

# What's New In DevTools (Chrome 65)



By Kayce Basques

Technical Writer for Chrome DevTools

New features coming to DevTools in Chrome 65 include:

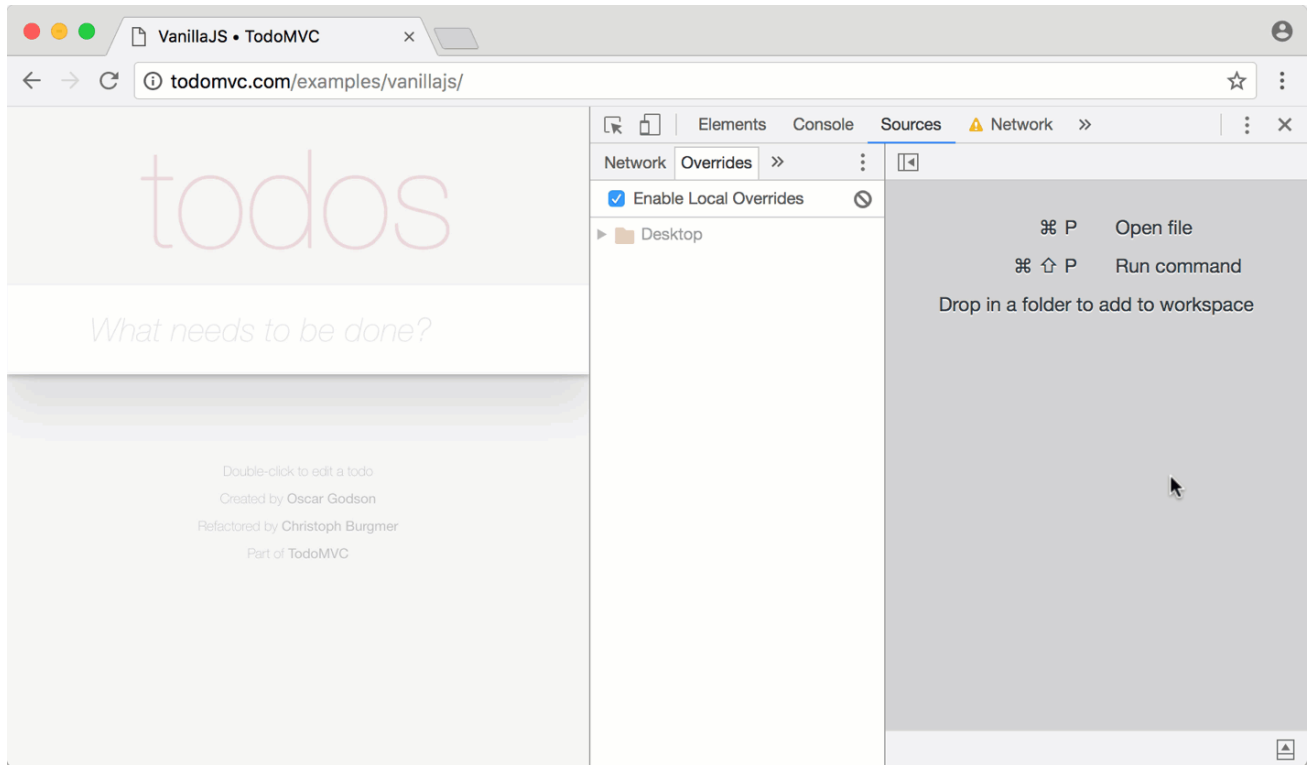
- **Local Overrides**
- New accessibility tools
- The **Changes** tab
- New SEO and performance audits
- Multiple recordings in the **Performance** panel
- Reliable code stepping with workers and asynchronous code

Read on, or watch the video version of these release notes, below.

**Note:** Check what version of Chrome you're running at <chrome://version>. If you're running an earlier version, these features won't exist. If you're running a later version, these features may have changed. Chrome auto-updates to a new major version about every 6 weeks.

# Local Overrides

**Local Overrides** let you make changes in DevTools, and keep those changes across page loads. Previously, any changes that you made in DevTools would be lost when you reloaded the page. **Local Overrides** work for most file types, with a couple of exceptions. See [Limitations](#).



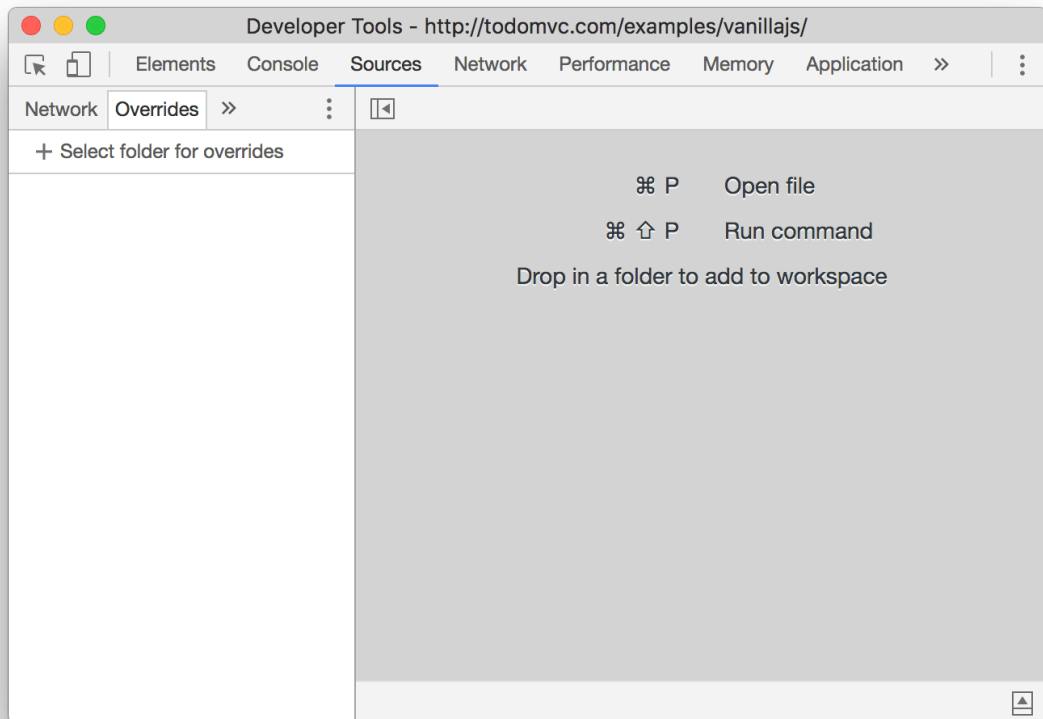
**Figure 1.** Persisting a CSS change across page loads with **Local Overrides**

How it works:

- You specify a directory where DevTools should save changes.
- When you make changes in DevTools, DevTools saves a copy of the modified file to your directory.
- When you reload the page, DevTools serves the local, modified file, rather than the network resource.

To set up **Local Overrides**:

1. Open the **Sources** panel.
2. Open the **Overrides** tab.



**Figure 2.** The **Overrides** tab

3. Click **Setup Overrides**.
4. Select which directory you want to save your changes to.
5. At the top of your viewport, click **Allow** to give DevTools read and write access to the directory.
6. Make your changes.

## Limitations

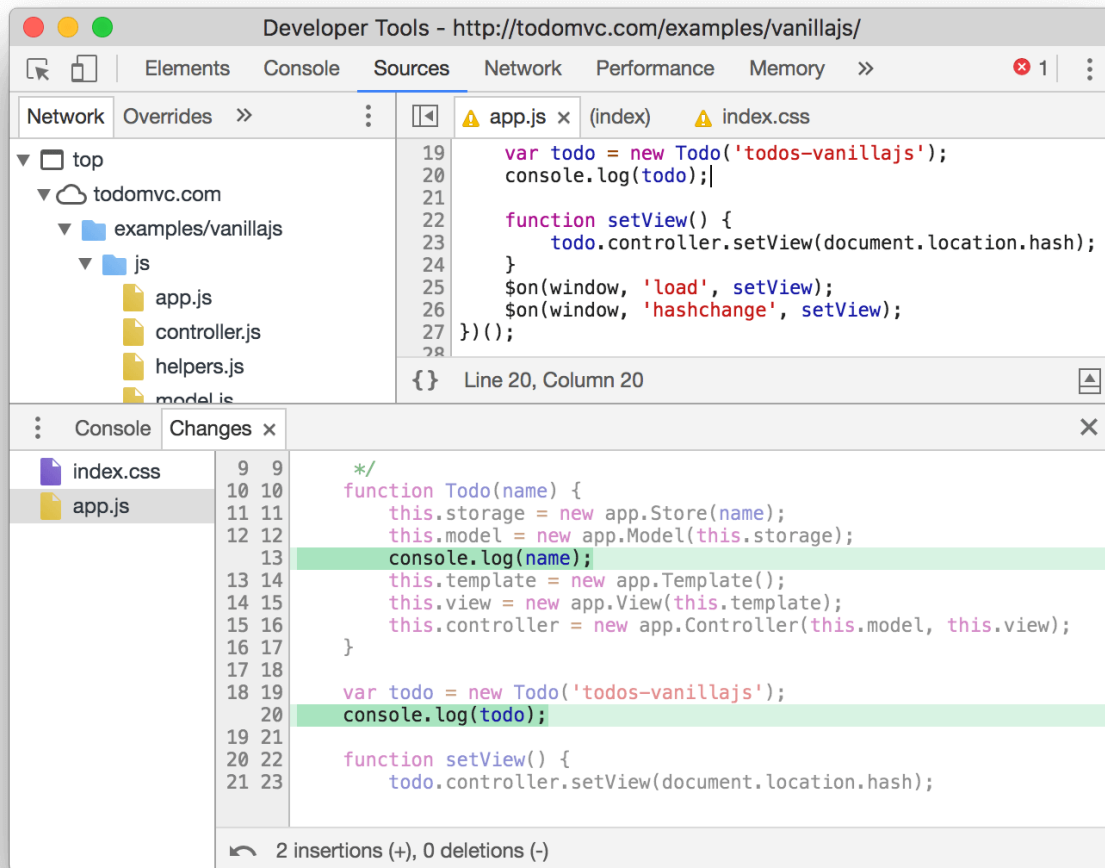
- DevTools doesn't save changes made in the **DOM Tree** of the **Elements** panel. Edit HTML in the **Sources** panel instead.
- If you edit CSS in the **Styles** pane, and the source of that CSS is an HTML file, DevTools won't save the change. Edit the HTML file in the **Sources** panel instead.

## Related features

- Workspaces. DevTools automatically maps network resources to a local repository. Whenever you make a change in DevTools, that change gets saved to your local repository, too.

## The Changes tab

Track changes that you make locally in DevTools via the new **Changes** tab.



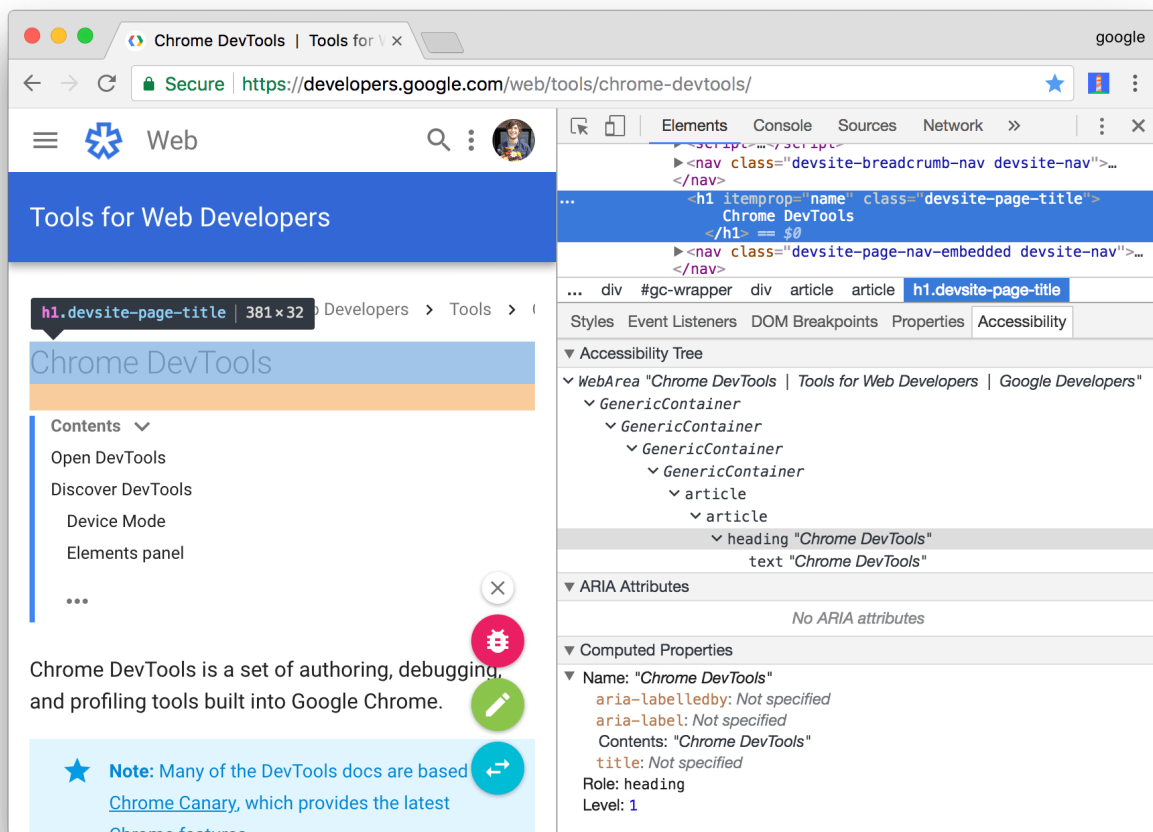
**Figure 3.** The **Changes** tab

## New accessibility tools

Use the new **Accessibility** pane to inspect the accessibility properties of an element, and inspect the contrast ratio of text elements in the **Color Picker** to ensure that they're accessible to users with low-vision impairments or color-vision deficiencies.

### Accessibility pane

Use the **Accessibility** pane on the **Elements** panel to investigate the accessibility properties of the currently-selected element.



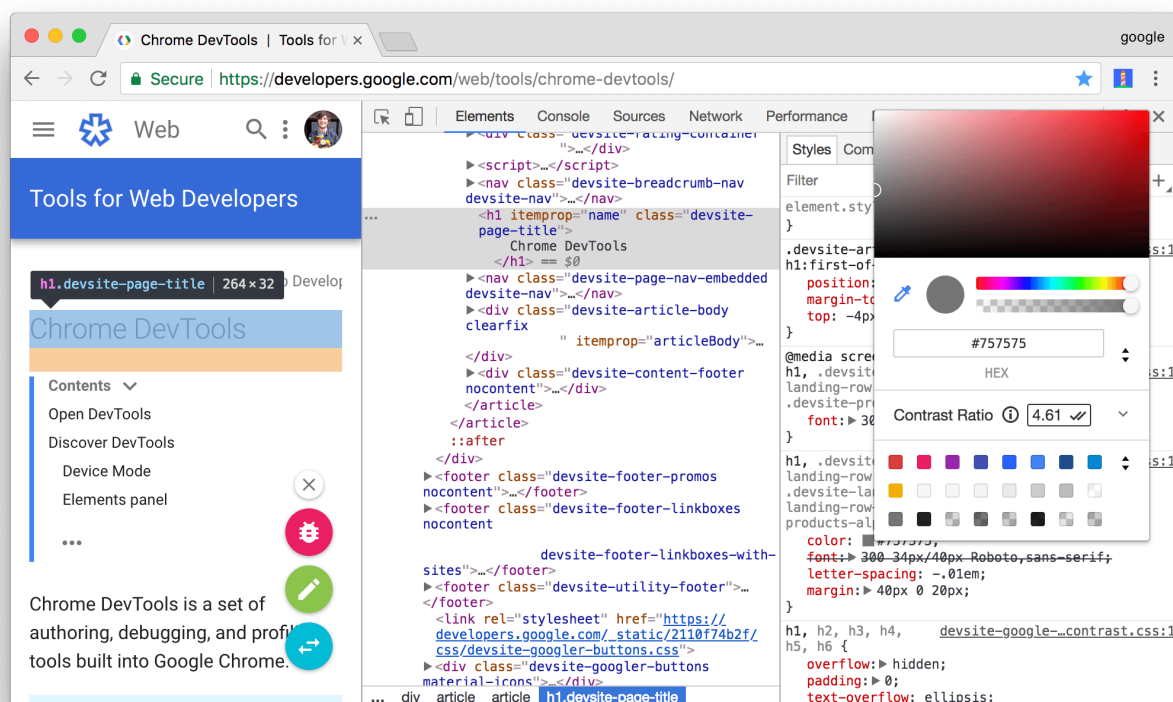
**Figure 4.** The **Accessibility** pane shows the ARIA attributes and computed properties for the element that's currently selected in the **DOM Tree** on the **Elements** panel, as well as its position in the accessibility tree

Check out Rob Dodson's A11ycast on labeling below to see the **Accessibility** pane in action.

## Contrast ratio in the Color Picker


The [Color Picker](#) now shows you the contrast ratio of text elements. Increasing the contrast ratio of text elements makes your site more accessible to users with low-vision impairments or color-vision deficiencies. See [Color and contrast](#) to learn more about how contrast ratio affects accessibility.

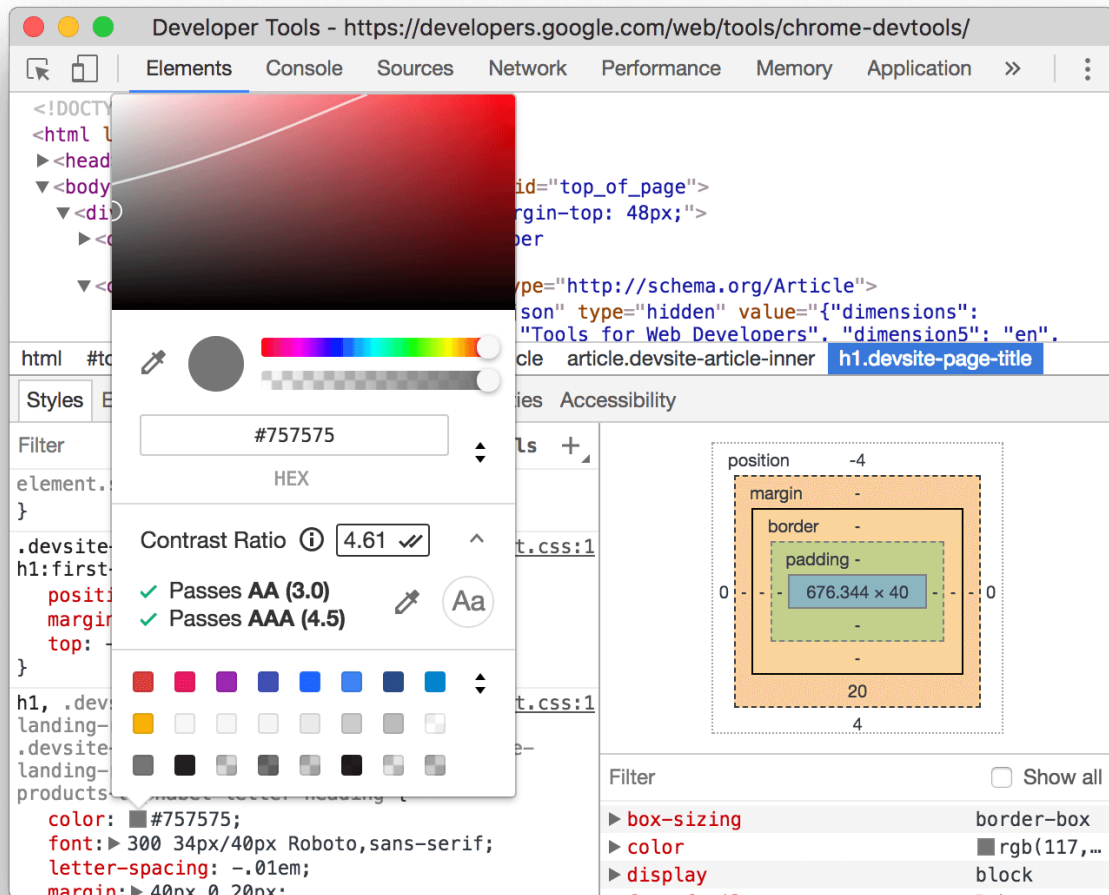
Improving the color contrast of your text elements makes your site more usable for *all* users. In other words, if your text is grey with a white background, that's hard for anyone to read.



**Figure 5.** Inspecting the contrast ratio of the highlighted h1 element

In **Figure 5**, the two checkmarks next to **4.61** means that this element meets the enhanced recommended contrast ratio (AAA). [\[1\]](#). If it only had one checkmark, that would mean it met the minimum recommended contrast ratio (AA). [\[2\]](#).

Click **Show More**  to expand the **Contrast Ratio** section. The white line in the **Color Spectrum** box represents the boundary between colors that meet the recommended contrast ratio, and those that don't. For example, since the grey color in **Figure 6** meets recommendations, that means that all of the colors below the white line also meet recommendations.



**Figure 6.** The expanded **Contrast Ratio** section

## Related features

The **Audits** panel has an automated accessibility audit for ensuring that every text element on a page has a sufficient contrast ratio.

See [Run Lighthouse in Chrome DevTools](#), or watch the A11ycast below, to learn how to use the **Audits** panel to test accessibility.

## New audits

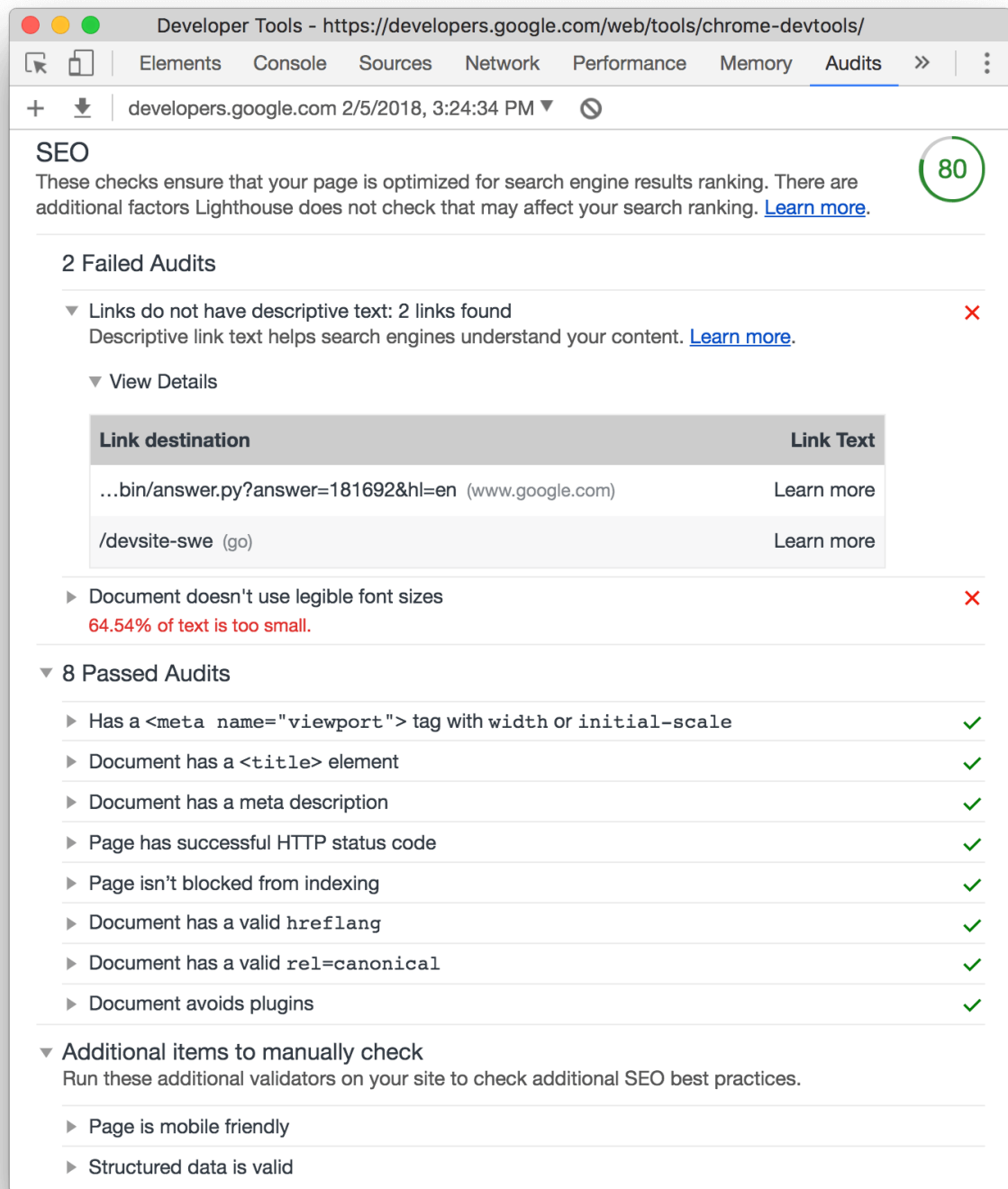
Chrome 65 ships with a whole new category of SEO audits, and many new performance audits.

**Note:** The **Audits** panel is powered by [Lighthouse](#). Chrome 64 runs Lighthouse version 2.5. Chrome 65 runs Lighthouse version 2.8. So this section is simply a summary of the Lighthouse updates from 2.6, 2.7, and 2.8.

## New SEO audits

Ensuring that your pages pass each of the audits in the new **SEO** category may help improve your search engine rankings.





**Figure 7.** The new **SEO** category of audits

## New performance audits

Chrome 65 also ships with many new performance audits:

- JavaScript boot-up time is high
- Uses inefficient cache policy on static assets

- Avoids page redirects
- Document uses plugins
- Minify CSS
- Minify JavaScript



**Perf matters!** After Mynet improved their page load speed by 4X, users spent 43% more time on the site, viewed 34% more pages, bounce rates dropped 24%, and revenue increased 25% per article pageview. [Learn more.](#)

**Tip!** If you want to improve the load performance of your pages, but don't know where to start, try the **Audits** panel. You give it a URL, and it gives you a detailed report on many different ways you can improve that page. [Get started.](#)

## Other updates

- New, manual accessibility audits
- Updates to the WebP audit to make it more inclusive of other next-generation image formats
- A rehaul of the accessibility score
- If an accessibility audit is not applicable for a page, that audit no longer counts towards the accessibility score
- Performance is now the top section in reports

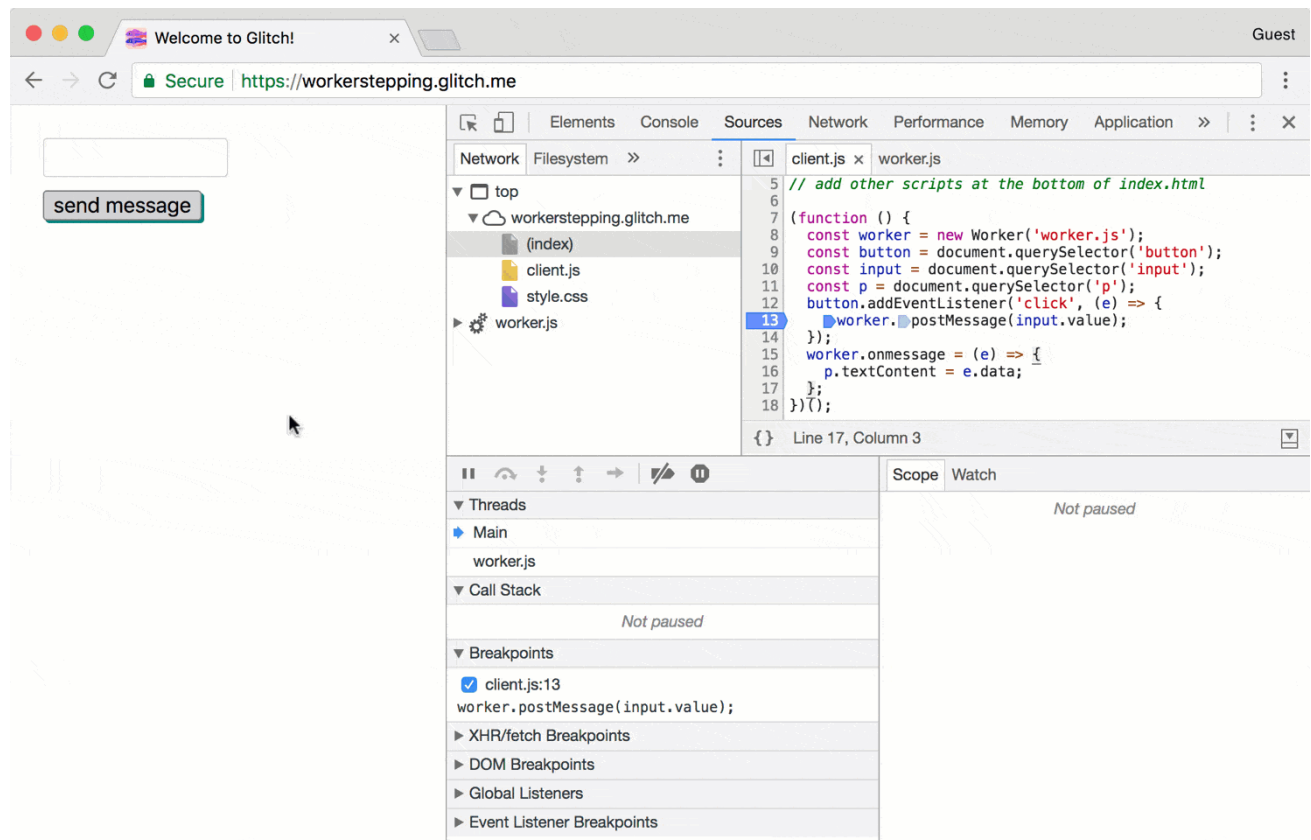
## Reliable code stepping with workers and asynchronous code

Chrome 65 brings updates to the **Step Into**  button when stepping into code that passes messages between threads, and asynchronous code. If you want the previous stepping behavior, you can use the new **Step**  button, instead.

## Stepping into code that passes messages between threads

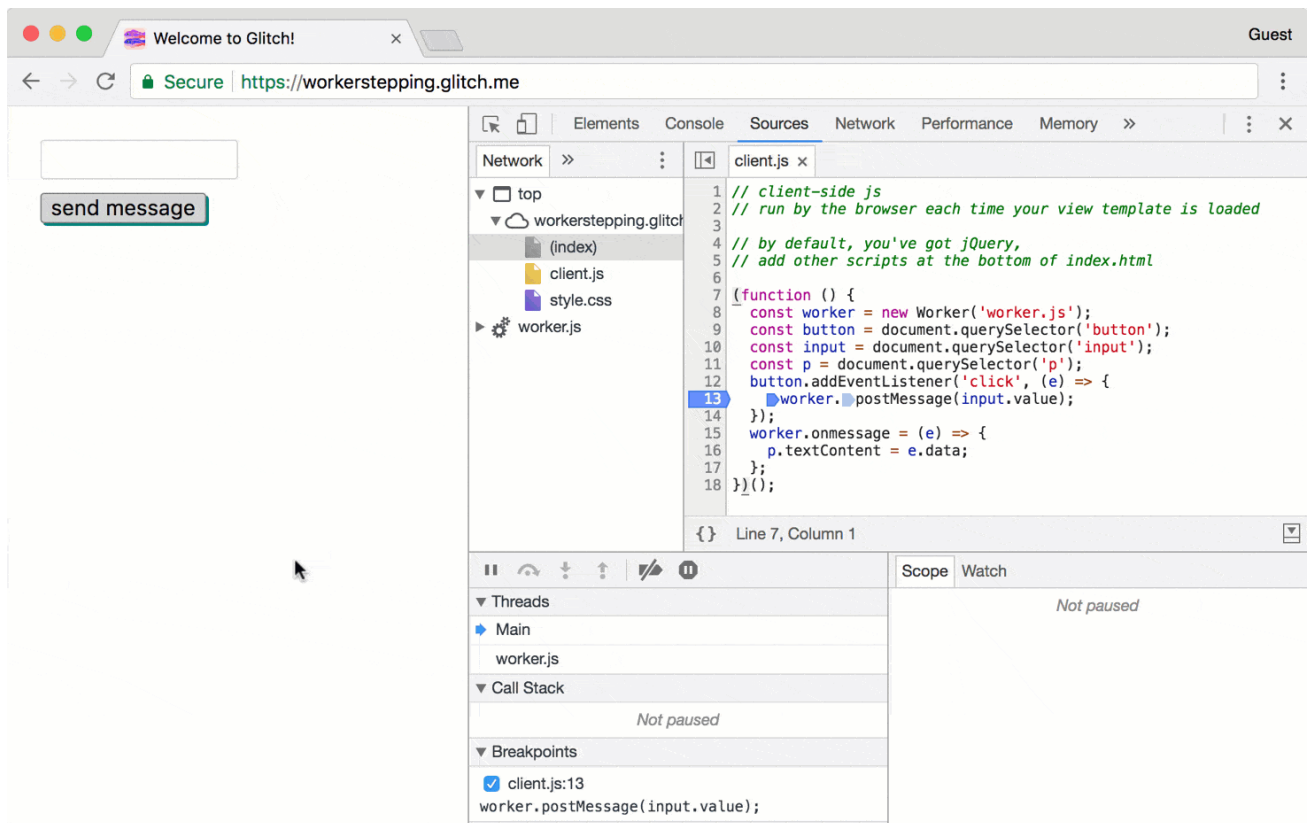
When you step into code that passes messages between threads, DevTools now shows you what happens in each thread.

For example, the app in **Figure 8** passes a message between the main thread and the worker thread. After stepping into the `postMessage()` call on the main thread, DevTools pauses in the `onmessage` handler in the worker thread. The `onmessage` handler itself posts a message back to the main thread. Stepping into *that* call pauses DevTools back in the main thread.



**Figure 8.** Stepping into message-passing code in Chrome 65

When you stepped into code like this in earlier versions of Chrome, Chrome only showed you the main-thread-side of the code, as you can see in **Figure 9**.

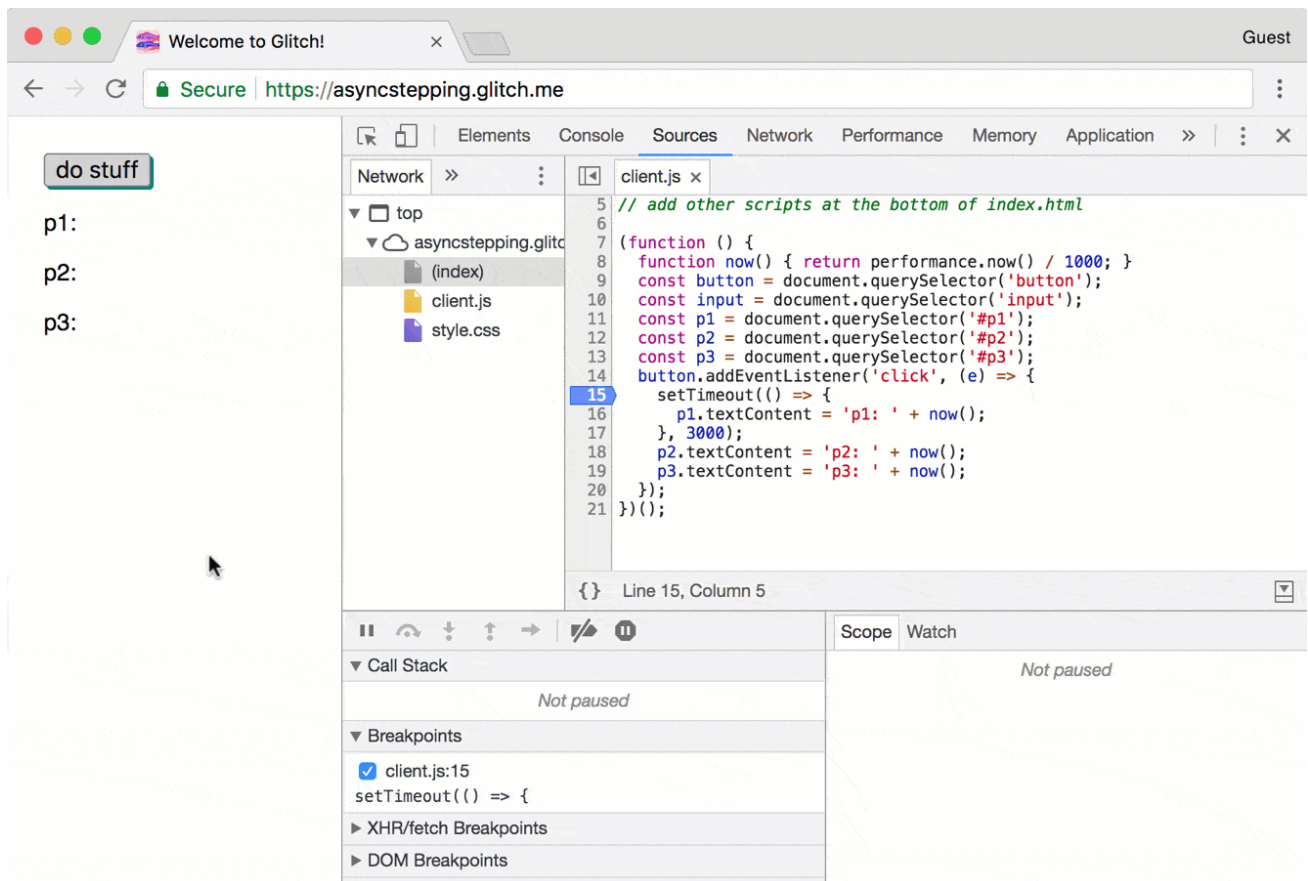


**Figure 9.** Stepping into message-passing code in Chrome 63

## Stepping into asynchronous code

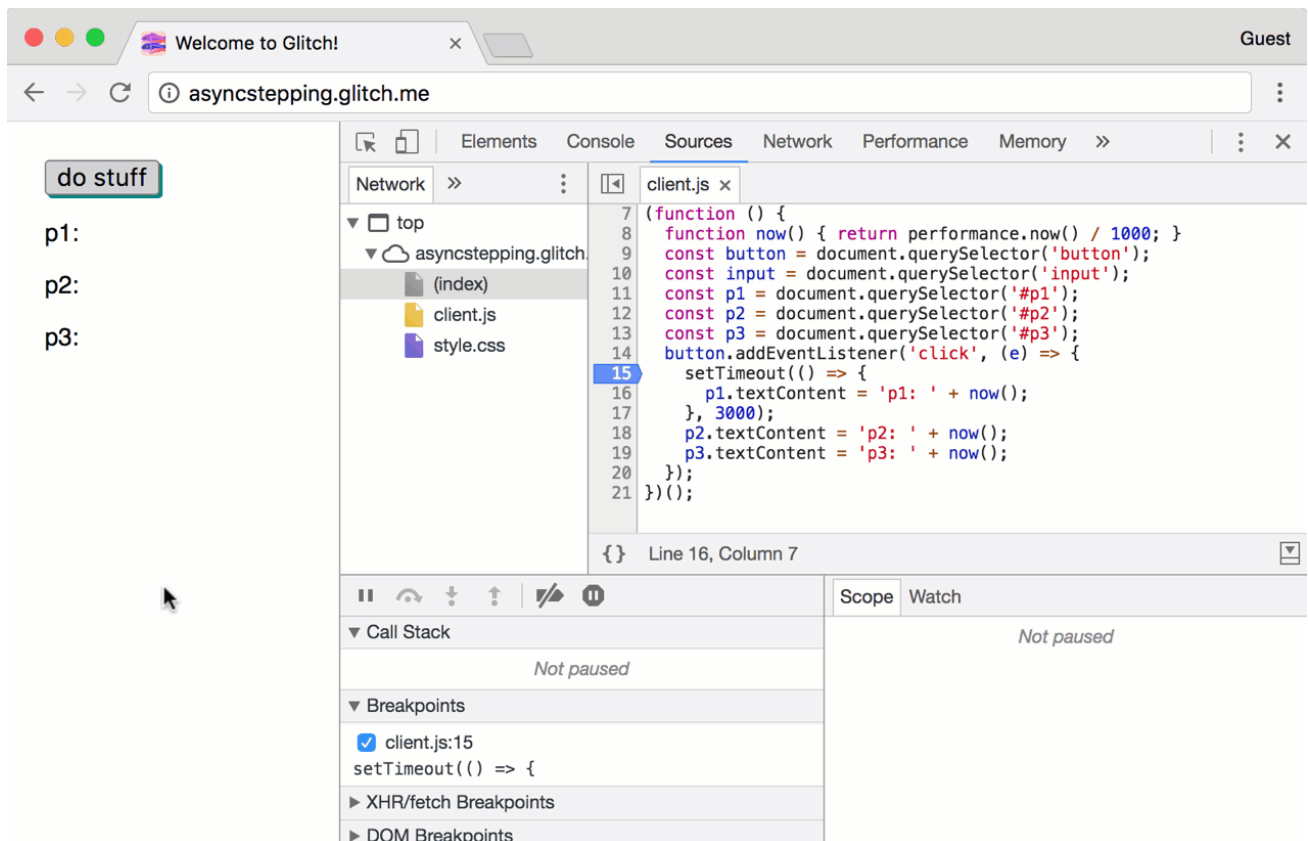
When stepping into asynchronous code, DevTools now assumes that you want to pause in the the asynchronous code that eventually runs.

For example, in **Figure 10** after stepping into `setTimeout()`, DevTools runs all of the code leading up to that point behind the scenes, and then pauses in the function that's passed to `setTimeout()`.



**Figure 10.** Stepping into asynchronous code in Chrome 65

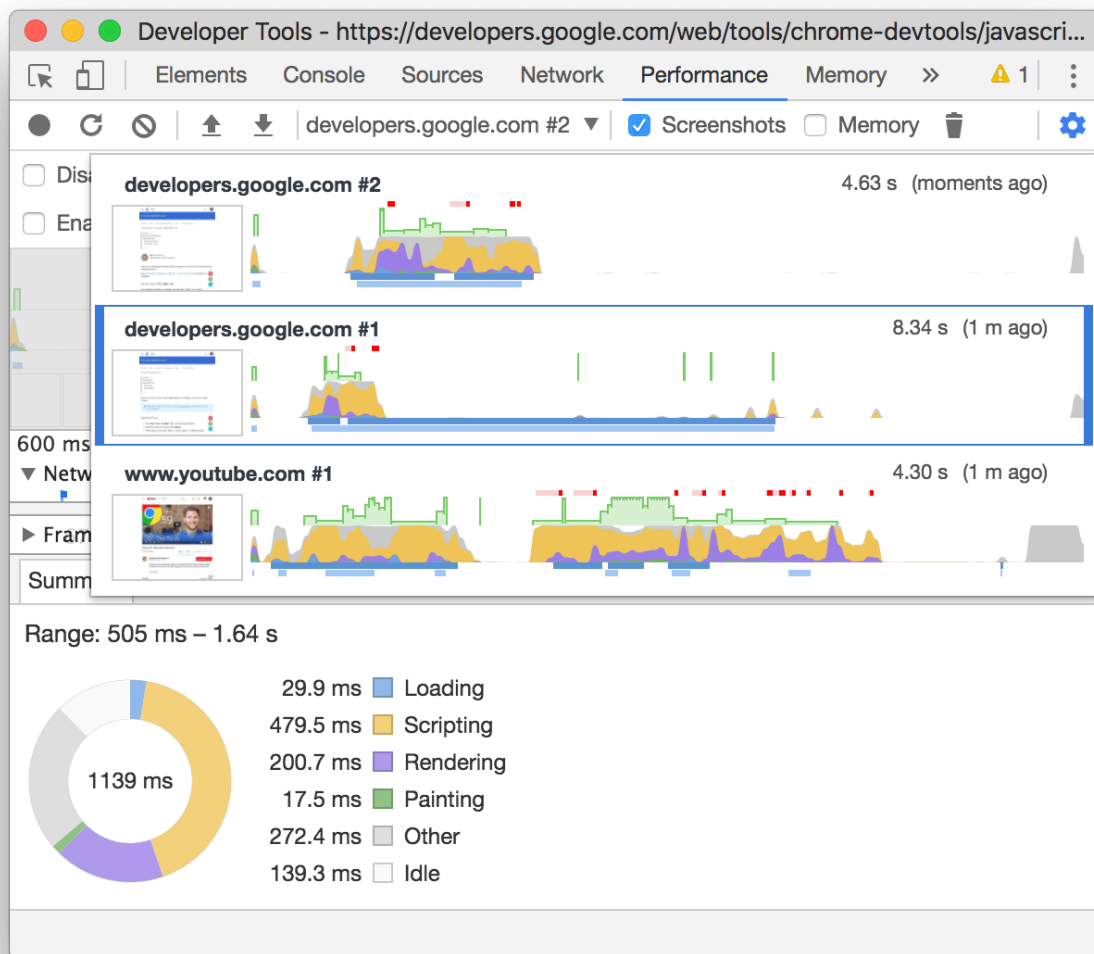
When you stepped into code like this in Chrome 63, DevTools paused in code as it chronologically ran, as you can see in **Figure 11**.



**Figure 11.** Stepping into asynchronous code in Chrome 63

## Multiple recordings in the Performance panel

The **Performance** panel now lets you temporarily save up to 5 recordings. The recordings are deleted when you close your DevTools window. See [Get Started with Analyzing Runtime Performance](#) to get comfortable with the **Performance** panel.



**Figure 12.** Selecting between multiple recordings in the **Performance** panel

## Bonus: Automate DevTools actions with Puppeteer 1.0

**Note:** This section isn't related to Chrome 65.

Version 1.0 of Puppeteer, a browser automation tool maintained by the Chrome DevTools team, is now out. You can use Puppeteer to automate many tasks that were previously only available via DevTools, such as capturing screenshots:



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

It also has APIs for lots of generally useful automation tasks, such as generating PDFs:



```
const puppeteer = require('puppeteer');
(async () => {
  const browser = await puppeteer.launch();
  const page = await browser.newPage();
  await page.goto('https://news.ycombinator.com', {waitUntil: 'networkidle2'});
  await page.pdf({path: 'hn.pdf', format: 'A4'});
  await browser.close();
})();
```

See [Quick Start](#) to learn more.

You can also use Puppeteer to expose DevTools features while browsing without ever explicitly opening DevTools. See [Using DevTools Features Without Opening DevTools](#) for an example.

## A request from the DevTools team: consider Canary

If you're on Mac or Windows, please consider using [Chrome Canary](#) as your default development browser. If you report a bug or a change that you don't like while it's still in Canary, the DevTools team can address your feedback significantly faster.

**Note:** Canary is the bleeding-edge version of Chrome. It's released as soon as it's built, without testing. This means that Canary breaks from time-to-time, about once-a-month, and it's usually fixed within a day. You can go back to using Chrome Stable when Canary breaks.

## Feedback

The best place to discuss any of the features or changes you see here is the [google-chrome-developer-tools@googlegroups.com mailing list](mailto:google-chrome-developer-tools@googlegroups.com). You can also tweet us at [@ChromeDevTools](https://twitter.com/ChromeDevTools) if you're short on time. If you're sure that you've encountered a bug in DevTools, please [open an issue](#).

## Previous release notes

See the [devtools-whatsnew](#) tag for links to all previous DevTools release notes.



Subscribe to our [RSS](#) or [Atom](#) feed and get the latest **updates** in your favorite feed reader!

---

*Except as otherwise noted, the content of this page is licensed under the [Creative Commons Attribution 3.0 License](#), and code samples are licensed under the [Apache 2.0 License](#). For details, see our [Site Policies](#). Java is a registered trademark of Oracle and/or its affiliates.*

*Last updated July 2, 2018.*