FAQ

Q: Who maintains Puppeteer?

The Chrome DevTools team maintains the library, but we'd love your help and expertise on the project! See Contributing.

Q: What are Puppeteer's goals and principles?

The goals of the project are:

- Provide a slim, canonical library that highlights the capabilities of the DevTools Protocol.
- Provide a reference implementation for similar testing libraries. Eventually, these other frameworks could adopt Puppeteer as their foundational layer.
- Grow the adoption of headless/automated browser testing.
- Help dogfood new DevTools Protocol features...and catch bugs!
- Learn more about the pain points of automated browser testing and help fill those gaps.

We adapt Chromium principles to help us drive product decisions: - **Speed**: Puppeteer has almost zero performance overhead over an automated page. - **Security**: Puppeteer operates off-process with respect to Chromium, making it safe to automate potentially malicious pages. - **Stability**: Puppeteer should not be flaky and should not leak memory. - **Simplicity**: Puppeteer provides a high-level API that's easy to use, understand, and debug.

Q: Is Puppeteer replacing Selenium/WebDriver?

No. Both projects are valuable for very different reasons: - Selenium/WebDriver focuses on cross-browser automation; its value proposition is a single standard API that works across all major browsers. - Puppeteer focuses on Chromium; its value proposition is richer functionality and higher reliability.

That said, you **can** use Puppeteer to run tests against Chromium, e.g. using the community-driven jest-puppeteer. While this probably shouldn't be your only testing solution, it does have a few good points compared to WebDriver:

- Puppeteer requires zero setup and comes bundled with the Chromium version it works best with, making it very easy to start with. At the end of the day, it's better to have a few tests running chromium-only, than no tests at all.

- Puppeteer has event-driven architecture, which removes a lot of potential flakiness. There's no need for evil "sleep(1000)" calls in puppeteer scripts.
- Puppeteer runs headless by default, which makes it fast to run. Puppeteer v1.5.0 also exposes browser contexts, making it possible to efficiently parallelize test execution.
- Puppeteer shines when it comes to debugging: flip the "headless" bit to false, add "slowMo", and you'll see what the browser is doing. You can even open Chrome DevTools to inspect the test environment.

Q: Why doesn't Puppeteer v.XXX work with Chromium v.YYY?

We see Puppeteer as an **indivisible entity** with Chromium. Each version of Puppeteer bundles a specific version of Chromium – **the only** version it is guaranteed to work with.

This is not an artificial constraint: A lot of work on Puppeteer is actually taking place in the Chromium repository. Here's a typical story: - A Puppeteer bug is reported: https://github.com/GoogleChrome/puppeteer/issues/2709 - It turned out this is an issue with the DevTools protocol, so we're fixing it in Chromium: https://chromium-review.googlesource.com/c/chromium/src/+/1102154 - Once the upstream fix is landed, we roll updated Chromium into Puppeteer: https://github.com/GoogleChrome/puppeteer/pull/2769

However, oftentimes it is desirable to use Puppeteer with the official Google Chrome rather than Chromium. For this to work, you should pick the version of Puppeteer that uses the Chromium version close enough to Chrome.

Q: Which Chromium version does Puppeteer use?

Look for `chromium_revision` in package.json.

Q: What's considered a "Navigation"?

From Puppeteer's standpoint, **"navigation" is anything that changes a page's URL**. Aside from regular navigation where the browser hits the network to fetch a new document from the web server, this includes anchor navigations and History API usage.

With this definition of "navigation," **Puppeteer works seamlessly with single-page applications.**

Q: What's the difference between a "trusted" and "untrusted" input event?

In browsers, input events could be divided into two big groups: trusted vs. untrusted.

- **Trusted events**: events generated by users interacting with the page, e.g. using a mouse or keyboard.
- **Untrusted event**: events generated by Web APIs, e.g. `document.createEvent` or `element.click()` methods.

Websites can distinguish between these two groups: - using an `Event.isTrusted` event flag - sniffing for accompanying events. For example, every trusted `'click'` event is preceded by `'mousedown'` and `'mouseup'` events.

For automation purposes it's important to generate trusted events. **All input events generated with Puppeteer are trusted and fire proper accompanying events.** If, for some reason, one needs an untrusted event, it's always possible to hop into a page context with `page.evaluate` and generate a fake event:

```
await page.evaluate(() => {
  document.querySelector('button[type=submit]').click();
});
```

Q: What features does Puppeteer not support?

You may find that Puppeteer does not behave as expected when controlling pages that incorporate audio and video. (For example, video playback/screenshots is likely to fail.) There are two reasons for this:

- Puppeteer is bundled with Chromium–not Chrome–and so by default, it inherits all of Chromium's media-related limitations. This means that Puppeteer does not support licensed formats such as AAC or H.264. (However, it is possible to force Puppeteer to use a separately-installed version Chrome instead of Chromium via the `executablePath` option to `puppeteer.launch`. You should only use this configuration if you need an official release of Chrome that supports these media formats.)
- Since Puppeteer (in all configurations) controls a desktop version of Chromium/Chrome, features that are only supported by the mobile version of Chrome are not supported. This means that Puppeteer does not support HTTP Live Streaming (HLS).

Q: I am having trouble installing / running Puppeteer in my test environment?

We have a troubleshooting guide for various operating systems that lists the required dependencies.

Q: How do I try/test a prerelease version of Puppeteer?

You can check out this repo or install the latest prerelease from npm:

```
npm i --save puppeteer@next
```

Please note that prerelease may be unstable and contain bugs.

Q: I have more questions! Where do I ask?

There are many ways to get help on Puppeteer: - bugtracker - stackoverflow - slack channel

Make sure to search these channels before posting your question.
