What's New In DevTools (Chrome 60)

By Kayce Basques

Technical Writer for Chrome DevTools

Welcome! New features and major changes coming to DevTools in Chrome 60 include:

- A new Audits panel, including tests for progressive web apps, performance, accessibility, and best practices.
- Third-party badges. Find out which third-parties are making network requests, logging to the Console, and executing JavaScript.
- Continue To Here. A new gesture that can speed up your JavaScript debugging workflow.
- Predictable debugging for asynchronous JavaScript.
- Object previews in the Console.
- Real-time updates in the Coverage tab.
- A new menu for selecting contexts in the Console.
- Simpler network throttling options.
- Async stack traces on by default.

Note: You can check what version of Chrome you're running at `chrome://version`. Chrome auto-updates to a new major version about every 6 weeks.

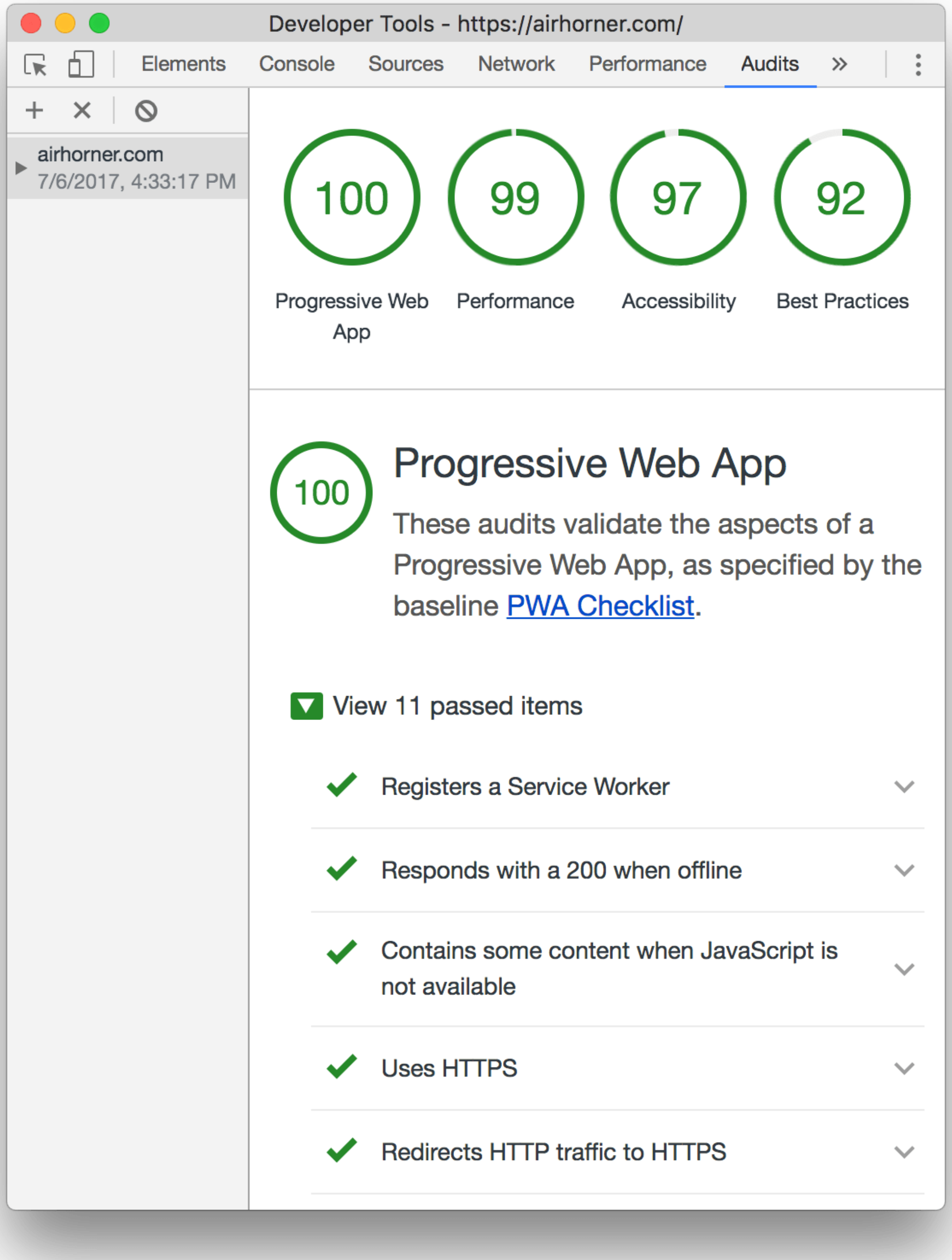
Check out the video version of these release notes below or read on to learn more.

New features

New Audits panel, powered by Lighthouse

The Audits panel is now powered by Lighthouse. Lighthouse provides a comprehensive set of tests for measuring the quality of your web pages.

The scores at the top for **Progressive Web App**, **Performance**, **Accessibility**, and **Best Practices** are your aggregate scores for each of those categories. The rest of the report is a breakdown of each of the tests that determined your scores. Improve the quality of your web page by fixing the failing tests.



Progressive Web App

These audits validate the aspects of a Progressive Web App, as specified by the baseline PWA Checklist.

View 11 passed items

- ✓ Registers a Service Worker
- ✓ Responds with a 200 when offline
- ✓ Contains some content when JavaScript is not available
- ✓ Uses HTTPS
- ✓ Redirects HTTP traffic to HTTPS

Figure 1. A Lighthouse report

To audit a page:

1. Click the **Audits** tab.
2. Click **Perform an audit**.

3. Click **Run audit**. Lighthouse sets up DevTools to emulate a mobile device, runs a bunch of tests against the page, and then displays the results in the **Audits** panel.

Lighthouse at Google I/O '17

Check out the DevTools talk from Google I/O '17 below to learn more about Lighthouse's integration in DevTools.

Note: The video should start playing at 32:30, which is when Paul discusses Lighthouse.

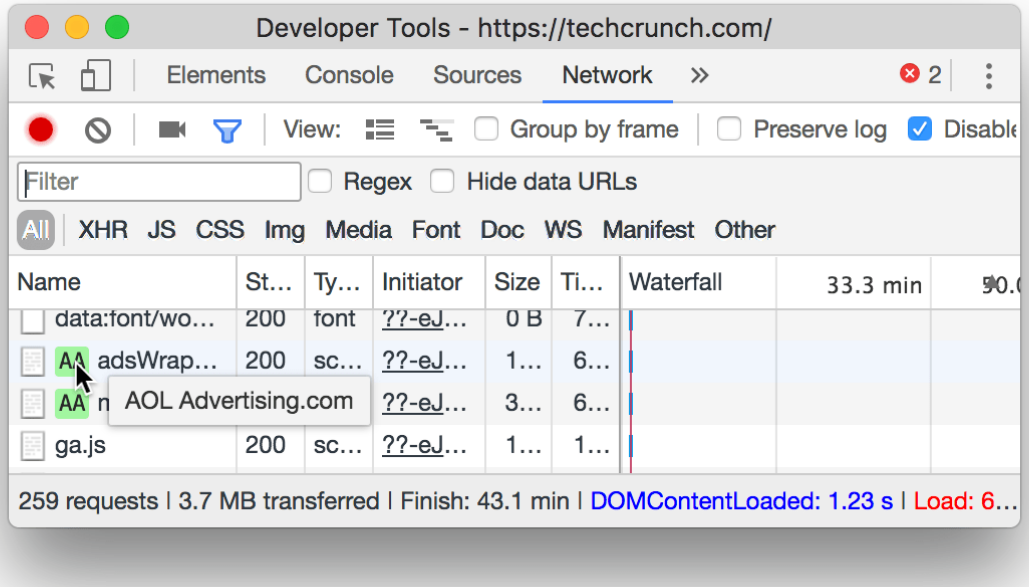
Contribute to Lighthouse

Lighthouse is an open-source project. To learn lots more about how it works and how to contribute to it, check out the Lighthouse talk from Google I/O '17 below.

Got an idea for a Lighthouse audit? Post it here!

Third-party badges

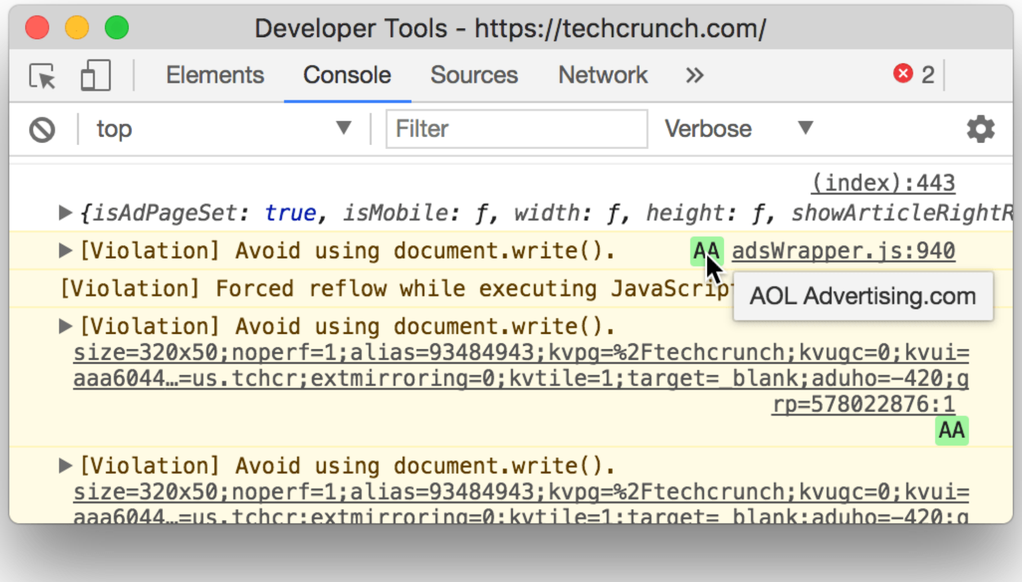
Use third-party badges to get more insight into the entities that are making network requests on a page, logging to the Console, and executing JavaScript.



Name	St…	Ty…	Initiator	Size	Ti…	Waterfall
data:font/wo…	200	font	??-eJ…	0 B	7…	
AA adsWrap…	200	sc…	??-eJ…	1…	6…	
AA n			??-eJ…	3…	6…	
ga.js	200	sc…	??-eJ…	1…	1…	

259 requests | 3.7 MB transferred | Finish: 43.1 min | DOMContentLoaded: 1.23 s | Load: 6…

Figure 2. Hovering over a third-party badge in the Network panel



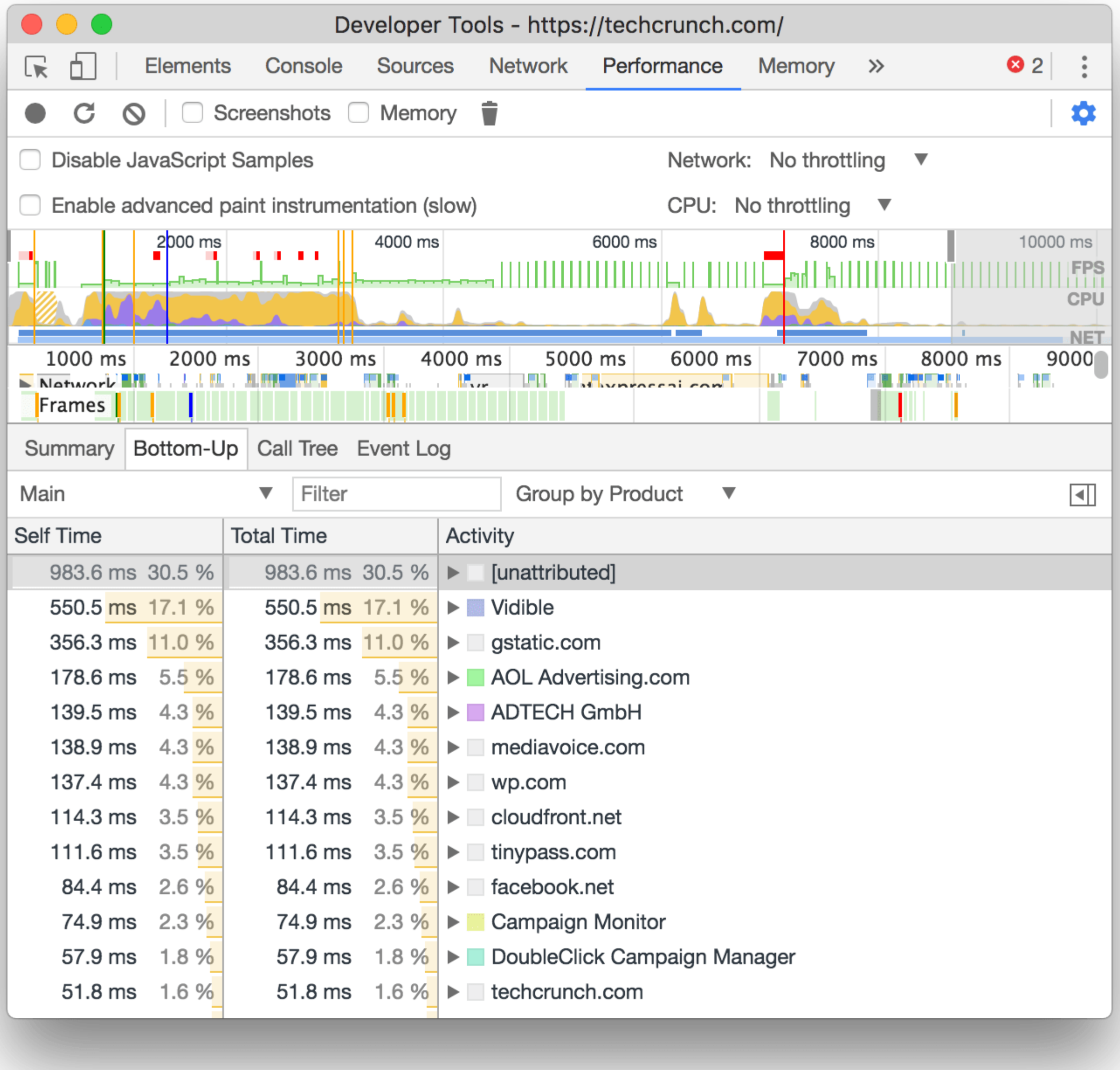
```
                                                               (index):443
▶ {isAdPageSet: true, isMobile: f, width: f, height: f, showArticleRightR
▶ [Violation] Avoid using document.write().             AA adsWrapper.js:940
[Violation] Forced reflow while executing JavaScrip
▶ [Violation] Avoid using document.write().
  size=320x50;noperf=1;alias=93484943;kvpg=%2Ftechcrunch;kvugc=0;kvui=
  aaa6044…=us.tchcr;extmirroring=0;kvtile=1;target=_blank;aduho=-420;g
                                                           rp=578022876:1
                                                                       AA
▶ [Violation] Avoid using document.write().
  size=320x50;noperf=1;alias=93484943;kvpg=%2Ftechcrunch;kvugc=0;kvui=
  aaa6044…=us.tchcr;extmirroring=0;kvtile=1;target=_blank;aduho=-420;g
```

Figure 3. Hovering over a third-party badge in the Console

To enable third-party badges:

1. Open the Command Menu.
2. Run the `Show third party badges` command.

Use the **Group by product** option in the **Call Tree** and **Bottom-Up** tabs to group performance recording activity by the third-party entities that caused the activities. See Get Started With Analyzing Runtime Performance to learn how to analyze performance with DevTools.



☐ Disable JavaScript Samples

☐ Enable advanced paint instrumentation (slow)

Network: No throttling ▼

CPU: No throttling ▼

Main ▼

Group by Product ▼

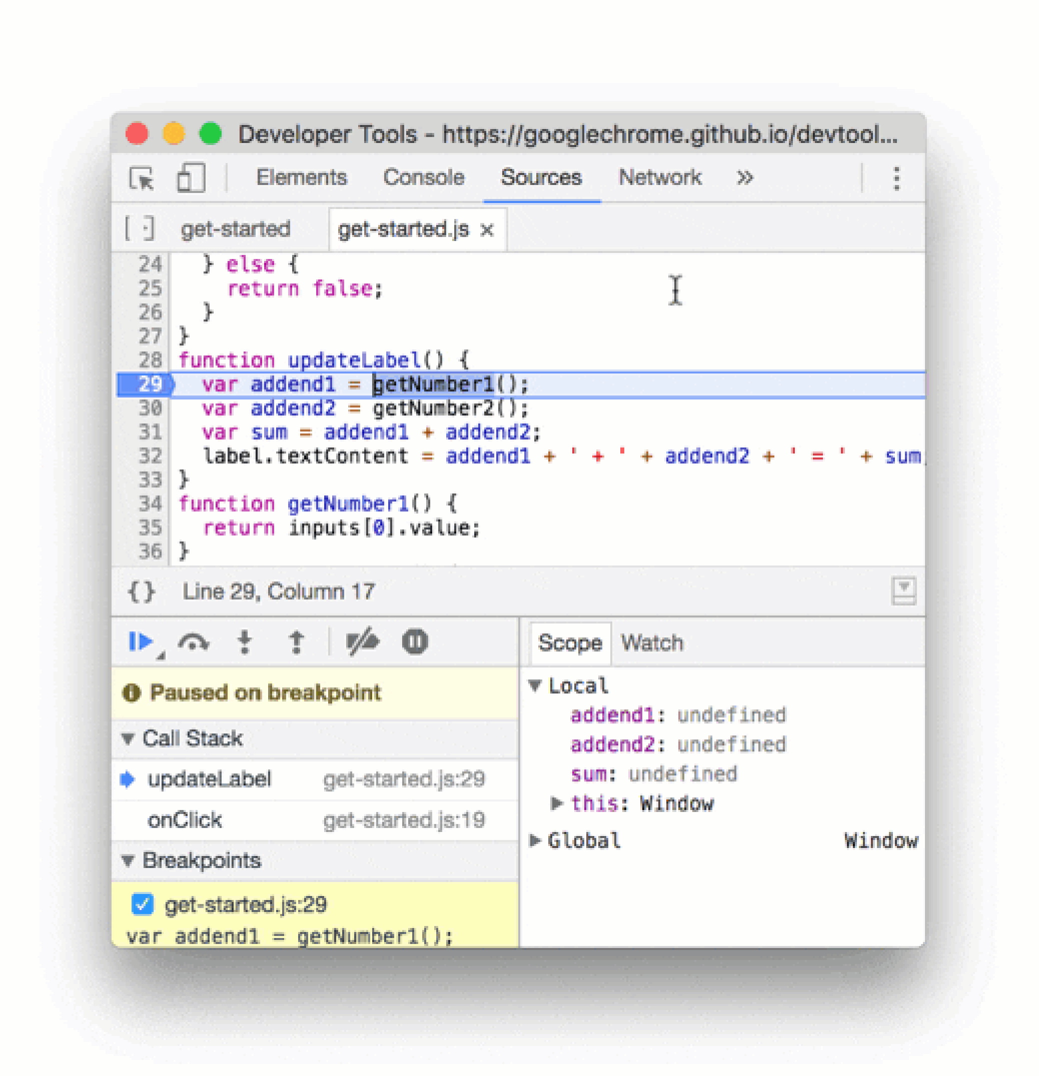
Self Time		Total Time		Activity
983.6 ms	30.5 %	983.6 ms	30.5 %	▶ [unattributed]
550.5 ms	17.1 %	550.5 ms	17.1 %	▶ Vidible
356.3 ms	11.0 %	356.3 ms	11.0 %	▶ gstatic.com
178.6 ms	5.5 %	178.6 ms	5.5 %	▶ AOL Advertising.com
139.5 ms	4.3 %	139.5 ms	4.3 %	▶ ADTECH GmbH
138.9 ms	4.3 %	138.9 ms	4.3 %	▶ mediavoice.com
137.4 ms	4.3 %	137.4 ms	4.3 %	▶ wp.com
114.3 ms	3.5 %	114.3 ms	3.5 %	▶ cloudfront.net
111.6 ms	3.5 %	111.6 ms	3.5 %	▶ tinypass.com
84.4 ms	2.6 %	84.4 ms	2.6 %	▶ facebook.net
74.9 ms	2.3 %	74.9 ms	2.3 %	▶ Campaign Monitor
57.9 ms	1.8 %	57.9 ms	1.8 %	▶ DoubleClick Campaign Manager
51.8 ms	1.6 %	51.8 ms	1.6 %	▶ techcrunch.com

Figure 4. Grouping by product in the **Bottom-Up** tab

A new gesture for Continue to Here

Say you're paused on line 25 of a script, and you want to jump to line 50. In the past, you could set a breakpoint on line 50, or right-click the line and select **Continue to here**. But now, there's a faster gesture for handling this workflow.

When stepping through code, hold Command (Mac) or Control (Windows, Linux) and then click to continue to that line of code. DevTools highlights the jumpable destinations in blue.



```
  } else {
    return false;
  }
}
function updateLabel() {
  var addend1 = getNumber1();
  var addend2 = getNumber2();
  var sum = addend1 + addend2;
  label.textContent = addend1 + ' + ' + addend2 + ' = ' + sum
}
function getNumber1() {
  return inputs[0].value;
}
```

Paused on breakpoint

▼ Call Stack

updateLabel	get-started.js:29
onClick	get-started.js:19

▼ Breakpoints

☑ get-started.js:29
`var addend1 = getNumber1();`

```
▼ Local
    addend1: undefined
    addend2: undefined
    sum: undefined
  ▶ this: Window
▶ Global                    Window
```

Figure 5. Continue To Here

See Get Started With Debugging JavaScript to learn the basics of debugging in DevTools.

Step into async

A big theme for the DevTools team in the near future is to make debugging asynchronous code predictable, and to provide you a complete history of asynchronous execution.

The new gesture for Continue to Here also works with asynchronous code. When you hold Command (Mac) or Control (Windows, Linux), DevTools highlights jumpable asynchronous destinations in green.

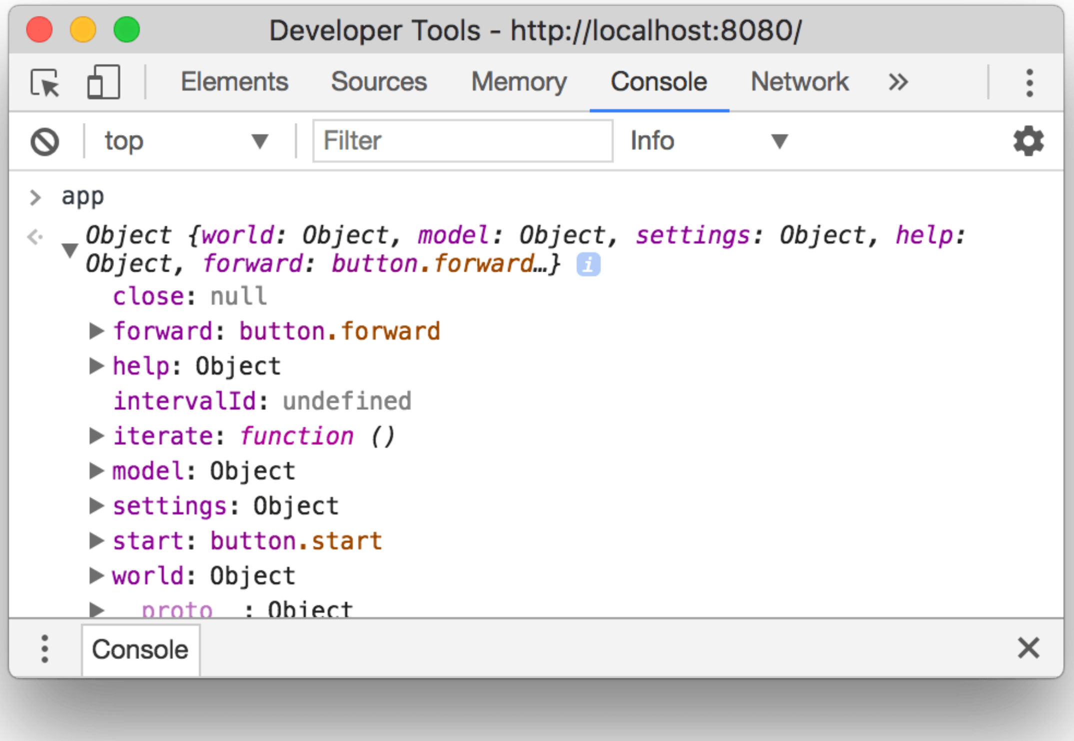
Check out the demo below from the DevTools talk at I/O for an example.

Note: The video should start playing at 17:40, which is when Paul discusses the feature.

Changes

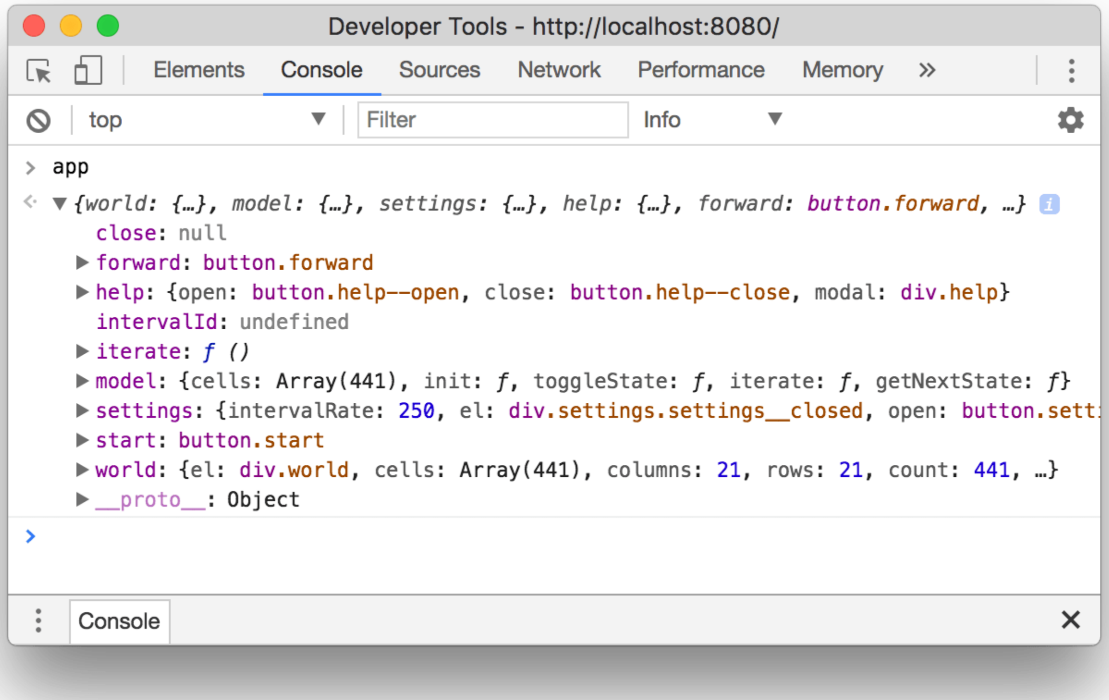
More informative object previews in the Console

Previously, when you logged or evaluated an object in the Console, the Console would only display `Object`, which is not particularly helpful. Now, the Console provides more information about the contents of the object.



```
> app
<- ▼Object {world: Object, model: Object, settings: Object, help:
    Object, forward: button.forward…}
      close: null
    ▶ forward: button.forward
    ▶ help: Object
      intervalId: undefined
    ▶ iterate: function ()
    ▶ model: Object
    ▶ settings: Object
    ▶ start: button.start
    ▶ world: Object
    ▶ __proto__: Object
```

Figure 6. How the Console used to preview objects



```
> app
<- ▼{world: {…}, model: {…}, settings: {…}, help: {…}, forward: button.forward, …}
      close: null
    ▶ forward: button.forward
    ▶ help: {open: button.help--open, close: button.help--close, modal: div.help}
      intervalId: undefined
    ▶ iterate: ƒ ()
    ▶ model: {cells: Array(441), init: ƒ, toggleState: ƒ, iterate: ƒ, getNextState: ƒ}
    ▶ settings: {intervalRate: 250, el: div.settings.settings__closed, open: button.sett
    ▶ start: button.start
    ▶ world: {el: div.world, cells: Array(441), columns: 21, rows: 21, count: 441, …}
    ▶ __proto__: Object
>
```

Figure 7. How the Console now previews objects

More informative context selection menu in the Console

The Console's Context Selection menu now provides more information about available contexts.

- The title describes what each item is.
- The subtitle below the title describes the domain where the item came from.
- Hover over an iframe context to highlight it in the viewport.

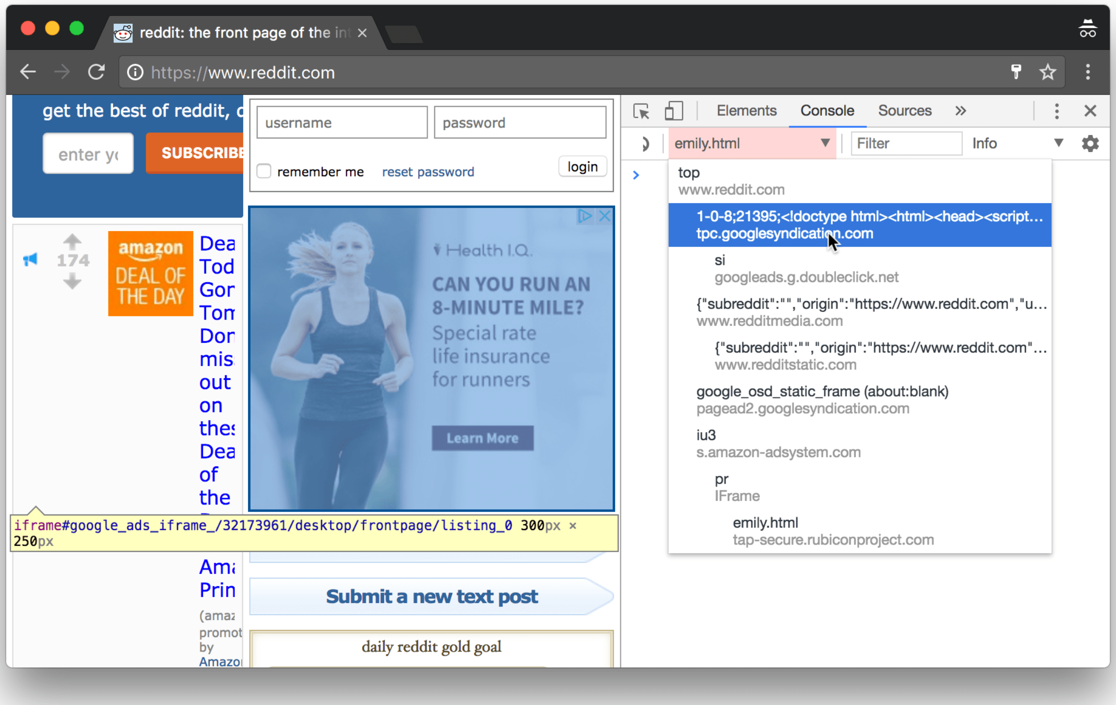
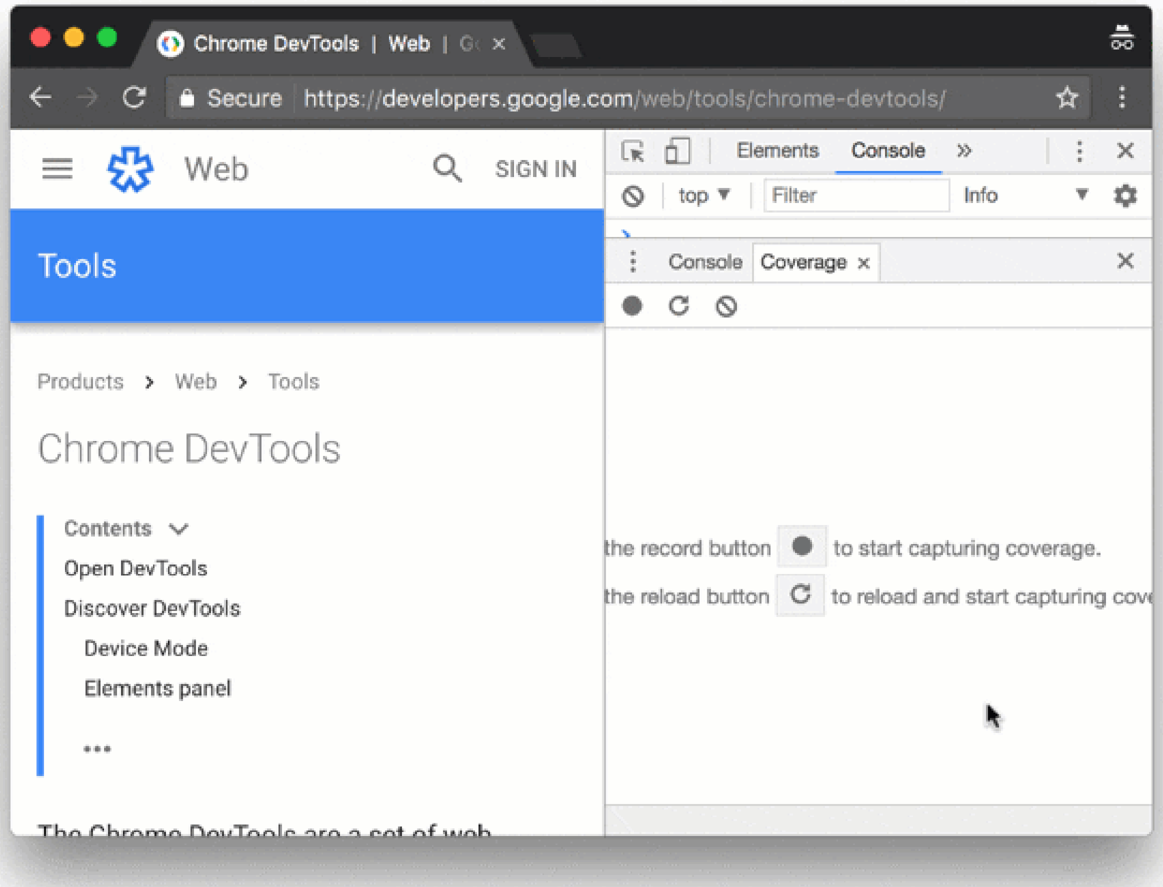

reddit: the front page of the in
https://www.reddit.com
get the best of reddit, d
enter yo
SUBSCRIBE
username
password
remember me
reset password
login
174
amazon
DEAL OF THE DAY
Health I.Q.
CAN YOU RUN AN 8-MINUTE MILE?
Special rate life insurance for runners
Learn More
iframe#google_ads_iframe_/32173961/desktop/frontpage/listing_0 300px × 250px
Submit a new text post
daily reddit gold goal
Elements
Console
Sources
emily.html
Filter
Info
top
www.reddit.com
1-0-8;21395;<!doctype html><html><head><script…
tpc.googlesyndication.com
si
googleads.g.doubleclick.net
{"subreddit":"","origin":"https://www.reddit.com","u…
www.redditmedia.com
{"subreddit":"","origin":"https://www.reddit.com"…
www.redditstatic.com
google_osd_static_frame (about:blank)
pagead2.googlesyndication.com
iu3
s.amazon-adsystem.com
pr
IFrame
emily.html
tap-secure.rubiconproject.com

Figure 8. Hovering over an iframe in the new Context Selection menu highlights it in the viewport

Real-time updates in the Coverage tab

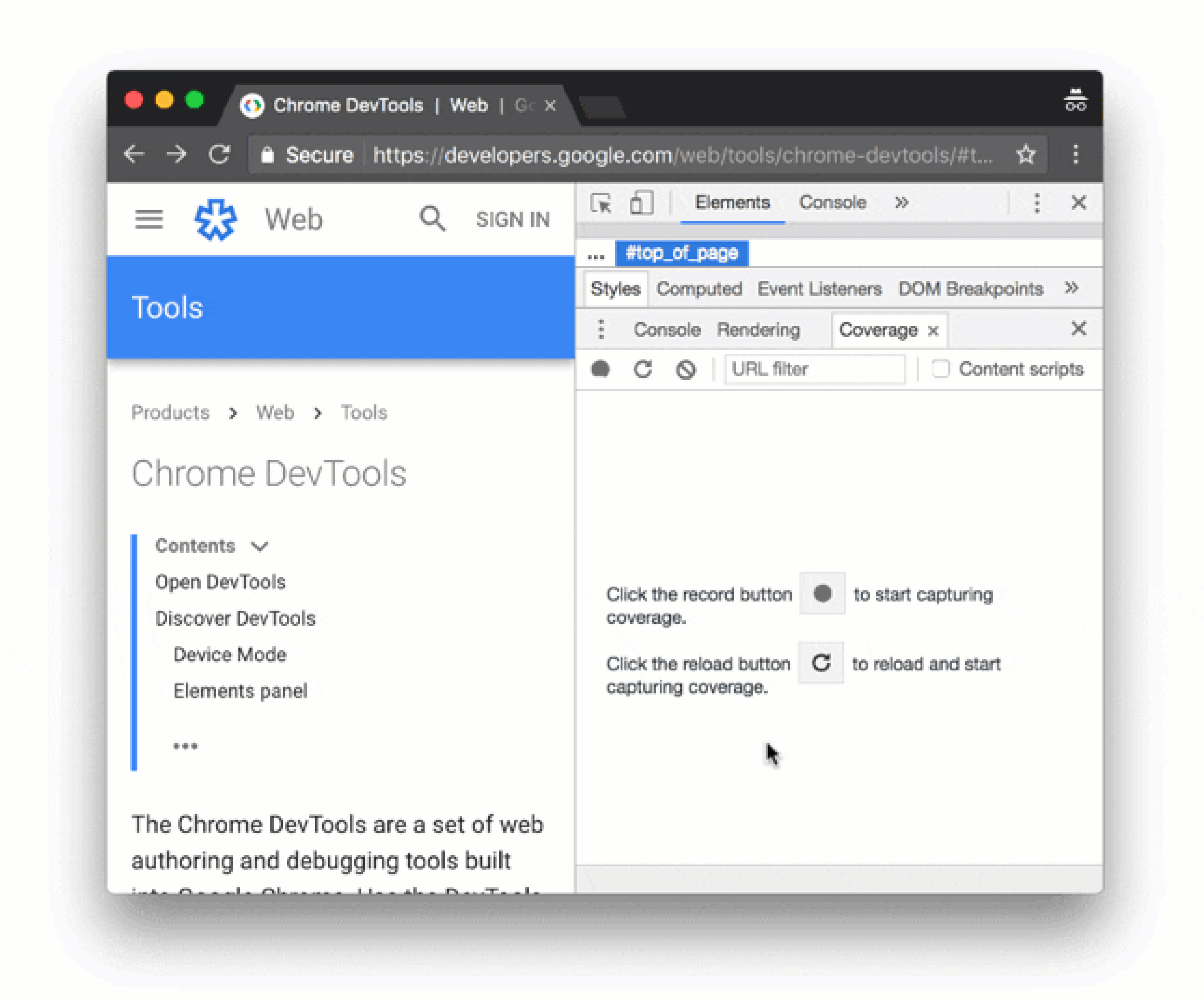
When recording code coverage in Chrome 59, the **Coverage** tab would just display "Recording...", with no visibility into what code was being used. Now, the **Coverage** tab shows you in real-time what code is being used.



the record button ● to start capturing coverage.

the reload button C to reload and start capturing cove

Figure 9. Loading and interacting with a page using the old **Coverage** tab



The Chrome DevTools are a set of web authoring and debugging tools built

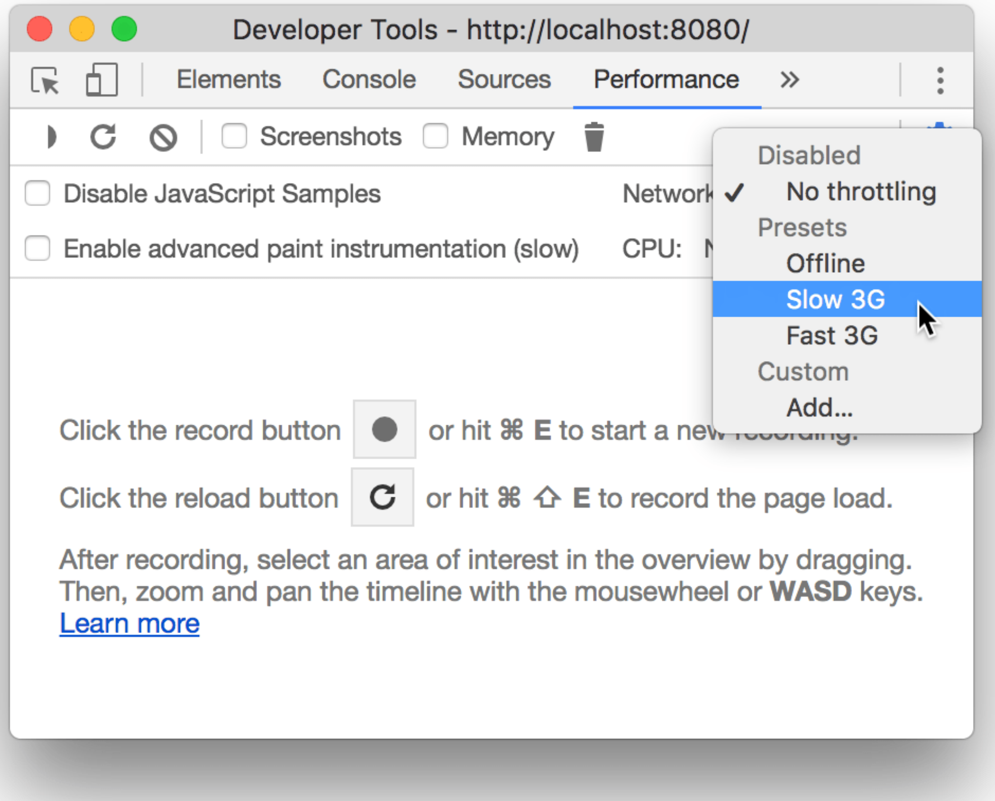
Click the record button ● to start capturing coverage.

Click the reload button to reload and start capturing coverage.

Figure 10. Loading and interacting with a page using the new **Coverage** tab

Simpler network throttling options

The network throttling menus in the **Network** and **Performance** panels have been simplified to include only three options: **Offline**, **Slow 3G**, which is common in places like India, and **Fast 3G**, which is common in places like the United States.



Developer Tools - http://localhost:8080/

Elements Console Sources Performance »

□ Screenshots □ Memory

□ Disable JavaScript Samples

□ Enable advanced paint instrumentation (slow)

Network

CPU:

Disabled

✓ No throttling

Presets

Offline

Slow 3G

Fast 3G

Custom

Add...

Click the record button ● or hit ⌘ E to start a new recording.

Click the reload button or hit ⌘ ⇧ E to record the page load.

After recording, select an area of interest in the overview by dragging. Then, zoom and pan the timeline with the mousewheel or **WASD** keys. Learn more

Figure 11. The new network throttling options

The throttling options have been tweaked to match other, kernel-level throttling tools. DevTools no longer shows the latency, download, and upload metrics next to each option, because those values were misleading. The goal is to match the true experience of each option.

Async stacks on by default

The **Async** checkbox has been removed from the **Sources** panel. Async stack traces are now on by default. In the past, this option was opt-in, because of performance overhead. The overhead is now minimal enough to enable the feature by default. If you prefer to have async stack traces disabled, you can turn them off in Settings or by running the `Do not capture async stack traces` command in the Command Menu.

DevTools at Google I/O '17

Check out the talk by the mythical Paul Irish below to learn more about what the DevTools team has been working on over the past year and the big themes that they're tackling in the near future.

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. You can also tweet us at @ChromeDevTools if you're short on time.

That's all for what's new in DevTools in Chrome 60. See you in 6 weeks for Chrome 61!

Links to previous release notes

- What's New In DevTools (Chrome 59)
- What's New In DevTools (Chrome 58)

Last updated July 2, 2018.