­­

Assignment 9.4 - Client-side Debugging

This window is the DevTools environment and is made up of many different tools split into various sections. I’ve found that the most important parts are:

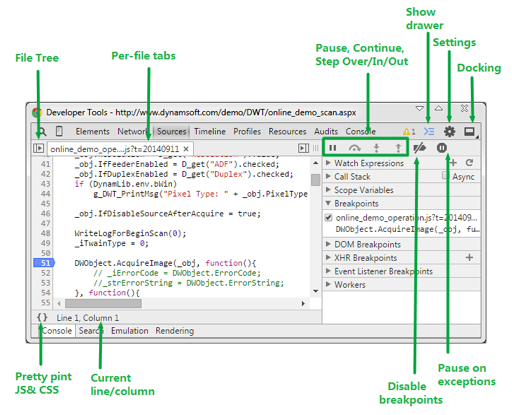
* Elements: What is the page made out of?
* Console: Your portal to interact with the page.
* Network: What’s going on in the background?
* Application: Explore cookies and more.

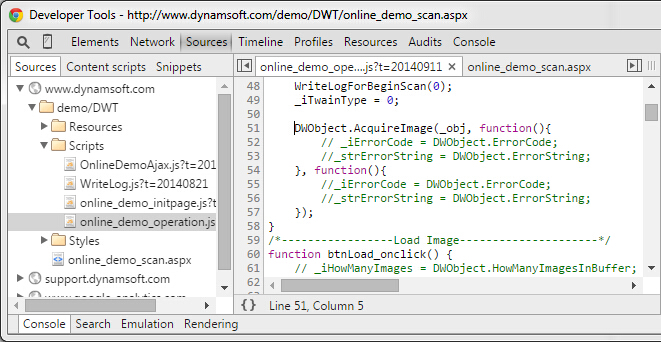
1. Step-by-step instructions on how one might set break points and "walk-through" a JavaScript application (include screen captures)

* Breakpoints help developers to find and fix the bugs.

Step 1: Replicate the Bug

Step 2: Open Chrome DevTools and setup for debugging.

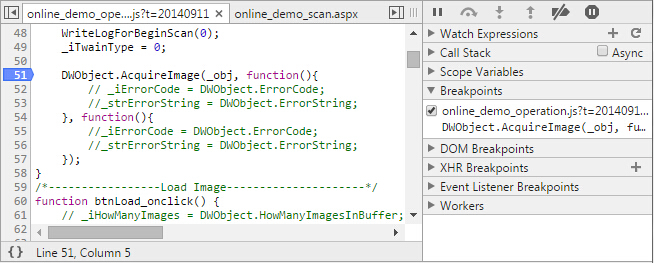




Step 3. Set Breakpoint in JavaScript code

Step 4. In the Sources panel, open a JavaScript file for debugging. In the example below, we are debugging the online\_demo\_operation.js

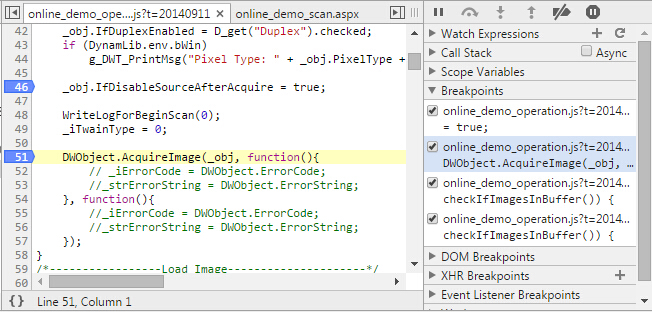
Step 5. Click the line gutter to set a breakpoint for that line of code. A blue tag will indicate if a breakpoint has been set:



Step 6. You can add multiple breakpoints. Click the line gutter of another line to set another breakpoint. All the breakpoints you have set appear under Breakpoints in the right-hand sidebar.

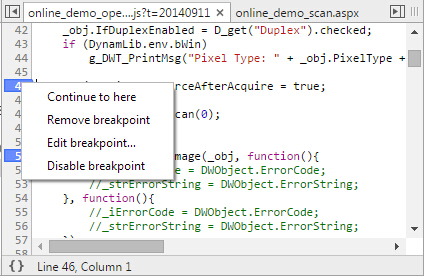
Step 7. Breakpoints can be enabled or disabled using the checkboxes in this sidebar. If a breakpoint is disabled, the blue tag will appear faded out.

Step 8. Click on a breakpoint entry to jump to that particular line in the source file:

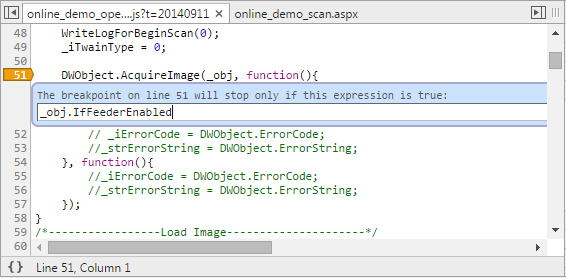


Step 9. Remove a breakpoint by clicking the blue tag breakpoint indicator.

Step 10. Right-click on the blue tag to access a menu with several options including: Continue to Here, Remove Breakpoint, Edit Breakpoint, and Disable Breakpoint.



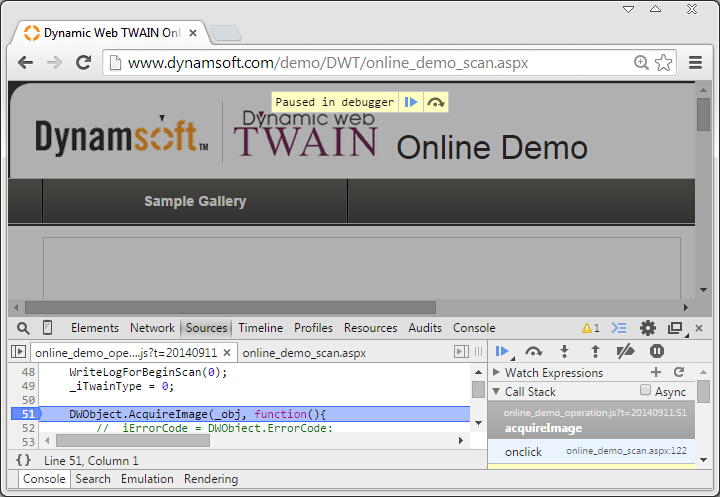
Step 11. To set a conditional breakpoint, choose Edit Breakpoint. Alternatively, right-click on the gutter line and choose Add Conditional Breakpoint. In the input field, type any expression that could resolve to true or false. The breakpoint will pause code execution only if the condition is true.



Step 12. Conditional breakpoints are especially useful when you're looking to analyze code in a loop or an event callback that fires often.

Step 13. Interact with paused breakpoints

Step 14. Once you have one or more breakpoints set, return to the browser window and interact with your page. In the example below, a breakpoint was added within DWObject.AcquireImage(). Now any attempts to scan in an image to our online demo will trigger a breakpoint pause:



Step 15. To resume code execution, click the Continue button or use the F8 keyboard shortcut in the DevTools window.

Step 16. While a script is paused, you can interact with the Watch Expressions, Call Stack, and Scope Variables panels in the right-hand side bar.

Video link

<https://developers.google.com/web/tools/chrome-devtools/javascript/animations/async-call-stack-demo.mp4>

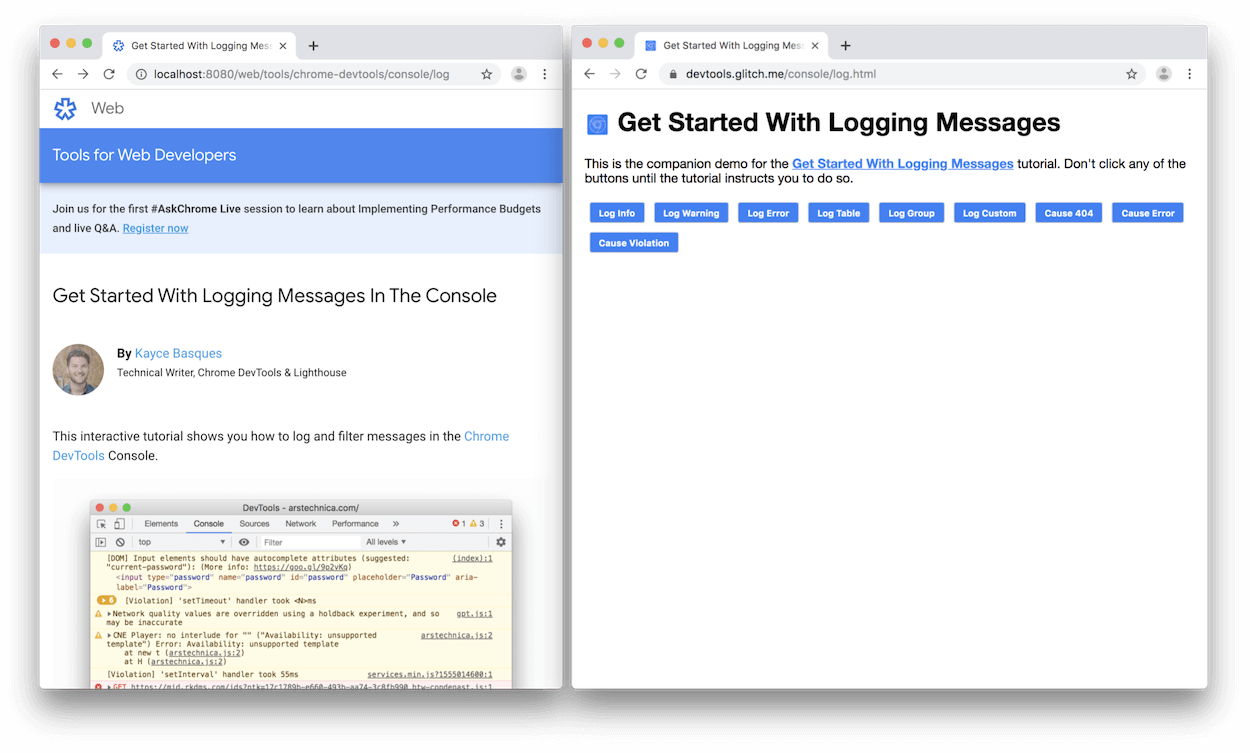
1. Hit the F12 key.
2. Select the Scripts , or Sources , tab in the developer tools.
3. Click the little folder icon in the top level.
4. Select your **JavaScript** file.
5. Add a breakpoint by clicking on the line number on the left (adds a little blue marker)
6. Execute your **JavaScript**.
7. Step-by-step instructions on how one might log and read messages from the JavaScript console window (include screen captures)

Step 1. Open The Console

(Fun fact: Console is a REPL which stands for Read, Evaluate, Print, and Loop

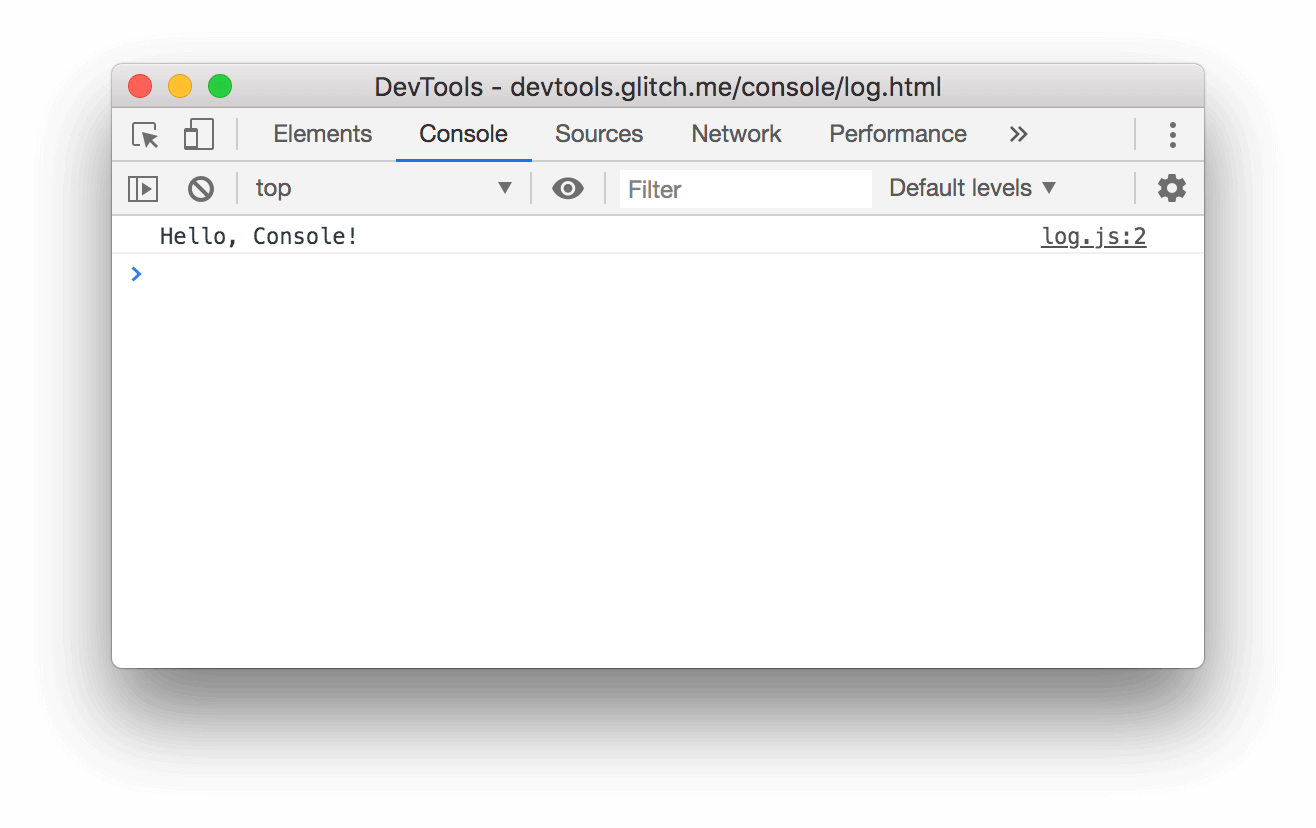
It reads the JavaScript that you type into it, evaluates your code, prints out the result of your expression, and then loops back to the first step.)

Step 2. Press Command + Option + J (Mac) or Control + Shift + J (Windows, Linux, Chrome OS) to open the Console, right here on this very page.



(When building or debugging a page, it's often useful to run statements in the Console in order to change how the page looks or runs)

Step 3. Type document.getElementById('hello').textContent = 'Hello, Console!' in the Console and then press Enter to evaluate the expression. Notice how the text inside the button changes.



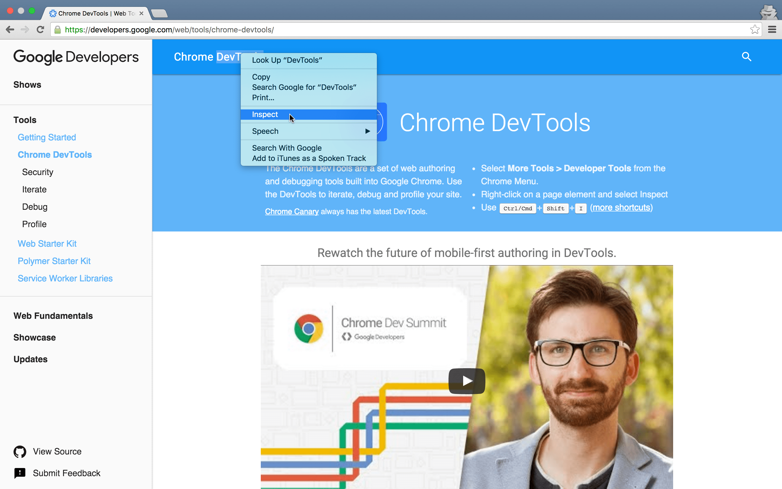
Step 4. Below the code that you evaluated, you see "Hello, Console!"

-Recall the 4 steps of REPL: read, evaluate, print, loop. After evaluating your code, a REPL prints the result of the expression.

So "Hello, Console!" must be the result of evaluating document.getElementById('hello').textContent = 'Hello, Console!'.

1. Step-by-step instructions on how one might select HTML elements in the JavaScript console window (include screen captures)

Step 1. Press Ctrl + Shift + C (Windows) or Cmd + Shift + C (Mac) to open DevTools in Inspect Element mode, then hover over an element.

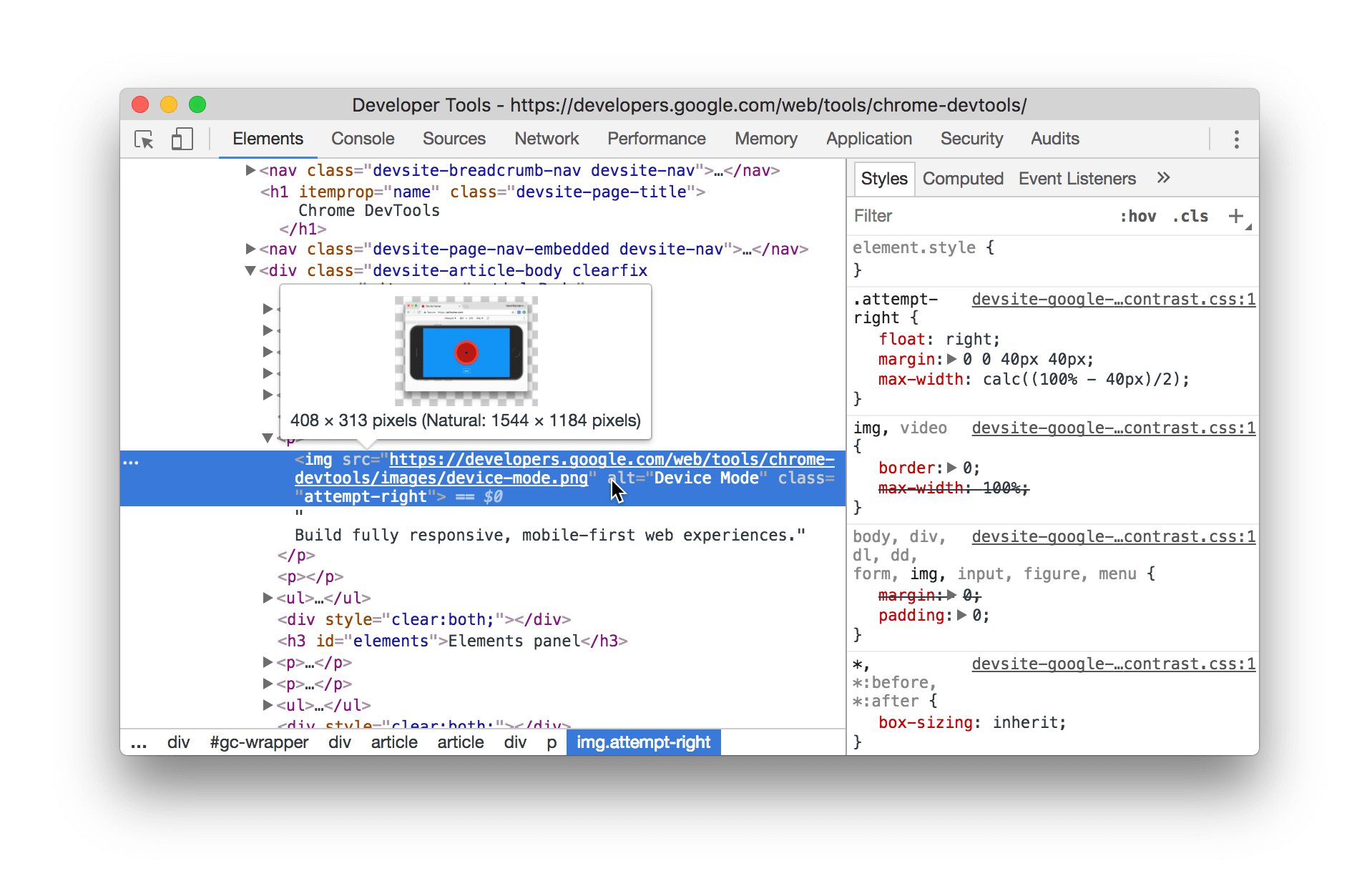


Step 2. Click the Inspect Element button Inspect icon to go into Inspect Element Mode, then click on an element.

Step 3. Use the inspect method in the console, such as inspect(document.body).

Step 4. View the rendered and n Step 1. atural sizes of an image

Step 5. Hover over an img tag in the DOM Tree to view the rendered and natural sizes of that image.



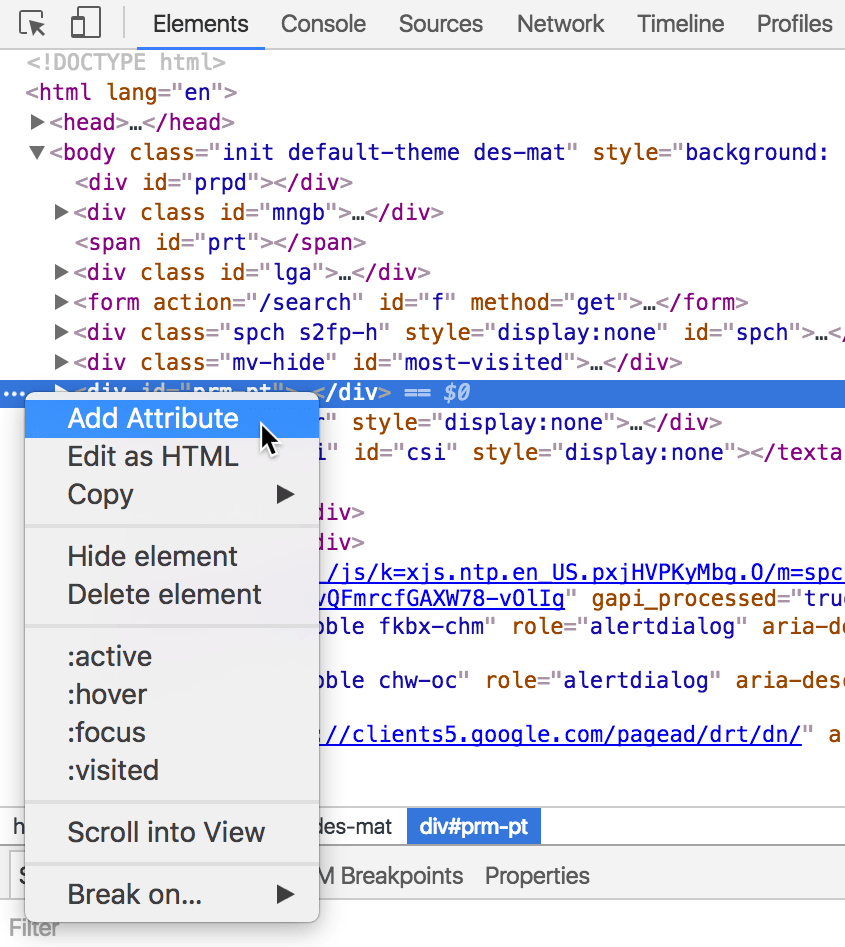
Step 6. Navigate through the DOM structure using your mouse or keyboard.

Step 7. A collapsed node has an arrow next to it pointing right: collapsed node

Step 8. An expanded node has an arrow next to it pointing down: expanded node

Step 9. The more actions menu lets you interact with a DOM node in a variety of ways. To view the menu, right-click on a node, or select a node and then press the more actions button

(more action
button)). The button is only displayed on the currently selected element.

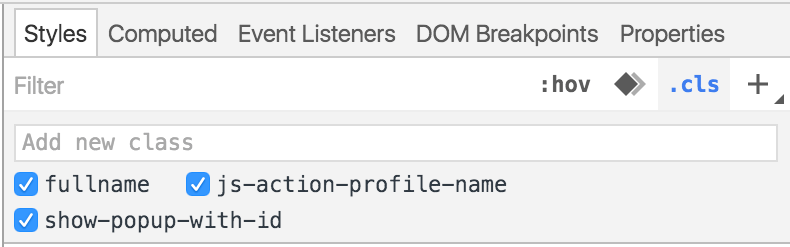


Step 10. Navigating back up the structure moves the highlight:

Step 11. Navigate up breadcrumb trail

Step 12. DevTools displays as many items as possible in the trail. If the entire trail doesn't fit in the status bar­­, an ellipsis (...) shows where the trail has been truncated. Click the ellipsis to show the hidden elements:

1. Step-by-step instructions on how one might simulate CSS states (hover, focus) on live HTML elements (include screen captures)



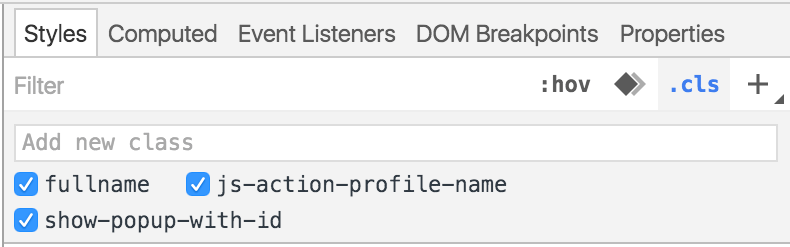
Step 1. Hover your mouse over a CSS selector in the Styles pane to view all elements that are affected by the selector.

Step 2. Add, enable, and disable CSS classes

Step 3. Click on the .cls button to view all of the CSS classes associated to the currently selected element. From there, you can:

Step 4. Enable or disable the classes currently associated to the element.

Add new classes to the element.



Step 5. Edit an existing property name or value

Step 6. Add a style rule

new style rule button

Step 7. Click the New Style Rule (new style rule button) button to add a new CSS rule.

Step 8. Click and hold the button to choose which stylesheet the rule is added to.