

Products Developers

Live for Teams

Pricing

Sign in

Guide

Categories

Search across Guide

Press /

Home

Testing on Cloud

Debugging

Best Practices

Tools & Frameworks

Tutorials

By Technocrat, Community Contributor - June 9, 2023









<u>Puppeteer</u> is a Node.js library that provides a high-level API to control Chrome or Chromium over the DevTools Protocol. It also provides a powerful headless mode that closely integrates with Chrome's headless mode. Developers prefer Puppeteer for applications that need extensive testing on Chrome. With Puppeteer, they are able to efficiently test, scrape, and interact with web pages programmatically.

You can use Puppeteer for multiple use cases:

- Single <u>browser automation</u>
- Generating screenshots and PDFs of web pages
- Scraping data from web pages
- Injecting JavaScript into web pages
- Controlling the browser's DevTools

Read More: What is Headless Browser Testing and Why is it Important?

- Run your first Puppeteer Test
- Best Practices for Creating Puppeteer Tests

Pre-requisites for Setting up Puppeteer

Before you can start running your tests using Puppeteer, you must install certain prerequisites.

- <u>Node.js</u> As Puppeteer is built on Node.js(node), Node.js must be installed. When you install Node.js, **npm** is auto installed.
- Google Chrome Download the Chrome version compatible with the latest <u>Puppeteer version</u>.

Install and Set up Puppeteer

You must create your Puppeteer project, initiate it, and then install Puppeteer with npm and Puppeteer.

- 1. Open the command line interface/terminal.
- 2. Create a new directory for your Puppeteer project:

mkdir puppeteer-project

3. Change to the Puppeteer project directory:

cd puppeteer-project

4. Run the following command to initiate a new **npm** project:

npm init

Run your first Puppeteer Test

In this test, we will launch a new browser page, open <u>bstackdemo.com</u>, take a screenshot, and then close the browser.

- 1. In VSCode, open the **puppeteer-project** directory.
- 2. Create a **testcase.js** file.
- 3. Copy the following code to the file:

```
//adding Puppeteer library
const pt = require('puppeteer');
pt.launch().then(async browser => {
//browser new page
const p = await browser.newPage();
//set viewpoint of browser page
await p.setViewport({ width: 1000, height: 500 })
//launch URL
await p.goto('https://bstackdemo.com')
//capture screenshot
await p.screenshot({
path: 'browserstack-demo.png'
});
//browser close
await browser.close()
})
```

4. Run the following command to test the script:

```
node testcase.js
```

After the script runs, the **browserstack-demo.png** is saved in your project directory.

Also Read: How to run UI Automation Testing using Puppeteer

Best Practices for Creating Puppeteer Tests

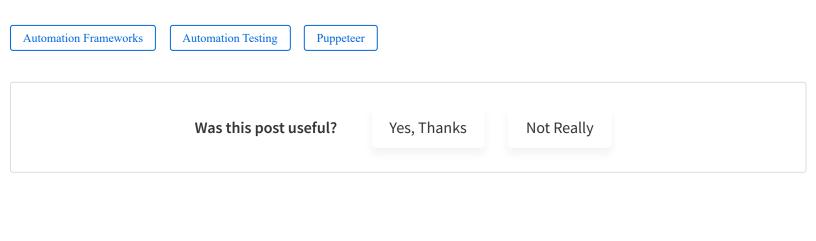
Here are a few best practices to consider while creating Puppeteer tests:

- It is recommended that you set the **headless** option to **true** when using Puppeteer with Chrome for better performance and automation.
- Add the **browser.close** method to close the browser at the end of your script to reduce resource consumption.
- When you are interacting with connected pages such as clicking links, submitting forms, ensure that you let Puppeteer know to wait for the next step using the **page.waitForNavigation** method.

Conclusion

BrowserStack gives you access to a real device cloud that supports Puppeteer and lets you run <u>cross browser Puppeteer</u> <u>tests in parallel</u> through 3000+ real devices and browsers on the cloud.

Run Puppeteeer Tests on Real Devices



Related Articles

Puppeteer Framework Tutorial: Basics and Setup

Learn the hasics advantages and a

Learn More

How to run UI Automation Testing using Puppeteer

Learn how to run automated tests using the Puppeteer framework. Step by step tutorial with images an...

Learn More

Cross Browser
Testing in Puppeteer:
Tutorial

Learn More

SOLUTIONS

Test on iPhone

Test on Galaxy

Android Testing

Cross Browser Testing

Emulators & Simulators

Android Emulators

Visual Testing

Test on iPad

Test In IE

iOS Testing

Selenium

Cypress

PRODUCTS

TOOLS

Automate

SpeedLab Screenshots

Automate TurboScale Beta

Responsive

Percy

Live

App Live

PLATFORM

App Automate

Browsers & Devices

App Percy New

Data Centers

Test Management New Test Observability New

Device Features

Accessibility Testing New

Security

Accessibility Automation Beta

Low Code Automation Beta

Nightwatch.js

RESOURCES

Enterprise

COMPANY

Test on Right Devices

About Us

Support

Customers

Status

Careers We're hiring!

Release Notes

Open Source

Case Studies

Partners

Press

Blog

Events

Test University Beta

Champions

Mobile Emulators

Guide

Responsive Design



Contact Us