UDC 004.4'234

THE INSIDE OF HEADLESS CHROME

M. BALABASH, A. OSKIN
Polotsk State University, Belarus

In this paper, we define headless web browsers and how to use them. The main components of the Headless Chrome web browser and the interaction protocol are considered.

Introduction. A headless browser is a great tool for automated testing and server environments where you don't need a visible UI shell. For example, you may want to run some tests against a real web page, create a PDF of it, or just inspect how the browser renders an URL.

Headless browsers provide automated web page management in an environment with all modern web platform features. They provide special programming interfaces for interacting with the runtime and allow you to control the web browser, execute code in its environment and receive any available information from the context of the web browser.

Chromium is an open browser implementation based on WebKit. Google Chrome is a branded version of Chromium, which has proprietary codecs, integrations with Google services, etc. Headless Chrome is a headless version of the Google Chrome browser. And that is we gonna explore.

Blink is a browser engine. Blink is a fork of the WebCore component of WebKit, which was originally a fork of the KHTML and KJS libraries from KDE. It is used in Chrome starting at version 28, Microsoft Edge starting at version 79, Opera (15+), Vivaldi, Amazon Silk and other Chromium-based browsers and frameworks.

The Chrome DevTools protocol is used as the management interface (API). Chrome DevTools is a set of tools for web developers, built directly into the Google Chrome browser.

Chromium Components. The lowest level is a Platform layer.

1. Ozone, the abstract window manager in Chrome, is what the window manager of the operating system interacts with. On Linux, it is either an X-server or Wayland. On Windows, it is a Windows window manager.

2. Scheduler - the same scheduler that deals with the synchronization of threads and processes, because Chrome is a multi-process application.

3. Net - a network component that parses HTTP, creates headers, etc.

The Content layer is the largest component Chrome has.

1. Blink - a WebCore-based web engine from WebKit for working with HTML and CSS; V8 (JavaScript engine); API for all extensions we use in Chrome. It also includes the DevTools protocol.

2. The Content API is an interface with which you can very easily use all the features of the web engine. Since there are a lot of things inside Blink (several hundred thousand interfaces), in order not to get lost in all these methods and functions, you need a Content API.

Headless layer level.

1. Headless library.

2. Embedder API interface for embedding Headless library in the application.

3. Client API is an interface that Puppeteer uses.

Application Layer

1. Your application (Embedding app).

2. Gadgets, for example, Headless shell.

Chrome DevTools protocol. All front-end developers (and not only they) came across the Chrome DevTools protocol, because they used the Chrome developer panel or the remote debugger, or Chrome development tools. If you run the developer tools remotely, communication with the browser occurs using the DevTools protocol. When you install debugger, see code coverage, use geolocation or something else - all this is controlled using DevTools.

The protocol has 2 components:

1. DevTools target - the tab that you inspect.

2. DevTools client - for example, this is a developer panel that is launched remotely.

They communicate using simple JSON:

1. There is an identifier for the command, the name of the method to be executed, and some parameters.

2. The answer also looks very simple: an identifier that is needed because all the commands that are executed using the protocol are asynchronous. In order for us to always be able to compare which response to which team we received, we need an identifier.

3. Result.

Puppeteer. This is a library developed by the Chrome team (available for several programming languages) that provides a high-level API for controlling Chrome or Chromium using the DevTools protocol.

It provides a high-level API to control headless (or full) Chrome. And hides away the complexities of the DevTools protocol and takes care of redundant tasks like launching a debug instance of Chrome. It's similar to other automated testing libraries like Phantom and NightmareJS, but it only works with the latest versions of Chrome.

Among other things, Puppeteer can be used to easily take screenshots, create PDFs, navigate pages, and fetch information about those pages, measure and diagnose performance indicators, intercept of network requests / responses, test Chrome extensions, automate form submission, user interface testing, keyboard input, etc.

Playwright. Playwright is a Node library to automate the Chromium, WebKit and Firefox browsers with a single API. It enables cross-browser web automation that is ever-green, capable, reliable and fast.

Headless is supported for all the browsers on all platforms. This solution has all abilities that Puppeteer has, but also supports WebKit and Firefox browsers. Playwright also has really similar to Puppeteer programming API.

Conclusion. Headless Chrome is a tool that allows you test our web applications and automate interactions with them. DevTools protocol has a rich set of features that allows you to develop complete and independent software solutions for a wide range of tasks (monitoring web applications, development environment VS Code). The ability to receive information, execute code, and automate browser actions opens up many possibilities.

REFERENCES

1. GitHub [Electronic resource] Headless Chrome Node.js API. Access Mode: https://github.com/ Google-Chrome / puppeteer. Access date: 09/27/19.
2. ChromeDevtools [Electronic resource] Chrome DevTools Protocol Viewer. Access Mode: https://chromedevtools.github.io/devtools-protocol/. Access date: 09/27/19.
2. Chromium [Electronic resource] The Chromium Projects. Access Mode: https: // www.chromium.org/. Access date: 09/27/19.