Joe Osborne



```
SELECT * FROM api_logs ORDER BY id DESC;
```

	id	api_key	ip_address	path	method	status_code	request_body	
1	42	[NULL]	35.230.46.208	/	GET	404	null	{"d
2	41	[NULL]	127.0.0.1	/	HEAD	404	null	{"d
3	40	3fa85f64-5717	127.0.0.1	/calculate	POST	200	{"home_price": 600000, "down_payment_amt": 5, "dov	{"to
4	39	3fa85f64-5717	127.0.0.1	/calculate	POST	200	{"home_price": 600000, "down_payment_amt": 5, "dov	{"to
5	38	3fa85f64-5717	127.0.0.1	/calculate	POST	200	{"home_price": 600000, "down_payment_amt": 5, "dov	{"to
6	37	3fa85f64-5717	127.0.0.1	/calculate	POST	200	{"home_price": 600000, "down_payment_amt": 5, "dov	{"to
7	36	[NULL]	54.82.119.14	/	GET	404	null	{"d
8	35	[NULL]	47.150.221.94	/	GET	404	null	{"d
9	34	[NULL]	47.150.221.94	/	GET	404	null	{"d
10	33	[NULL]	54.82.119.14	/	GET	404	null	{"d
11	32	[NULL]	35.230.46.208	/	GET	404	null	{"d
12	31	[NULL]	127.0.0.1	/	HEAD	404	null	{"d
13	30	3fa85f64-5717	47.150.221.94	/calculate	POST	200	{"home_price": 600000, "down_payment_amt": 5, "dov	{"to
14	29	3fa85f64-5717	47.150.221.94	/rates	GET	200	null	{"te
15	28	3fa85f64-5717	47.150.221.94	/calculate	POST	200	{"home_price": 600000, "down_payment_amt": 5, "dov	{"to

```
42 row(s) fetched - 0.072s (0.002s fetch), on 2024-12-10 at 22:24:08
```

Joe Osborne

## How to Log Every Request and Response in FastAPI

📕 A guide on efficiently storing logs for every request and response in your FastAPI...

Dec 13, 2024 👏 227 💬 3

Joe Osborne

## FastAPI with API Key Authentication

📘 How to set up your FastAPI app with a simple API key auth system.

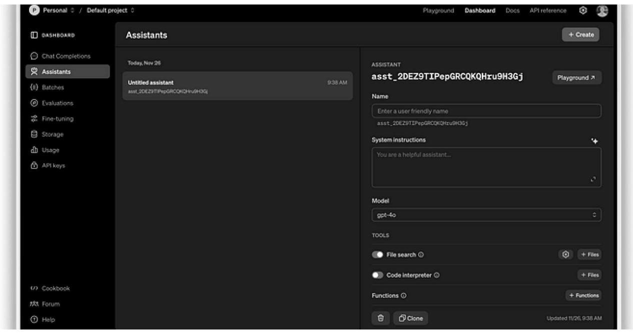
Dec 27, 2024 👏 98

Joe Osborne

## Web Scraping with Puppeteer Extra, Typescript, & AWS Lambda

An in-depth guide on how to build a web scraper using the best tools available.

Jan 21, 2024 👏 159 💬 3



Joe Osborne

## Guide: How to Build a Customized AI Assistant using OpenAI's...

An in-depth guide on how to create an assistant, create vector stores with your own...

Nov 27, 2024 👏 24 💬 1

See all from Joe Osborne

Recommended from Medium

alejandro

Logging + LLM + FastAPI

Simple Observability from Scratch for a FastAPI LLM Endpoint

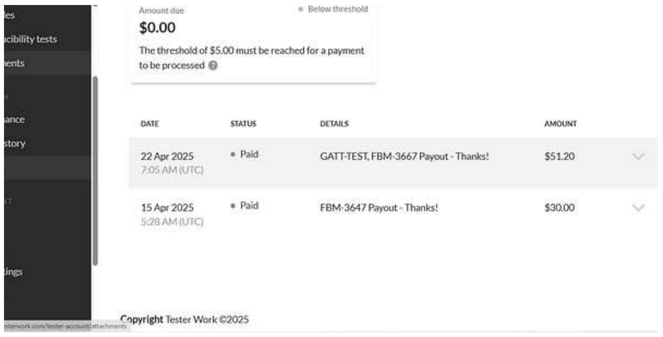
Jun 17 24

In Python in Plain English by Sanjay Prajapati

10 Killer Python Projects for 2025 That'll Level Up Your Skills Fast!

10 real Python projects in 2025 that go beyond to-do apps—build tools, bots, game...

May 24 114 1



Amount due · Below threshold

$0.00

The threshold of $5.00 must be reached for a payment to be processed

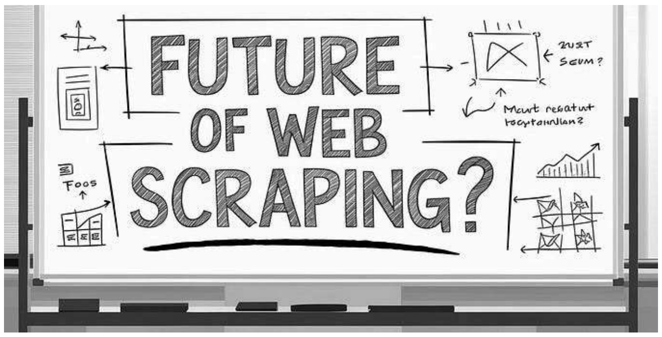
DATE	STATUS	DETAILS	AMOUNT
22 Apr 2025 7:05 AM (UTC)	Paid	GATT-TEST, FBM-3667 Payout - Thanks!	$51.20
15 Apr 2025 5:28 AM (UTC)	Paid	FBM-3647 Payout - Thanks!	$30.00

Copyright Tester Work ©2025

Healing Pathways

How I Made $81.20 in Just 2 Days

A Real Side Hustle, Anyone Can Do it (Just it needs your phone)



MD Kawsar

What Makes Jina.ai the Future of Web Scraping? - Free (for Now!)

A Guide to Getting Started with Jina.ai for Scraping

Apr 23 6.7K 274

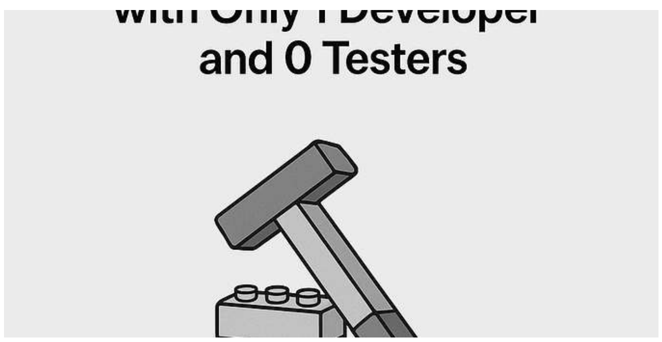
Feb 22 727 7

Joe Osborne

Easy Web Scraping: Scrape an Entire Website in Under 10 Minutes

📘 A simple guide on building and running a web scraper.

Mar 10 35 1



In QEvolution by Igor Goldshmidt

How I Would Build a QA Process in a Startup with Only 1 Developer a...

A practical guide to building just enough quality assurance when you're outnumbered...

Apr 4 53 1

See more recommendations