

# Pause Your Code With Breakpoints



By Kayce Basques

Technical Writer for Chrome DevTools

Use breakpoints to pause your JavaScript code. This guide explains each type of breakpoint that's available in DevTools, as well as when to use and how to set each type. For a hands-on tutorial of the debugging process, see [Get Started with Debugging JavaScript in Chrome DevTools](#).

## Overview of when to use each breakpoint type

The most well-known type of breakpoint is line-of-code. But line-of-code breakpoints can be inefficient to set, especially if you don't know exactly where to look, or if you are working with a large codebase. You can save yourself time when debugging by knowing how and when to use the other types of breakpoints.

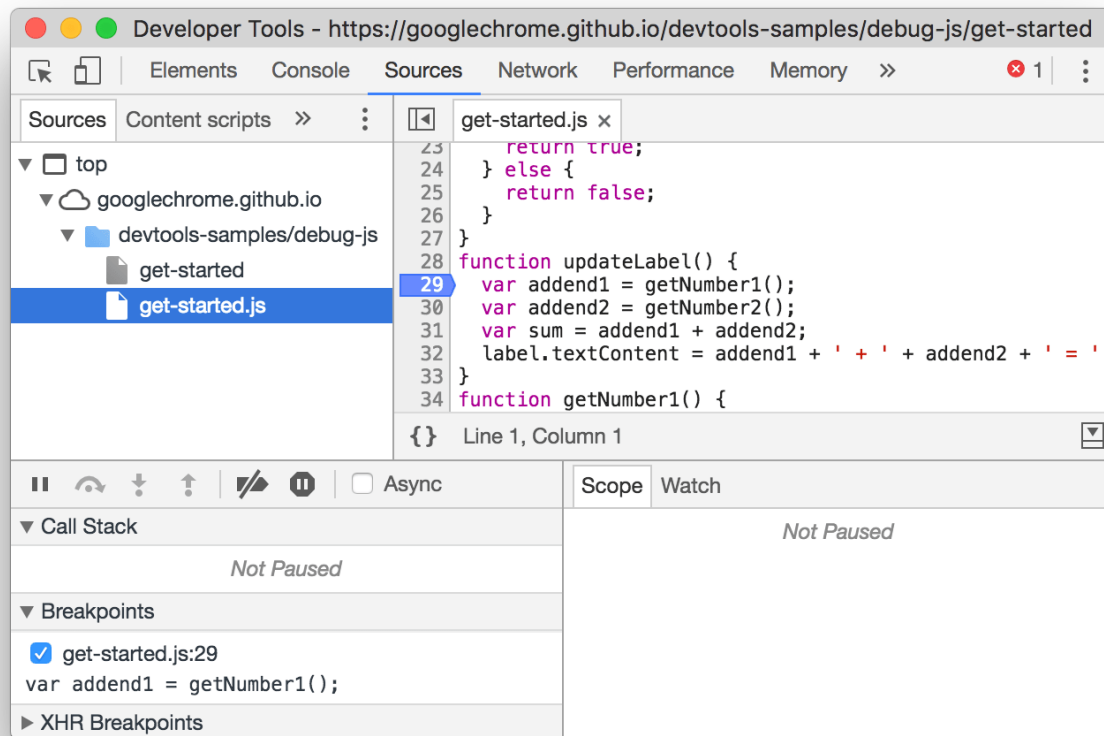
Breakpoint Type	Use This When You Want To Pause...
<u>Line-of-code</u>	On an exact region of code.
<u>Conditional line-of-code</u>	On an exact region of code, but only when some other condition is true.
<u>DOM</u>	On the code that changes or removes a specific DOM node, or its children.
<u>XHR</u>	When an XHR URL contains a string pattern.
<u>Event listener</u>	On the code that runs after an event, such as <code>click</code> , is fired.
<u>Exception</u>	On the line of code that is throwing a caught or uncaught exception.
<u>Function</u>	Whenever a specific function is called.

## Line-of-code breakpoints

Use a line-of-code breakpoint when you know the exact region of code that you need to investigate. DevTools *always* pauses before this line of code is executed.

To set a line-of-code breakpoint in DevTools:

1. Click the **Sources** tab.
2. Open the file containing the line of code you want to break on.
3. Go the line of code.
4. To the left of the line of code is the line number column. Click on it. A blue icon appears on top of the line number column.



**Figure 1:** A line-of-code breakpoint set on line 29

## Line-of-code breakpoints in your code

Call `debugger` from your code to pause on that line. This is equivalent to a line-of-code breakpoint, except that the breakpoint is set in your code, not in the DevTools UI.

```
console.log('a');  
console.log('b');  
debugger;  
console.log('c');
```

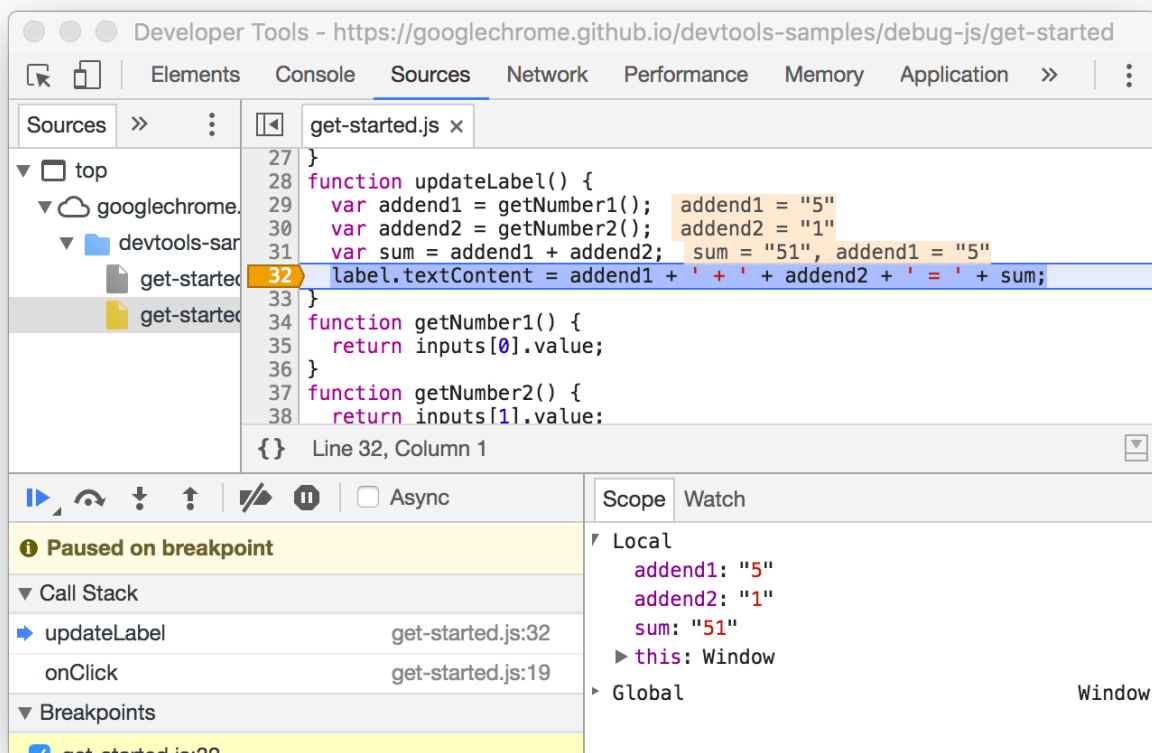


## Conditional line-of-code breakpoints

Use a conditional line-of-code breakpoint when you know the exact region of code that you need to investigate, but you want to pause only when some other condition is true.

To set a conditional line-of-code breakpoint:

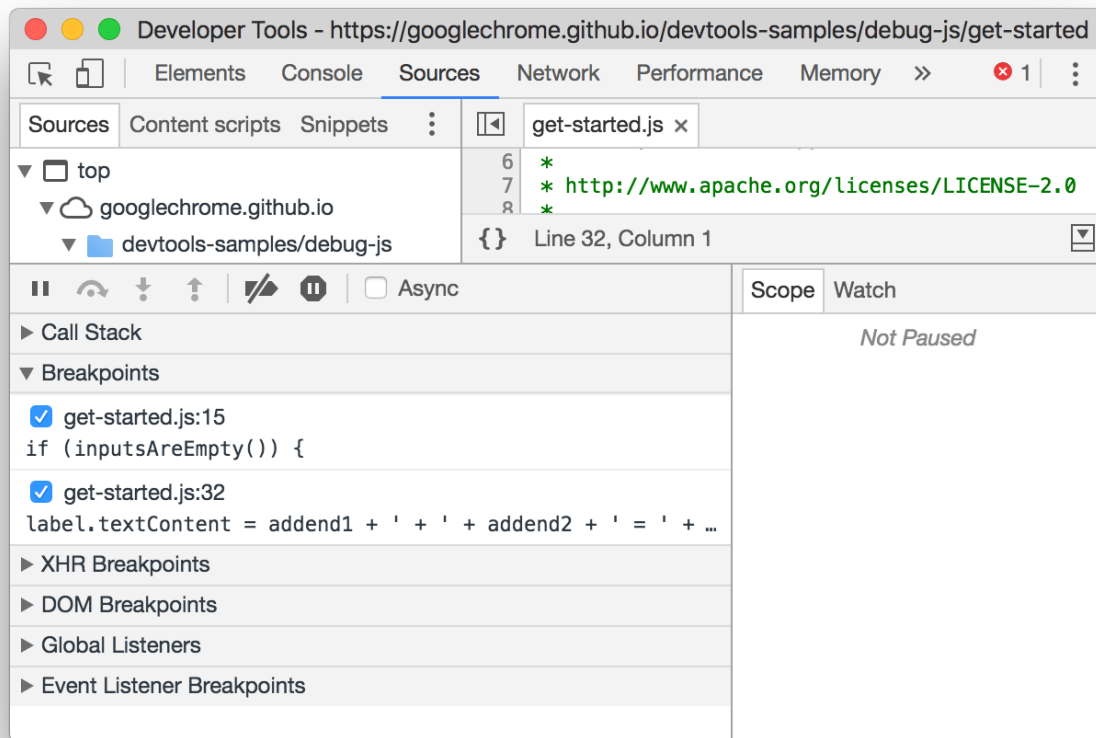
1. Click the **Sources** tab.
2. Open the file containing the line of code you want to break on.
3. Go the line of code.
4. To the left of the line of code is the line number column. Right-click it.
5. Select **Add conditional breakpoint**. A dialog displays underneath the line of code.
6. Enter your condition in the dialog.
7. Press Enter to activate the breakpoint. An orange icon appears on top of the line number column.



**Figure 2:** A conditional line-of-code breakpoint set on line 32

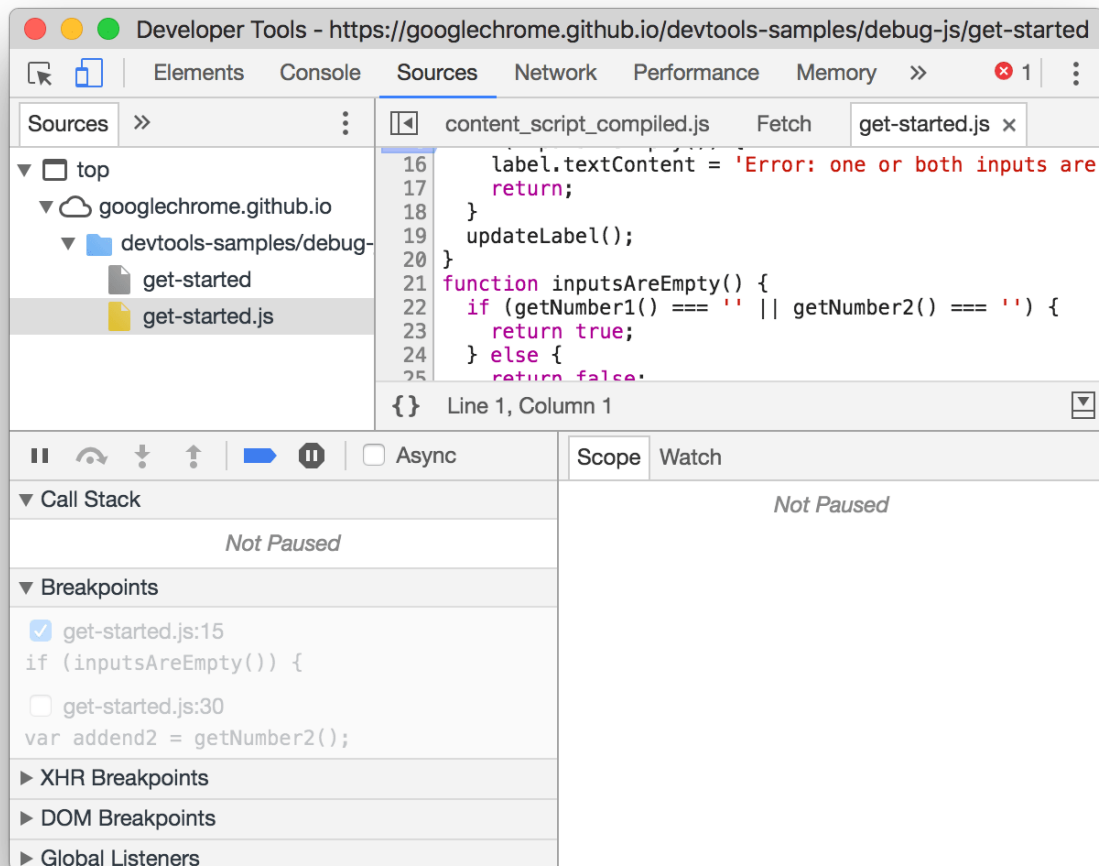
## Manage line-of-code breakpoints

Use the **Breakpoints** pane to disable or remove line-of-code breakpoints from a single location.



**Figure 3:** The **Breakpoints** pane showing two line-of-code breakpoints: one on line 15 of `get-started.js`, another on line 32

- Check the checkbox next to an entry to disable that breakpoint.
- Right-click an entry to remove that breakpoint.
- Right-click anywhere in the **Breakpoints** pane to deactivate all breakpoints, disable all breakpoints, or remove all breakpoints. Disabling all breakpoints is equivalent to unchecking each one. Deactivating all breakpoints instructs DevTools to ignore all line-of-code breakpoints, but to also maintain preserve their enabled state so that they are in the same state as before when you reactivate them.



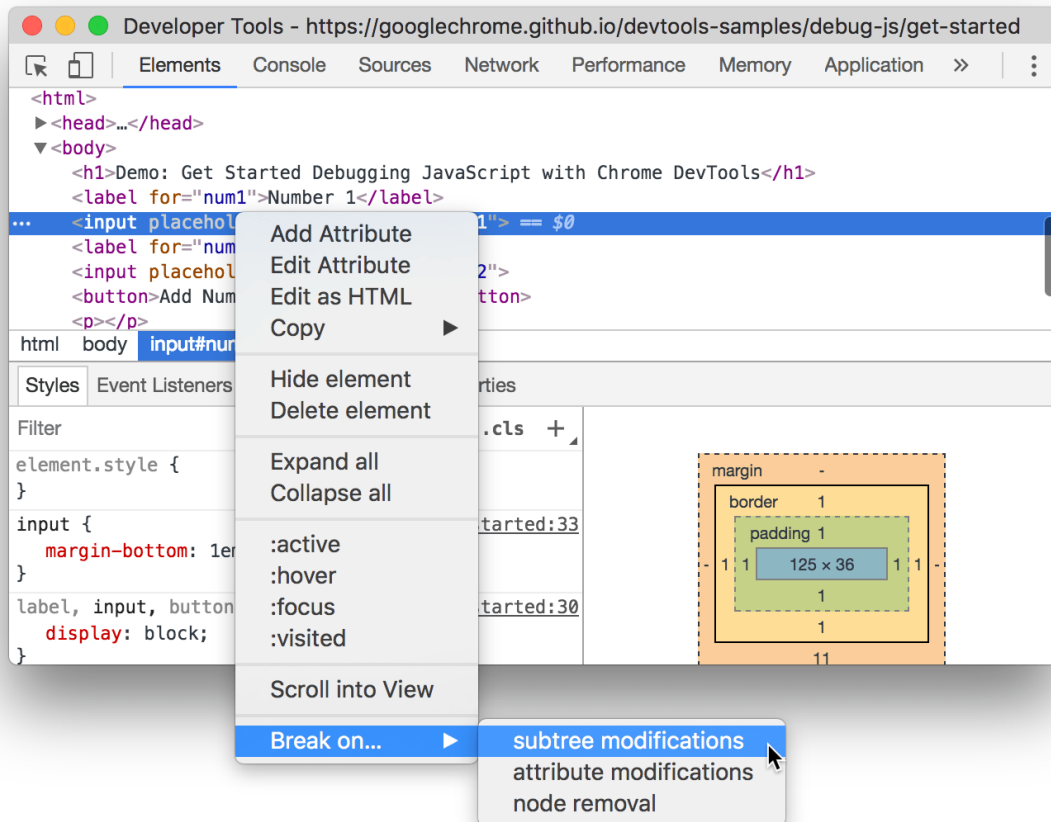
**Figure 4:** Deactivated breakpoints in the **Breakpoints** pane are disabled and transparent

## DOM change breakpoints

Use a DOM change breakpoint when you want to pause on the code that changes a DOM node or its children.

To set a DOM change breakpoint:

1. Click the **Elements** tab.
2. Go to the element that you want to set the breakpoint on.
3. Right-click the element.
4. Hover over **Break on** then select **Subtree modifications**, **Attribute modifications**, or **Node removal**.



**Figure 5:** The context menu for creating a DOM change breakpoint

## Types of DOM change breakpoints

- **Subtree modifications.** Triggered when a child of the currently-selected node is removed or added, or the contents of a child are changed. Not triggered on child node attribute changes, or on any changes to the currently-selected node.
- **Attributes modifications:** Triggered when an attribute is added or removed on the currently-selected node, or when an attribute value changes.
- **Node Removal:** Triggered when the currently-selected node is removed.

## XHR/Fetch breakpoints

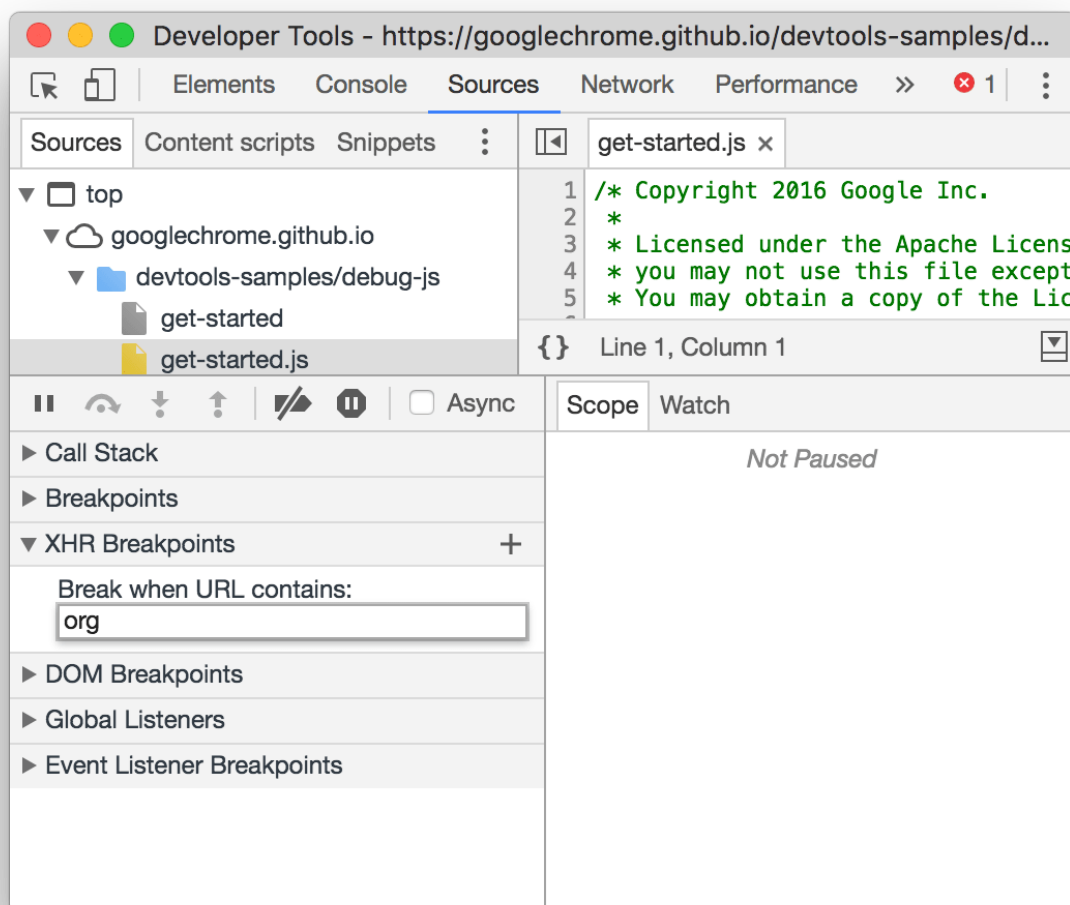
Use an XHR breakpoint when you want to break when the request URL of an XHR contains a specified string. DevTools pauses on the line of code where the XHR calls `send()`.

**Note:** This feature also works with [Fetch](#) requests.

One example of when this is helpful is when you see that your page is requesting an incorrect URL, and you want to quickly find the AJAX or Fetch source code that is causing the incorrect request.

To set an XHR breakpoint:

1. Click the **Sources** tab.
2. Expand the **XHR Breakpoints** pane.
3. Click **Add breakpoint**.
4. Enter the string which you want to break on. DevTools pauses when this string is present anywhere in an XHR's request URL.
5. Press Enter to confirm.

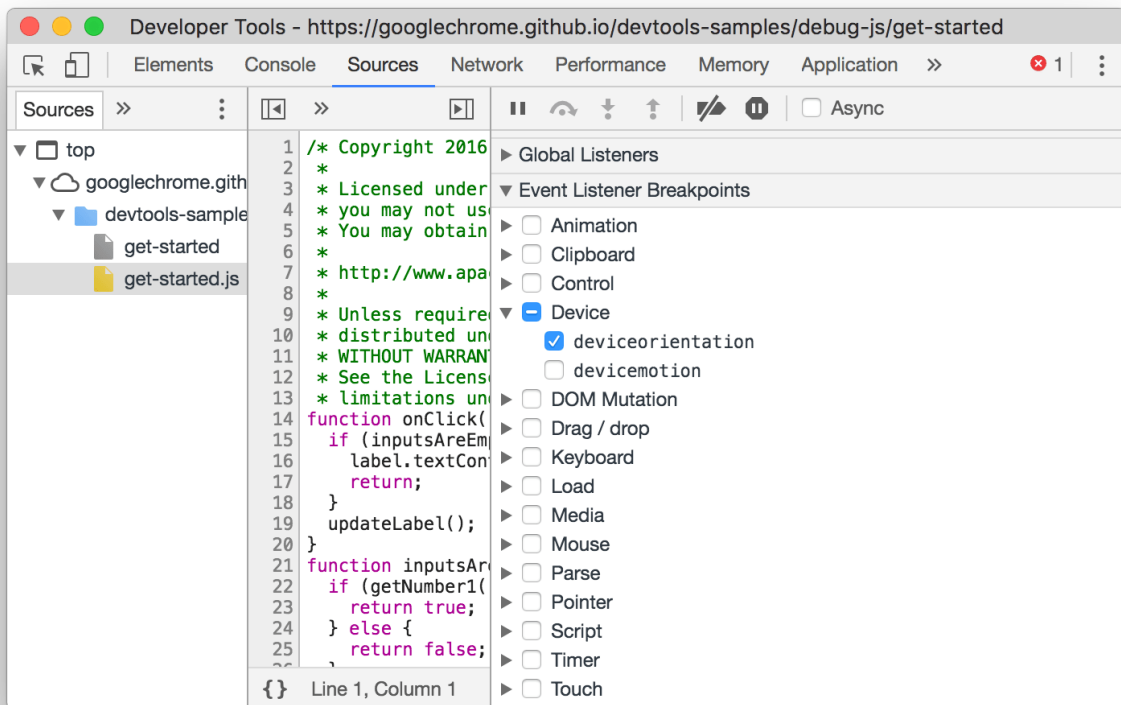


**Figure 6:** Creating an XHR breakpoint in the **XHR Breakpoints** for any request that contains **org** in the URL

## Event listener breakpoints

Use event listener breakpoints when you want to pause on the event listener code that runs after an event is fired. You can select specific events, such as `click`, or categories of events, such as all mouse events.

1. Click the **Sources** tab.
2. Expand the **Event Listener Breakpoints** pane. DevTools shows a list of event categories, such as **Animation**.
3. Check one of these categories to pause whenever any event from that category is fired, or expand the category and check a specific event.




**Figure 7:** Creating an event listener breakpoint for `deviceorientation`

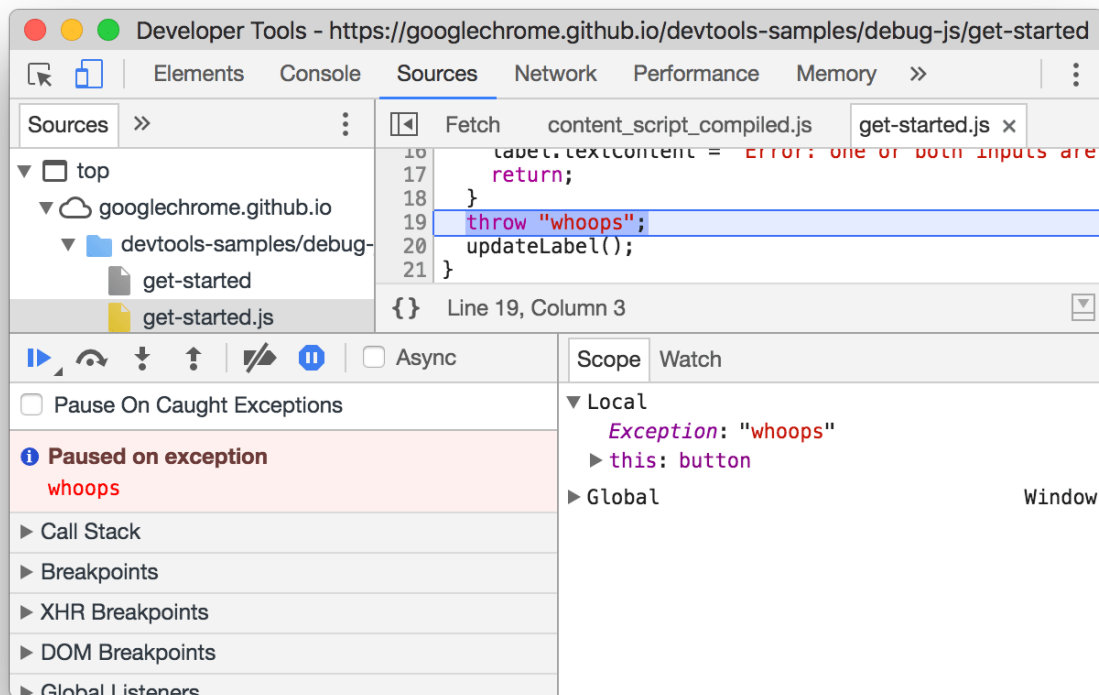
## Exception breakpoints

Use exception breakpoints when you want to pause on the line of code that's throwing a caught or uncaught exception.

1. Click the **Sources** tab.



2. Click **Pause on exceptions** . It turns blue when enabled.
3. (Optional) Check the **Pause On Caught Exceptions** checkbox if you also want to pause on caught exceptions, in addition to uncaught ones.



**Figure 7:** Paused on an uncaught exception

## Function breakpoints

Call `debug(functionName)`, where `functionName` is the function you want to debug, when you want to pause whenever a specific function is called. You can insert `debug()` into your code (like a `console.log()` statement) or call it from the DevTools Console. `debug()` is equivalent to setting a line-of-code breakpoint on the first line of the function.

```
function sum(a, b) {  
  let result = a + b; // DevTools pauses on this line.  
  return result;  
}  
debug(sum); // Pass the function object, not a string.  
sum();
```



## Make sure the target function is in scope

DevTools throws a `ReferenceError` if the function you want to debug is not in scope.

```
(function () {  
  function hey() {  
    console.log('hey');  
  }  
  function yo() {  
    console.log('yo');  
  }  
  debug(yo); // This works.  
  yo();  
})();  
debug(hey); // This doesn't work. hey() is out of scope.
```



Ensuring the target function is in scope can be tricky if you're calling `debug()` from the DevTools Console. Here's one strategy:

1. Set a line-of-code breakpoint somewhere where the function is scope.
2. Trigger the breakpoint.
3. Call `debug()` in the DevTools Console while the code is still paused on your line-of-code breakpoint.

---

*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.*