Edit the DOM

By Kayce Basques
Technical Writer for Chrome DevTools

By Meggin Kearney
Meggin is a Tech Writer

The DOM tree view in the Chrome DevTools Elements panel displays the DOM structure of the current web page. Live-edit the content and structure of your page through DOM updates.

TL;DR

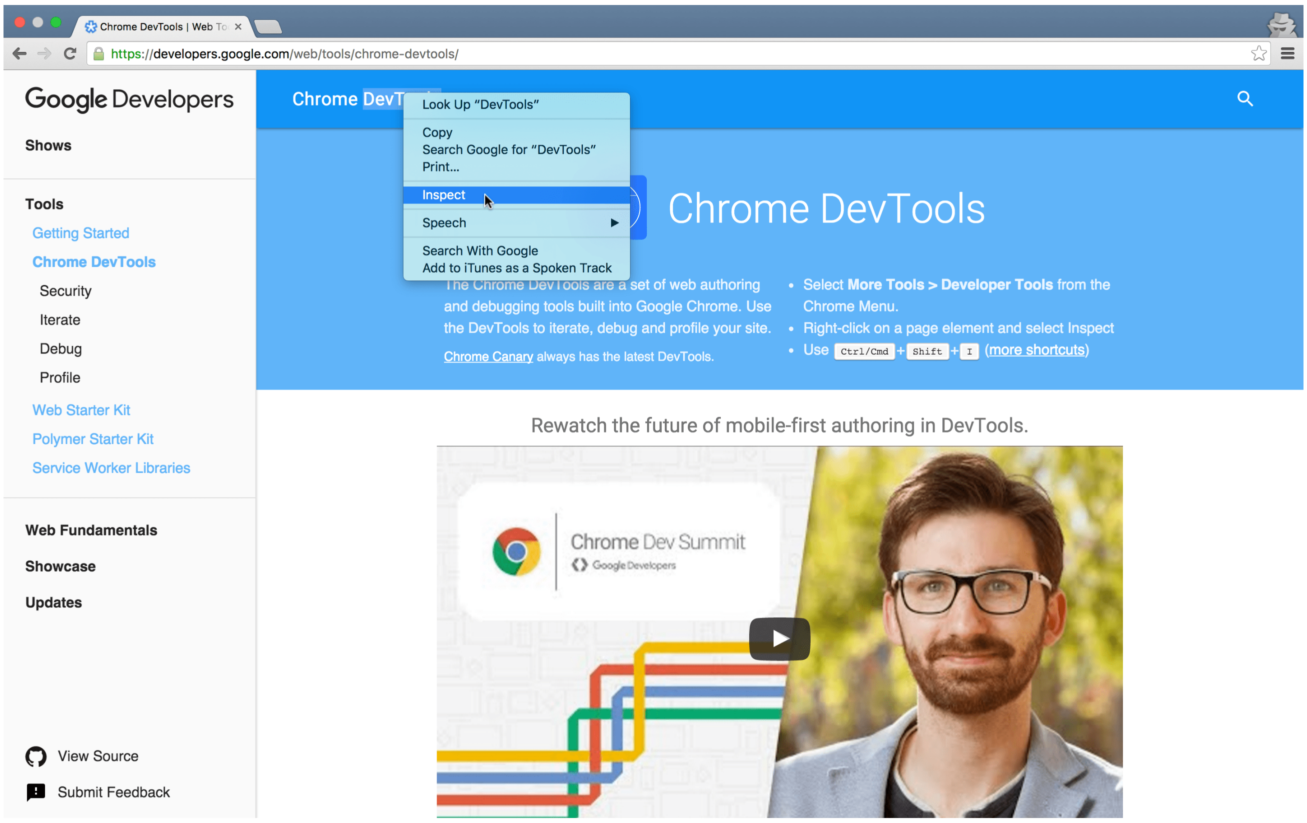
- The DOM defines your page structure. Each DOM node is a page element, for example, a header node, paragraph node.
- Live-edit the content and structure of your pages through the rendered DOM.
- But remember, you can't modify source files through DOM changes in the Elements panel. Reloading the page erases any DOM tree modifications.
- Watch for changes to the DOM using DOM breakpoints.

Inspect an element

Use the **Elements panel** to inspect all elements in your page in one DOM tree. Select any element and inspect the styles applied to it.

There are several ways to inspect an element:

Right-click any element on the page and select **Inspect**.



The Chrome DevTools are a set of web authoring and debugging tools built into Google Chrome. Use the DevTools to iterate, debug and profile your site.

Chrome Canary always has the latest DevTools.

- Select **More Tools > Developer Tools** from the Chrome Menu.
- Right-click on a page element and select Inspect
- Use `Ctrl/Cmd` + `Shift` + `I` (more shortcuts)

Rewatch the future of mobile-first authoring in DevTools.

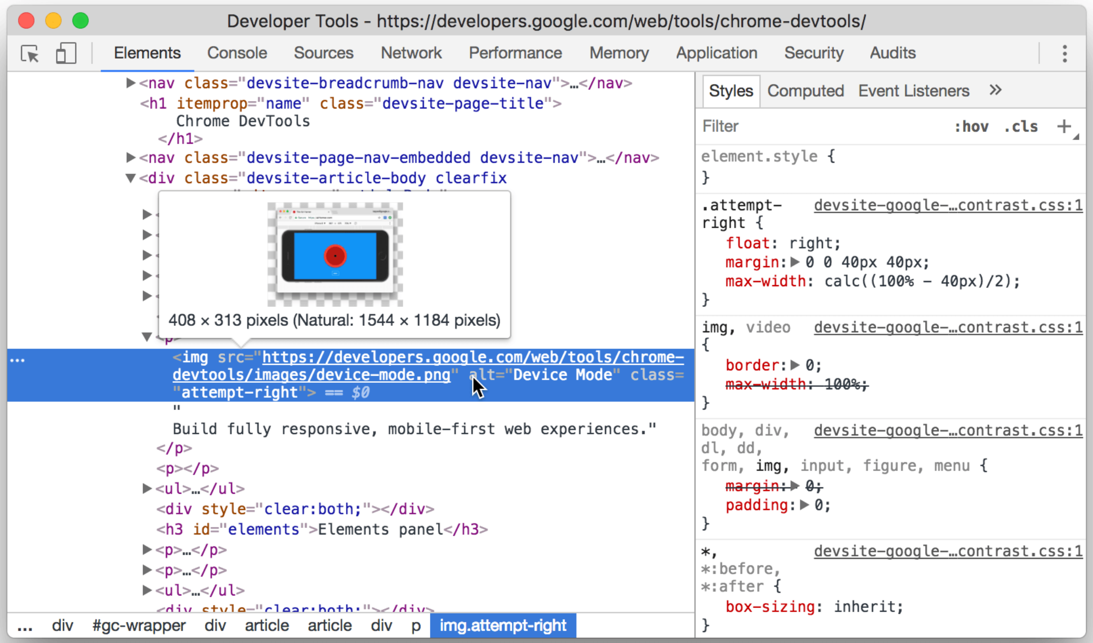
Press Ctrl + Shift + C (Windows) or Cmd + Shift + C (Mac) to open DevTools in Inspect Element mode, then hover over an element. DevTools automatically highlights the element that you are hovering over in the **Elements** panel. Click on the element to exit inspect mode while keeping the element highlighted within the **Elements** panel.

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

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

View the rendered and natural sizes of an image

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


Developer Tools - https://developers.google.com/web/tools/chrome-devtools/
Elements Console Sources Network Performance Memory Application Security Audits
Styles Computed Event Listeners »
Filter :hov .cls +
408 × 313 pixels (Natural: 1544 × 1184 pixels)
... div #gc-wrapper div article article div p img.attempt-right

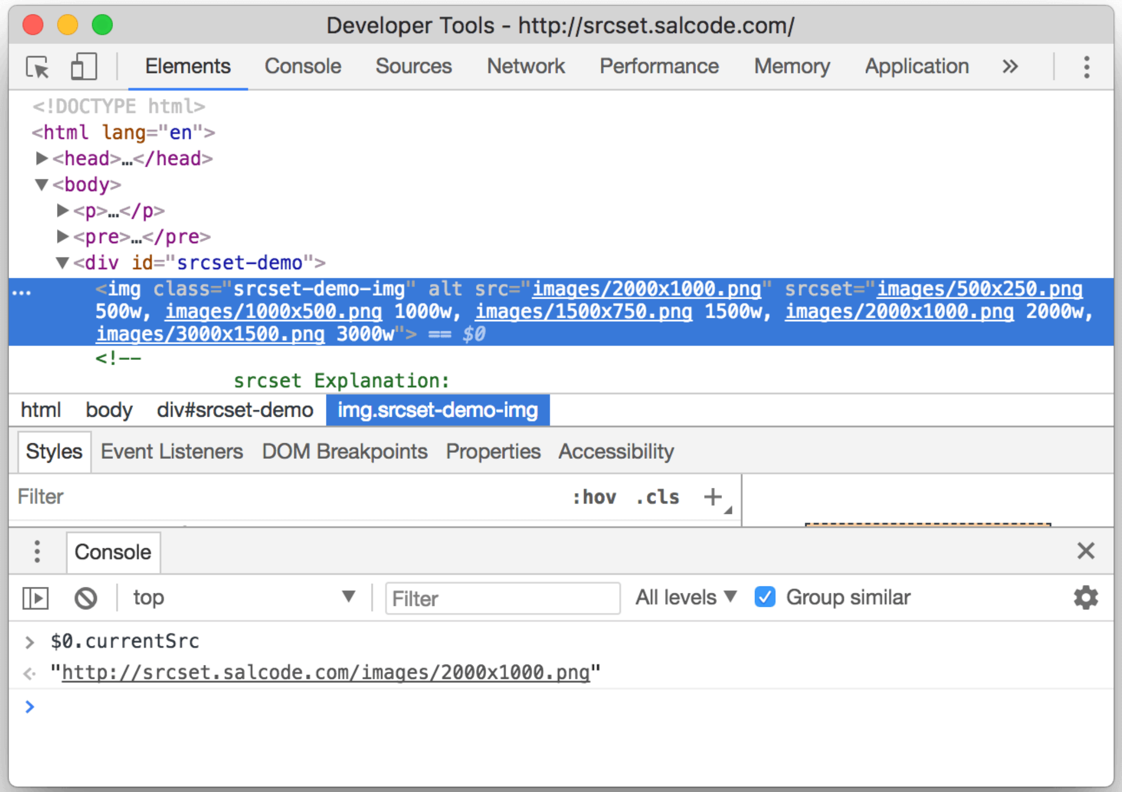
```
▶<nav class="devsite-breadcrumb-nav devsite-nav">…</nav>
 <h1 itemprop="name" class="devsite-page-title">
     Chrome DevTools
   </h1>
▶<nav class="devsite-page-nav-embedded devsite-nav">…</nav>
▼<div class="devsite-article-body clearfix
      <img src="https://developers.google.com/web/tools/chrome-
      devtools/images/device-mode.png" alt="Device Mode" class=
      "attempt-right"> == $0
      "
      Build fully responsive, mobile-first web experiences."
    </p>
    <p></p>
  ▶<ul>…</ul>
    <div style="clear:both;"></div>
    <h3 id="elements">Elements panel</h3>
  ▶<p>…</p>
  ▶<p>…</p>
  ▶<ul>…</ul>
    <div style="clear:both;"></div>
```

```
element.style {
}
.attempt-     devsite-google-…contrast.css:1
right {
   float: right;
   margin:▶ 0 0 40px 40px;
   max-width: calc((100% - 40px)/2);
}
img, video    devsite-google-…contrast.css:1
{
   border:▶ 0;
   ~~max-width: 100%;~~
}
body, div,    devsite-google-…contrast.css:1
dl, dd,
form, img, input, figure, menu {
   ~~margin:▶ 0;~~
   padding:▶ 0;
}
*,            devsite-google-…contrast.css:1
*:before,
*:after {
   box-sizing: inherit;
}
```

View which image in a source set (srcset) is being used

To view which version of an image in a `srcset` was loaded, select the `img` element, then evaluate `$0.currentSrc` in the **Console**.

Note: See Enhance `imgs` with `srcset` for high DPI devices to learn more about image optimization using `srcset`.

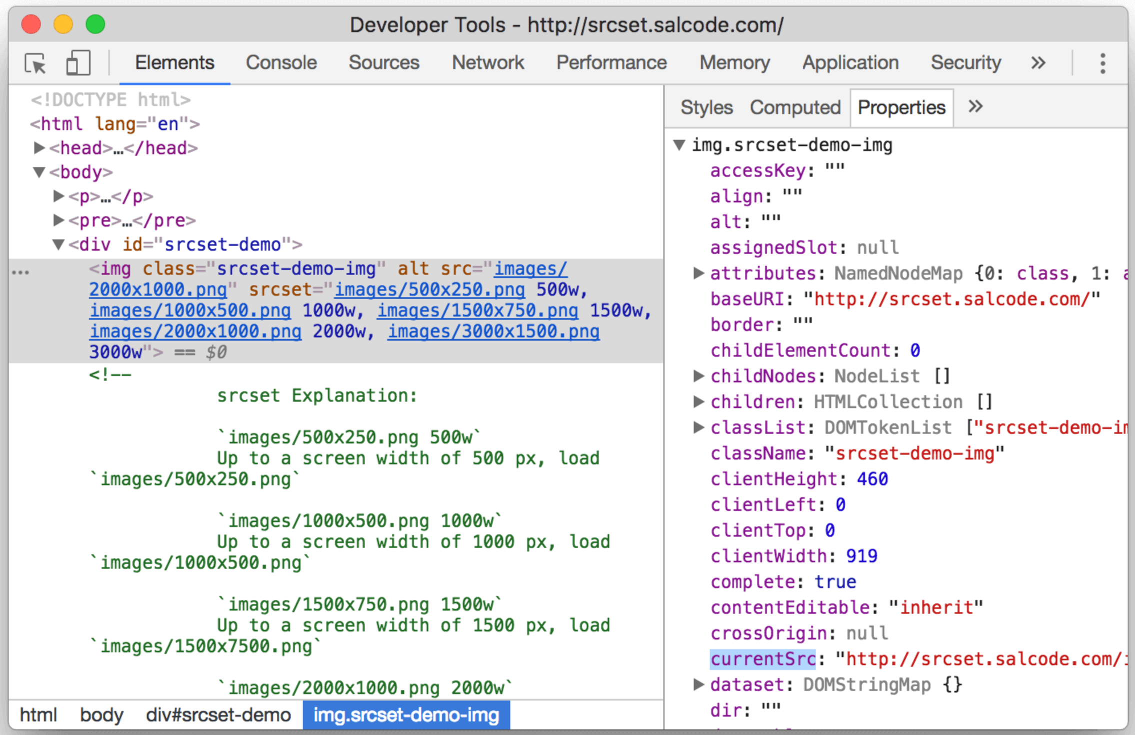


```
<!DOCTYPE html>
<html lang="en">
 ▶<head>…</head>
 ▼<body>
   ▶<p>…</p>
   ▶<pre>…</pre>
   ▼<div id="srcset-demo">
...     <img class="srcset-demo-img" alt src="images/2000x1000.png" srcset="images/500x250.png
        500w, images/1000x500.png 1000w, images/1500x750.png 1500w, images/2000x1000.png 2000w,
        images/3000x1500.png 3000w"> == $0
        <!--
                   srcset Explanation:
```

```
> $0.currentSrc
< "http://srcset.salcode.com/images/2000x1000.png"
>
```

Note: `$0` is a shortcut in the DevTools **Console**. It provides a reference to the currently-selected element in the **DOM Tree**.

You can also view `currentSrc` via the **Properties** tab. The **Properties** tab only displays properties for the currently-selected element, so make sure that you've selected the correct element before viewing.



```
<!DOCTYPE html>
<html lang="en">
 ▶<head>…</head>
 ▼<body>
   ▶<p>…</p>
   ▶<pre>…</pre>
   ▼<div id="srcset-demo">
...     <img class="srcset-demo-img" alt src="images/
        2000x1000.png" srcset="images/500x250.png 500w,
        images/1000x500.png 1000w, images/1500x750.png 1500w,
        images/2000x1000.png 2000w, images/3000x1500.png
        3000w"> == $0
        <!--
                    srcset Explanation:

                    `images/500x250.png 500w`
                    Up to a screen width of 500 px, load
        `images/500x250.png`

                    `images/1000x500.png 1000w`
                    Up to a screen width of 1000 px, load
        `images/1000x500.png`

                    `images/1500x750.png 1500w`
                    Up to a screen width of 1500 px, load
        `images/1500x7500.png`

                    `images/2000x1000.png 2000w`
```

```
▼ img.srcset-demo-img
    accessKey: ""
    align: ""
    alt: ""
    assignedSlot: null
  ▶ attributes: NamedNodeMap {0: class, 1: a
    baseURI: "http://srcset.salcode.com/"
    border: ""
    childElementCount: 0
  ▶ childNodes: NodeList []
  ▶ children: HTMLCollection []
  ▶ classList: DOMTokenList ["srcset-demo-in
    className: "srcset-demo-img"
    clientHeight: 460
    clientLeft: 0
    clientTop: 0
    clientWidth: 919
    complete: true
    contentEditable: "inherit"
    crossOrigin: null
    currentSrc: "http://srcset.salcode.com/i
  ▶ dataset: DOMStringMap {}
    dir: ""
```

Navigate the DOM

Navigate through the DOM structure using your mouse or keyboard.

A collapsed node has an arrow next to it pointing right: ▶ <head>…</head>

An expanded node has an arrow next to it pointing down: ▼ <head>

Using your mouse:

- Click once to highlight a node.
- To expand a node, double-click anywhere on it or click on the arrow next to it.
- To collapse a node, click on the arrow next to it.

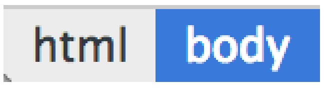
Using your keyboard:

- Press the **Up Arrow** key to select the node above the current one.

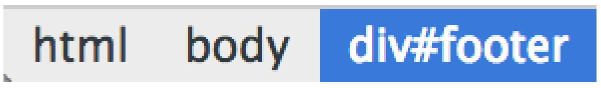
- Press the **Down Arrow** to select the node below the current one.
- Press the **Right Arrow** key to expand a collapsed node. Press it again to move to the first child of the (now-expanded) node. You can use this technique to quickly navigate deeply-nested nodes.

Navigate the breadcrumb trail

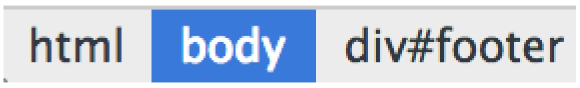
At the bottom of the Elements panel is a breadcrumb trail.



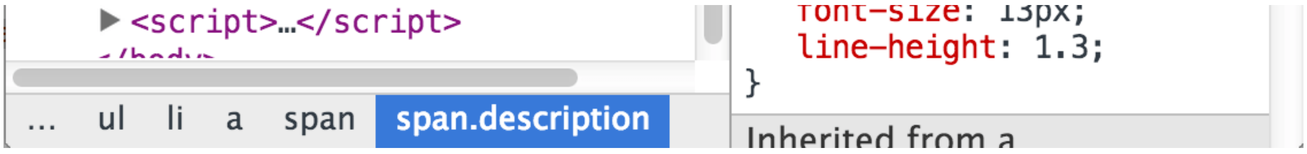
The currently selected node is highlighted in blue. The node to the left is the current node's parent. And to the left of that is the parent's parent. And so on, all the way up the tree.



Navigating back up the structure moves the highlight:



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:


```
▶ <script>…</script>
```
```
  line-height: 1.3;
}
```

Edit DOM nodes and attributes

To edit a DOM node name or attribute:

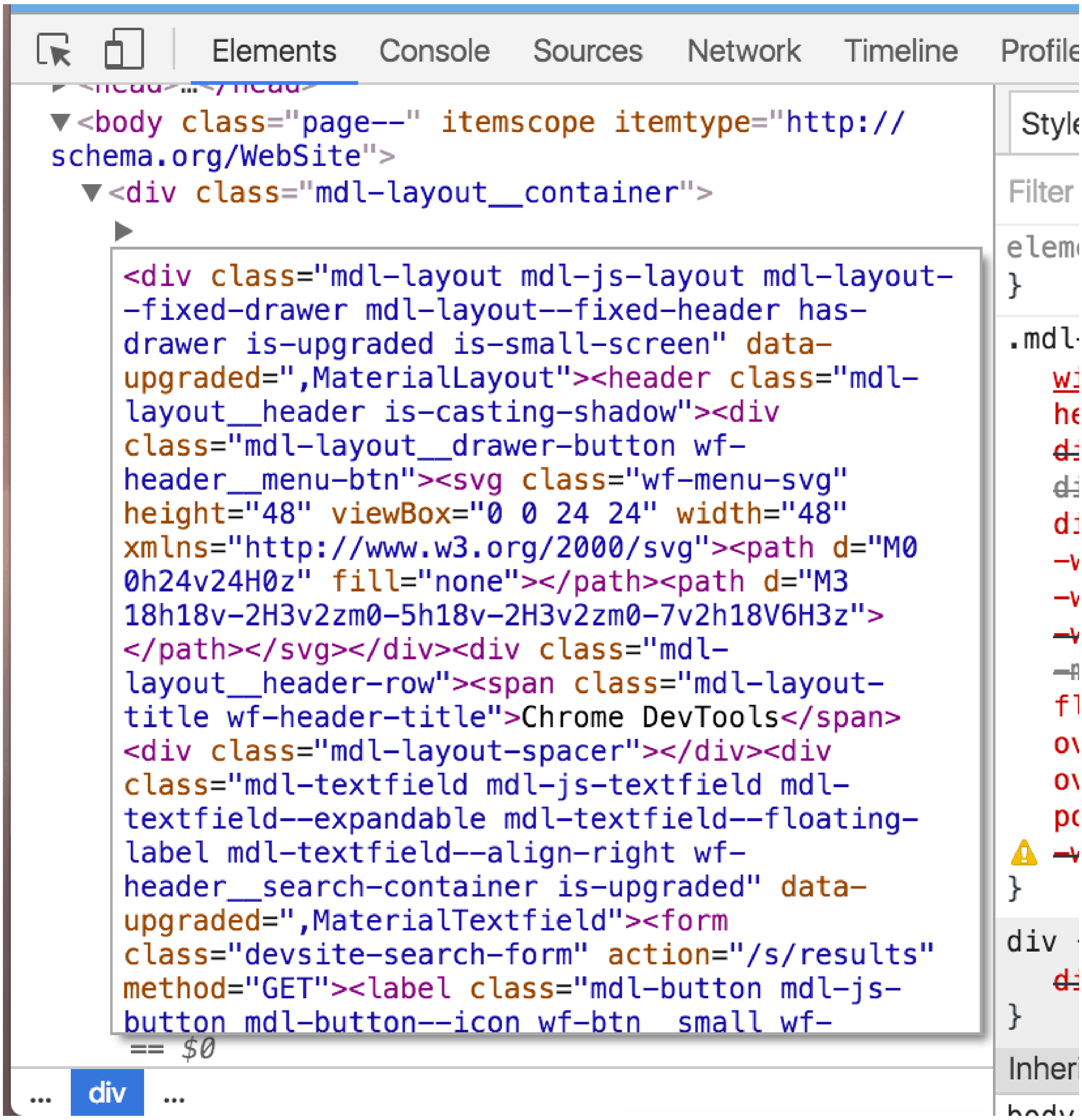
- Double-click directly on the node name or attribute.
- Highlight the node, press Enter, and then press Tab until the name or attribute is selected.
- Open the more actions menu and select **Add Attribute** or **Edit Attribute**. **Edit Attribute** is context-sensitive; the portion you click on determines what gets edited.

The closing tag is automatically updated when you're finished.

Edit DOM node and its children as HTML

To edit a DOM node and its children as HTML:

- Open the more actions menu and select **Edit as HTML**.
- Press F2 (Windows / Linux) or Fn+F2 (Mac).
- Press Ctrl+Enter (Windows / Linux) or Cmd+Enter (Mac) to save your changes.
- Press Esc to exit the editor without saving.



```
▼<body class="page--" itemscope itemtype="http://
schema.org/WebSite">
  ▼<div class="mdl-layout__container">
    ▶
    <div class="mdl-layout mdl-js-layout mdl-layout-
    -fixed-drawer mdl-layout--fixed-header has-
    drawer is-upgraded is-small-screen" data-
    upgraded=",MaterialLayout"><header class="mdl-
    layout__header is-casting-shadow"><div
    class="mdl-layout__drawer-button wf-
    header__menu-btn"><svg class="wf-menu-svg"
    height="48" viewBox="0 0 24 24" width="48"
    xmlns="http://www.w3.org/2000/svg"><path d="M0
    0h24v24H0z" fill="none"></path><path d="M3
    18h18v-2H3v2zm0-5h18v-2H3v2zm0-7v2h18V6H3z">
    </path></svg></div><div class="mdl-
    layout__header-row"><span class="mdl-layout-
    title wf-header-title">Chrome DevTools</span>
    <div class="mdl-layout-spacer"></div><div
    class="mdl-textfield mdl-js-textfield mdl-
    textfield--expandable mdl-textfield--floating-
    label mdl-textfield--align-right wf-
    header__search-container is-upgraded" data-
    upgraded=",MaterialTextfield"><form
    class="devsite-search-form" action="/s/results"
    method="GET"><label class="mdl-button mdl-js-
    button mdl-button--icon wf-btn  small wf-
     == $0
```

Move DOM node

Click, hold, and drag a node to move it.

Delete DOM node

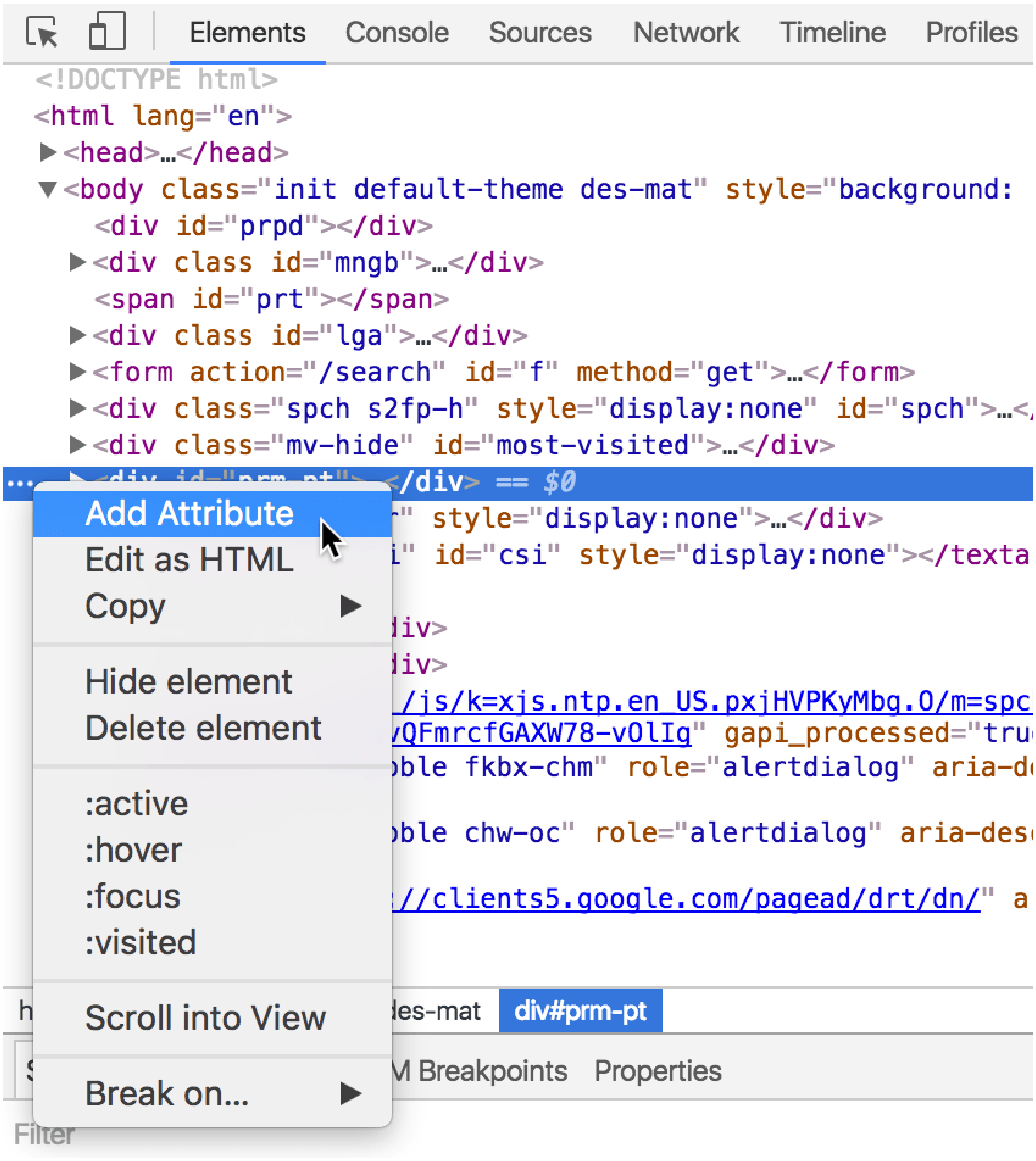
To delete a DOM node:

- Open the more actions menu and select **Delete Node**.
- Select the node and press the Delete key.

Note: If you delete a node by accident, Ctrl + Z (or Cmd + Z on Mac) to undo your last action.

Show more actions menu

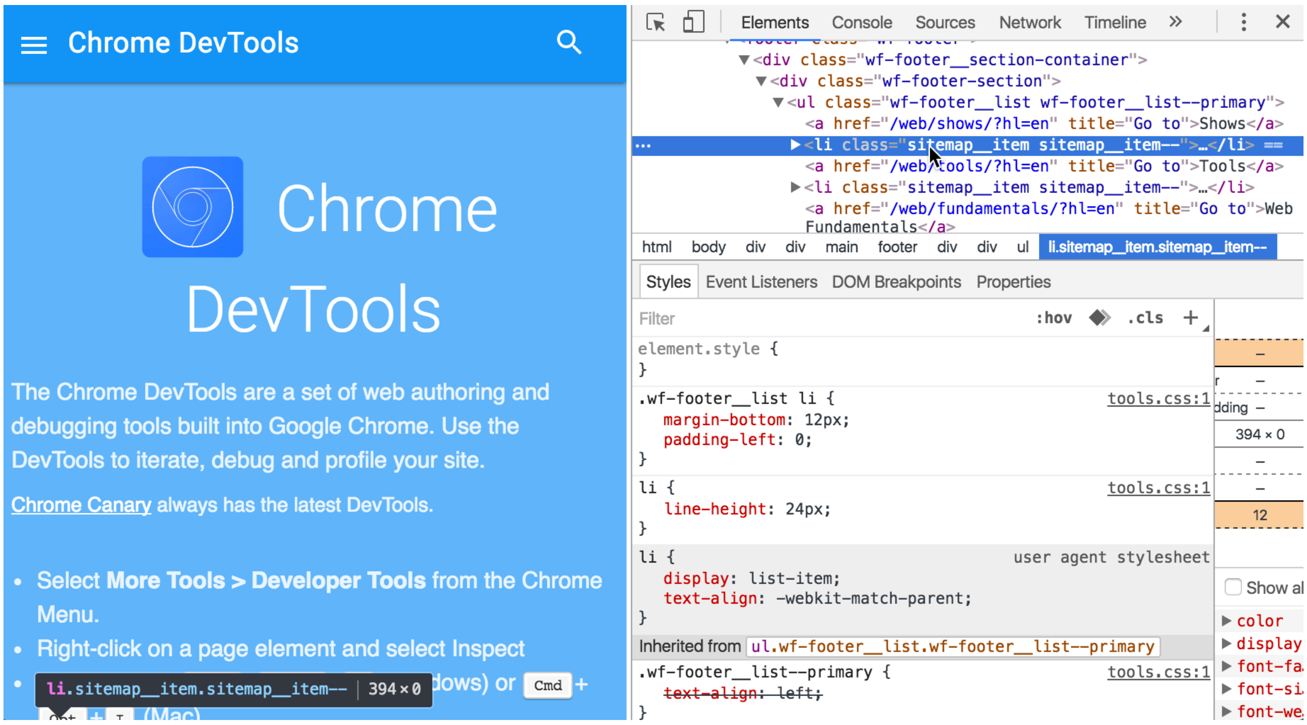
The **more actions** menu lets you interact with a DOM node in a variety of ways. To view the menu, right-click on a node, or select a node and then press the **more actions** button (•••)). The button is only displayed on the currently selected element.



```
<!DOCTYPE html>
<html lang="en">
 ▶<head>…</head>
 ▼<body class="init default-theme des-mat" style="background:
    <div id="prpd"></div>
   ▶<div class id="mngb">…</div>
    <span id="prt"></span>
   ▶<div class id="lga">…</div>
   ▶<form action="/search" id="f" method="get">…</form>
   ▶<div class="spch s2fp-h" style="display:none" id="spch">…<
   ▶<div class="mv-hide" id="most-visited">…</div>
…  ▶<div id="prm-pt">…</div> == $0
                     " style="display:none">…</div>
                     i" id="csi" style="display:none"></texta

                     div>
                     div>
                     /js/k=xjs.ntp.en_US.pxjHVPKyMbg.O/m=spc
                     vQFmrcfGAXW78-vOlIg" gapi_processed="tru
                     ble fkbx-chm" role="alertdialog" aria-d

                     ble chw-oc" role="alertdialog" aria-des

                     //clients5.google.com/pagead/drt/dn/" a
```

Scroll into view

When you hover over or select a DOM node, the rendered node is highlighted in the viewport. If the node is scrolled offscreen, you'll see a tooltip at the top of the viewport if the node is above the current viewport, and a tooltip at the bottom if the node is below the current viewport. For example, in the screenshot below DevTools is indicating that the currently selected element in the **Elements** panel is below the viewport.



The Chrome DevTools are a set of web authoring and debugging tools built into Google Chrome. Use the DevTools to iterate, debug and profile your site.

Chrome Canary always has the latest DevTools.

- Select More Tools > Developer Tools from the Chrome Menu.
- Right-click on a page element and select Inspect

```
▼<div class="wf-footer__section-container">
  ▼<div class="wf-footer__section">
    ▼<ul class="wf-footer__list wf-footer__list--primary">
        <a href="/web/shows/?hl=en" title="Go to">Shows</a>
      ▶<li class="sitemap__item sitemap__item--">…</li> ==
        <a href="/web/tools/?hl=en" title="Go to">Tools</a>
      ▶<li class="sitemap__item sitemap__item--">…</li>
        <a href="/web/fundamentals/?hl=en" title="Go to">Web
        Fundamentals</a>
```

html body div div main footer div div ul li.sitemap__item.sitemap__item--

```
element.style {
}
.wf-footer__list li {                                  tools.css:1
  margin-bottom: 12px;
  padding-left: 0;
}
li {                                                   tools.css:1
  line-height: 24px;
}
li {                                          user agent stylesheet
  display: list-item;
  text-align: -webkit-match-parent;
}
Inherited from ul.wf-footer__list.wf-footer__list--primary
.wf-footer__list--primary {                            tools.css:1
  ~~text-align: left;~~
}
```

li.sitemap__item.sitemap__item-- | 394 × 0

To scroll the page so the node appears in the viewport, **Right-click** the node and select **Scroll into View**.

Set DOM breakpoints

Set DOM breakpoints to debug complex JavaScript applications. For example, if your JavaScript is changing the styling of a DOM element, set a DOM breakpoint to fire when the element's attributes are modified. Trigger a breakpoint on one of the following DOM changes: subtree change, attribute change, node removal.

Subtree Modifications

A subtree modification breakpoint is triggered when a child element is added, removed, or moved. For example, if you set a subtree modification breakpoint on the **`main-content`** element, the following code triggers the breakpoint:

```
var element = document.getElementById('main-content');
//modify the element's subtree.
var mySpan = document.createElement('span');
element.appendChild( mySpan );
```

Attribute Modifications

An attribute modification occurs when the attribute of an element (`class, id, name`) is changed dynamically:

```
var element = document.getElementById('main-content');
// class attribute of element has been modified.
element.className = 'active';
```

Node Removal

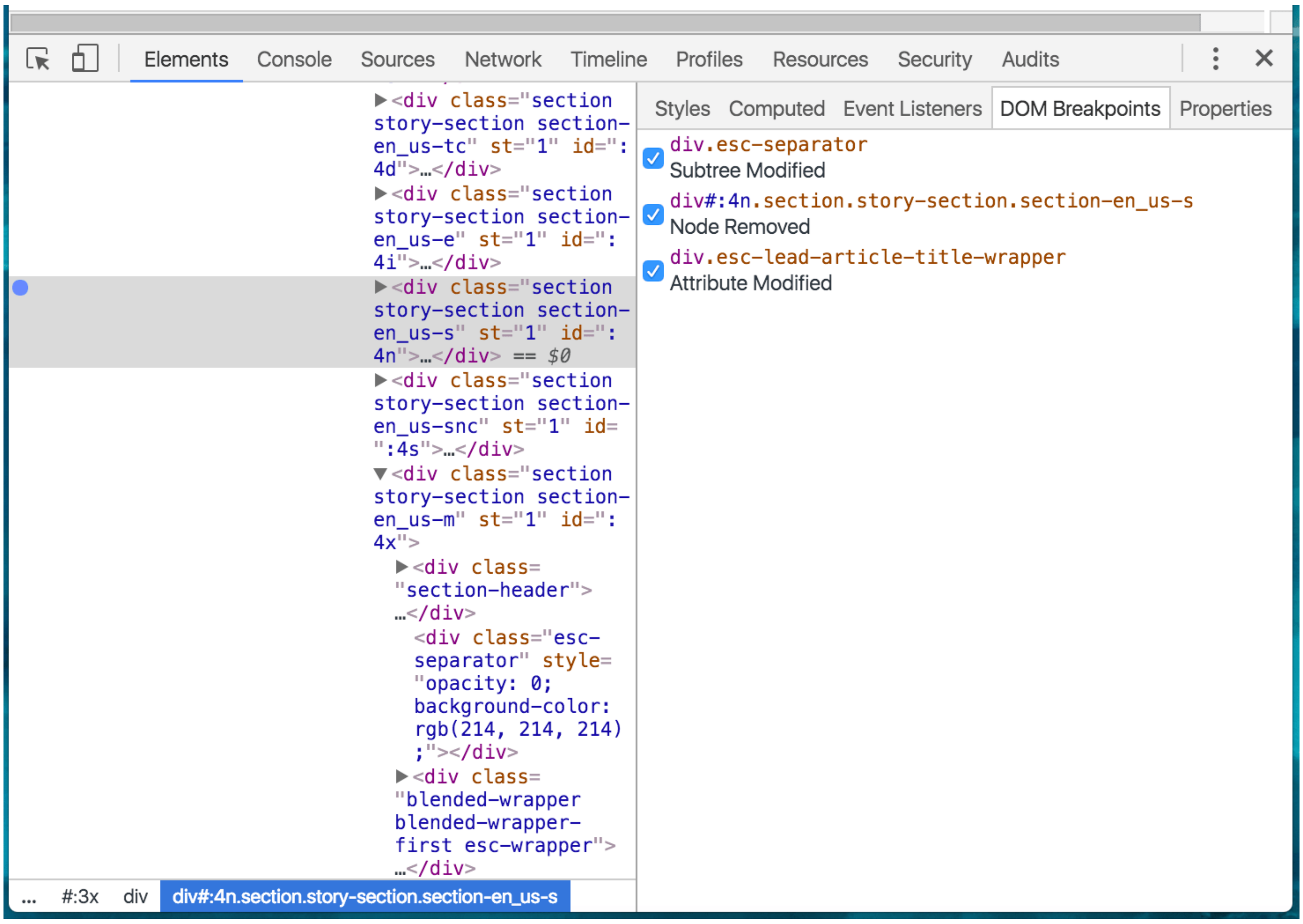
A node removal modification is triggered when the node in question is removed from the DOM:

```
document.getElementById('main-content').remove();
```

Interact with DOM breakpoints

The Elements and Sources panels both include a pane for managing your DOM breakpoints.

Each breakpoint is listed with an element identifier and the breakpoint type.



```
▶<div class="section story-section section-en_us-tc" st="1" id=":4d">…</div>
▶<div class="section story-section section-en_us-e" st="1" id=":4i">…</div>
▶<div class="section story-section section-en_us-s" st="1" id=":4n">…</div> == $0
▶<div class="section story-section section-en_us-snc" st="1" id=":4s">…</div>
▼<div class="section story-section section-en_us-m" st="1" id=":4x">
  ▶<div class="section-header">…</div>
   <div class="esc-separator" style="opacity: 0; background-color: rgb(214, 214, 214);"></div>
  ▶<div class="blended-wrapper blended-wrapper-first esc-wrapper">…</div>
```

✓ div.esc-separator
Subtree Modified

✓ div#:4n.section.story-section.section-en_us-s
Node Removed

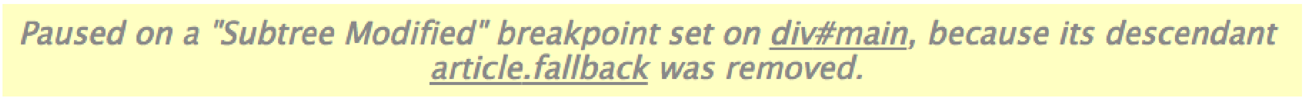
✓ div.esc-lead-article-title-wrapper
Attribute Modified

… #:3x div div#:4n.section.story-section.section-en_us-s

Interact with each listed breakpoint in any of the following ways:

- **Hover** over the element identifier to show the element's corresponding position on the page (similar to hovering over nodes in the Elements panel).
- **Click** an element to select it in the Elements panel.
- **Toggle** the checkbox to enable or disable the breakpoint.

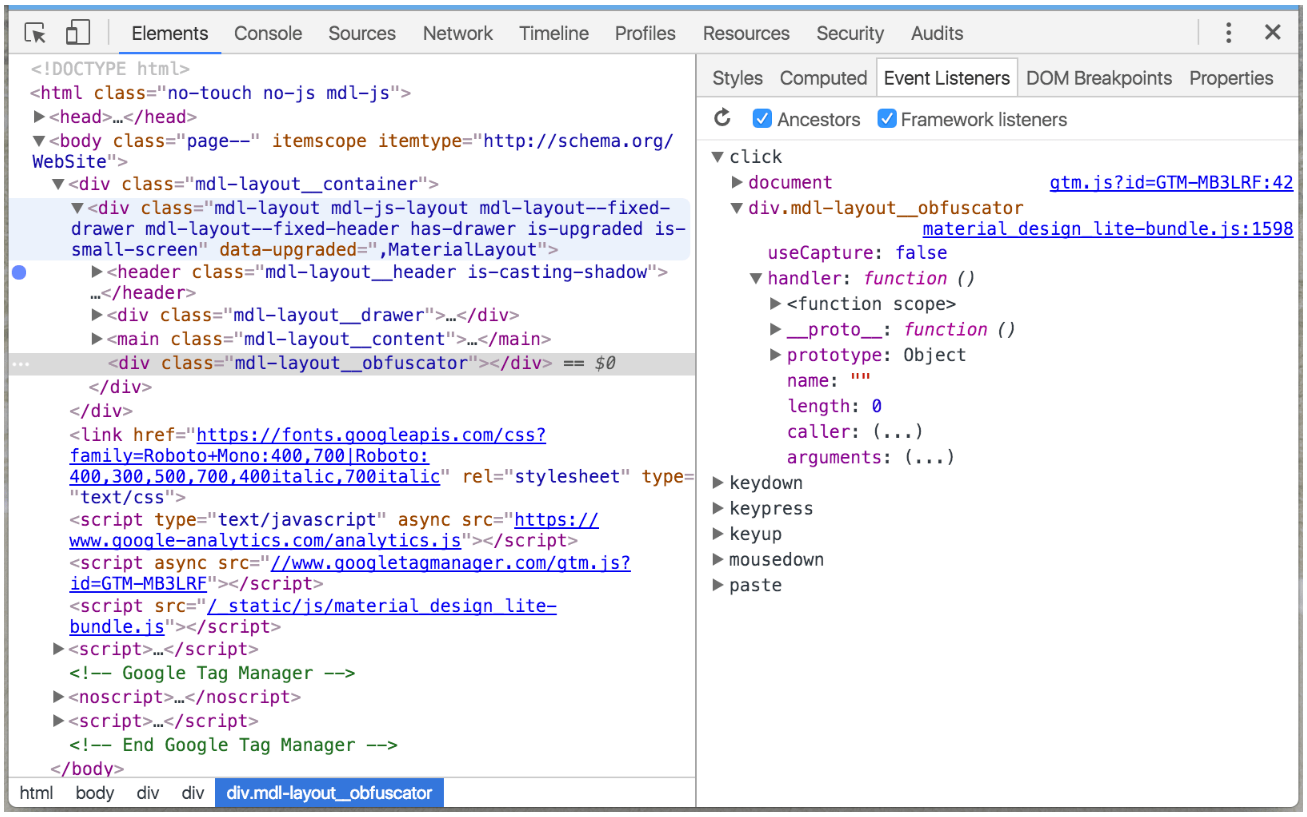
When you trigger a DOM breakpoint, the breakpoint is highlighted in the DOM Breakpoints pane. The **Call Stack** pane displays the **reason** for a debugger pause:



Paused on a "Subtree Modified" breakpoint set on div#main, because its descendant article.fallback was removed.

View element event listeners

View JavaScript event listeners associated with a DOM node in the **Event Listeners** pane.



```
<!DOCTYPE html>
<html class="no-touch no-js mdl-js">
  ▶<head>…</head>
  ▼<body class="page--" itemscope itemtype="http://schema.org/
  WebSite">
    ▼<div class="mdl-layout__container">
      ▼<div class="mdl-layout mdl-js-layout mdl-layout--fixed-
      drawer mdl-layout--fixed-header has-drawer is-upgraded is-
      small-screen" data-upgraded=",MaterialLayout">
        ▶<header class="mdl-layout__header is-casting-shadow">
        …</header>
        ▶<div class="mdl-layout__drawer">…</div>
        ▶<main class="mdl-layout__content">…</main>
         <div class="mdl-layout__obfuscator"></div> == $0
       </div>
     </div>
     <link href="https://fonts.googleapis.com/css?
     family=Roboto+Mono:400,700|Roboto:
     400,300,500,700,400italic,700italic" rel="stylesheet" type=
     "text/css">
     <script type="text/javascript" async src="https://
     www.google-analytics.com/analytics.js"></script>
     <script async src="//www.googletagmanager.com/gtm.js?
     id=GTM-MB3LRF"></script>
     <script src="/_static/js/material_design_lite-
     bundle.js"></script>
    ▶<script>…</script>
     <!-- Google Tag Manager -->
    ▶<noscript>…</noscript>
    ▶<script>…</script>
     <!-- End Google Tag Manager -->
   </body>
```

```
▼ click
  ▶ document                              gtm.js?id=GTM-MB3LRF:42
  ▼ div.mdl-layout__obfuscator
                              material_design_lite-bundle.js:1598
      useCapture: false
    ▼ handler: function ()
      ▶ <function scope>
      ▶ __proto__: function ()
      ▶ prototype: Object
        name: ""
        length: 0
        caller: (...)
        arguments: (...)
▶ keydown
▶ keypress
▶ keyup
▶ mousedown
▶ paste
```

The top-level items in the Event Listeners pane show the event types that have registered listeners.

Click the arrow next to the event type (for example `click`) to see a list of registered event handlers. Each handler is identified by a CSS selector-like element identifier, such as `document` or `button#call-to-action`. If more than one handler is registered for the same element, the element is listed repeatedly.

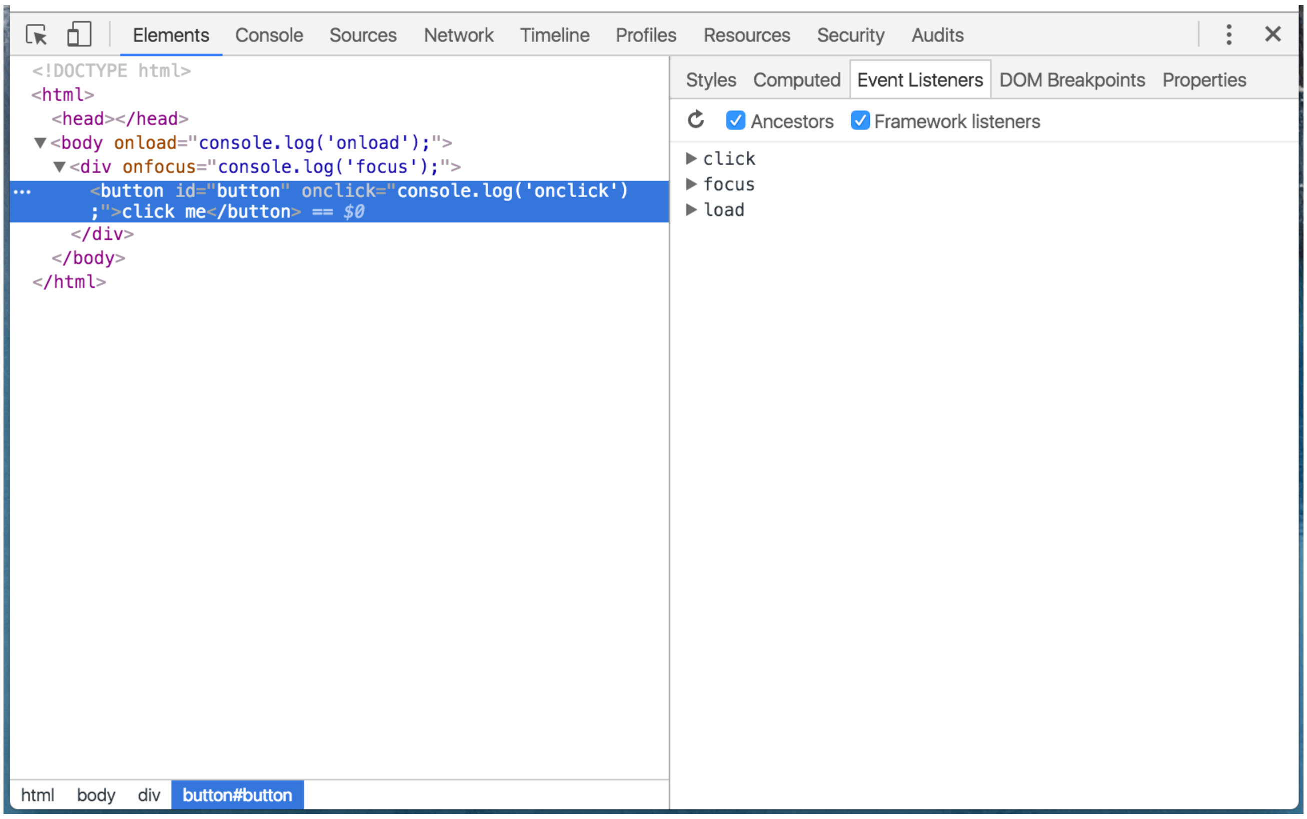
Click the expander arrow next to an element identifier to see the properties of the event handler. The Event Listeners pane lists the following properties for each listener:

Event Listener Properties & Description	
`handler`	Contains a callback function. Right-click on the function and select **Show Function Definition** to view where the function is defined (if source code is available).
`useCapture`	A boolean value stating whether the useCapture flag on `addEventListener` was set.

Note: Many Chrome extensions add their own event listeners onto the DOM. If you see a number of event listeners that aren't set by your code, you may want to reopen your page in an Incognito window. Incognito windows prevent extensions from running by default.

View ancestor event listeners

When the **Ancestors** checkbox is enabled, the event listeners for the ancestors of the currently selected node are displayed, in addition to the currently selected node's event listeners.

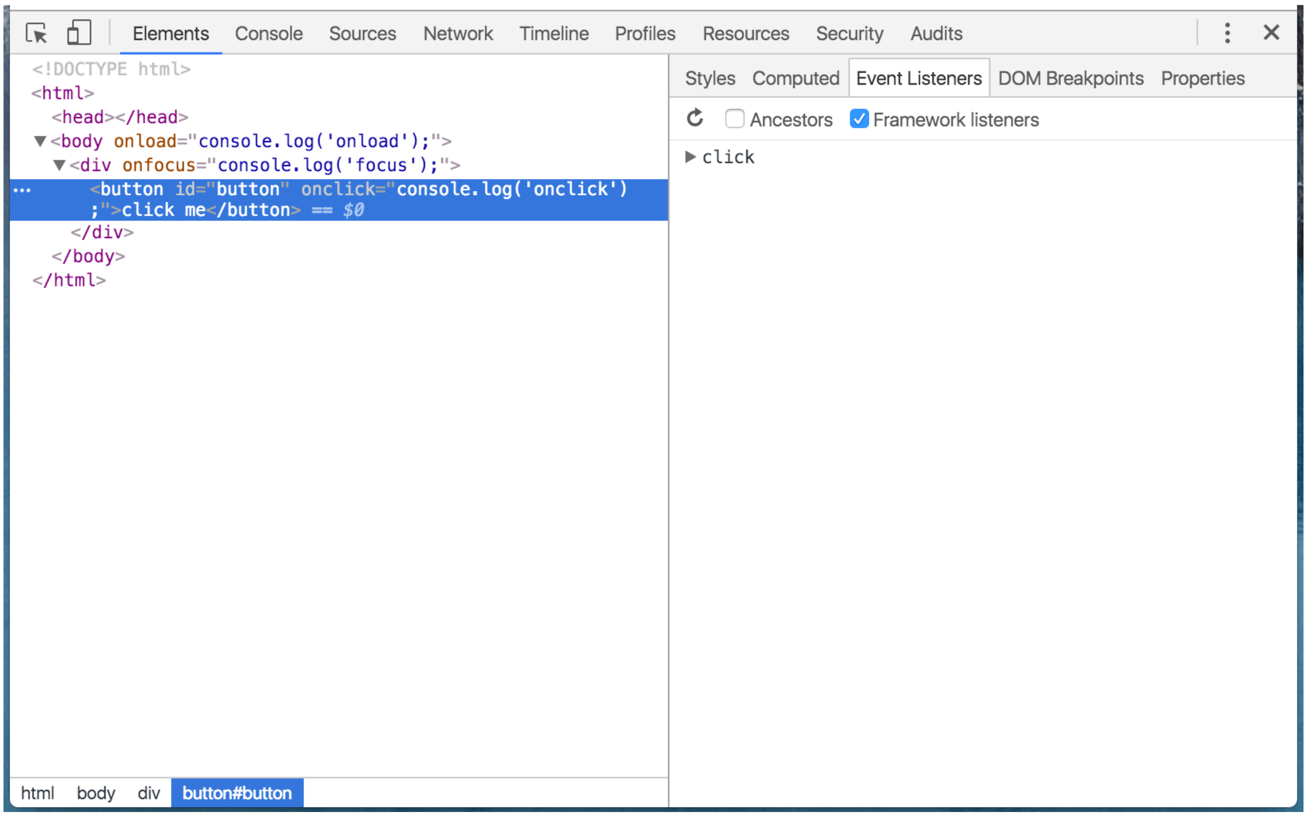


```
<!DOCTYPE html>
<html>
  <head></head>
▼<body onload="console.log('onload');">
  ▼<div onfocus="console.log('focus');">
      <button id="button" onclick="console.log('onclick')
      ;">click me</button> == $0
    </div>
  </body>
</html>
```

☑ Ancestors ☑ Framework listeners

```
▶ click
▶ focus
▶ load
```

When the checkbox is disabled, only the event listeners for the currently selected node are displayed.



```
<!DOCTYPE html>
<html>
  <head></head>
▼<body onload="console.log('onload');">
  ▼<div onfocus="console.log('focus');">
      <button id="button" onclick="console.log('onclick')
      ;">click me</button> == $0
    </div>
  </body>
</html>
```

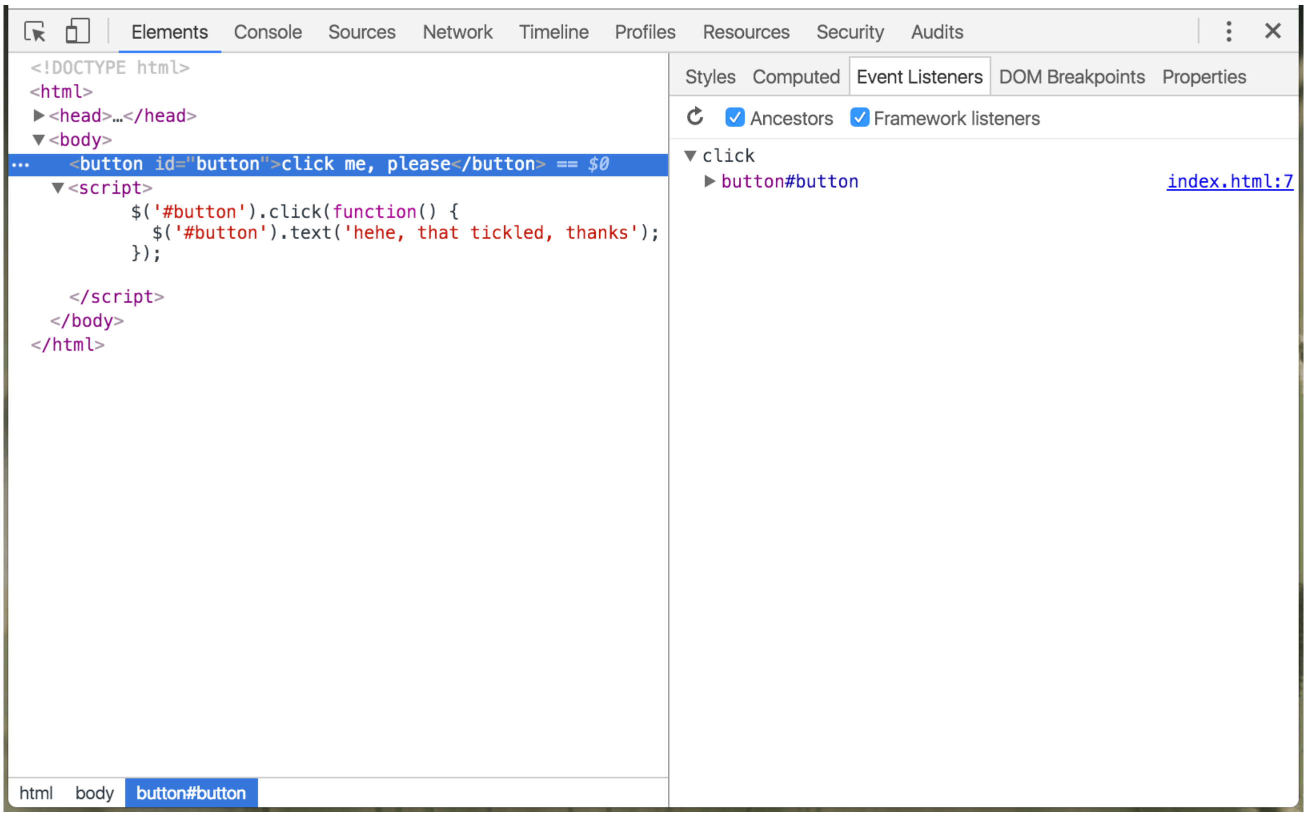
☐ Ancestors ☑ Framework listeners

```
▶ click
```

View framework listeners

Some JavaScript frameworks and libraries wrap native DOM events into their custom event APIs. In the past this made it hard to inspect the event listeners with DevTools, because the function definition would just reference back to the framework or library code. The **Framework listeners** feature solves this problem.

When the **Framework listeners** checkbox is enabled, DevTools automatically resolves the framework or library wrapping portion of the event code, and then tells you where you actually bound the event in your own code.

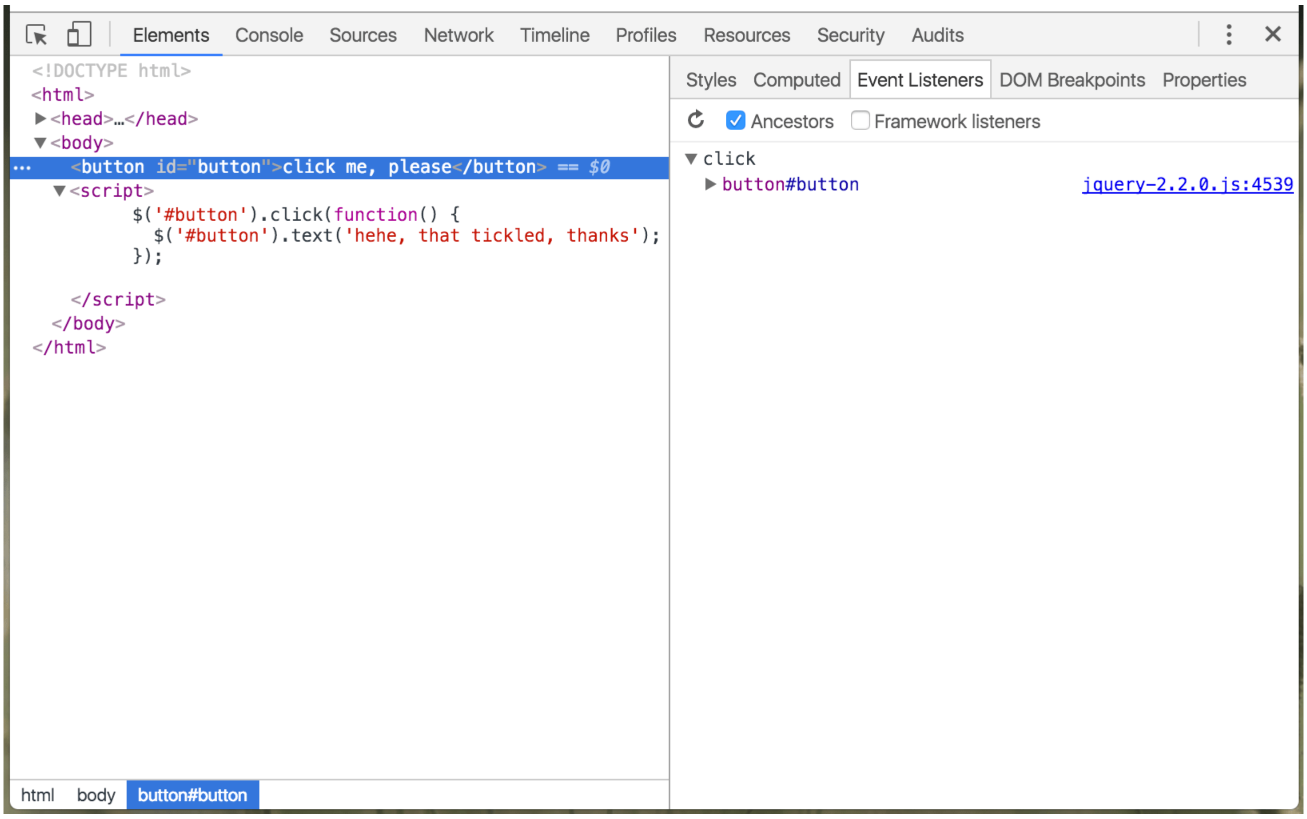


```
<!DOCTYPE html>
<html>
▶<head>…</head>
▼<body>
    <button id="button">click me, please</button> == $0
  ▼<script>
        $('#button').click(function() {
          $('#button').text('hehe, that tickled, thanks');
        });

    </script>
  </body>
</html>
```

✓ Ancestors ✓ Framework listeners

```
▼ click
  ▶ button#button                                index.html:7
```

When the **Framework listeners** checkbox is disabled, the event listener code will probably resolve somewhere in the framework or library code.



```
<!DOCTYPE html>
<html>
▶<head>…</head>
▼<body>
    <button id="button">click me, please</button> == $0
  ▼<script>
          $('#button').click(function() {
            $('#button').text('hehe, that tickled, thanks');
          });

    </script>
  </body>
</html>
```

```
▼ click
  ▶ button#button                    jquery-2.2.0.js:4539
```

Show HTML comments

To show or hide HTML comments in the Elements panel:

1. Open Settings.
2. Click the **Preferences** tab.
3. Under the **Elements** section, check the **Show HTML comments** checkbox.

To show or hide HTML comments in the **Elements** panel, open **Settings**, go to the **Preferences** panel, find the **Elements** section, and then toggle the **Show HTML comments** checkbox.