What's New In DevTools (Chrome 66)

By Kayce Basques

Technical Writer for Chrome DevTools

New features and major changes coming to DevTools in Chrome 66 include:

- Blackboxing in the **Network** panel
- Auto-adjust zooming in **Device Mode**
- Pretty-printing in the **Preview** and **Response** tabs
- Previewing HTML content in the **Preview** tab
- **Local Overrides** with styles inside of HTML

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.

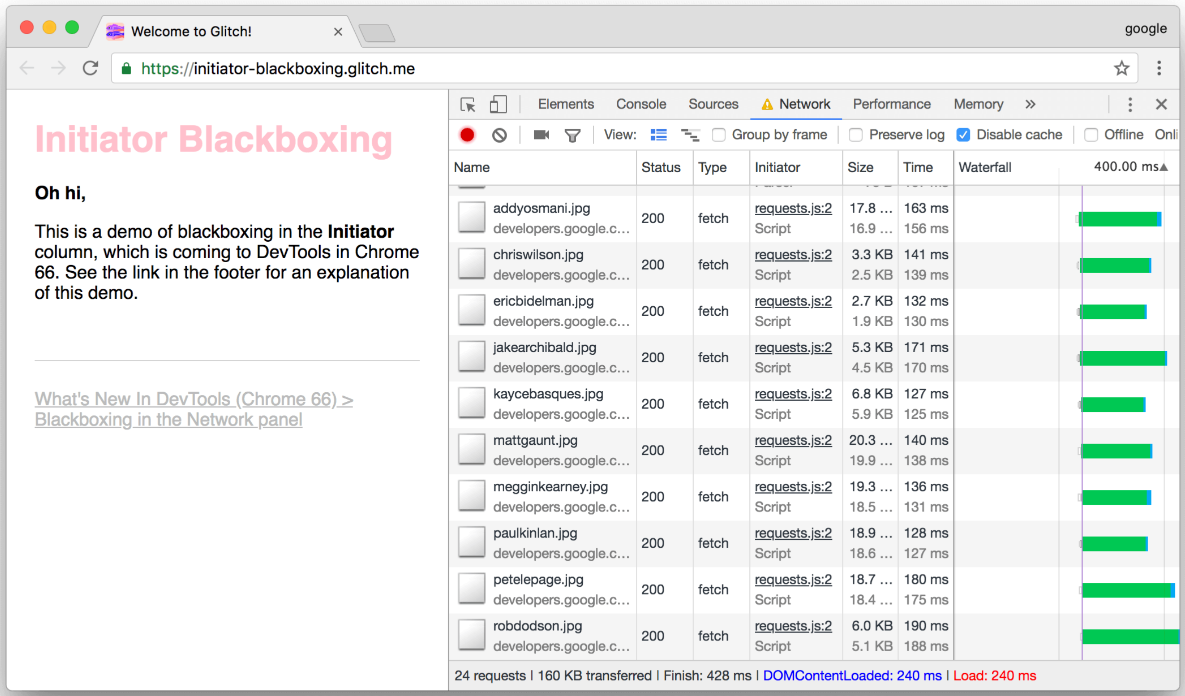
Read on, or watch the video version of the release notes below.

Blackboxing in the Network panel

The **Initiator** column in the **Network** panel tells you why a resource was requested. For example, if JavaScript causes an image to be fetched, the **Initiator** column shows you the line of JavaScript code that caused the request.

Note: You can hide or show columns in the **Network** panel by right-clicking the table header.

Previously, if your framework wrapped network requests in a wrapper, the **Initiator** column wouldn't be that helpful. All network requests pointed to the same line of wrapper code.



Initiator Blackboxing

Oh hi,

This is a demo of blackboxing in the **Initiator** column, which is coming to DevTools in Chrome 66. See the link in the footer for an explanation of this demo.

What's New In DevTools (Chrome 66) > Blackboxing in the Network panel

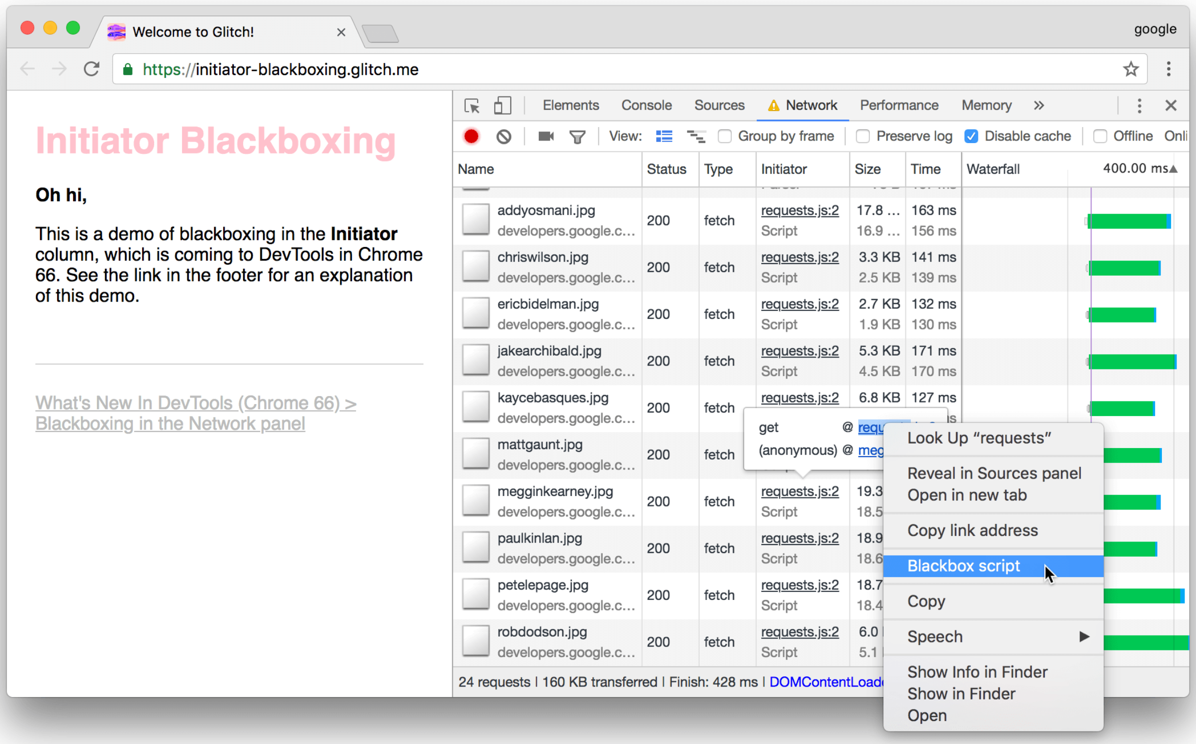
Name	Status	Type	Initiator	Size	Time	Waterfall
addyosmani.jpg developers.google.c...	200	fetch	requests.js:2 Script	17.8 ... 16.9 ...	163 ms 156 ms	
chriswilson.jpg developers.google.c...	200	fetch	requests.js:2 Script	3.3 KB 2.5 KB	141 ms 139 ms	
ericbidelman.jpg developers.google.c...	200	fetch	requests.js:2 Script	2.7 KB 1.9 KB	132 ms 130 ms	
jakearchibald.jpg developers.google.c...	200	fetch	requests.js:2 Script	5.3 KB 4.5 KB	171 ms 170 ms	
kaycebasques.jpg developers.google.c...	200	fetch	requests.js:2 Script	6.8 KB 5.9 KB	127 ms 125 ms	
mattgaunt.jpg developers.google.c...	200	fetch	requests.js:2 Script	20.3 ... 19.9 ...	140 ms 138 ms	
megginkearney.jpg developers.google.c...	200	fetch	requests.js:2 Script	19.3 ... 18.5 ...	136 ms 131 ms	
paulkinlan.jpg developers.google.c...	200	fetch	requests.js:2 Script	18.9 ... 18.6 ...	128 ms 127 ms	
petelepage.jpg developers.google.c...	200	fetch	requests.js:2 Script	18.7 ... 18.4 ...	180 ms 175 ms	
robdodson.jpg developers.google.c...	200	fetch	requests.js:2 Script	6.0 KB 5.1 KB	190 ms 188 ms	

24 requests | 160 KB transferred | Finish: 428 ms | DOMContentLoaded: 240 ms | Load: 240 ms

Figure 1. The **Initiator** column shows that all of the requests were initiated by line 2 of `requests.js`

What you really want in this scenario is to see the application code that causes the request. That's now possible:

1. Hover over the **Initiator** column. The call stack that caused the request appears in a pop-up.
2. Right-click the call that you want to hide from the initiator results.
3. Select **Blackbox script**. The **Initiator** column now hides any calls from the script that you blackboxed.



Welcome to Glitch!

google

https://initiator-blackboxing.glitch.me

Initiator Blackboxing

Oh hi,

This is a demo of blackboxing in the **Initiator** column, which is coming to DevTools in Chrome 66. See the link in the footer for an explanation of this demo.

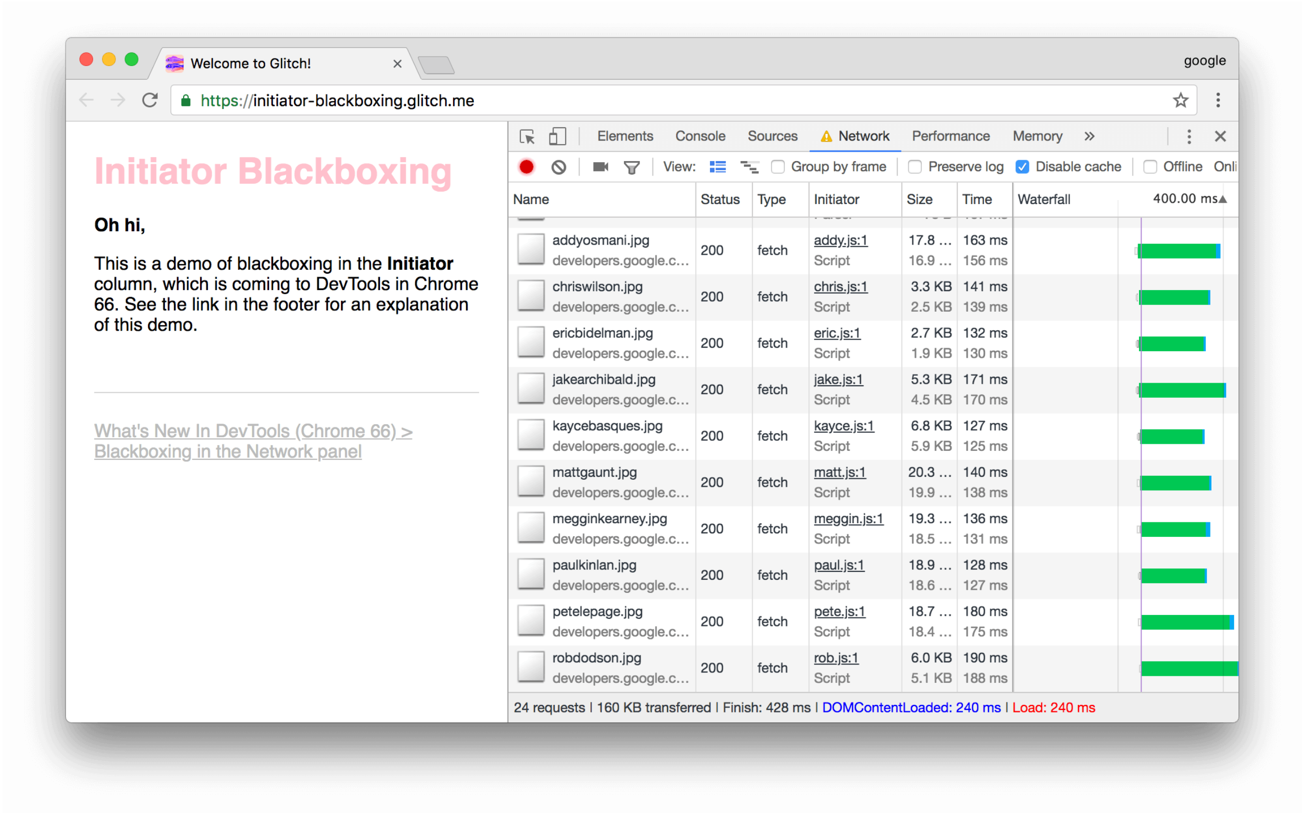
What's New In DevTools (Chrome 66) > Blackboxing in the Network panel

Name	Status	Type	Initiator	Size	Time	Waterfall 400.00 ms
addyosmani.jpg developers.google.c...	200	fetch	requests.js:2 Script	17.8 ... 16.9 ...	163 ms 156 ms	
chriswilson.jpg developers.google.c...	200	fetch	requests.js:2 Script	3.3 KB 2.5 KB	141 ms 139 ms	
ericbidelman.jpg developers.google.c...	200	fetch	requests.js:2 Script	2.7 KB 1.9 KB	132 ms 130 ms	
jakearchibald.jpg developers.google.c...	200	fetch	requests.js:2 Script	5.3 KB 4.5 KB	171 ms 170 ms	
kaycebasques.jpg developers.google.c...	200	fetch	requests.js:2	6.8 KB	127 ms	
mattgaunt.jpg developers.google.c...	200	fetch				
megginkearney.jpg developers.google.c...	200	fetch	requests.js:2 Script	19.3 18.5		
paulkinlan.jpg developers.google.c...	200	fetch	requests.js:2 Script	18.9 18.6		
petelepage.jpg developers.google.c...	200	fetch	requests.js:2 Script	18.7 18.4		
robdodson.jpg developers.google.c...	200	fetch	requests.js:2 Script	6.0 5.1		

get @ requ
(anonymous) @ meg

24 requests | 160 KB transferred | Finish: 428 ms | DOMContentLoad

Figure 2. Blackboxing `requests.js`



Welcome to Glitch!

google

https://initiator-blackboxing.glitch.me

Initiator Blackboxing

Oh hi,

This is a demo of blackboxing in the **Initiator** column, which is coming to DevTools in Chrome 66. See the link in the footer for an explanation of this demo.

What's New In DevTools (Chrome 66) > Blackboxing in the Network panel

Name	Status	Type	Initiator	Size	Time	Waterfall 400.00 ms
addyosmani.jpg developers.google.c...	200	fetch	addy.js:1 Script	17.8 ... 16.9 ...	163 ms 156 ms	
chriswilson.jpg developers.google.c...	200	fetch	chris.js:1 Script	3.3 KB 2.5 KB	141 ms 139 ms	
ericbidelman.jpg developers.google.c...	200	fetch	eric.js:1 Script	2.7 KB 1.9 KB	132 ms 130 ms	
jakearchibald.jpg developers.google.c...	200	fetch	jake.js:1 Script	5.3 KB 4.5 KB	171 ms 170 ms	
kaycebasques.jpg developers.google.c...	200	fetch	kayce.js:1 Script	6.8 KB 5.9 KB	127 ms 125 ms	
mattgaunt.jpg developers.google.c...	200	fetch	matt.js:1 Script	20.3 ... 19.9 ...	140 ms 138 ms	
megginkearney.jpg developers.google.c...	200	fetch	meggin.js:1 Script	19.3 ... 18.5 ...	136 ms 131 ms	
paulkinlan.jpg developers.google.c...	200	fetch	paul.js:1 Script	18.9 ... 18.6 ...	128 ms 127 ms	
petelepage.jpg developers.google.c...	200	fetch	pete.js:1 Script	18.7 ... 18.4 ...	180 ms 175 ms	
robdodson.jpg developers.google.c...	200	fetch	rob.js:1 Script	6.0 KB 5.1 KB	190 ms 188 ms	

24 requests | 160 KB transferred | Finish: 428 ms | DOMContentLoaded: 240 ms | Load: 240 ms

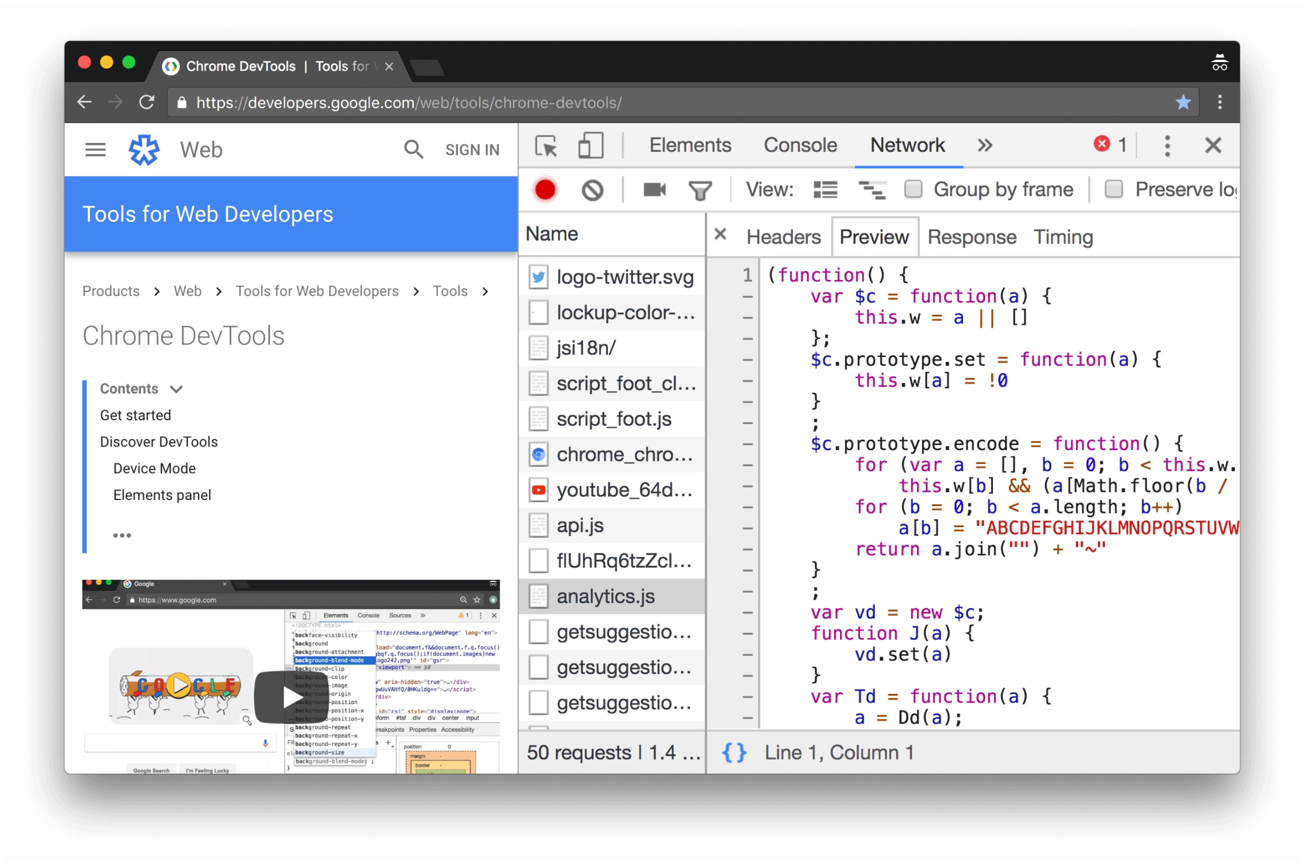
Figure 3. After blackboxing `requests.js`, the **Initiator** column now shows more helpful results

Manage your blackboxed scripts from the **Blackboxing** tab in Settings.

See Ignore a script or pattern of scripts to learn more about blackboxing.

Pretty-printing in the Preview and Response tabs

The **Preview** tab in the **Network** panel now pretty-prints resources by default when it detects that those resources have been minified.



```
(function() {
    var $c = function(a) {
        this.w = a || []
    };
    $c.prototype.set = function(a) {
        this.w[a] = !0
    }
    ;
    $c.prototype.encode = function() {
        for (var a = [], b = 0; b < this.w.
            this.w[b] && (a[Math.floor(b /
        for (b = 0; b < a.length; b++)
            a[b] = "ABCDEFGHIJKLMNOPQRSTUVW
        return a.join("") + "~"
    }
    ;
    var vd = new $c;
    function J(a) {
        vd.set(a)
    }
    var Td = function(a) {
        a = Dd(a);
```

Figure 4. The **Preview** tab pretty-printing the contents of `analytics.js` by default

To view the unminified version of a resource, use the **Response** tab. You can also manually pretty-print resources from the **Response** tab, via the new **Format** button.

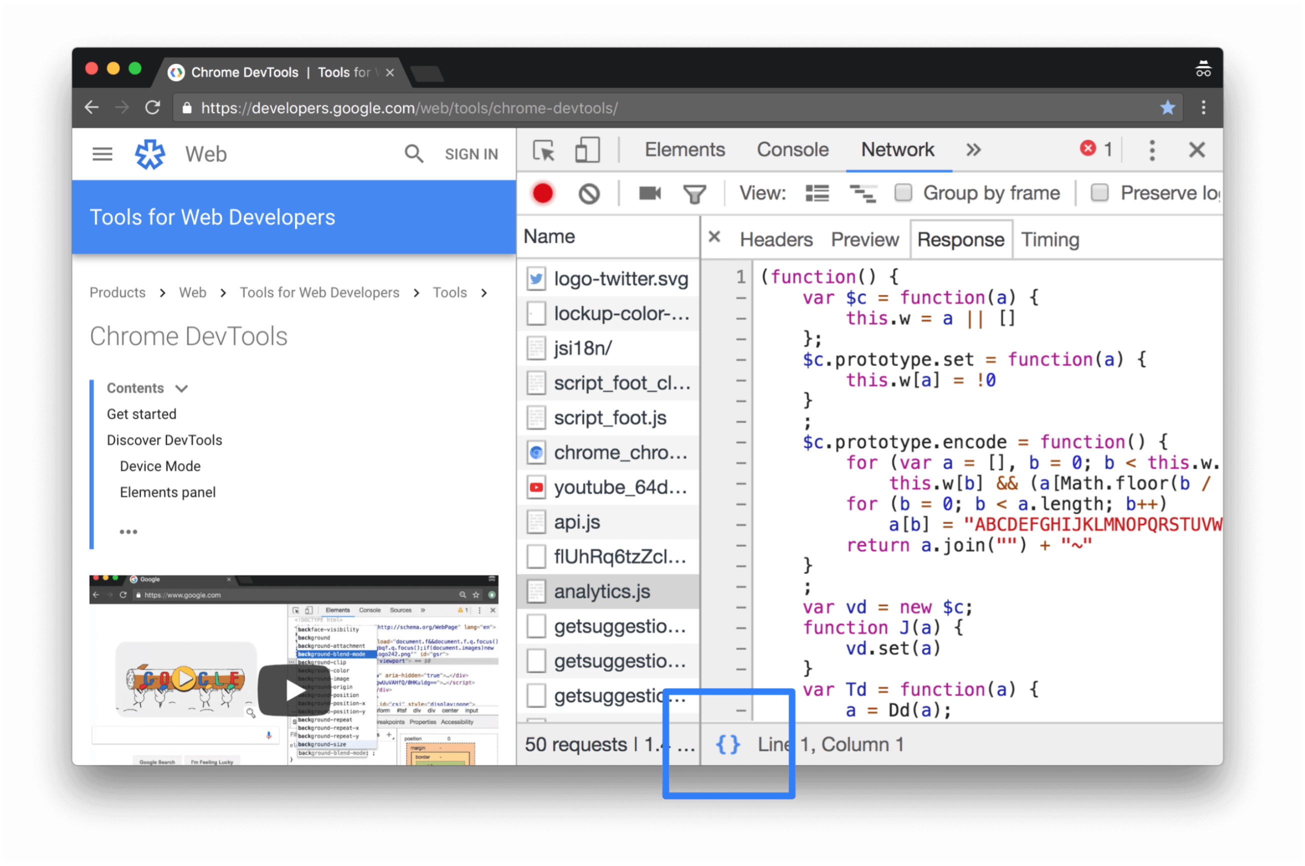
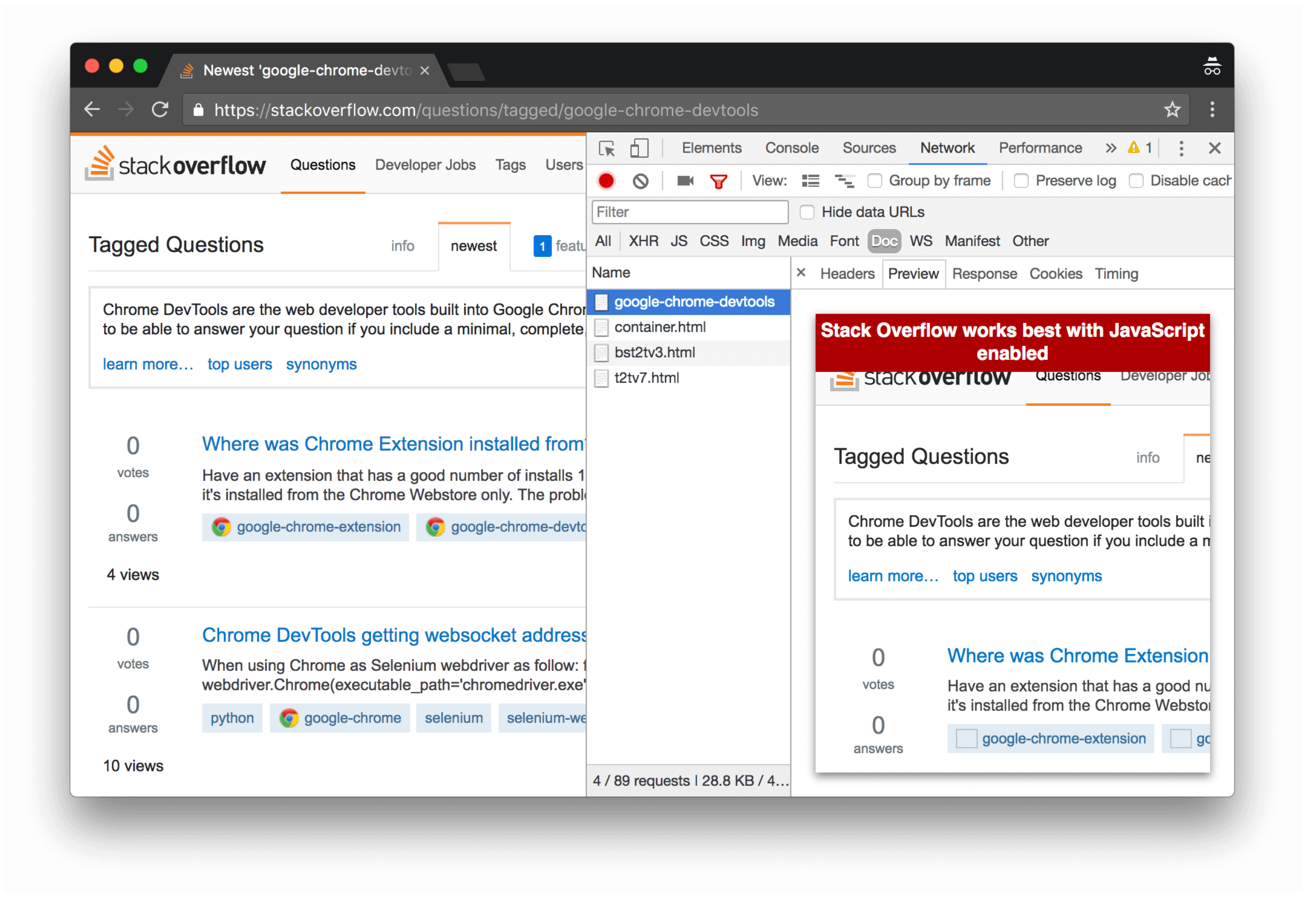

```
(function() {
    var $c = function(a) {
        this.w = a || []
    };
    $c.prototype.set = function(a) {
        this.w[a] = !0
    }
    ;
    $c.prototype.encode = function() {
        for (var a = [], b = 0; b < this.w.
            this.w[b] && (a[Math.floor(b /
        for (b = 0; b < a.length; b++)
            a[b] = "ABCDEFGHIJKLMNOPQRSTUVW
        return a.join("") + "~"
    }
    ;
    var vd = new $c;
    function J(a) {
        vd.set(a)
    }
    var Td = function(a) {
        a = Dd(a);
```

Figure 5. Manually pretty-printing the contents of `analytics.js` via the **Format** button

Previewing HTML content in the Preview tab

Previously, the **Preview** tab in the **Network** panel showed the code of an HTML resource in certain situations, while rendering a preview of the HTML in others. The **Preview** tab now always does a basic rendering of the HTML. It's not intended to be a full browser, so it may not display HTML exactly as you expect. If you want to see the HTML code, click the **Response** tab, or right-click a resource and select **Open in Sources panel**.



Stack Overflow works best with JavaScript enabled

Chrome DevTools are the web developer tools built into Google Chro
to be able to answer your question if you include a minimal, complete

Where was Chrome Extension installed from

Have an extension that has a good number of installs 1
it's installed from the Chrome Webstore only. The probl

Chrome DevTools getting websocket address

When using Chrome as Selenium webdriver as follow: f
webdriver.Chrome(executable_path='chromedriver.exe'

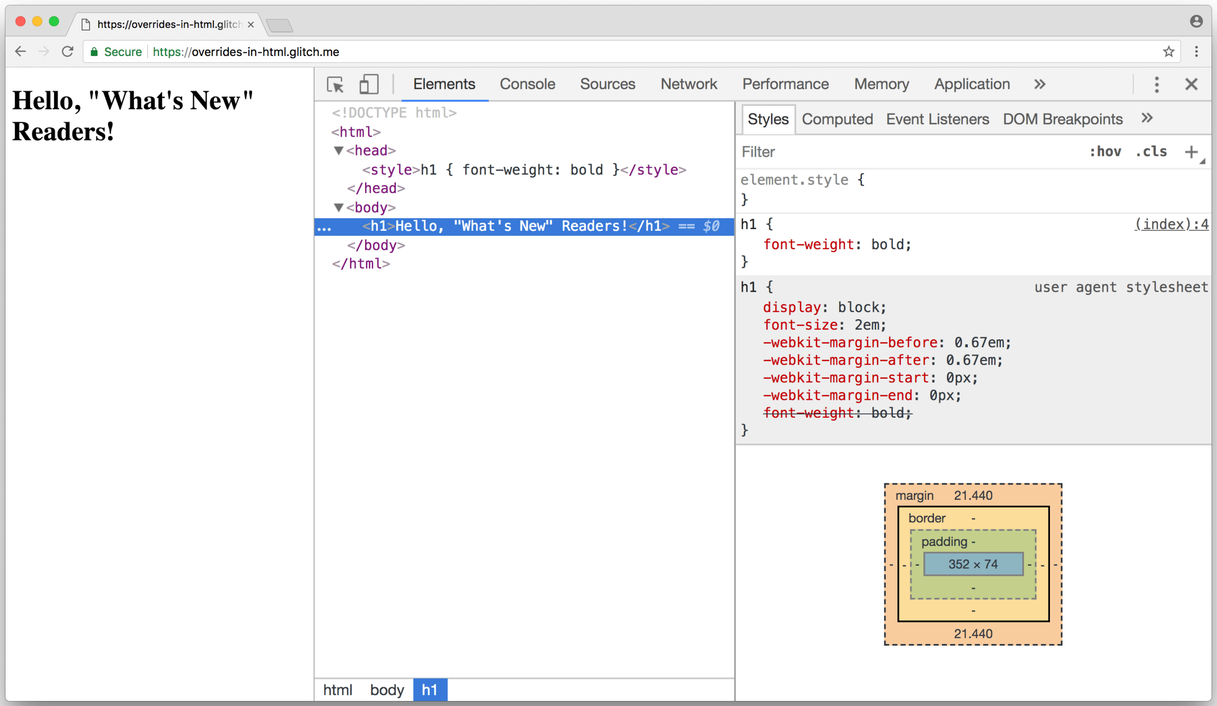
Figure 6. Previewing HTML in the **Preview** tab

Auto-adjust zooming in Device Mode

When in **Device Mode**, open the **Zoom** dropdown and select **Auto-adjust zoom** to automatically resize the viewport whenever you change device orientation.

Local Overrides now works with some styles defined in HTML

Back when DevTools launched **Local Overrides** in Chrome 65, one limitation was that it couldn't track changes to styles defined within HTML. For example, in **Figure 7** there's a style rule in the **head** of the document that declares `font-weight: bold` for **h1** elements.



Hello, "What's New" Readers!

```
<!DOCTYPE html>
<html>
▼<head>
    <style>h1 { font-weight: bold }</style>
  </head>
▼<body>
... <h1>Hello, "What's New" Readers!</h1> == $0
  </body>
</html>
```

```
element.style {
}
h1 {                                          (index):4
  font-weight: bold;
}
h1 {                               user agent stylesheet
  display: block;
  font-size: 2em;
  -webkit-margin-before: 0.67em;
  -webkit-margin-after: 0.67em;
  -webkit-margin-start: 0px;
  -webkit-margin-end: 0px;
  ~~font-weight: bold;~~
}
```

Figure 7. An example of styles defined within HTML

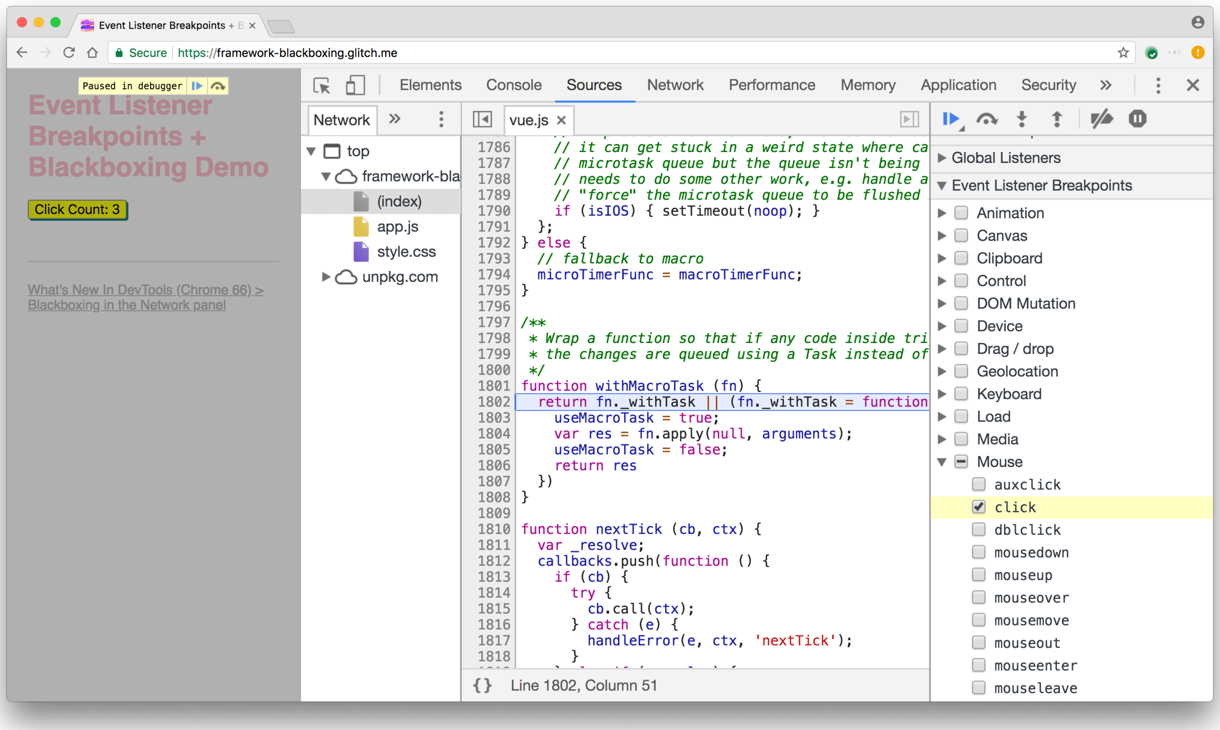
In Chrome 65, if you changed the `font-weight` declaration via the DevTools **Style** pane, **Local Overrides** wouldn't track the change. In other words, on the next reload, the style would revert back to `font-weight: bold`. But in Chrome 66, changes like this now persist across page loads.

Caution: Local Overrides can track changes like this *so long as the style is defined in the HTML document that was sent over the network*. If you have a script that dynamically adds styles to an HTML document, **Local Overrides** still won't be able to detect those changes.

Bonus tip: Blackbox framework scripts to make Event Listener Breakpoints more useful

Note: This section is not related to Chrome 66. It's just a bonus tip about an existing feature that you may find useful.

Back when I created the Get Started With Debugging JavaScript video, some viewers commented that event listener breakpoints aren't useful for apps built on top of frameworks, because the event listeners are often wrapped in framework code. For example, in **Figure 8** I've set up a **`click`** breakpoint in DevTools. When I click the button in the demo, DevTools automatically pauses in the first line of listener code. In this case, it pauses in Vue.js's wrapper code on line 1802, which isn't that helpful.

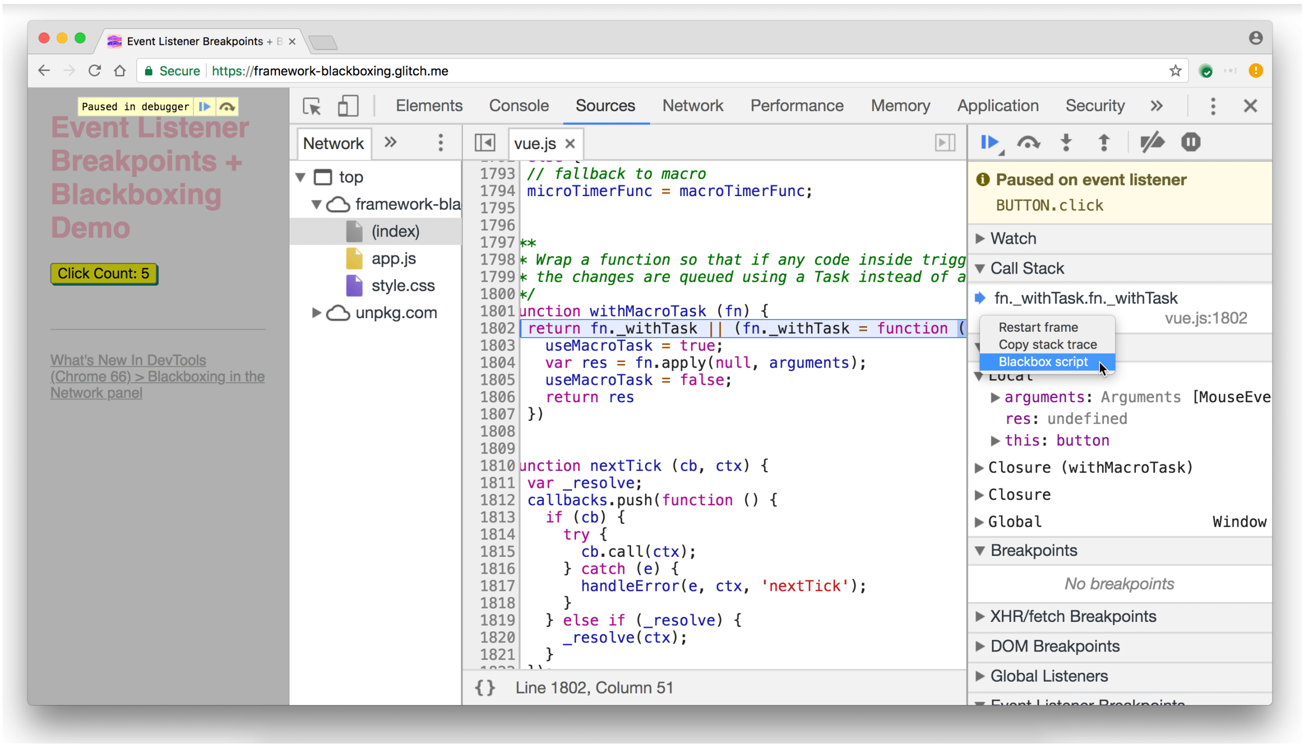


```
1786     // it can get stuck in a weird state where ca
1787     // microtask queue but the queue isn't being
1788     // needs to do some other work, e.g. handle a
1789     // "force" the microtask queue to be flushed
1790     if (isIOS) { setTimeout(noop); }
1791   };
1792 } else {
1793   // fallback to macro
1794   microTimerFunc = macroTimerFunc;
1795 }
1796
1797 /**
1798  * Wrap a function so that if any code inside tri
1799  * the changes are queued using a Task instead of
1800  */
1801 function withMacroTask (fn) {
1802   return fn._withTask || (fn._withTask = function
1803     useMacroTask = true;
1804     var res = fn.apply(null, arguments);
1805     useMacroTask = false;
1806     return res
1807   })
1808 }
1809
1810 function nextTick (cb, ctx) {
1811   var _resolve;
1812   callbacks.push(function () {
1813     if (cb) {
1814       try {
1815         cb.call(ctx);
1816       } catch (e) {
1817         handleError(e, ctx, 'nextTick');
1818       }
```

Line 1802, Column 51

Figure 8. The **`click`** breakpoint pauses in Vue.js' wrapper code

Since the Vue.js script is in a separate file, I can blackbox that script from the **Call Stack** pane in order to make this **`click`** breakpoint more useful.


Event Listener Breakpoints + B
Secure https://framework-blackboxing.glitch.me
Paused in debugger
Event Listener Breakpoints + Blackboxing Demo
Click Count: 5
What's New In DevTools (Chrome 66) > Blackboxing in the Network panel
Elements Console Sources Network Performance Memory Application Security
Network
top
framework-bla
(index)
app.js
style.css
unpkg.com
vue.js
Restart frame
Copy stack trace
Blackbox script
Watch
Call Stack
fn._withTask.fn._withTask vue.js:1802
Local
Closure (withMacroTask)
Closure
Global Window
Breakpoints
No breakpoints
XHR/fetch Breakpoints
DOM Breakpoints
Global Listeners
Line 1802, Column 51

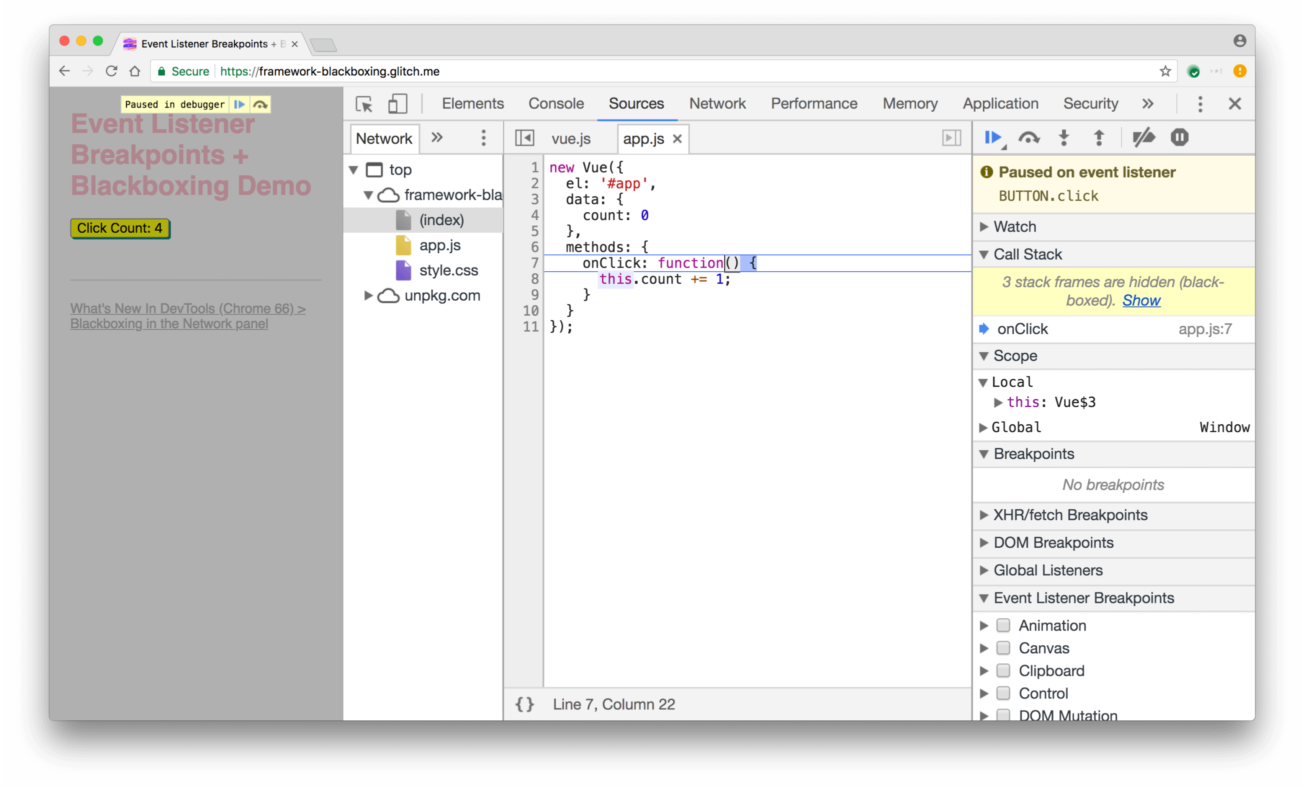
```
// fallback to macro
microTimerFunc = macroTimerFunc;


**
* Wrap a function so that if any code inside trigg
* the changes are queued using a Task instead of a
*/
unction withMacroTask (fn) {
 return fn._withTask || (fn._withTask = function (
   useMacroTask = true;
   var res = fn.apply(null, arguments);
   useMacroTask = false;
   return res
 })


unction nextTick (cb, ctx) {
 var _resolve;
 callbacks.push(function () {
   if (cb) {
     try {
       cb.call(ctx);
     } catch (e) {
       handleError(e, ctx, 'nextTick');
     }
   } else if (_resolve) {
     _resolve(ctx);
   }
```

Paused on event listener
BUTTON.click

```
arguments: Arguments [MouseEve
res: undefined
this: button
```

Figure 9. Blackboxing the Vue.js script from the **Call Stack** pane

The next time I click the button and trigger the `click` breakpoint, it executes the Vue.js code without pausing in it, and then pauses on the first line of code in my app's listener, which is where I really wanted to pause all along.



```
new Vue({
  el: '#app',
  data: {
    count: 0
  },
  methods: {
    onClick: function() {
      this.count += 1;
    }
  }
});
```

Paused on event listener
BUTTON.click

3 stack frames are hidden (black-boxed). Show

```
this: Vue$3
```

Figure 10. The `click` breakpoint now pauses on the app's listener code

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 its 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 while Canary is broken.

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. 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!