DevTools Digest: DevTools in 2016 and Beyond

By Kayce Basques
Technical Writer for Chrome DevTools

Google I/O 2016 is a wrap. DevTools had a strong presence at I/O, including a talk by Paul Bakaus, Paul Irish, and Seth Thompson outlining the future of DevTools. Check out the video below or read on to learn more about where DevTools is headed in 2016 and beyond.

Authoring

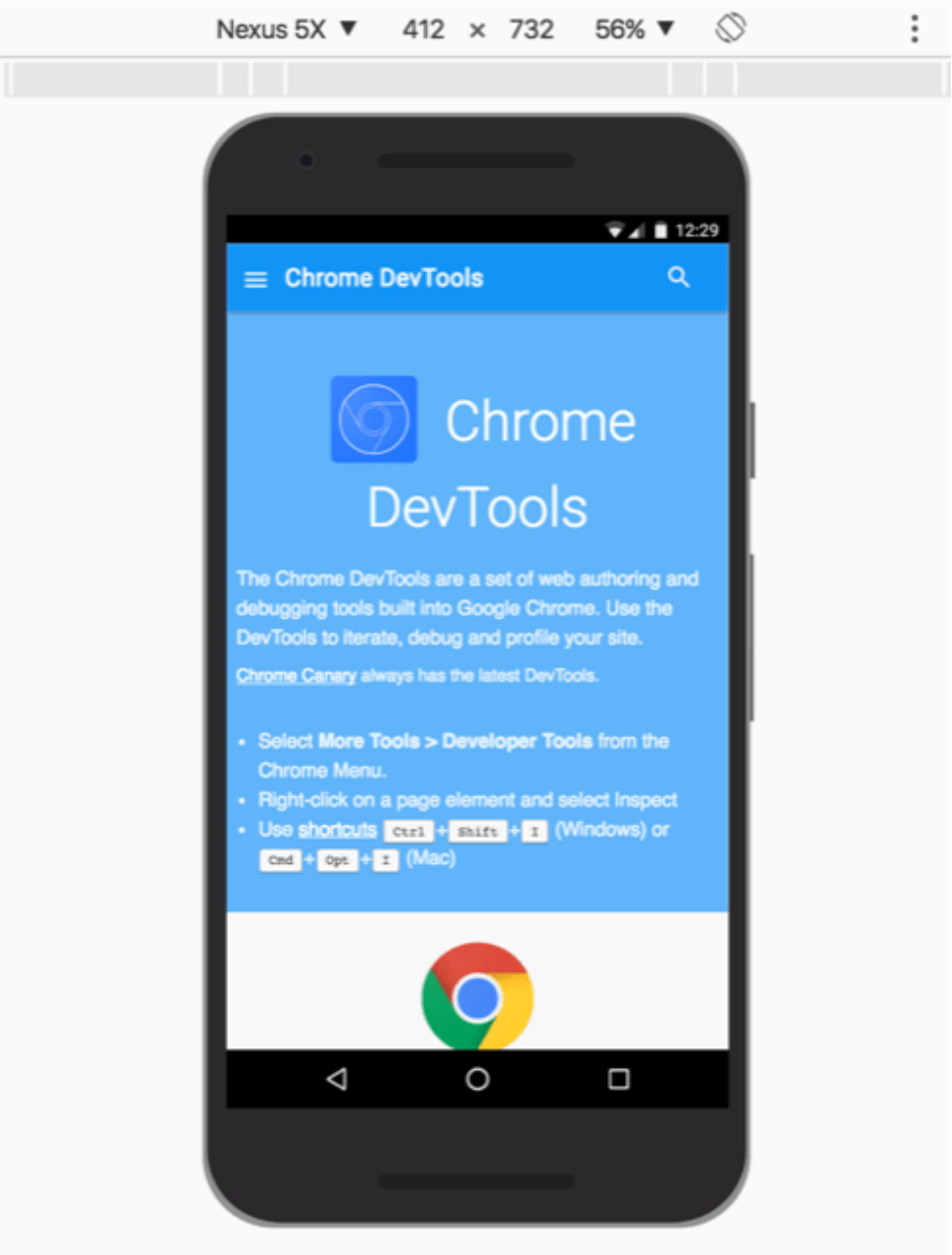
DevTools aims to make every stage of the web development lifecycle easier. You probably know that DevTools can help you debug or profile a website, but you may not know how to use it to help you author a site. By "authoring" we mean the act of creating a site. One of the goals in the foreseeable future is to make it easier for you to iterate, experiment, and emulate your site across multiple devices.

Device mode

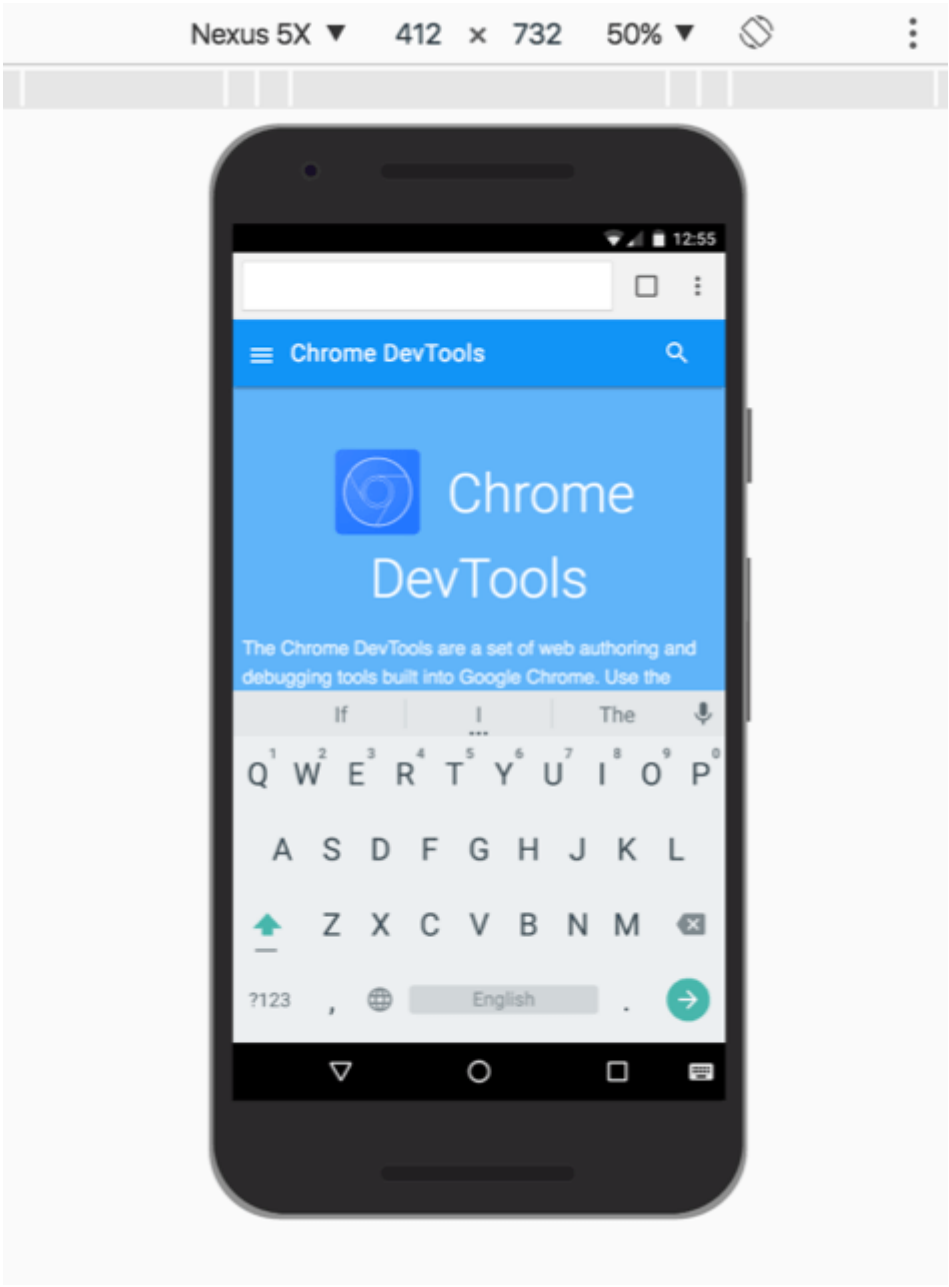
The DevTools team continues to iterate on the Device Mode experience, which received its first major upgrade in Chrome 49. Check out the post from March (A new Device Mode for a mobile-first world) for an overview of the updates. The overarching goal is to provide a seamless workflow for viewing and emulating your site across all form factors.

Here's a quick summary of some Device Mode updates that have landed in Canary since we posted the article back in March.

When viewing a page as a specific device, you can immerse yourself more in the experience by showing the device hardware around your page.

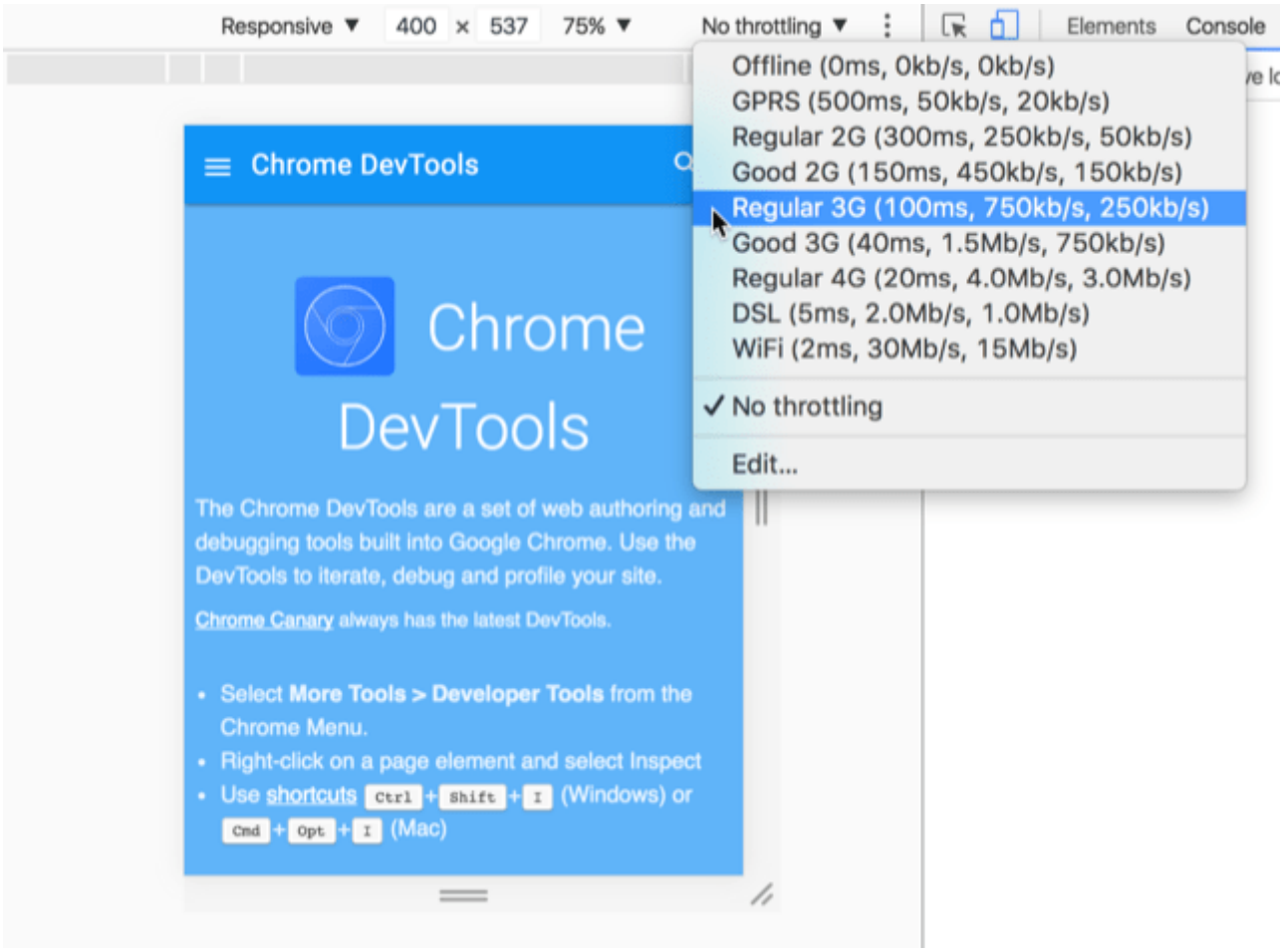


The device orientation menu lets you view your page when different system UI elements, such as the navigation bar and keyboard, are active.



The desktop story has improved, too. Thanks to Device Mode's zoom feature, you can emulate desktop screens larger than the screen that you're actually working on, such as a 4K (3840px x 2160px) screen.

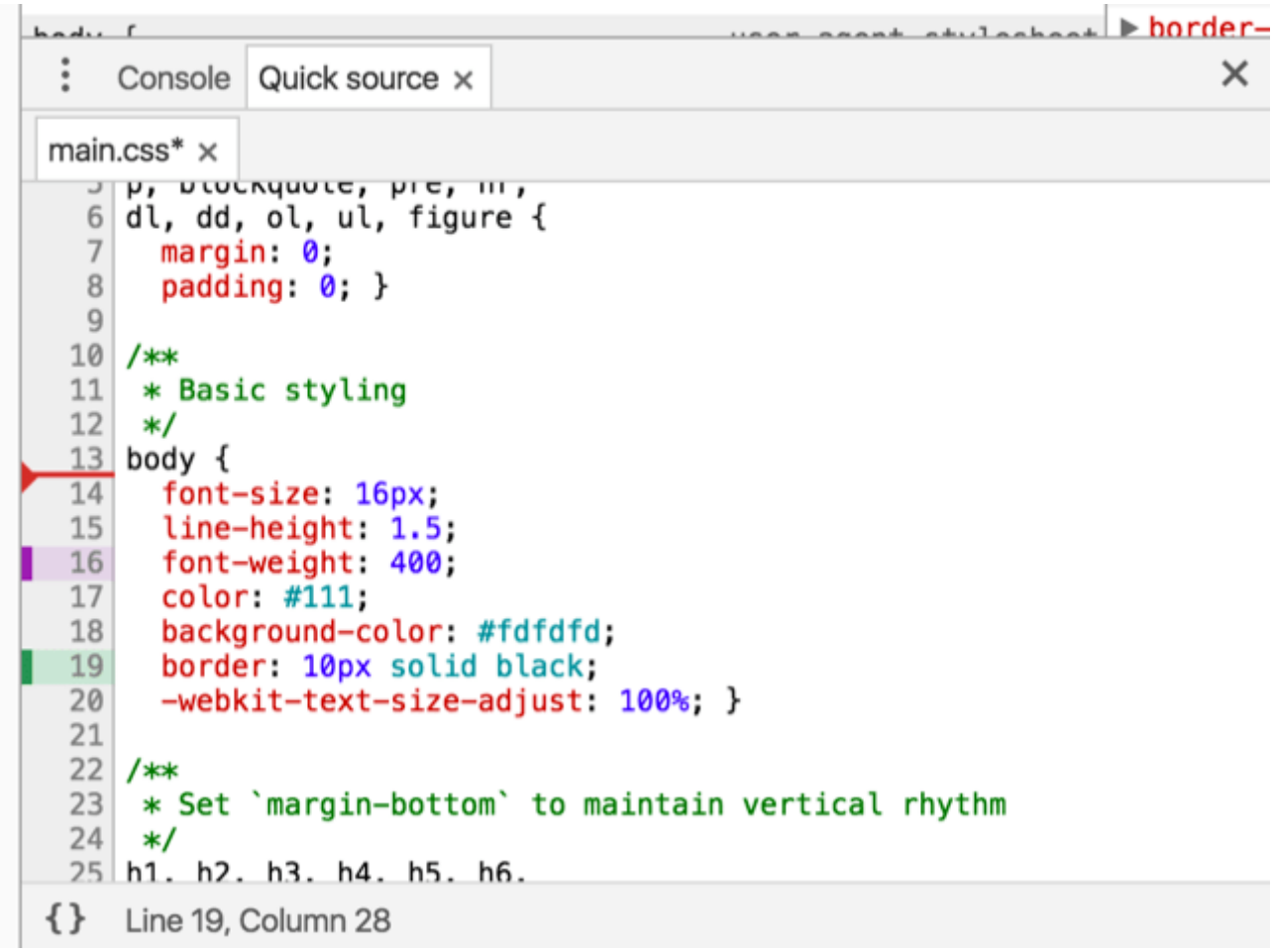
You can throttle the network directly from the Device Mode UI, rather than having to switch to the Network panel.



Source diffs

When you iterate upon a site's styling, it's easy to lose track of your changes. To fix this, DevTools is going to use visual cues on the line number gutter of the Sources panel to help you keep track of your changes. Deleted lines are indicated with a red line, modified lines are highlighted purple, and new lines are highlighted green.

You'll also be able to keep track of your changes in the new Quick Source drawer tab:



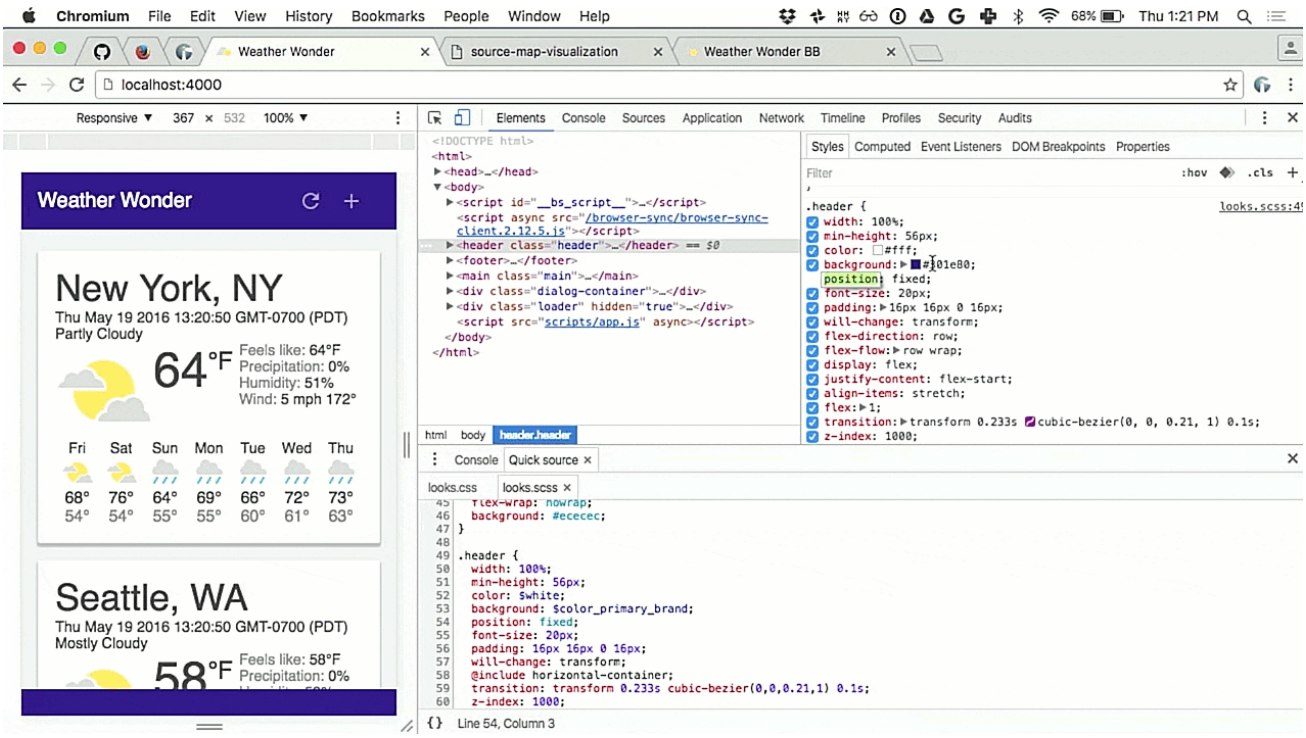
For the first time, the Quick Source tab lets you focus on your HTML or JavaScript source code at the same time as your CSS rules. Also, when you click a CSS property in the Styles pane, the Quick Source tab automatically jumps to and highlights the definition in the source.

Enable the **sources diff** experiment in Chrome Canary to try it out today.

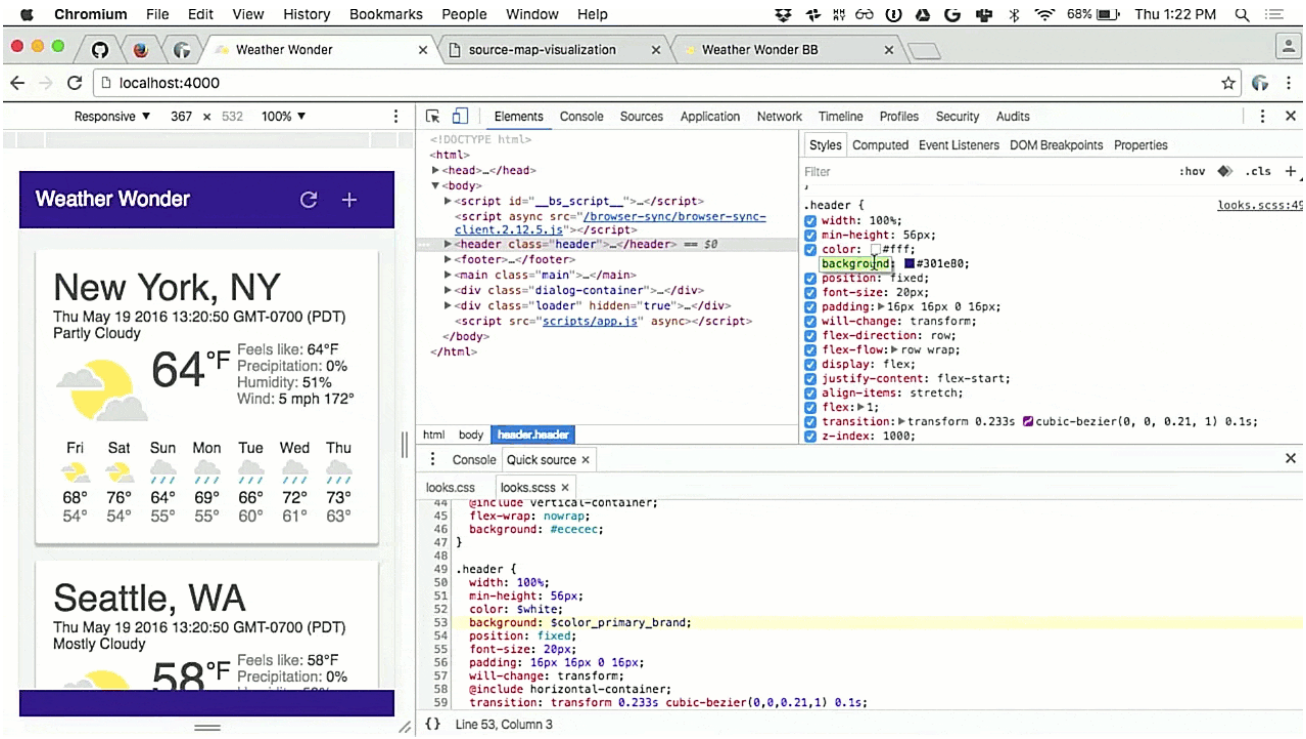
Live Sass editing

Here's a sneak peek of some upcoming major improvements to the Sass editing workflow. These features are very early in the experimental phase. We'll follow up in a later post when they're ready for you to try out.

Basically, DevTools is going to let you view and edit Sass variables like you always hoped it would. Click on a value that was compiled from a Sass variable, and the new Quick Sources tab jumps to the definition of the variable.



When editing a value that was compiled from a Sass variable, your edit updates the Sass variable, not just the individual property that you selected.

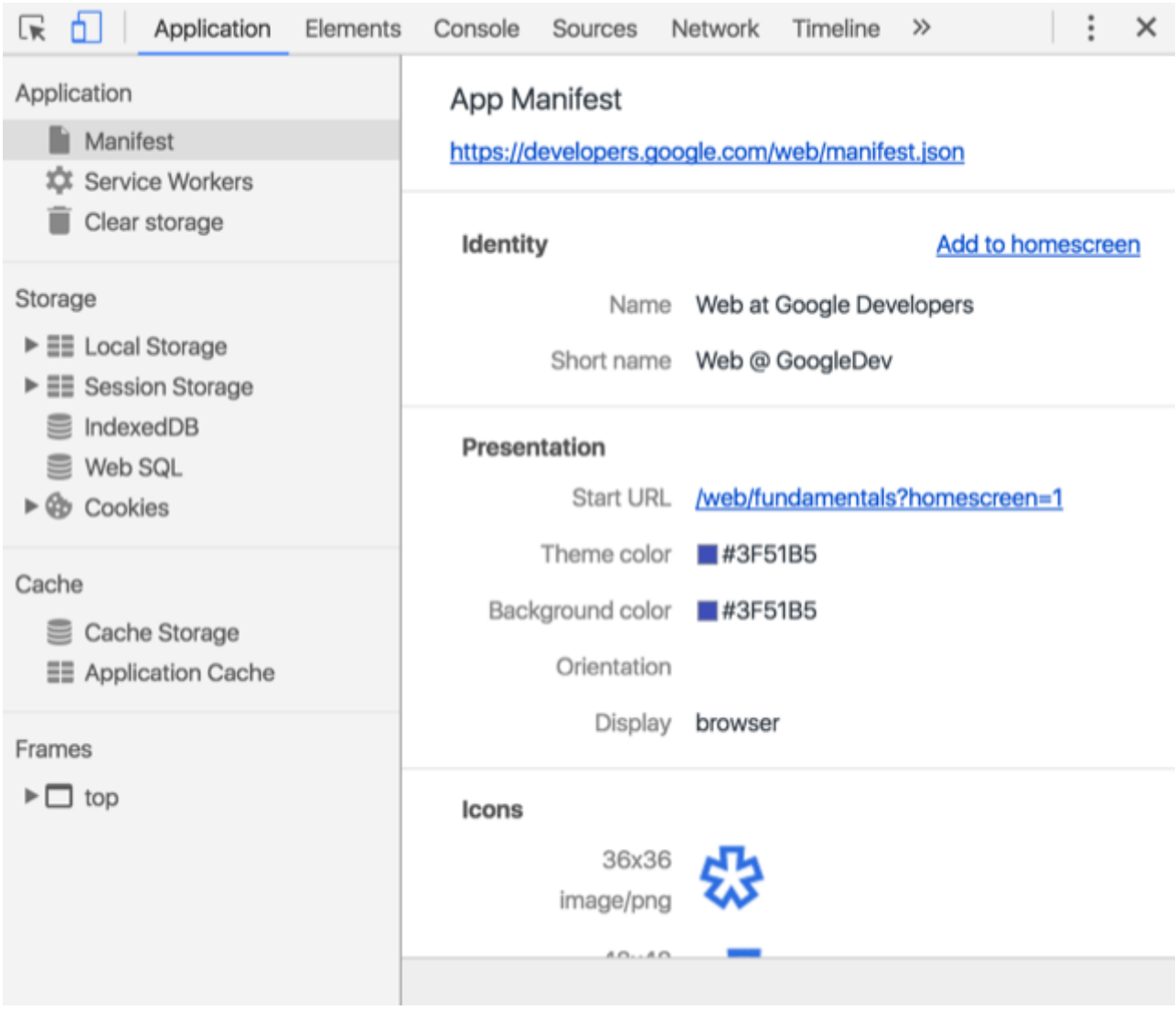


Progressive web apps

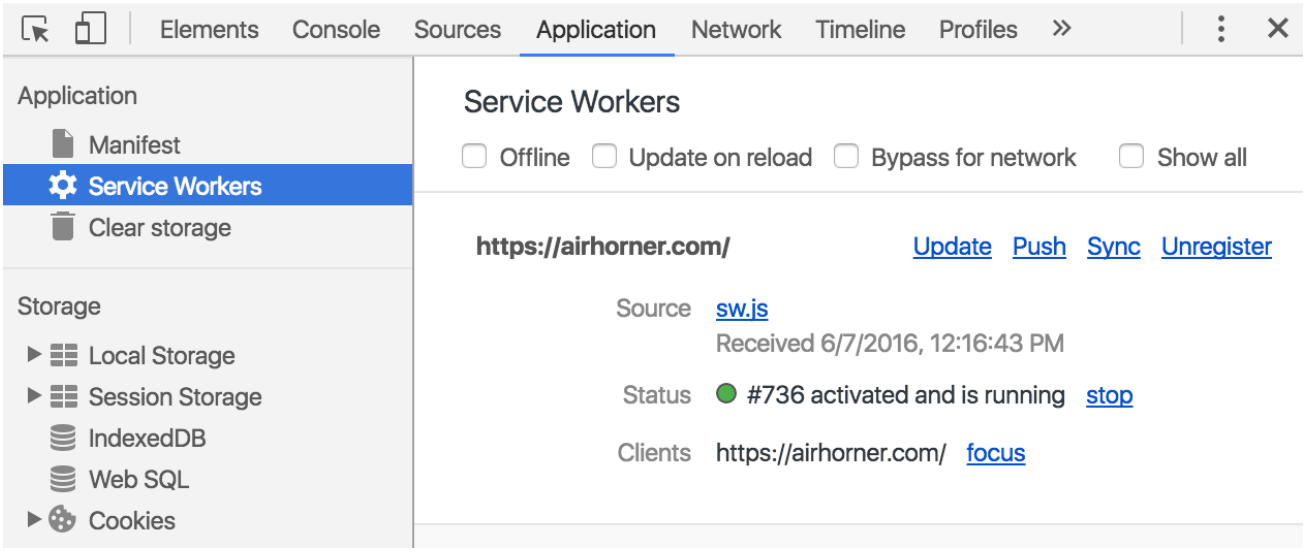
Look at the list of web and Chrome talks at Google I/O 2016 and you'll see a huge theme emerging in the world of web development: Progressive Web Apps.

As the Progressive Web App model emerges, DevTools is iterating rapidly to provide the tools developers need to create great progressive web apps. We realized that building and debugging these modern applications often comes with unique requirements, so DevTools has dedicated an entire panel to Progressive Web Application development. Open up Chrome Canary and you'll see that the Resources panel has been replaced with the Application panel. All of the previous functionality from the Resources panel is still there. There's just a few new panes designed specifically for Progressive Web App development:

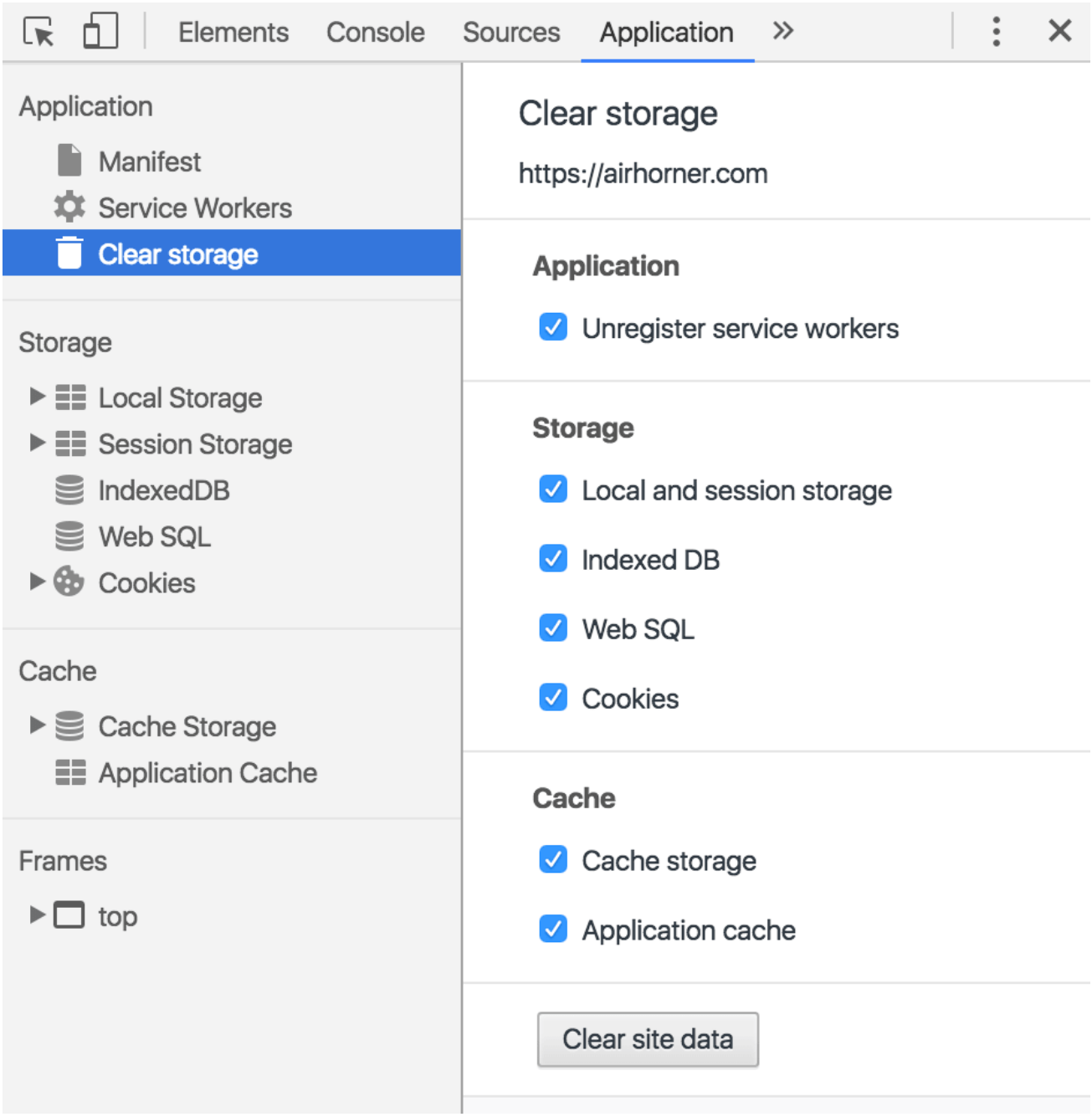
The Manifest pane gives you a visual representation of the app manifest file. From here you can manually trigger the "Add to home screen" workflow.



The Service Workers pane lets you inspect and interact with the service worker registered for the current page.



And the Clear Storage pane lets you wipe all sorts of data so that you can view a page with a clean slate.



JavaScript

Crossing the boundary between frontend and backend is a difficult part of fullstack web development. If you discover that your backend is returning a 500 status code while debugging a web app, you have effectively reached the limit of DevTools' usefulness, requiring you to change contexts and fire up your backend development environment to debug the problem.

With backends written in Node.js, however, the boundaries between frontend and backend are starting to blur. Since Node.js runs on the V8 JavaScript engine (the same engine that

powers Google Chrome) we wanted to make it possible to debug Node.js from DevTools. Thanks to the V8, DevTools, and Google Cloud Platform for Node.js teams, you can now use all of DevTools' powerful debugging features to introspect a Node.js app. The functionality has already reached Node.js nightly builds, although DevTools integration is still being polished before being included in a major release. Debugging your Node.js app from DevTools will someday be as simple as passing `node --inspect app.js` and connecting from DevTools in any Chrome window.

Although existing tools such as Node Inspector provide GUI-based debugging experiences, the new Node.js DevTools integration will remain up-to-date with DevTools' latest debuging features, such as async stack debugging, blackboxing, and ES6 support.

Last updated July 2, 2018.