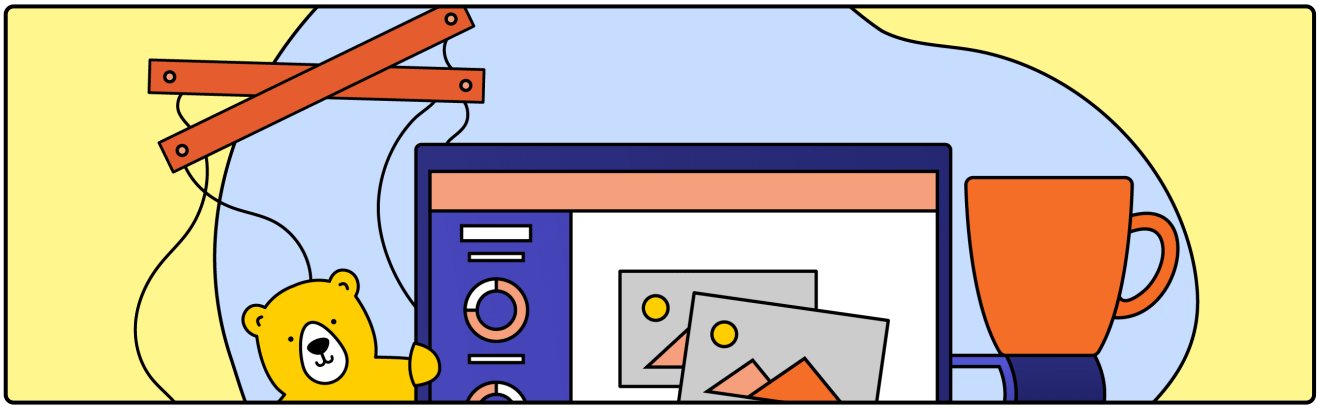
How to Download Images from a Website Using Puppeteer

This article will take you through steps to download images from a website using Puppeteer.

by Josephine Loo · June 2022



Contents

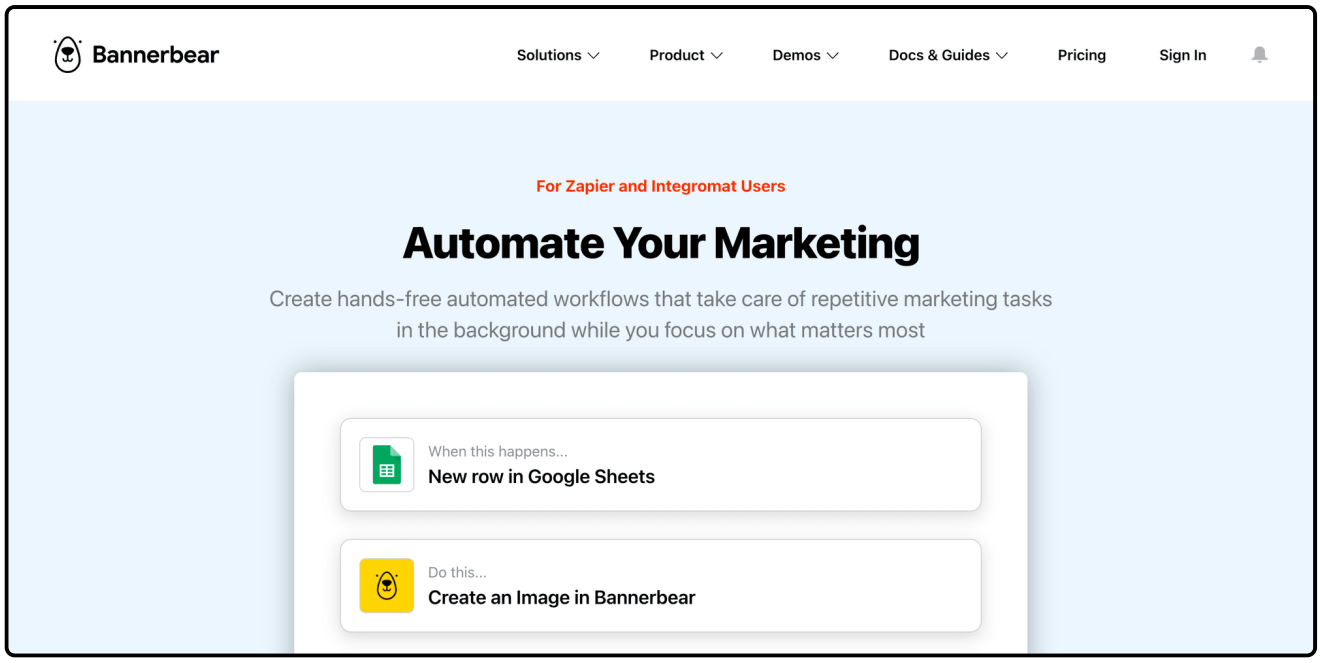
- What is Puppeteer
- Pre-requisites
- Getting Started
  - Step 1. Create a New Node.js Project

    - Step 2. Install Puppeteer
- Testing Puppeteer
    - Step 1. Write the Code
    - Step 2. Run the Code
- Downloading Images from a Website
    - Step 1. Create a New File
    - Step 2. Import Modules
    - Step 3. Write the Code for Downloading Images
    - Step 4. Create a New Folder for Images
    - Step 5. Run the Code
- Using the Bannerbear API

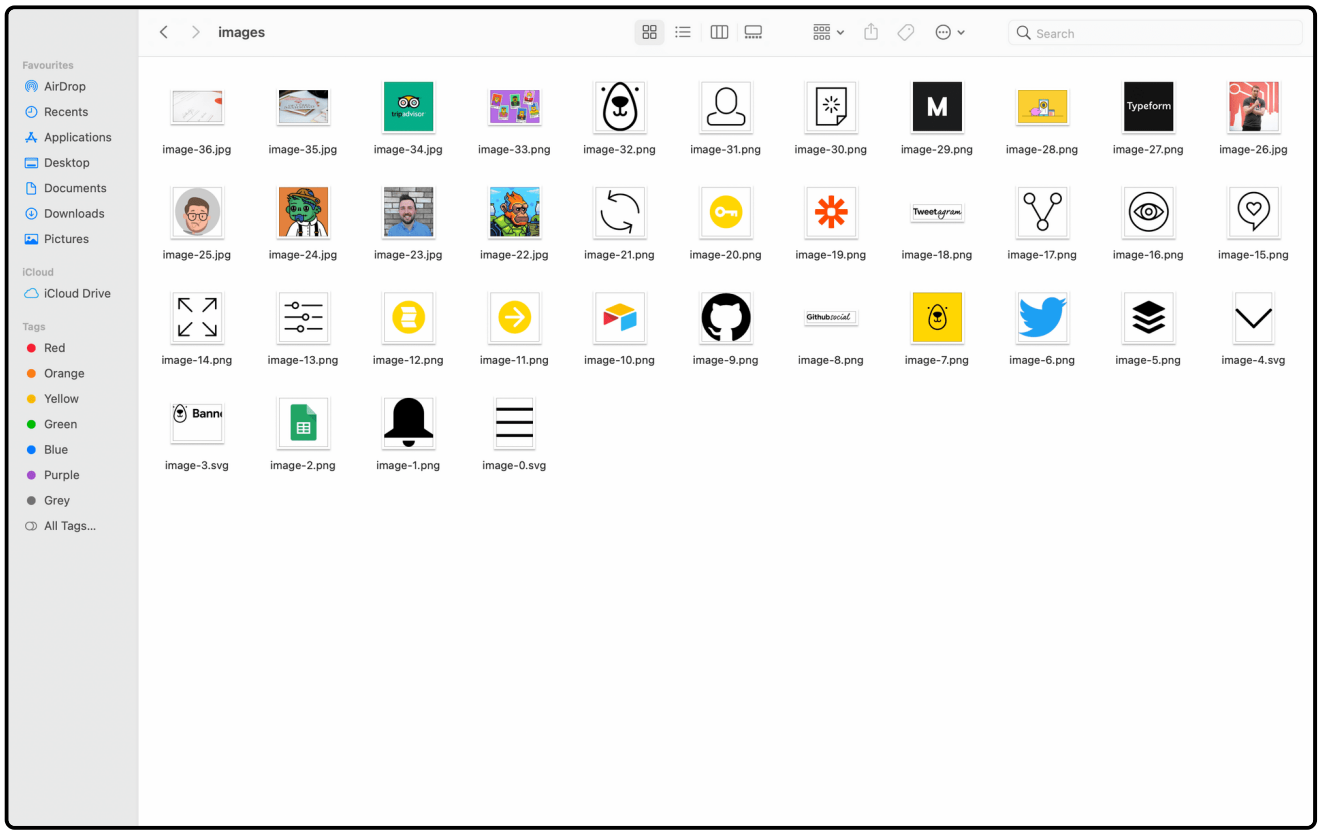
Why use Puppeteer to download images? We can just right-click and save. It's simple.

That's only true if you're only downloading a few images. Imagine if you're downloading 100+ images from a website manually, that's a dreadful task. 😰 Fret not! We can do this by using automation and save all images from a website to a folder IN ONE GO.

In this tutorial, we will be guiding you step-by-step on how to download images (.jpg, .png, .svg, .gif) from a website using an automation tool called Puppeteer. You can use it on any website that you want but we will be using this Bannerbear page for this tutorial:



At the end of this tutorial, you will have images from a website downloaded to a folder:



Images downloaded from the Bannerbear page using Puppeteer

What is Puppeteer

Puppeteer is a Node library which provides a high-level API to control Chrome or Chromium over the DevTools Protocol. It is very useful for automating the Chrome browser to run website tests. Puppeteer runs headless by default, which means you won't see the browser running but it can be configured to run full (non-headless) Chrome or Chromium.

Pre-requisites

To use Puppeteer to download images from a website, you will need to have Node.js and npm installed.

For reference, the version of Node.js and npm we are using for this tutorial are 14.17.3 and 6.14.13 respectively. Please check the official documentation to check your version compatibility.

Getting Started

Step 1. Create a New Node.js Project

Create a new folder for your project and go to the directory.

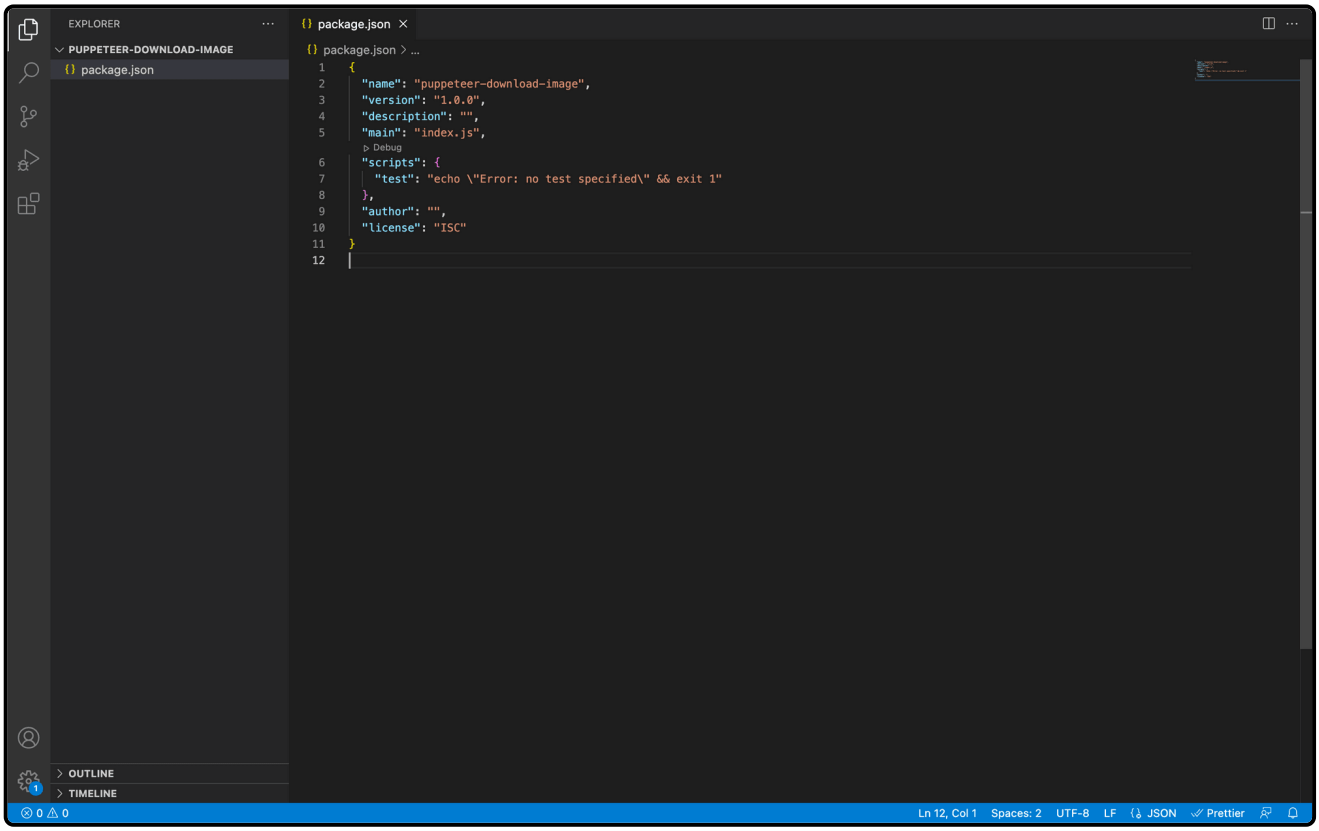
```
mkdir puppeteer-download-images
cd puppeteer-download-images
```

Init a new Node.js project in the folder.

```
npm init
```

It will prompt you for input for a few aspects of the project, just press enter if you want to use the default values.

Once you run through the `npm init` steps above, a `package.json` file will be generated and placed in the current directory.

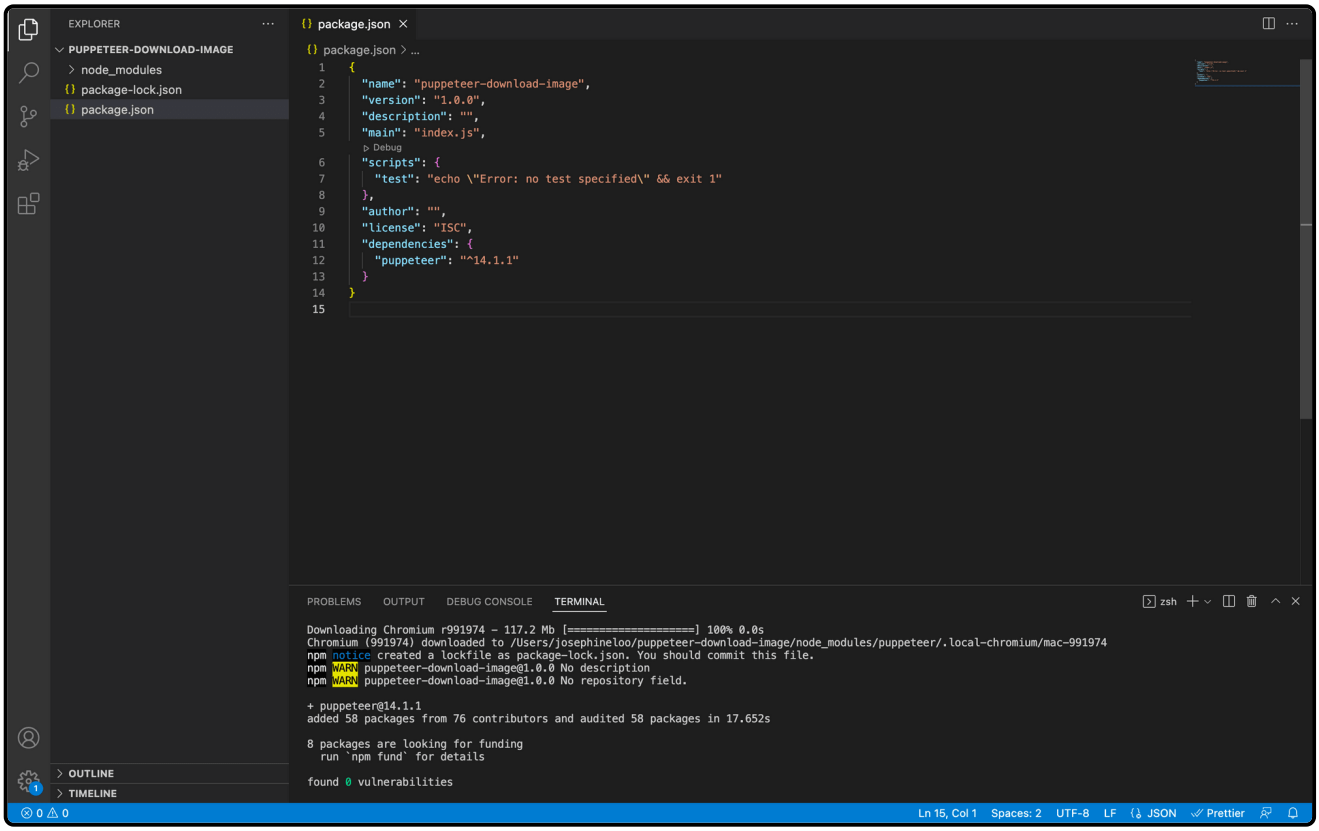


Step 2. Install Puppeteer

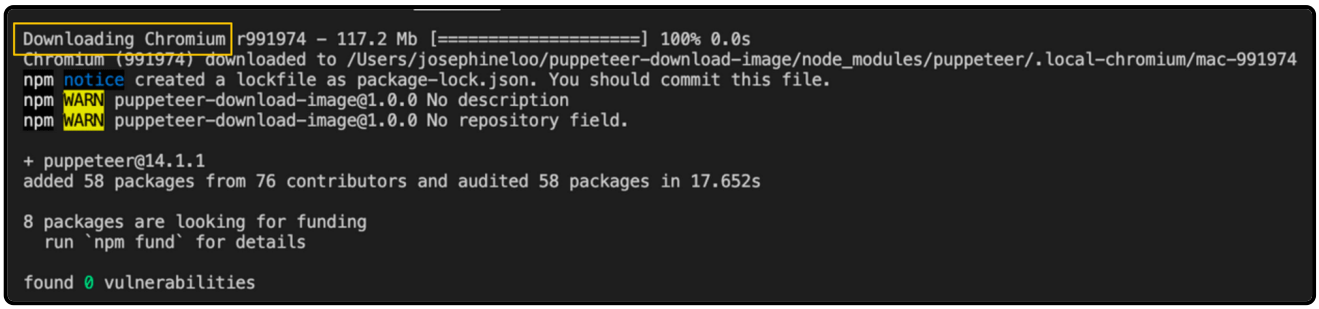
Run the command below to install Puppeteer.

```
npm i puppeteer
```

A folder named `node_modules` and a file named `package-lock.json` will be added to your project after running the command.



When you run the command above to install Puppeteer, a recent version of Chromium which is guaranteed to work with the Puppeteer API is also downloaded.



Testing Puppeteer

Before we start writing codes to download images from a website, let's try whether Puppeteer is working properly. We will use a simple example from the official documentation to take a screenshot of a website.

Step 1. Write the Code

Create a new `example.js` file and paste the following code:

```
const puppeteer = require('puppeteer');

(async () => {
  const browser = await puppeteer.launch();
  const page = await browser.newPage();
  await page.goto('https://google.com');
  await page.screenshot({ path: 'example.png' });

  await browser.close();
})();
```

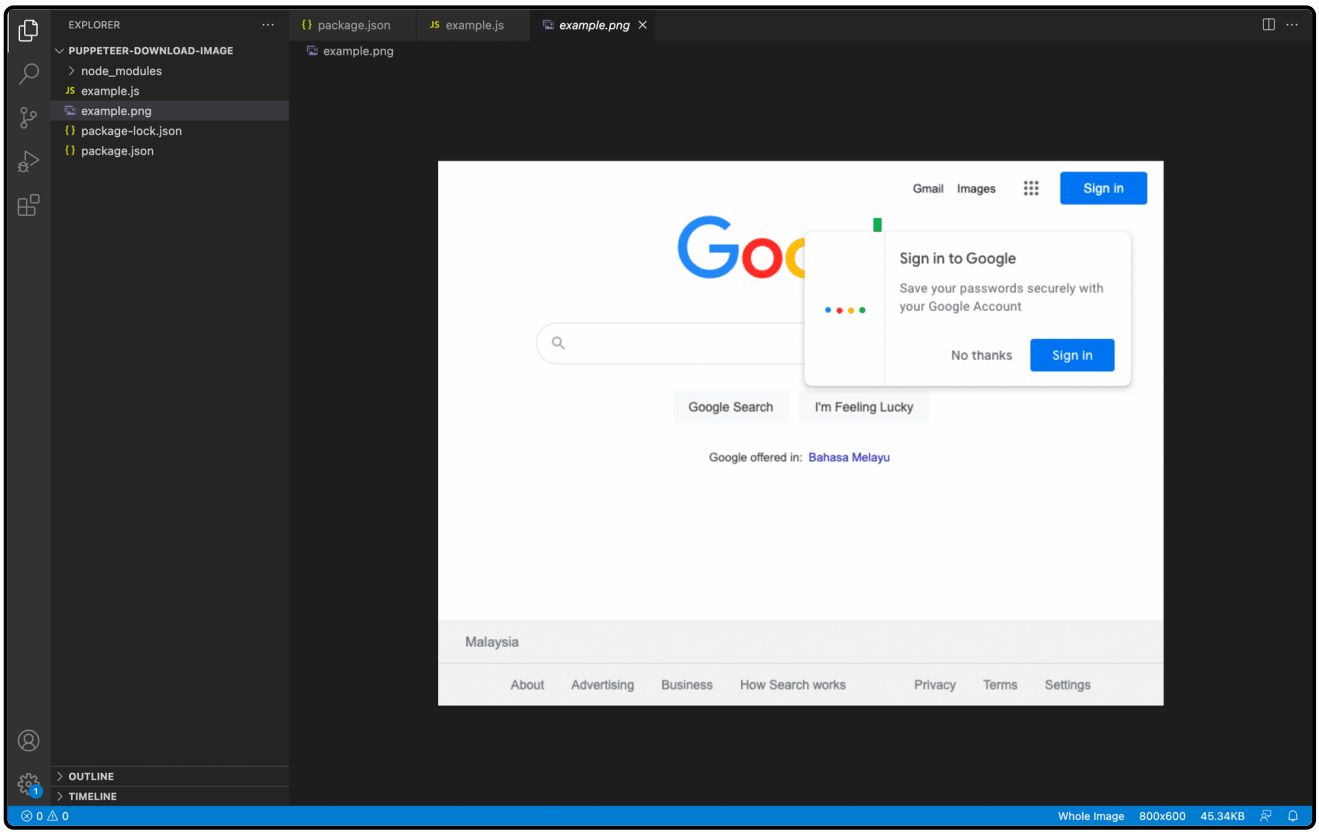
This example creates a page, navigates it to a URL, and then saves a screenshot.

Step 2. Run the Code

Run the `example.js` file to execute the code.

```
node example.js
```

You will find a new image `example.png` created inside your folder. This is the screenshot of the page visited by Puppeteer. 😀

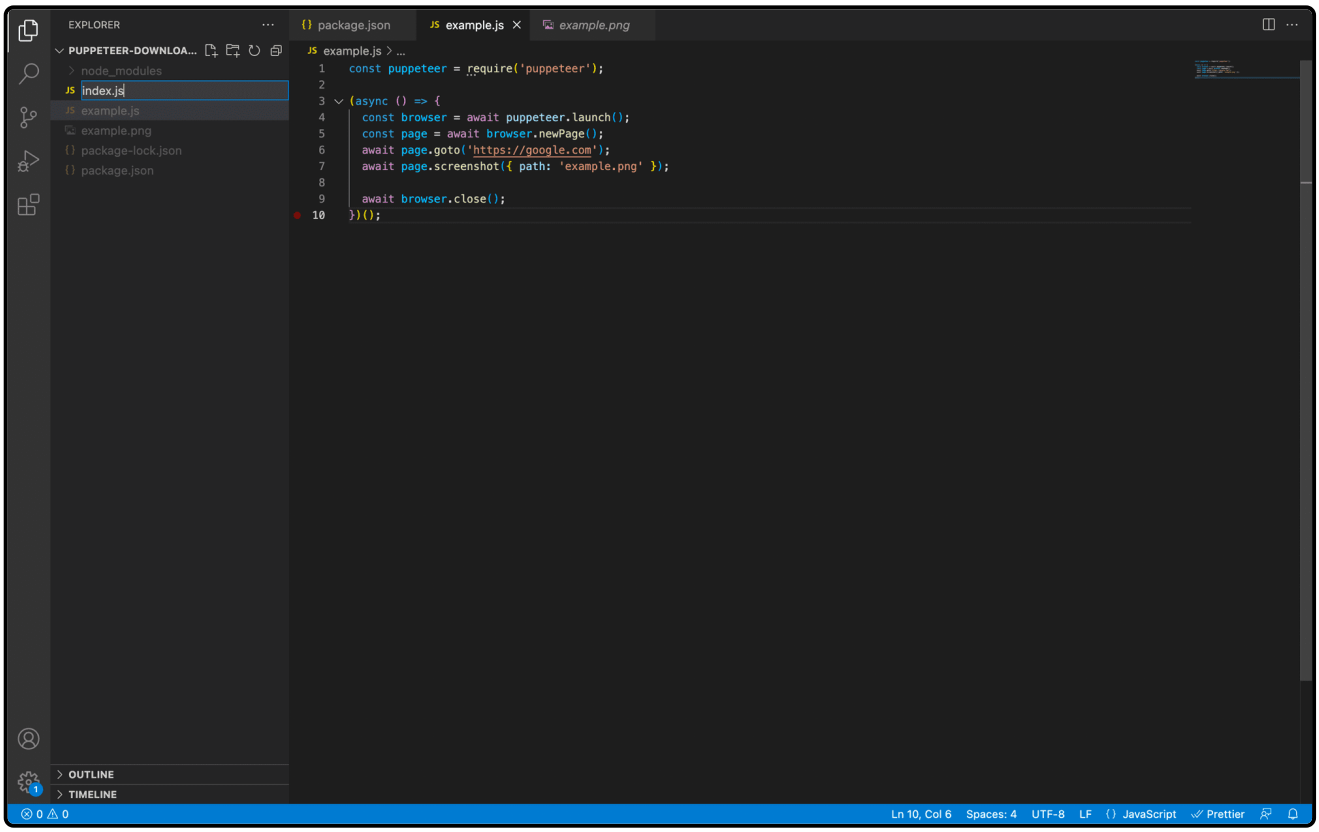


Now we know that Puppeteer is working, we can start writing codes to download images!

Downloading Images from a Website

Step 1. Create a New File

In the same project, create `index.js` file. This is where we will be writing our code to download images from the Bannerbear page.



Step 2. Import Modules

Inside the `index.js` file, we need to require `puppeteer` and the `fs` (file system) module. The `fs` module will allow you to write data fetched from the website into a file.

```
const puppeteer = require('puppeteer');
const fs = require('fs');
```

Step 3. Write the Code for Downloading Images

Then, write the following code:

```
(async () => {
  const browser = await puppeteer.launch();
  const page = await browser.newPage();

  let counter = 0;
  page.on('response', async (response) => {
    const matches = /.*\.(jpg|png|svg|gif)$/.exec(response.url()
    console.log(matches);
    if (matches && (matches.length === 2)) {
      const extension = matches[1];
      const buffer = await response.buffer();
      fs.writeFileSync(`images/image-${counter}.${extension}`, bu
      counter += 1;
    }
  });

  await page.goto('https://www.bannerbear.com/solutions/automate
```

```
  await browser.close();
})();
```

Similar to the previous example, Puppeteer will open a page and navigate to the URL. Then, it will catch responses which match the image file extensions (.jpg, .png, .svg, .gif), rename it and save it to a folder named `/images`.

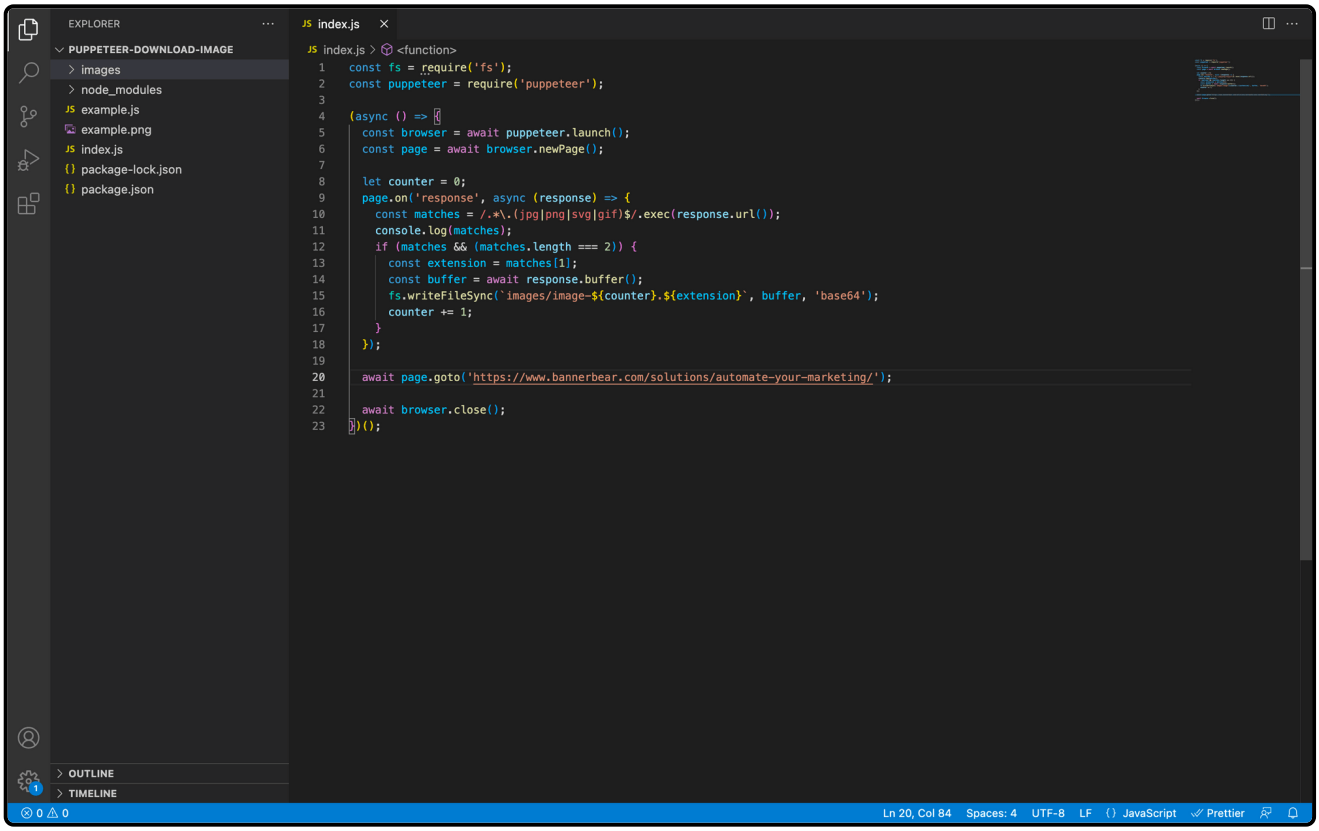
The complete code looks like this.

```
const puppeteer = require('puppeteer');
const fs = require('fs');

(async () => {
  const browser = await puppeteer.launch();
  const page = await browser.newPage();

  let counter = 0;
  page.on('response', async (response) => {
    const matches = /.*\.(jpg|png|svg|gif)$/.exec(response.url()
    console.log(matches);
    if (matches && (matches.length === 2)) {
      const extension = matches[1];
      const buffer = await response.buffer();
      fs.writeFileSync(`images/image-${counter}.${extension}`, bu
      counter += 1;
    }
  });

  await page.goto('https://www.bannerbear.com/solutions/automate
```

```
  await browser.close();
})();
```

Step 4. Create a New Folder for Images

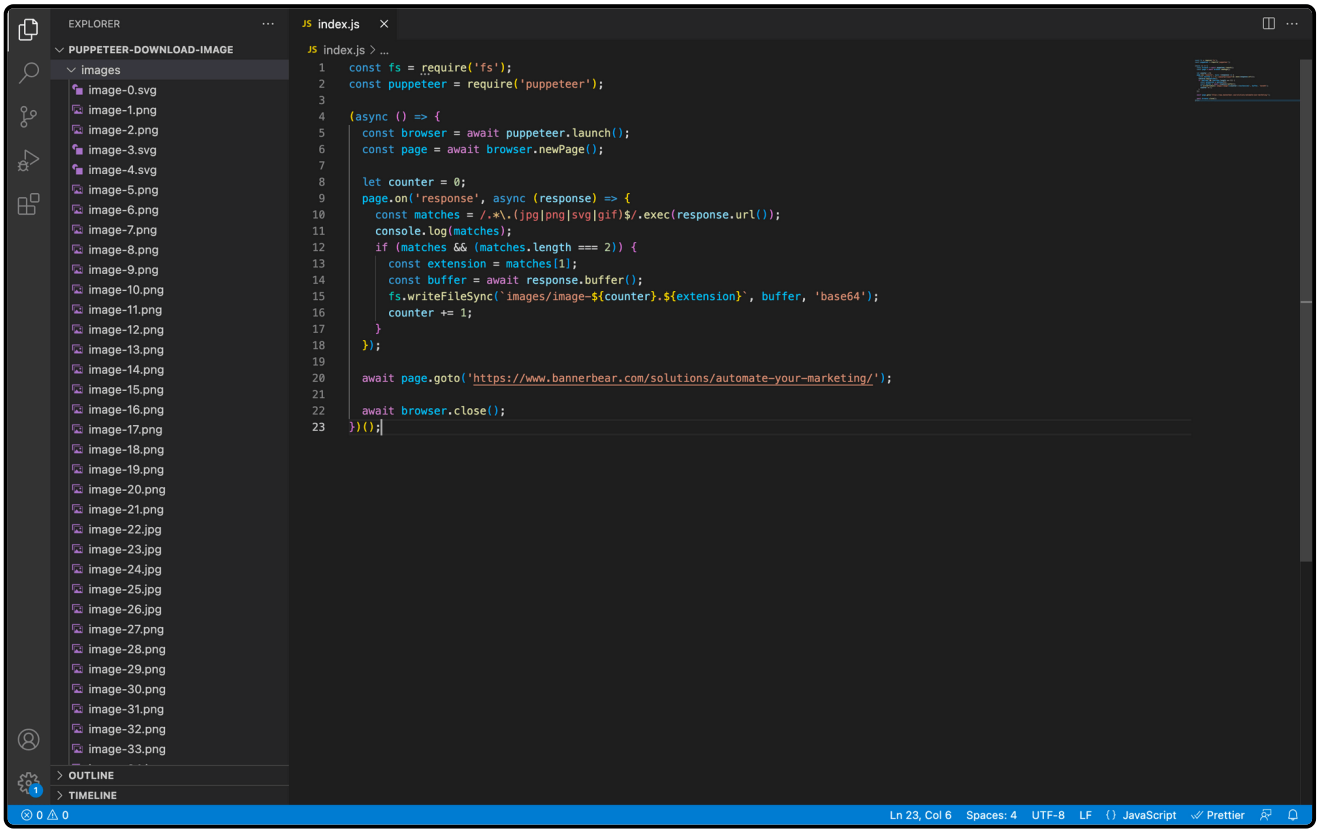
Before executing the code, create a new `/images` subfolder in the current directory. This is where the images downloaded from the page will be saved.



Step 5. Run the Code

Now, run `index.js` and see all images from the page get downloaded into the `/images` folder.

```
node index.js
```



That's it! All images from the Bannearbear page are now downloaded to the `/images` folder. 🥳

You can try this on other websites as well. Simply replace the Bannerbear URL with another URL and then run the `node index.js` command.

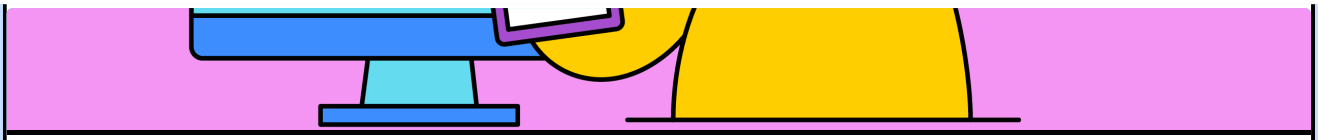
Using the Bannerbear API

If you want to process the images like applying an overlay or watermark to them, you can try using the Bannerbear API. The Bannerbear API allows you to create a template that can be applied to all images and generate the images in a few seconds by sending modification requests to the API endpoint. There are also tons of templates in our template library to choose from if you don't want to create your own template. All you need to do is sign up for an account and you can start generating images immediately for free! 😉

About the author

Josephine Loo

Josephine is an automation enthusiast. She loves automating stuff and helping people to increase productivity with automation.

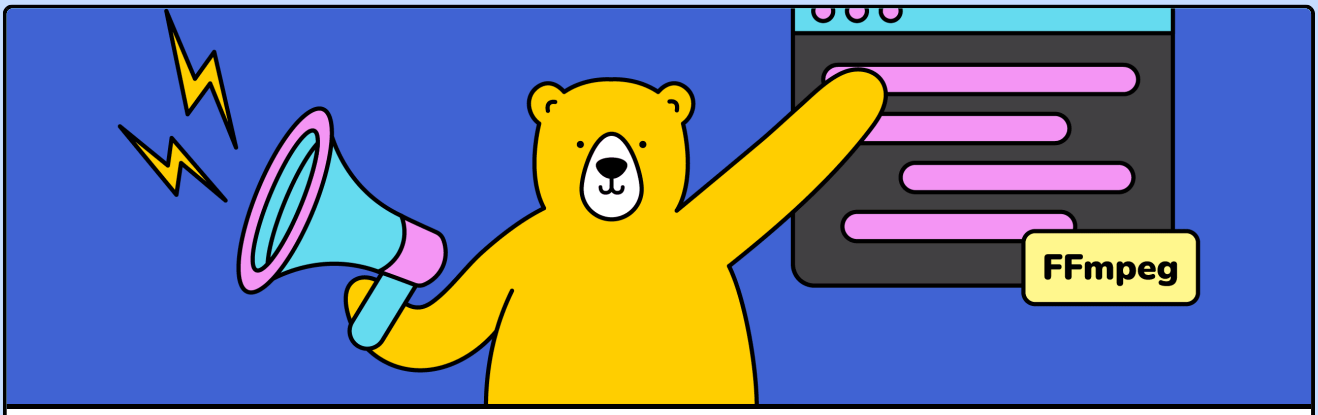


March 2023

How to Make a Device Mockup Generator for Websites using Bannerbear

Device mockups are a valuable tool to display website designs across multiple devices as they offer a realistic preview of a website. This tutorial demonstrates how to build a device mockup generator using Bannerbear, allowing the automatic generation of device mockups by entering a website's URL.

api bannerbear developers

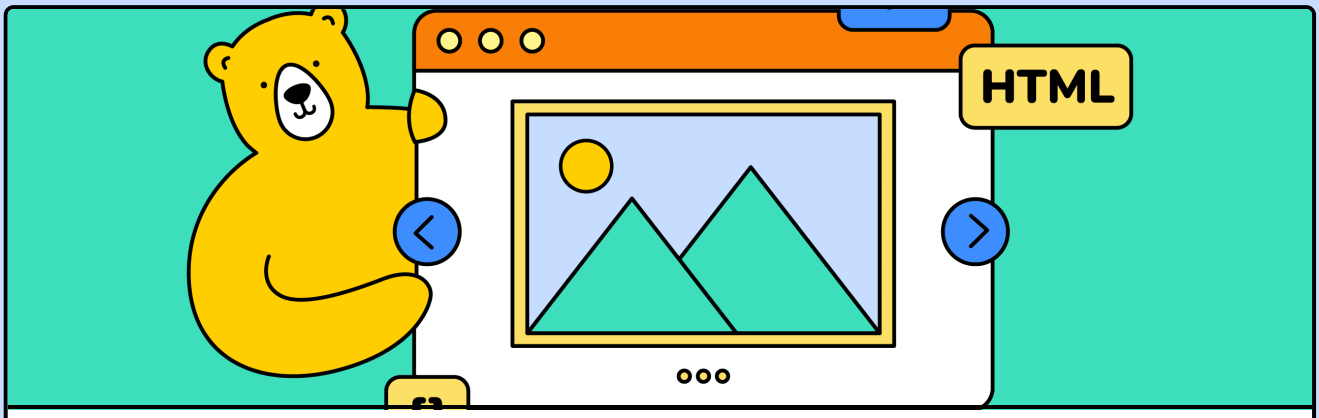


February 2023

FFmpeg 101: Top 10 Command Options You Need to Know (with Examples)

FFmpeg is a complete, cross-platform solution to edit media files programatically. With hundreds of command options available, it is easy to feel overwhelmed. In this article, we'll go through 10 essential FFmpeg options you should know.

developers ffmpeg



January 2023

How to Create an Image Slider in HTML, CSS, and Javascript

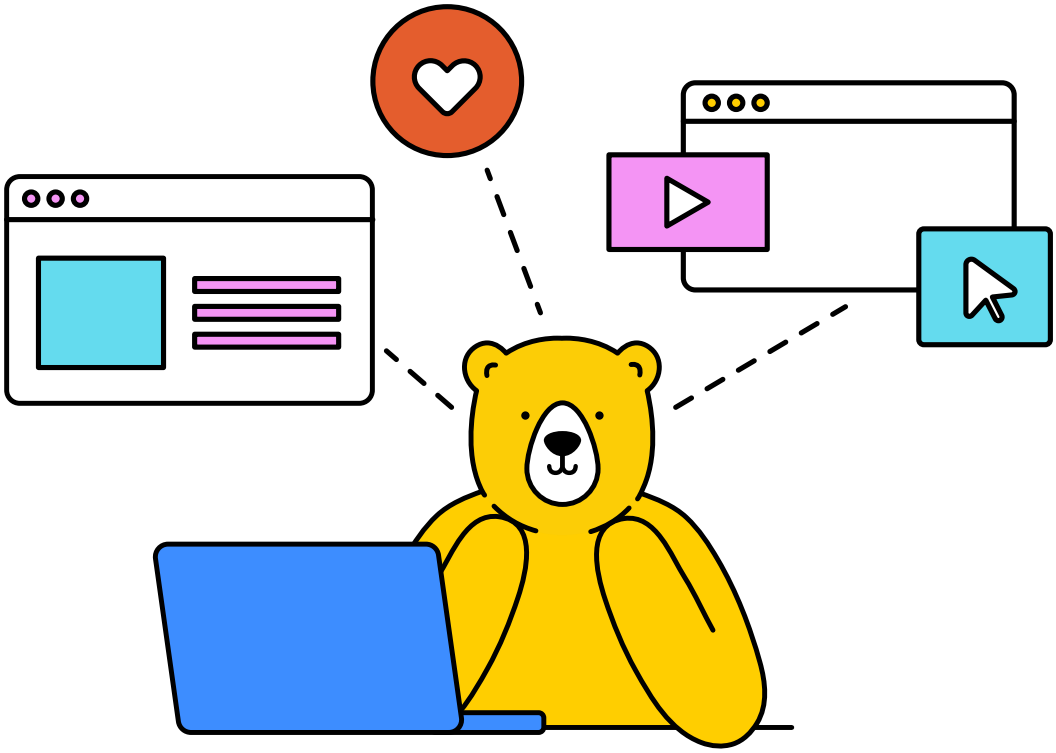
Many JavaScript image slider libraries are available but which is the best for your project? Instead of using an external library, let's create one from scratch using simple HTML, CSS, and JavaScript.

api developers

Follow the Journey

Sign up for our once a fortnight newsletter update on new Bannerbear features and our business journey

Subscribe



Use Cases

Generate Images via API
Watermark Videos via API
Generate PDFs via API
Generate Images with Zapier
Watermark Videos with Zapier
Generate PDFs with Zapier
More Use Cases

Product

Image Generation API
Multi Image Generation API
Video Generation API
PDF Generation API

Template Library
Bannerbear for Enterprise

Integrations

Airtable Integration
Zapier Integration
Integromat Integration
Forms
URLs
WordPress

Demos

Multi Image Demo
AI Face Detect Demo
Twitter to Instagram
Github Social
Smart Crop Demo
Online Certificate Maker
Online Wedding Invite Maker
Online Event ID Card Maker
Online Photo Collage Maker
Online Invoice Maker

Docs & Guides

Help Articles
Blog
eBooks
API Quick Start
API Reference

Other

System Status
About Bannerbear
Open Startup
↳ $10K MRR SaaS Journey
↳ $10K to $20K MRR
↳ Journey to $1MM ARR
Affiliate Program
Facebook Preview Tool
Twitter Preview Tool
Changelog
Pricing

Copyright ©2023 Bannerbear

Privacy Policy

Terms and Conditions