

# Inspecting the Developer Tools

Ana Risteska / a.risteska@gmail.com



## About me

■ UI design / front-end development / UX @ **netcetera**

■ Member of Free Software Macedonia

■ Part of the Front-End meet ups @ Hacklab KIKA in Skopje

■ Part of the Macedonian Mozilla community

# What's in this talk?

■ Developer Tools - what and why?

■ Overview of some of the Firefox and Chrome Developer Tools and their panels

■ ... and what can we do with them to speed up our workflow

■ ... or to learn how our page works

**So what are the DevTools exactly?**

**Visual and on-demand tools** that help you  
**build** and **debug** your webpage

**DevTools give us control over the building blocks of every web page**



# What can we do with the DevTools?

- Manipulate the DOM tree
- Manipulate the CSS styles
- Debug JavaScript: find and fix errors
- Analyze performance - loading time of scripts
- Test the responsivity of a web page
- **Learn how our webpage works, so we can know how to make it better!**

# **Disclaimer: Automatic save-to-disk is not possible**

■ ...unless we set up some kind of authoring and saving tools

■ Firefox: Scratchpad, Style Editor etc for writing.  
Add-ons for saving: Firediff and others.

■ Chrome: Enabling Workspaces

## A bit of history

- 2001 - Netscape Navigator 7 had a JavaScript debugger called Venkman. It couldn't inspect the DOM or show network traffic.
- 2005 - Microsoft Issues IE Developer Toolbar in 2005. DOM inspector and styles
- 2006 - Firebug for Mozilla Firefox - second generation of devtools: has DOM and styles inspector and JavaScript console and debugger
- A few years later Google Chrome and its Developer Tools came around

# Finding the devtools in the browser

**Right-click menu** (context menu) and choose the options:  
**Inspect Element**, or just **Inspect**

Shortcuts: **F12, ctrl+shift+i**  
cmd instead of ctrl for Mac users

# The DOM tree

## On the GUI:

1. First panel: “Inspector” on FF, “Elements” on Chrome
2. Easy to see on which element I am focused on
3. HTML breadcrumbs
4. Searching for nodes in the HTML tree (ctrl+f)
5. Simulate mobile devices (ctrl+shift+m on both browsers)
6. 3D View on Firefox

## Actions:

Edit the DOM: add and delete nodes, change text and attributes

2. 5. 1.

Chrome

The screenshot shows the Chrome DevTools interface with the 'Elements' tab selected. The left pane displays the DOM tree of the page, with the 'wrapper' div highlighted. The right pane shows the 'Styles' tab of the styles panel, listing CSS rules applied to the selected element. The 'element.style' section contains rules from 'www.softwareishard.com/style.css:30'. The 'media="screen"' section contains rules from 'www.softwareishard.com/style.css:9'. The 'user agent stylesheet' section contains rules from the user agent's style sheet. The 'Inherited from body' section contains rules from 'www.softwareishard.com/style.css:10'. A green oval highlights the 'Elements' tab and the 'wrapper' node in the DOM tree. A green circle highlights the 'div#Wrapper' entry in the bottom navigation bar.

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  
<html xmlns="http://www.w3.org/1999/xhtml">  
 <head profile="http://gmpg.org/xfn/11">...</head>  
 <body cz-shortcut-listen="true">  
 <script type="text/javascript">...</script>  
 ...  
 <div id="wrapper">...</div>  
 <div id="sk2-footer" style="color:#FFF; background-color:#444;  
padding: 3px 2px 3px 2px; border-top: #888 solid 1px;">...</div>  
 <script src="http://www.google-analytics.com/urchin.js" type=  
"text/javascript">  
 </script>  
 <script type="text/javascript">  
 \_uacct = "UA-3586722-1";  
 urchinTracker();  
 </script>  
 <div id="overlay" style="display: none;"></div>  
 <div id="lightbox" style="display: none;">...</div>  
 </body>  
</html>

Styles Computed Event Listeners DOM Breakpoints Properties

Filter

element.style {  
}

media="screen" [www.softwareishard.com/  
style.css:30](http://www.softwareishard.com/style.css:30)  
#wrapper {  
 margin: 0 auto;  
 text-align: left;  
 width: 760px;  
 background: #FFFFFF;  
 font-size: 0.9em;  
 line-height: 1.6em;  
 padding: 20px 10px 0 10px;  
}  
media="screen" [www.softwareishard.com/  
style.css:9](http://www.softwareishard.com/style.css:9)  
\* {  
 padding: 0;  
 margin: 0;  
}  
div {  
 display: block;  
}  
Inherited from body  
media="screen" [www.softwareishard.com/  
style.css:10](http://www.softwareishard.com/style.css:10)  
body {  
 padding: 0px;  
 text-align: left;  
 font-family: Verdana, Geneva, Arial, Helvetica, sans-serif;  
 font-size: small;  
 color: #222222;  
 line-height: 150%;  
 background: #4A525A;  
 background-color: #fff;  
}

3.

html body div#Wrapper

## Firefox

2. 1.

The screenshot shows the Firefox Developer Tools Inspector panel. The top navigation bar includes tabs for Inspector, Console, Debugger, Style Editor, Performance, Network, Rules, Computed, Fonts, Box Model, and Animations. The Inspector tab is selected and highlighted with a red circle (1). The left sidebar displays the page's DOM structure. A specific input element, `<input id="livesearch" class="search" type="search" ...>`, is selected and highlighted with a red box (2). The status bar at the bottom shows the URL `https://www.google.com/search?rlz=1C1GCEU_enUS904US904&q=site%3Aexample.com+example.com`. The main content area shows the element's properties and styles. Several UI elements are circled with red circles: the search icon in the top right (3), the search bar in the top right (4), the Rules tab (5), and the Box Model tab (6).

# CSS Modification

- Inspect styles applied to an element
- Understanding how cascading and inheritance works
- Editing CSS / Designing in the browser
- Ctrl+click on a color value and that will take you to the file where it is defined

The screenshot shows the Webkit Inspector interface with two main tabs: 'Rules' and 'Computed'. The 'Rules' tab displays the raw CSS code, while the 'Computed' tab shows the final computed styles.

**Rules Tab:**

```
element { inline
}
#ghx-header h2 { batch.css:88
  color: #333;
  font-size: 20px;
  font-weight: normal;
  line-height: 1.5;
  margin: 0px;
  word-wrap: break-word;
}

h1:first-child, h2:first-child, h3:first-child, h4:first-child, h5:first-child, h6:first-child { batch.css:23
  margin-top: 0px;
}

p:first-child, ul:first-child, ol:first-child, dl:first-child, h1:first-child, h2:first-child, h3:first-child, h4:first-child, h5:first-child, h6:first-child, blockquote: pre:first-child, form.aui:first-child, table.aui:first-child, .aui-tabs:first-child, .aui-panel:first-child, .aui-group:f {
  margin-top: 0px;
}
```

**Computed Tab:**

Style	Value
color	#333
font-family	Arial, sans-serif
font-size	20px
font-weight	400
line-height	30px
margin-bottom	0px
margin-left	0px
margin-right	0px
margin-top	0px
padding-bottom	0px
padding-left	0px
padding-right	0px
padding-top	0px
text-transform	none
word-wrap	break-word

Firefox

Styles Computed Event Listeners DOM Breakpoints Properties

Filter

```

element.style {
}

.index>div {
    padding: ▶ 2rem 0;
}

* {
    -moz-box-sizing: border-box;
    -webkit box-sizing: border-box;
    box-sizing: border-box;
}

div {
    display: block;
}

Inherited from header.index
.index {
    text-shadow: 1px 1px 1px ■#0c0d0d;
}

.index, .text-center {
    text-align: center;
}

Inherited from body#gfs
body, button, input, select, textarea {
    font-family: sans-serif;
    color: ■#e3d7bf;
}

```

## Chrome

Styles Computed Event Listeners DOM Breakpoints Properties

margin	-
border	-
padding	32
	709.094 × 293
	32
	-
	-

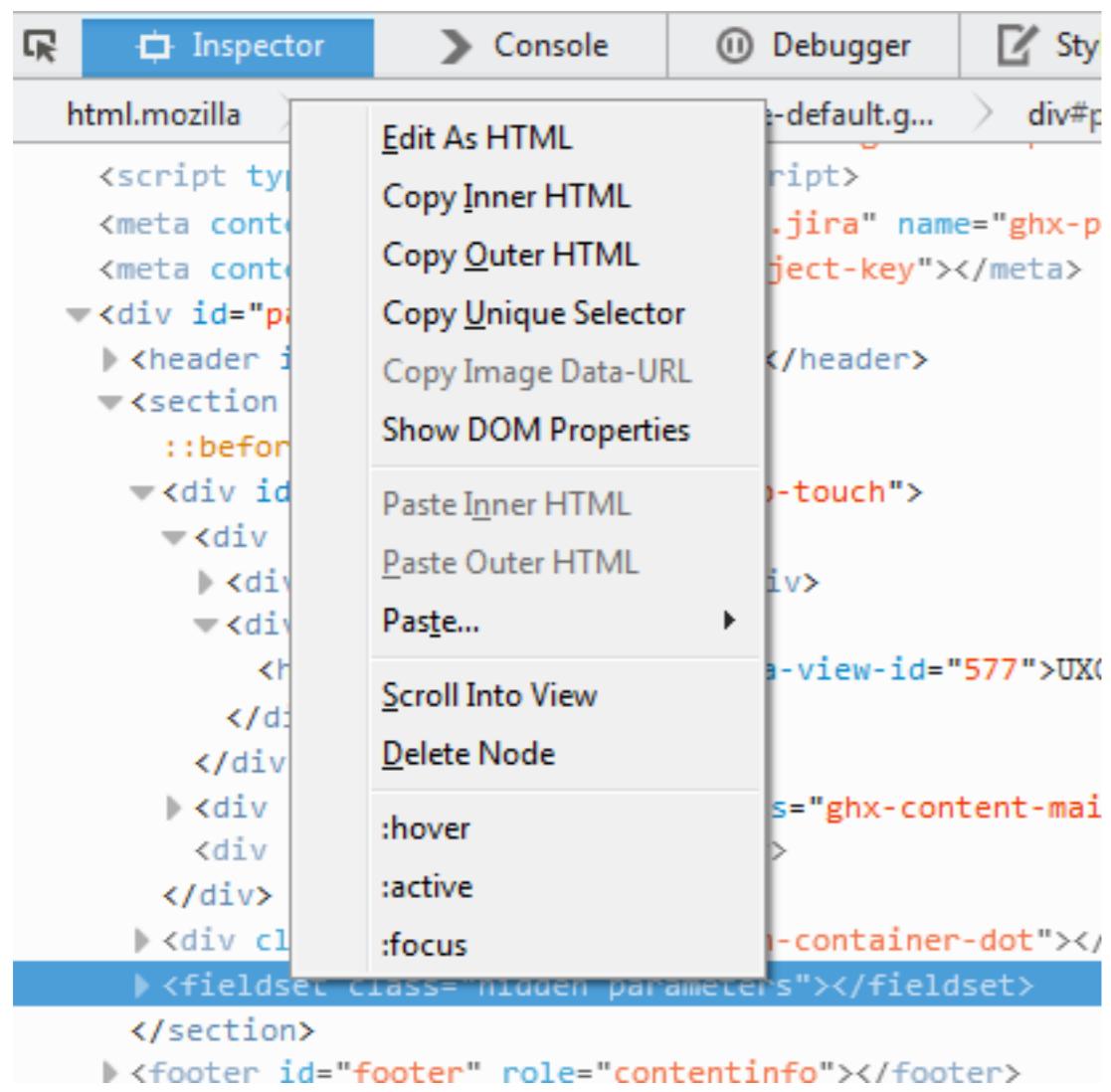
Filter  Show all

- ▶ **box-sizing** border-box
- ▶ **color** ■rgb(227, 215, 191)
- ▶ **display** block
- ▶ **font-family** sans-serif
- ▶ **font-size** 16px
- ▶ **font-style** normal
- ▶ **font-variant** normal
- ▶ **font-weight** normal
- ▶ **height** 357px
- ▶ **line-height** 24px
- ▶ **padding-bottom** 32px
- ▶ **padding-left** 0px
- ▶ **padding-right** 0px
- ▶ **padding-top** 32px
- ▶ **text-align** center

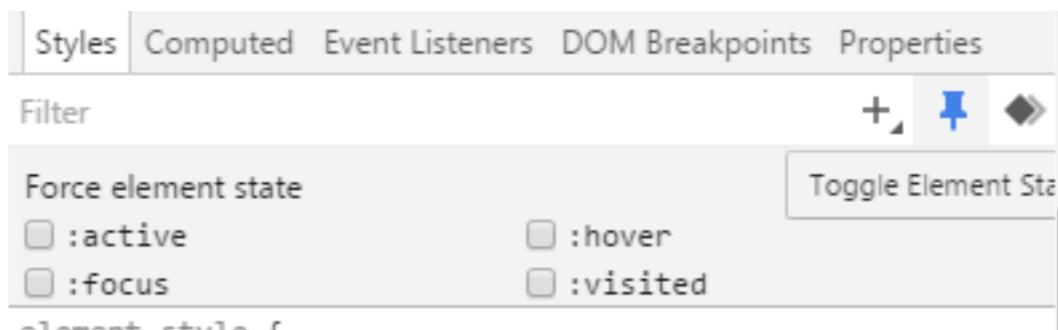
# CSS Modification

## Pseudo classes

### Firefox



### Chrome



# CSS Modification

The preprocessor workflow - Experimental in Chrome, but by default it shows the file names where the style comes from

## Chrome

The screenshot shows the Chrome DevTools Styles tab. The top navigation bar has 'Styles' selected. Below the navigation bar is a 'Filter' input field and three icons: a plus sign, a magnifying glass, and a diamond. The main area displays a list of CSS rules:

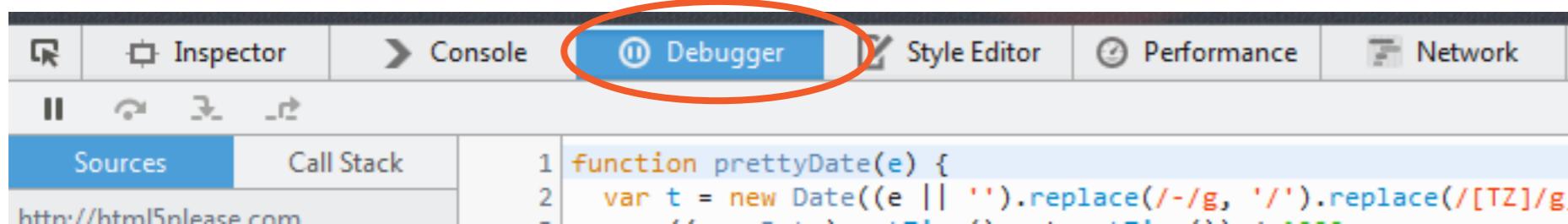
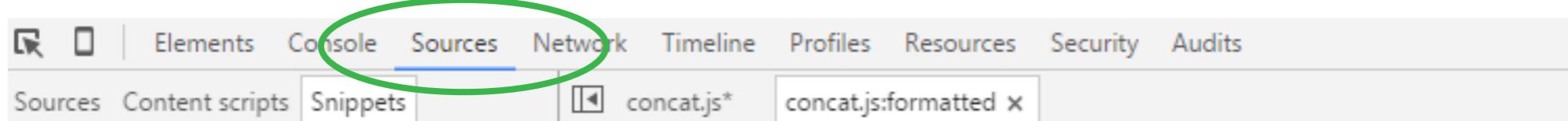
- `element.style { }`
- `#mc-embedded-subscribe-form { padding: 20px 0 5px 0; background: #ccc; text-align: center; }`  footer.scss:15
- `form { margin: 0; }`  normalize.scss:84
- `*, *::after, *::before { -moz-box-sizing: border-box; box-sizing: border-box; }`  normalize.scss:1
- `form { display: block; margin_top: 0em; }`  user agent stylesheet
- `Inherited from html`
- `html { font-family: 'Source Sans Pro', sans-serif; font-size: 17px; font-weight: 300; line-height: 1.45; }`  normalize.scss:169

# Debugging scripts

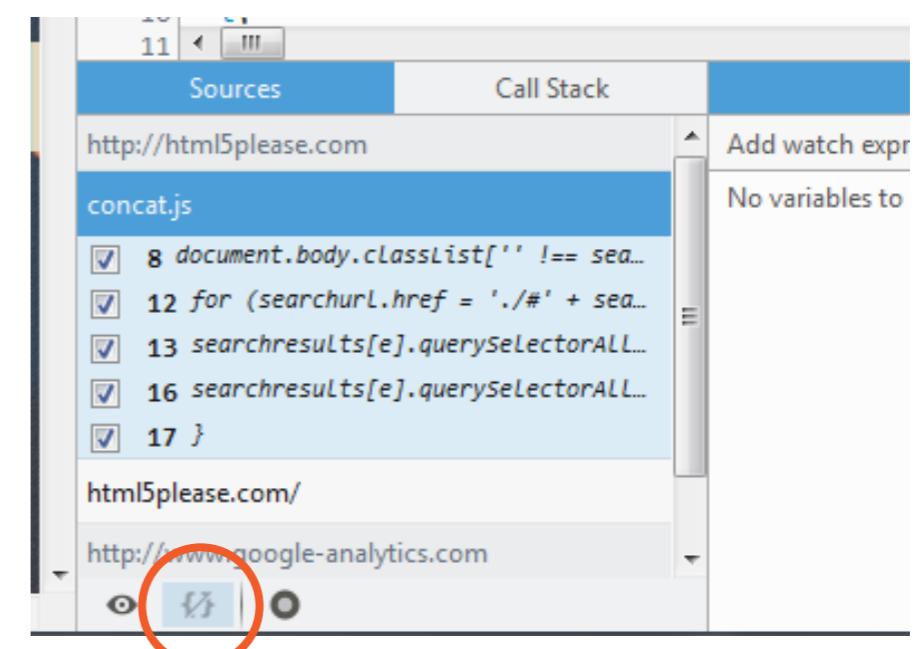
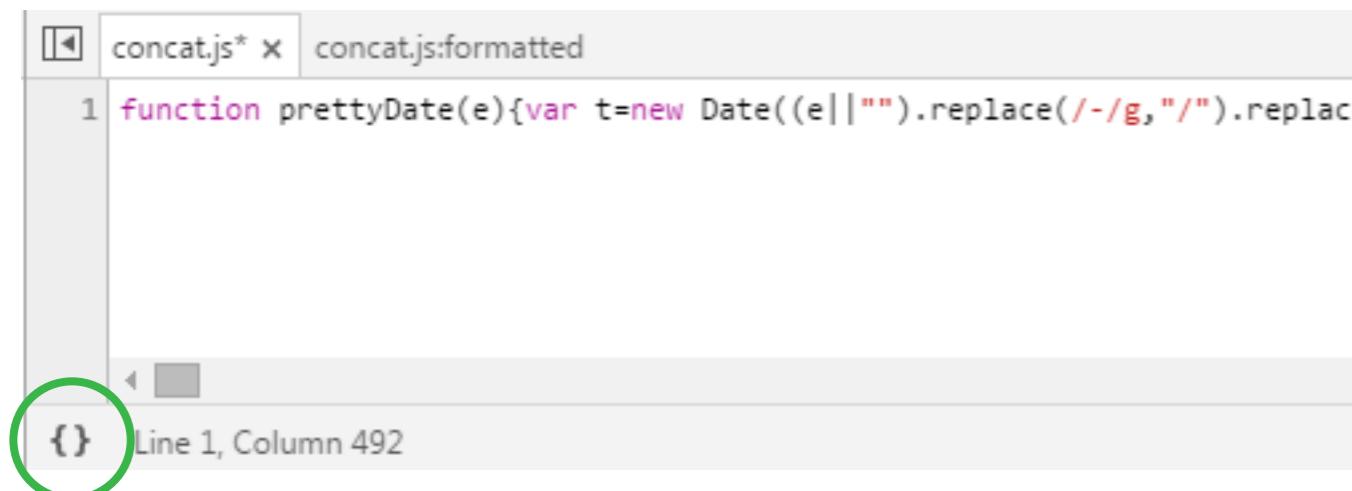
- Debugging stands for finding and fixing logic in our code.
- Pause running JavaScript at various points so you can determine its progress or examine its variable values.
- no need to use brute force alerts like: `alert("ok so far");`

# JS in the DevTools

## Sources tab in Chrome and Debugger tab in FF

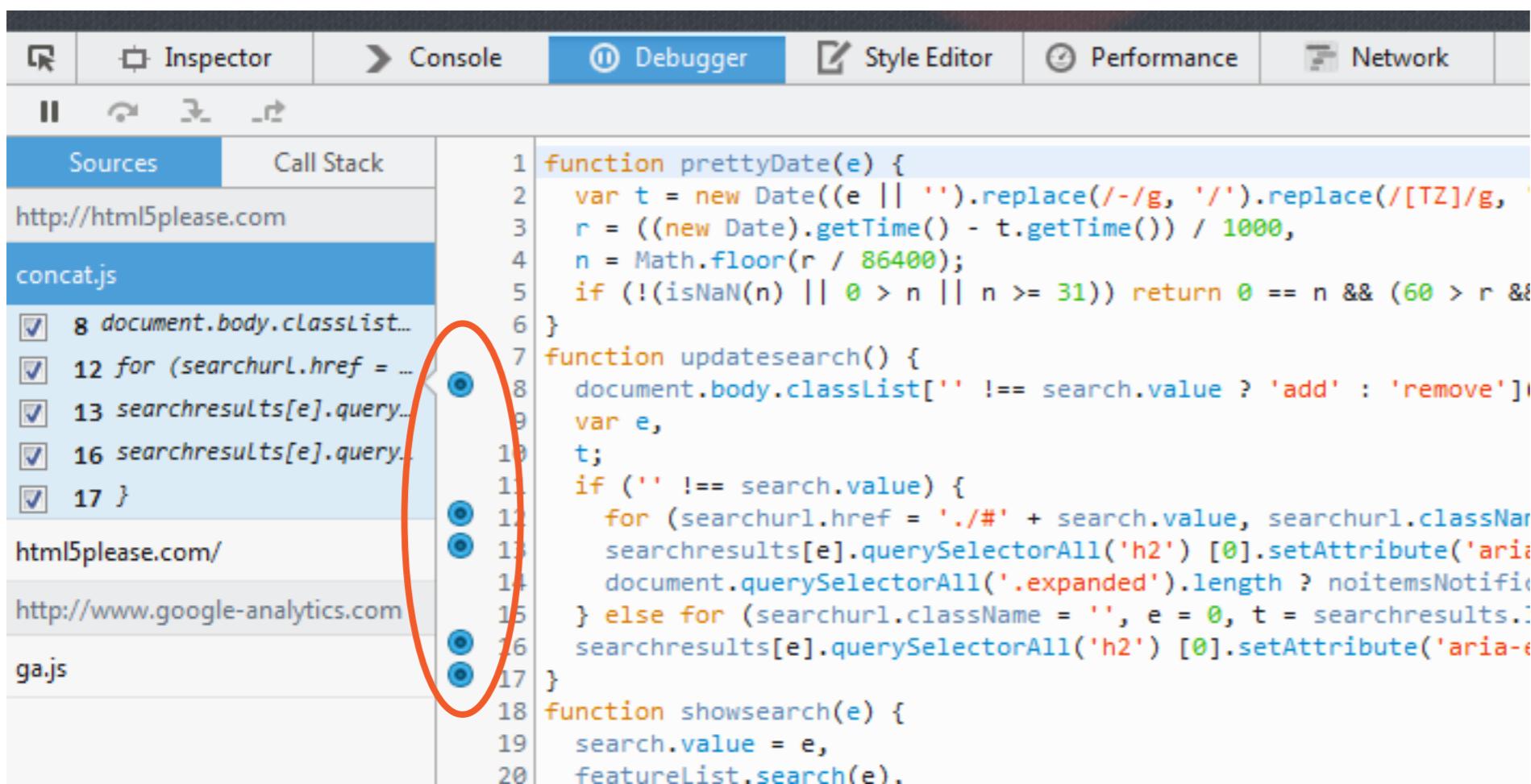


## Prettify the JS



# Breakpoints: Quintessential feature of debuggers

1. Find the line of source you want to investigate
2. Set a breakpoint
3. Run the program
4. Debugger stops the execution at that point, you poke around



The screenshot shows the 'Debugger' tab of a browser's developer tools. The left sidebar lists several scripts: 'concat.js' (selected), 'document.body.classList...', 'searchresults[e].query...', 'searchresults[e].query...', 'html5please.com/' (disabled), 'http://www.google-analytics.com', and 'ga.js'. The main pane displays the code of 'concat.js' with line numbers 1 through 20. A red oval highlights the first four lines of code, which define the `prettyDate` function. Each highlighted line has a blue circular breakpoint icon to its left.

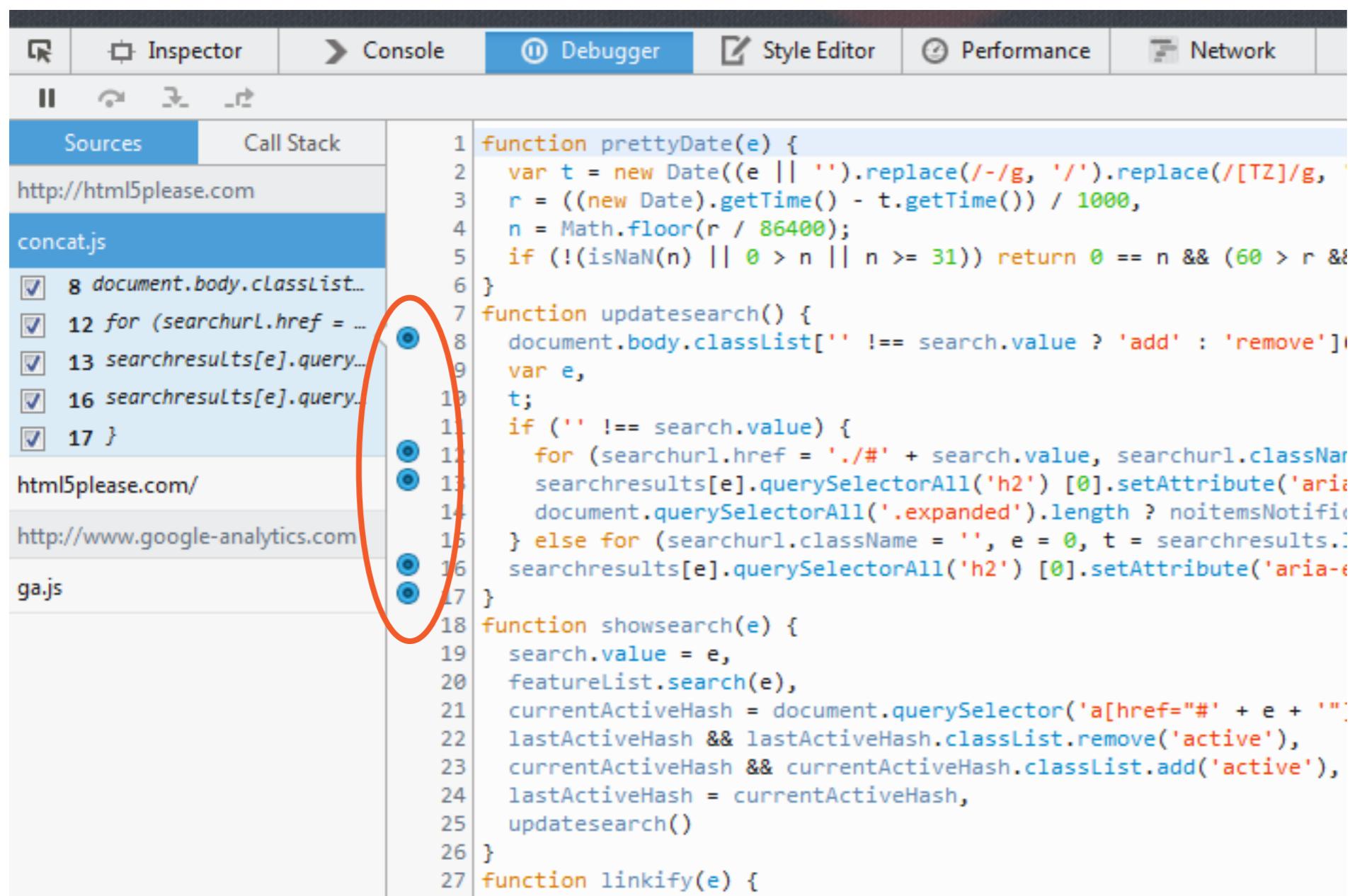
```
function prettyDate(e) {
  var t = new Date((e || '')).replace(/-/g, '/').replace(/[TZ]/g, '')
  r = ((new Date).getTime() - t.getTime()) / 1000,
  n = Math.floor(r / 86400);
  if (!(isNaN(n) || 0 > n || n >= 31)) return 0 == n && (60 > r &&
}
function updateSearch() {
  document.body.classList['' != search.value ? 'add' : 'remove']
  var e,
  t;
  if ('' != search.value) {
    for (searchurl.href = './#' + search.value, searchurl.className =
      searchresults[e].querySelectorAll('h2')[0].setAttribute('aria-
      document.querySelectorAll('.expanded').length ? noitemsNotific
    } else for (searchurl.className = '', e = 0, t = searchresults.
      searchresults[e].querySelectorAll('h2')[0].setAttribute('aria-
    }
  function showSearch(e) {
    search.value = e,
    featureList.search(e),
  }
}
```

# Types of breakpoints and how to use them

Locate a specific method with **ctrl+shift+o** in Chrome / **ctrl + d** in FF

Add breakpoint in the JS (or debugging inside-out)

Firefox



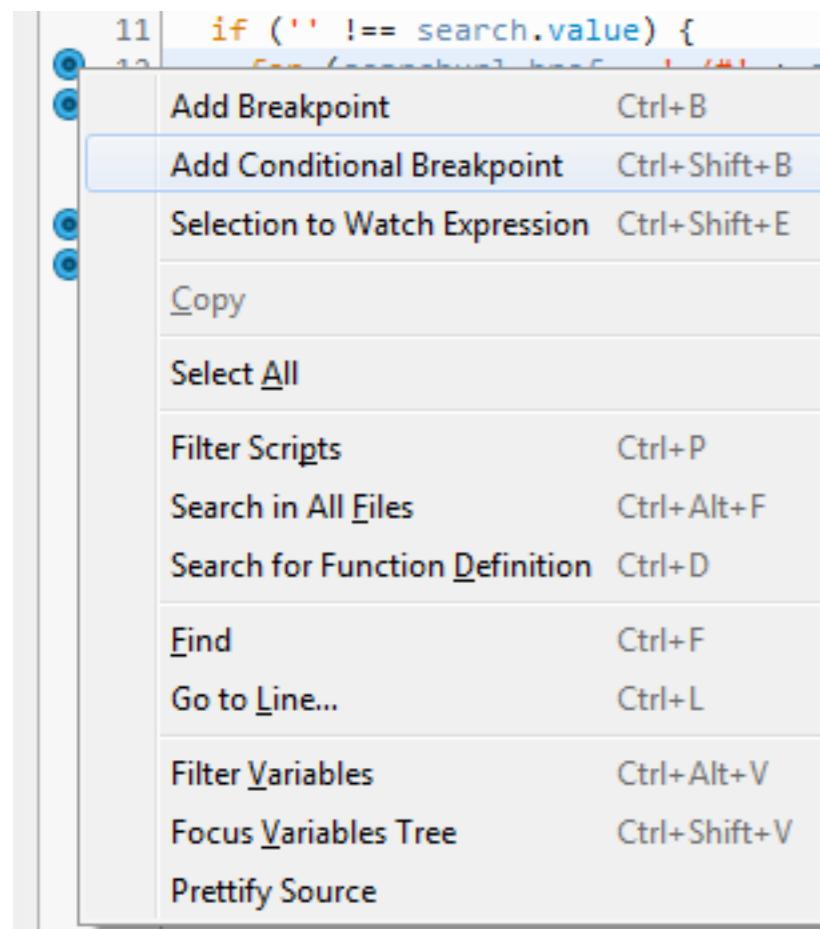
The screenshot shows the Firefox Developer Tools interface with the 'Debugger' tab selected. In the left sidebar under 'Sources', several files are listed: 'concat.js' (selected), 'document.body.classList...', 'searchresults[e].query...', 'searchresults[e].query...', and 'ga.js'. The 'concat.js' file is expanded, showing its source code. A red oval highlights a group of five circular breakpoints placed on the first few lines of code. The code itself is a snippet of JavaScript related to date formatting and search functionality.

```
function prettyDate(e) {
  var t = new Date((e || '')).replace(/-/g, '/').replace(/[TZ]/g, '')
  r = ((new Date).getTime() - t.getTime()) / 1000,
  n = Math.floor(r / 86400);
  if (!(isNaN(n) || 0 > n || n >= 31)) return 0 == n && (60 > r &&
}
function updateSearch() {
  document.body.classList['' != search.value ? 'add' : 'remove']
  var e,
  t;
  if ('' != search.value) {
    for (searchurl.href = './#' + search.value, searchurl.className =
      searchresults[e].querySelectorAll('h2')[0].setAttribute('aria-
      document.querySelectorAll('.expanded').length ? noitemsNotific
    } else for (searchurl.className = '', e = 0, t = searchresults.
      searchresults[e].querySelectorAll('h2')[0].setAttribute('aria-
    }
  }
  function showSearch(e) {
    search.value = e,
    featureList.search(e),
    currentActiveHash = document.querySelector('a[href="#' + e + '"]
    lastActiveHash && lastActiveHash.classList.remove('active'),
    currentActiveHash && currentActiveHash.classList.add('active'),
    lastActiveHash = currentActiveHash,
    updateSearch()
  }
  function linkify(e) {
```

# Types of breakpoints and how to use them

## Add conditional breakpoint

### Firefox

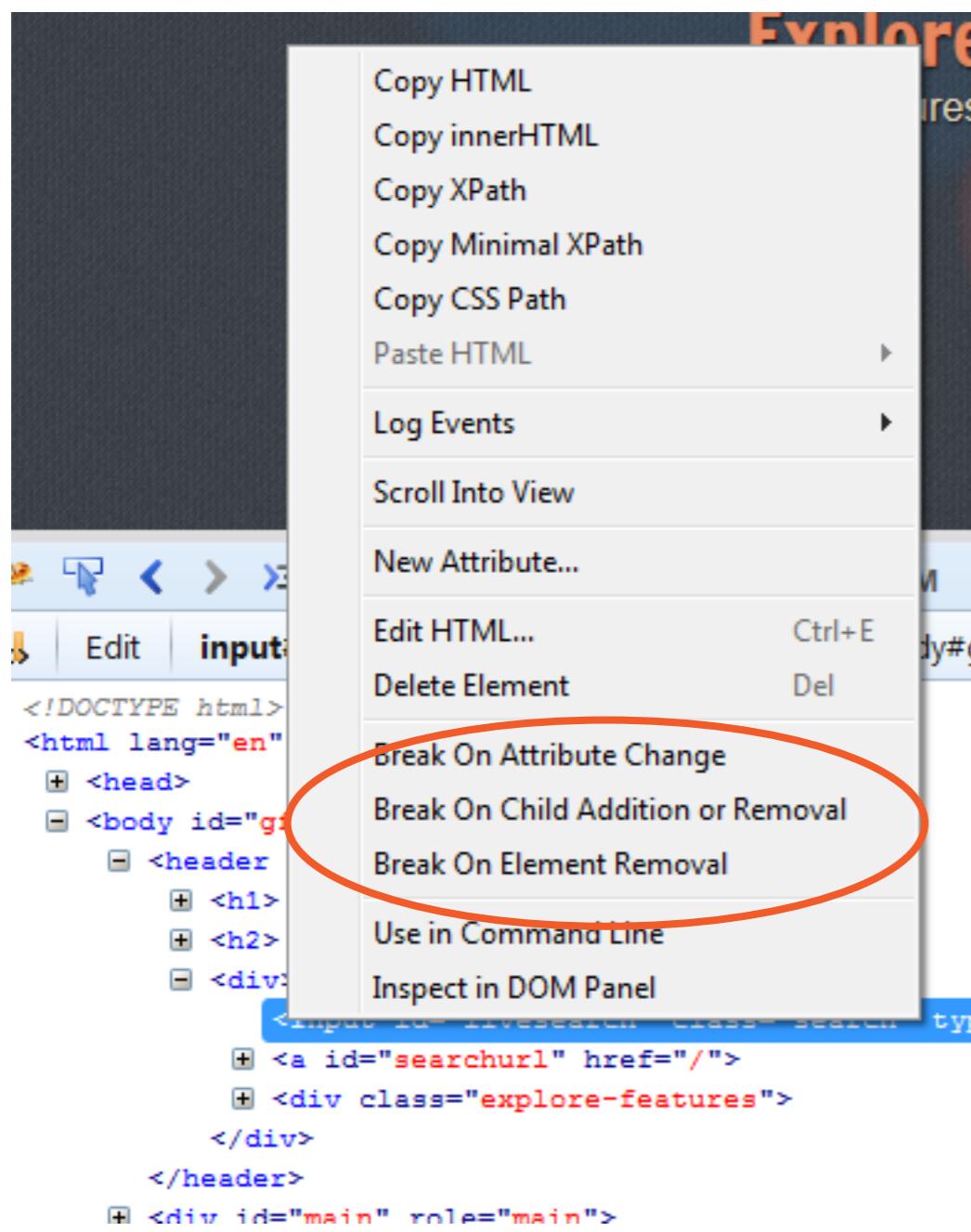


```
11 if ('' !== search.value) {  
12   e = document.querySelector('h2').classList.add('active');  
13   search.value, searchurl.classList[0].setAttribute('aria-expanded', true);  
14 }  
15  
16 function prettyDate(e) {  
17   e.innerHTML = Date.now() - e.getAttribute('time') / 1000 / 60 / 60 / 24 / 365;  
18   e.innerHTML = e.innerHTML + ' days ago';  
19 }  
20  
21 function updateSearch() {  
22   document.body.classList['' !== search.value] ?  
23     document.body.classList.remove('search-active');  
24   else document.body.classList.add('search-active');  
25 }  
26  
27 var e, t;  
28  
29 if ('' !== search.value) {  
30   e = document.querySelector('a[href="#' + e + '"]').classList.remove('active');  
31   Hash.classList.add('active');  
32 }  
33  
34 function e(t, r, n) {  
35   var s = e.resolve(t);  
36   s.then(r, n);  
37 }  
38  
39 e.resolve = function (t) {  
40   return new Promise((r, n) => {  
41     t(r, n);  
42   });  
43 }  
44  
45 e.all = function (t) {  
46   return new Promise((r, n) => {  
47     t(r, n);  
48   });  
49 }  
50  
51 e.all = function (t) {  
52   return new Promise((r, n) => {  
53     t(r, n);  
54   });  
55 }  
56  
57 e.all = function (t) {  
58   return new Promise((r, n) => {  
59     t(r, n);  
60   });  
61 }  
62  
63 e.all = function (t) {  
64   return new Promise((r, n) => {  
65     t(r, n);  
66   });  
67 }  
68  
69 e.all = function (t) {  
70   return new Promise((r, n) => {  
71     t(r, n);  
72   });  
73 }  
74  
75 e.all = function (t) {  
76   return new Promise((r, n) => {  
77     t(r, n);  
78   });  
79 }  
80  
81 e.all = function (t) {  
82   return new Promise((r, n) => {  
83     t(r, n);  
84   });  
85 }  
86  
87 e.all = function (t) {  
88   return new Promise((r, n) => {  
89     t(r, n);  
90   });  
91 }  
92  
93 e.all = function (t) {  
94   return new Promise((r, n) => {  
95     t(r, n);  
96   });  
97 }  
98  
99 e.all = function (t) {  
100   return new Promise((r, n) => {  
101     t(r, n);  
102   });  
103 }  
104  
105 e.all = function (t) {  
106   return new Promise((r, n) => {  
107     t(r, n);  
108   });  
109 }  
110  
111 e.all = function (t) {  
112   return new Promise((r, n) => {  
113     t(r, n);  
114   });  
115 }  
116  
117 e.all = function (t) {  
118   return new Promise((r, n) => {  
119     t(r, n);  
120   });  
121 }  
122  
123 e.all = function (t) {  
124   return new Promise((r, n) => {  
125     t(r, n);  
126   });  
127 }  
128  
129 e.all = function (t) {  
130   return new Promise((r, n) => {  
131     t(r, n);  
132   });  
133 }  
134  
135 e.all = function (t) {  
136   return new Promise((r, n) => {  
137     t(r, n);  
138   });  
139 }  
140  
141 e.all = function (t) {  
142   return new Promise((r, n) => {  
143     t(r, n);  
144   });  
145 }  
146  
147 e.all = function (t) {  
148   return new Promise((r, n) => {  
149     t(r, n);  
150   });  
151 }  
152  
153 e.all = function (t) {  
154   return new Promise((r, n) => {  
155     t(r, n);  
156   });  
157 }  
158  
159 e.all = function (t) {  
160   return new Promise((r, n) => {  
161     t(r, n);  
162   });  
163 }  
164  
165 e.all = function (t) {  
166   return new Promise((r, n) => {  
167     t(r, n);  
168   });  
169 }  
170  
171 e.all = function (t) {  
172   return new Promise((r, n) => {  
173     t(r, n);  
174   });  
175 }  
176  
177 e.all = function (t) {  
178   return new Promise((r, n) => {  
179     t(r, n);  
180   });  
181 }  
182  
183 e.all = function (t) {  
184   return new Promise((r, n) => {  
185     t(r, n);  
186   });  
187 }  
188  
189 e.all = function (t) {  
190   return new Promise((r, n) => {  
191     t(r, n);  
192   });  
193 }  
194  
195 e.all = function (t) {  
196   return new Promise((r, n) => {  
197     t(r, n);  
198   });  
199 }  
200  
201 e.all = function (t) {  
202   return new Promise((r, n) => {  
203     t(r, n);  
204   });  
205 }  
206  
207 e.all = function (t) {  
208   return new Promise((r, n) => {  
209     t(r, n);  
210   });  
211 }  
212  
213 e.all = function (t) {  
214   return new Promise((r, n) => {  
215     t(r, n);  
216   });  
217 }  
218  
219 e.all = function (t) {  
220   return new Promise((r, n) => {  
221     t(r, n);  
222   });  
223 }  
224  
225 e.all = function (t) {  
226   return new Promise((r, n) => {  
227     t(r, n);  
228   });  
229 }  
230  
231 e.all = function (t) {  
232   return new Promise((r, n) => {  
233     t(r, n);  
234   });  
235 }  
236  
237 e.all = function (t) {  
238   return new Promise((r, n) => {  
239     t(r, n);  
240   });  
241 }  
242  
243 e.all = function (t) {  
244   return new Promise((r, n) => {  
245     t(r, n);  
246   });  
247 }  
248  
249 e.all = function (t) {  
250   return new Promise((r, n) => {  
251     t(r, n);  
252   });  
253 }  
254  
255 e.all = function (t) {  
256   return new Promise((r, n) => {  
257     t(r, n);  
258   });  
259 }  
260  
261 e.all = function (t) {  
262   return new Promise((r, n) => {  
263     t(r, n);  
264   });  
265 }  
266  
267 e.all = function (t) {  
268   return new Promise((r, n) => {  
269     t(r, n);  
270   });  
271 }  
272  
273 e.all = function (t) {  
274   return new Promise((r, n) => {  
275     t(r, n);  
276   });  
277 }  
278  
279 e.all = function (t) {  
280   return new Promise((r, n) => {  
281     t(r, n);  
282   });  
283 }  
284  
285 e.all = function (t) {  
286   return new Promise((r, n) => {  
287     t(r, n);  
288   });  
289 }  
290  
291 e.all = function (t) {  
292   return new Promise((r, n) => {  
293     t(r, n);  
294   });  
295 }  
296  
297 e.all = function (t) {  
298   return new Promise((r, n) => {  
299     t(r, n);  
300   });  
301 }  
302  
303 e.all = function (t) {  
304   return new Promise((r, n) => {  
305     t(r, n);  
306   });  
307 }  
308  
309 e.all = function (t) {  
310   return new Promise((r, n) => {  
311     t(r, n);  
312   });  
313 }  
314  
315 e.all = function (t) {  
316   return new Promise((r, n) => {  
317     t(r, n);  
318   });  
319 }  
320  
321 e.all = function (t) {  
322   return new Promise((r, n) => {  
323     t(r, n);  
324   });  
325 }  
326  
327 e.all = function (t) {  
328   return new Promise((r, n) => {  
329     t(r, n);  
330   });  
331 }  
332  
333 e.all = function (t) {  
334   return new Promise((r, n) => {  
335     t(r, n);  
336   });  
337 }  
338  
339 e.all = function (t) {  
340   return new Promise((r, n) => {  
341     t(r, n);  
342   });  
343 }  
344  
345 e.all = function (t) {  
346   return new Promise((r, n) => {  
347     t(r, n);  
348   });  
349 }  
350  
351 e.all = function (t) {  
352   return new Promise((r, n) => {  
353     t(r, n);  
354   });  
355 }  
356  
357 e.all = function (t) {  
358   return new Promise((r, n) => {  
359     t(r, n);  
360   });  
361 }  
362  
363 e.all = function (t) {  
364   return new Promise((r, n) => {  
365     t(r, n);  
366   });  
367 }  
368  
369 e.all = function (t) {  
370   return new Promise((r, n) => {  
371     t(r, n);  
372   });  
373 }  
374  
375 e.all = function (t) {  
376   return new Promise((r, n) => {  
377     t(r, n);  
378   });  
379 }  
380  
381 e.all = function (t) {  
382   return new Promise((r, n) => {  
383     t(r, n);  
384   });  
385 }  
386  
387 e.all = function (t) {  
388   return new Promise((r, n) => {  
389     t(r, n);  
390   });  
391 }  
392  
393 e.all = function (t) {  
394   return new Promise((r, n) => {  
395     t(r, n);  
396   });  
397 }  
398  
399 e.all = function (t) {  
400   return new Promise((r, n) => {  
401     t(r, n);  
402   });  
403 }  
404  
405 e.all = function (t) {  
406   return new Promise((r, n) => {  
407     t(r, n);  
408   });  
409 }  
410  
411 e.all = function (t) {  
412   return new Promise((r, n) => {  
413     t(r, n);  
414   });  
415 }  
416  
417 e.all = function (t) {  
418   return new Promise((r, n) => {  
419     t(r, n);  
420   });  
421 }  
422  
423 e.all = function (t) {  
424   return new Promise((r, n) => {  
425     t(r, n);  
426   });  
427 }  
428  
429 e.all = function (t) {  
430   return new Promise((r, n) => {  
431     t(r, n);  
432   });  
433 }  
434  
435 e.all = function (t) {  
436   return new Promise((r, n) => {  
437     t(r, n);  
438   });  
439 }  
440  
441 e.all = function (t) {  
442   return new Promise((r, n) => {  
443     t(r, n);  
444   });  
445 }  
446  
447 e.all = function (t) {  
448   return new Promise((r, n) => {  
449     t(r, n);  
450   });  
451 }  
452  
453 e.all = function (t) {  
454   return new Promise((r, n) => {  
455     t(r, n);  
456   });  
457 }  
458  
459 e.all = function (t) {  
460   return new Promise((r, n) => {  
461     t(r, n);  
462   });  
463 }  
464  
465 e.all = function (t) {  
466   return new Promise((r, n) => {  
467     t(r, n);  
468   });  
469 }  
470  
471 e.all = function (t) {  
472   return new Promise((r, n) => {  
473     t(r, n);  
474   });  
475 }  
476  
477 e.all = function (t) {  
478   return new Promise((r, n) => {  
479     t(r, n);  
480   });  
481 }  
482  
483 e.all = function (t) {  
484   return new Promise((r, n) => {  
485     t(r, n);  
486   });  
487 }  
488  
489 e.all = function (t) {  
490   return new Promise((r, n) => {  
491     t(r, n);  
492   });  
493 }  
494  
495 e.all = function (t) {  
496   return new Promise((r, n) => {  
497     t(r, n);  
498   });  
499 }  
500  
501 e.all = function (t) {  
502   return new Promise((r, n) => {  
503     t(r, n);  
504   });  
505 }  
506  
507 e.all = function (t) {  
508   return new Promise((r, n) => {  
509     t(r, n);  
510   });  
511 }  
512  
513 e.all = function (t) {  
514   return new Promise((r, n) => {  
515     t(r, n);  
516   });  
517 }  
518  
519 e.all = function (t) {  
520   return new Promise((r, n) => {  
521     t(r, n);  
522   });  
523 }  
524  
525 e.all = function (t) {  
526   return new Promise((r, n) => {  
527     t(r, n);  
528   });  
529 }  
530  
531 e.all = function (t) {  
532   return new Promise((r, n) => {  
533     t(r, n);  
534   });  
535 }  
536  
537 e.all = function (t) {  
538   return new Promise((r, n) => {  
539     t(r, n);  
540   });  
541 }  
542  
543 e.all = function (t) {  
544   return new Promise((r, n) => {  
545     t(r, n);  
546   });  
547 }  
548  
549 e.all = function (t) {  
550   return new Promise((r, n) => {  
551     t(r, n);  
552   });  
553 }  
554  
555 e.all = function (t) {  
556   return new Promise((r, n) => {  
557     t(r, n);  
558   });  
559 }  
560  
561 e.all = function (t) {  
562   return new Promise((r, n) => {  
563     t(r, n);  
564   });  
565 }  
566  
567 e.all = function (t) {  
568   return new Promise((r, n) => {  
569     t(r, n);  
570   });  
571 }  
572  
573 e.all = function (t) {  
574   return new Promise((r, n) => {  
575     t(r, n);  
576   });  
577 }  
578  
579 e.all = function (t) {  
580   return new Promise((r, n) => {  
581     t(r, n);  
582   });  
583 }  
584  
585 e.all = function (t) {  
586   return new Promise((r, n) => {  
587     t(r, n);  
588   });  
589 }  
590  
591 e.all = function (t) {  
592   return new Promise((r, n) => {  
593     t(r, n);  
594   });  
595 }  
596  
597 e.all = function (t) {  
598   return new Promise((r, n) => {  
599     t(r, n);  
600   });  
601 }  
602  
603 e.all = function (t) {  
604   return new Promise((r, n) => {  
605     t(r, n);  
606   });  
607 }  
608  
609 e.all = function (t) {  
610   return new Promise((r, n) => {  
611     t(r, n);  
612   });  
613 }  
614  
615 e.all = function (t) {  
616   return new Promise((r, n) => {  
617     t(r, n);  
618   });  
619 }  
620  
621 e.all = function (t) {  
622   return new Promise((r, n) => {  
623     t(r, n);  
624   });  
625 }  
626  
627 e.all = function (t) {  
628   return new Promise((r, n) => {  
629     t(r, n);  
630   });  
631 }  
632  
633 e.all = function (t) {  
634   return new Promise((r, n) => {  
635     t(r, n);  
636   });  
637 }  
638  
639 e.all = function (t) {  
640   return new Promise((r, n) => {  
641     t(r, n);  
642   });  
643 }  
644  
645 e.all = function (t) {  
646   return new Promise((r, n) => {  
647     t(r, n);  
648   });  
649 }  
650  
651 e.all = function (t) {  
652   return new Promise((r, n) => {  
653     t(r, n);  
654   });  
655 }  
656  
657 e.all = function (t) {  
658   return new Promise((r, n) => {  
659     t(r, n);  
660   });  
661 }  
662  
663 e.all = function (t) {  
664   return new Promise((r, n) => {  
665     t(r, n);  
666   });  
667 }  
668  
669 e.all = function (t) {  
670   return new Promise((r, n) => {  
671     t(r, n);  
672   });  
673 }  
674  
675 e.all = function (t) {  
676   return new Promise((r, n) => {  
677     t(r, n);  
678   });  
679 }  
680  
681 e.all = function (t) {  
682   return new Promise((r, n) => {  
683     t(r, n);  
684   });  
685 }  
686  
687 e.all = function (t) {  
688   return new Promise((r, n) => {  
689     t(r, n);  
690   });  
691 }  
692  
693 e.all = function (t) {  
694   return new Promise((r, n) => {  
695     t(r, n);  
696   });  
697 }  
698  
699 e.all = function (t) {  
700   return new Promise((r, n) => {  
701     t(r, n);  
702   });  
703 }  
704  
705 e.all = function (t) {  
706   return new Promise((r, n) => {  
707     t(r, n);  
708   });  
709 }  
710  
711 e.all = function (t) {  
712   return new Promise((r, n) => {  
713     t(r, n);  
714   });  
715 }  
716  
717 e.all = function (t) {  
718   return new Promise((r, n) => {  
719     t(r, n);  
720   });  
721 }  
722  
723 e.all = function (t) {  
724   return new Promise((r, n) => {  
725     t(r, n);  
726   });  
727 }  
728  
729 e.all = function (t) {  
730   return new Promise((r, n) => {  
731     t(r, n);  
732   });  
733 }  
734  
735 e.all = function (t) {  
736   return new Promise((r, n) => {  
737     t(r, n);  
738   });  
739 }  
740  
741 e.all = function (t) {  
742   return new Promise((r, n) => {  
743     t(r, n);  
744   });  
745 }  
746  
747 e.all = function (t) {  
748   return new Promise((r, n) => {  
749     t(r, n);  
750   });  
751 }  
752  
753 e.all = function (t) {  
754   return new Promise((r, n) => {  
755     t(r, n);  
756   });  
757 }  
758  
759 e.all = function (t) {  
760   return new Promise((r, n) => {  
761     t(r, n);  
762   });  
763 }  
764  
765 e.all = function (t) {  
766   return new Promise((r, n) => {  
767     t(r, n);  
768   });  
769 }  
770  
771 e.all = function (t) {  
772   return new Promise((r, n) => {  
773     t(r, n);  
774   });  
775 }  
776  
777 e.all = function (t) {  
778   return new Promise((r, n) => {  
779     t(r, n);  
780   });  
781 }  
782  
783 e.all = function (t) {  
784   return new Promise((r, n) => {  
785     t(r, n);  
786   });  
787 }  
788  
789 e.all = function (
```

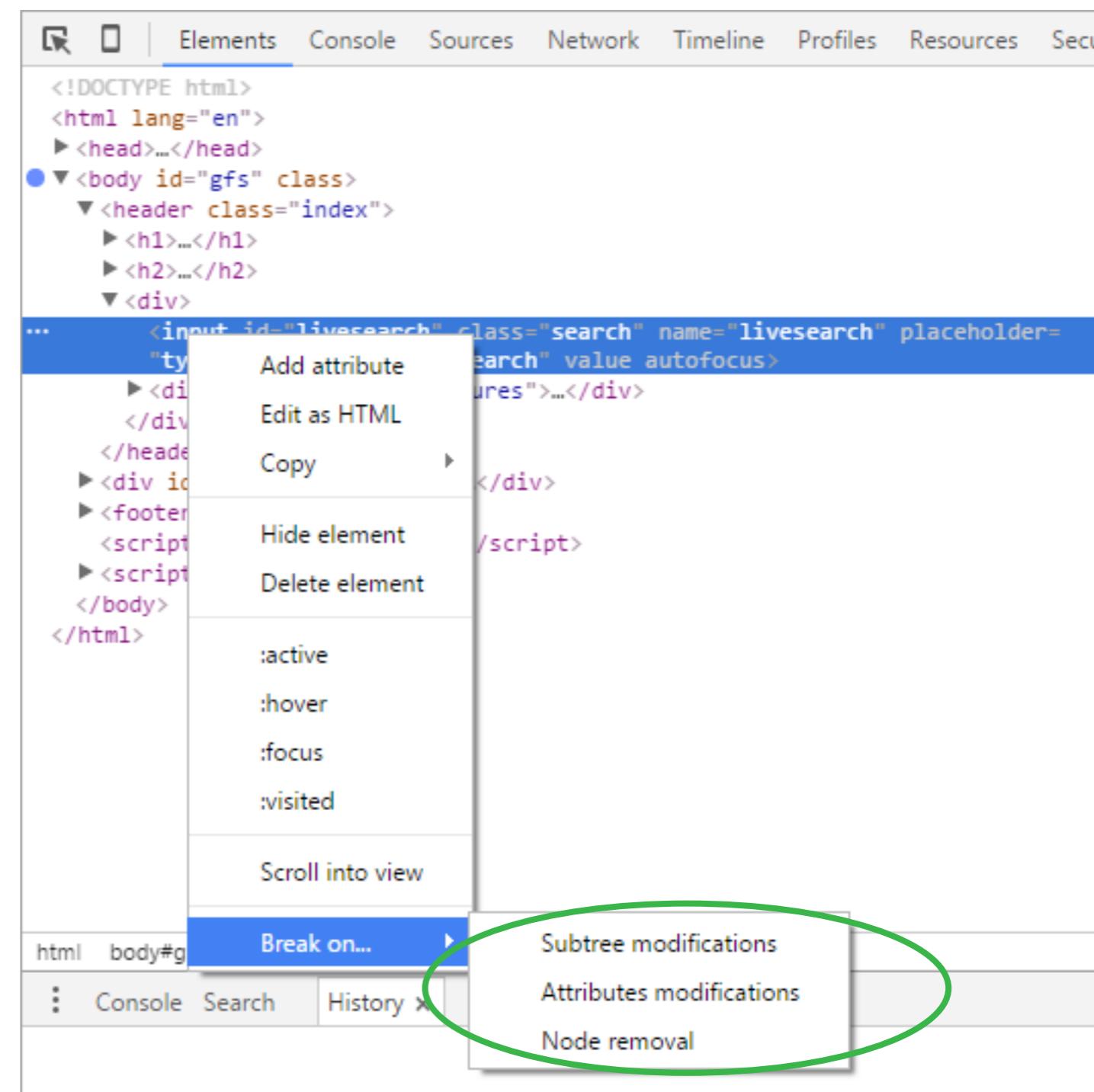
# Types of breakpoints and how to use them

## Add DOM breakpoint (or debugging outside-in)

Firefox

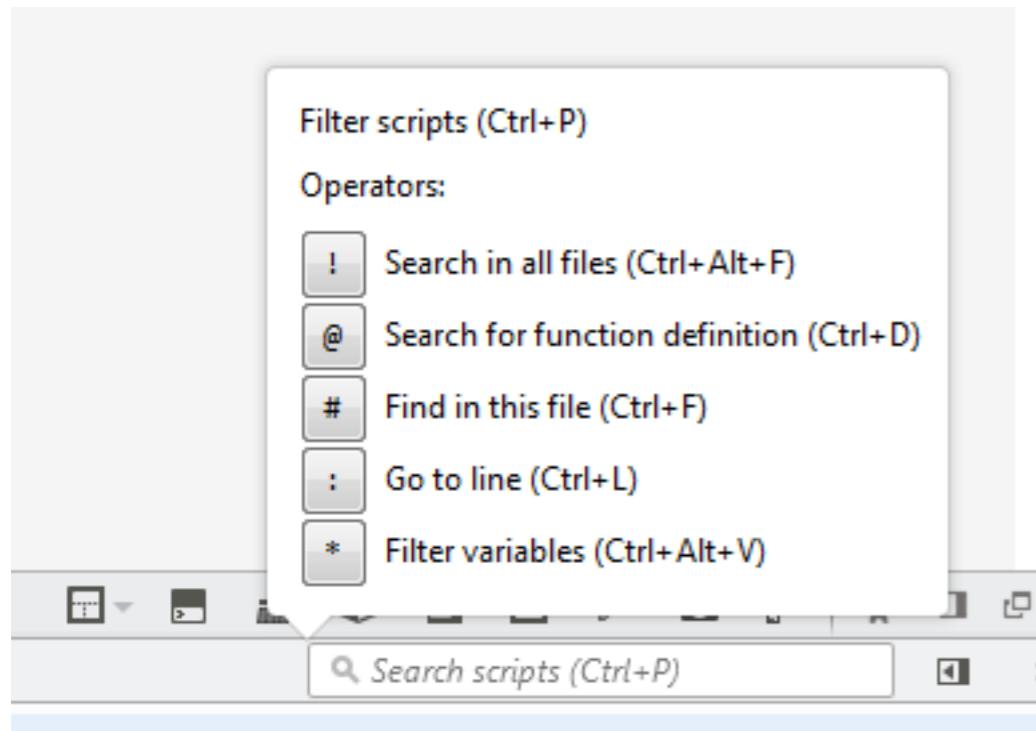


Chrome



# Shortcuts

## Firefox



## Chrome

Ctrl+o gives the list of files that are available for that particular project (also css)

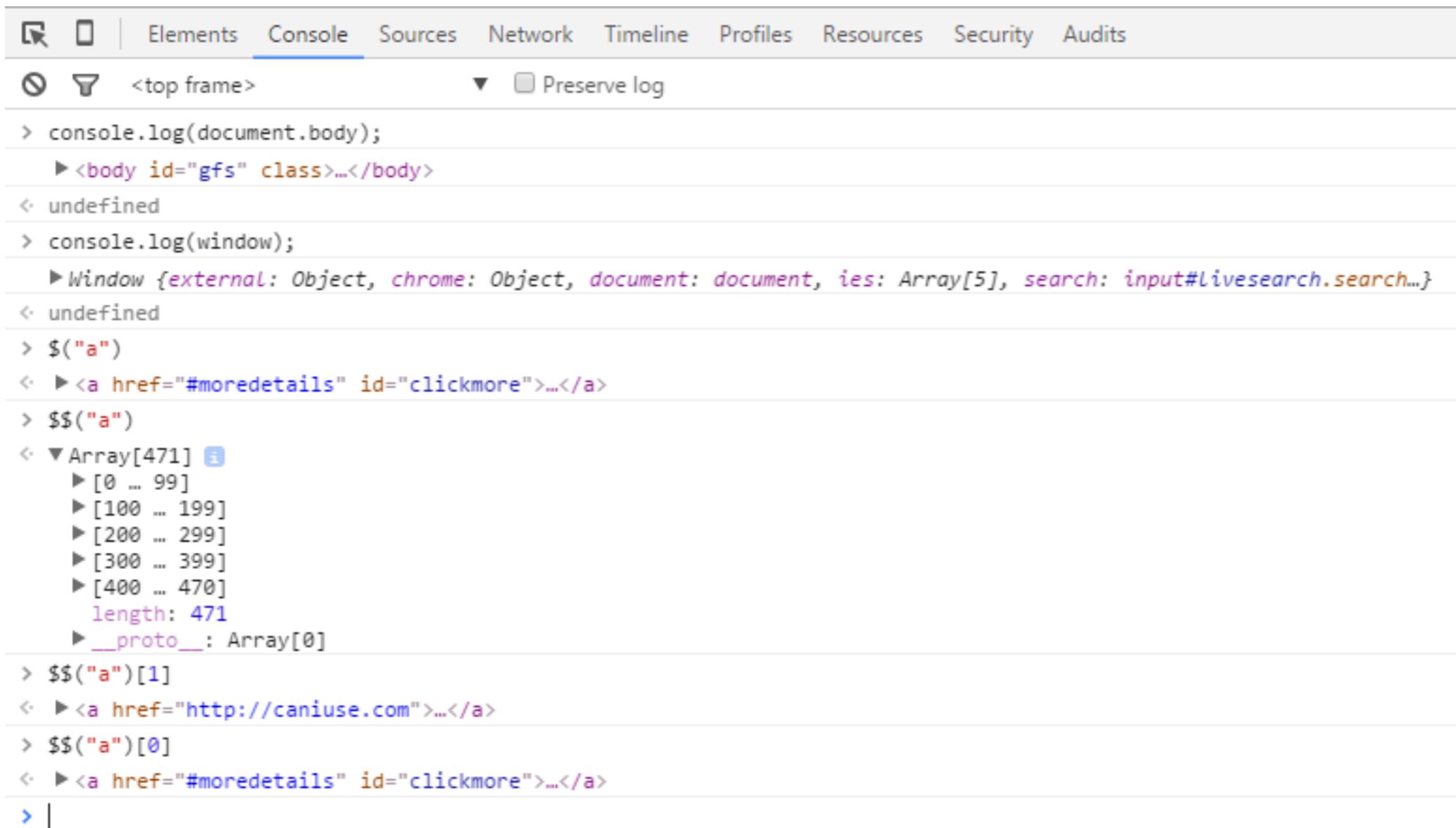
Ctrl+l go to line

Ctrl+f= searches for every instance of a method or variable name

# Using the console

Console lets you use standard JavaScript statements and Console-specific commands while a page is live in the browser to help you debug the page.

## Chrome



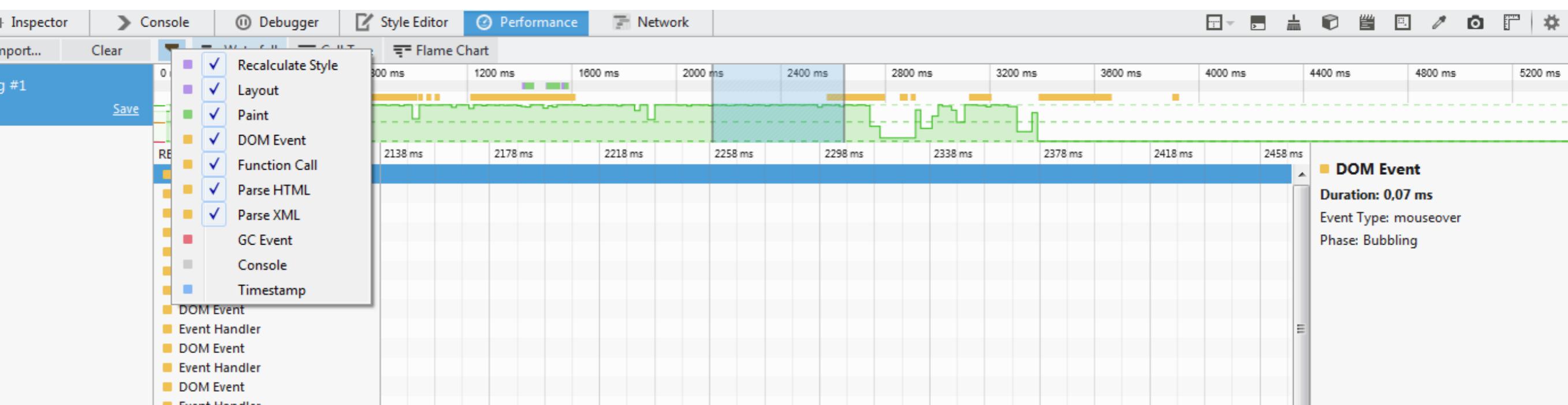
The screenshot shows the Chrome DevTools interface with the 'Console' tab selected. The console output is as follows:

```
> console.log(document.body);
▶ <body id="gfs" class="...</body>
<- undefined
> console.log(window);
▶ Window {external: Object, chrome: Object, document: document, ies: Array[5], search: input#livesearch.search...}
<- undefined
> $("a")
<- ▶ <a href="#moredetails" id="clickmore">...</a>
> $$("a")
<- ▶ Array[471] ⓘ
▶ [0 ... 99]
▶ [100 ... 199]
▶ [200 ... 299]
▶ [300 ... 399]
▶ [400 ... 470]
  length: 471
▶ __proto__: Array[0]
> $$("a")[1]
<- ▶ <a href="http://caniuse.com">...</a>
> $$("a")[0]
<- ▶ <a href="#moredetails" id="clickmore">...</a>
> |
```

# Performance

Every marker in the timeline represents the type of work a browser is doing at a particular point of time

## Firefox



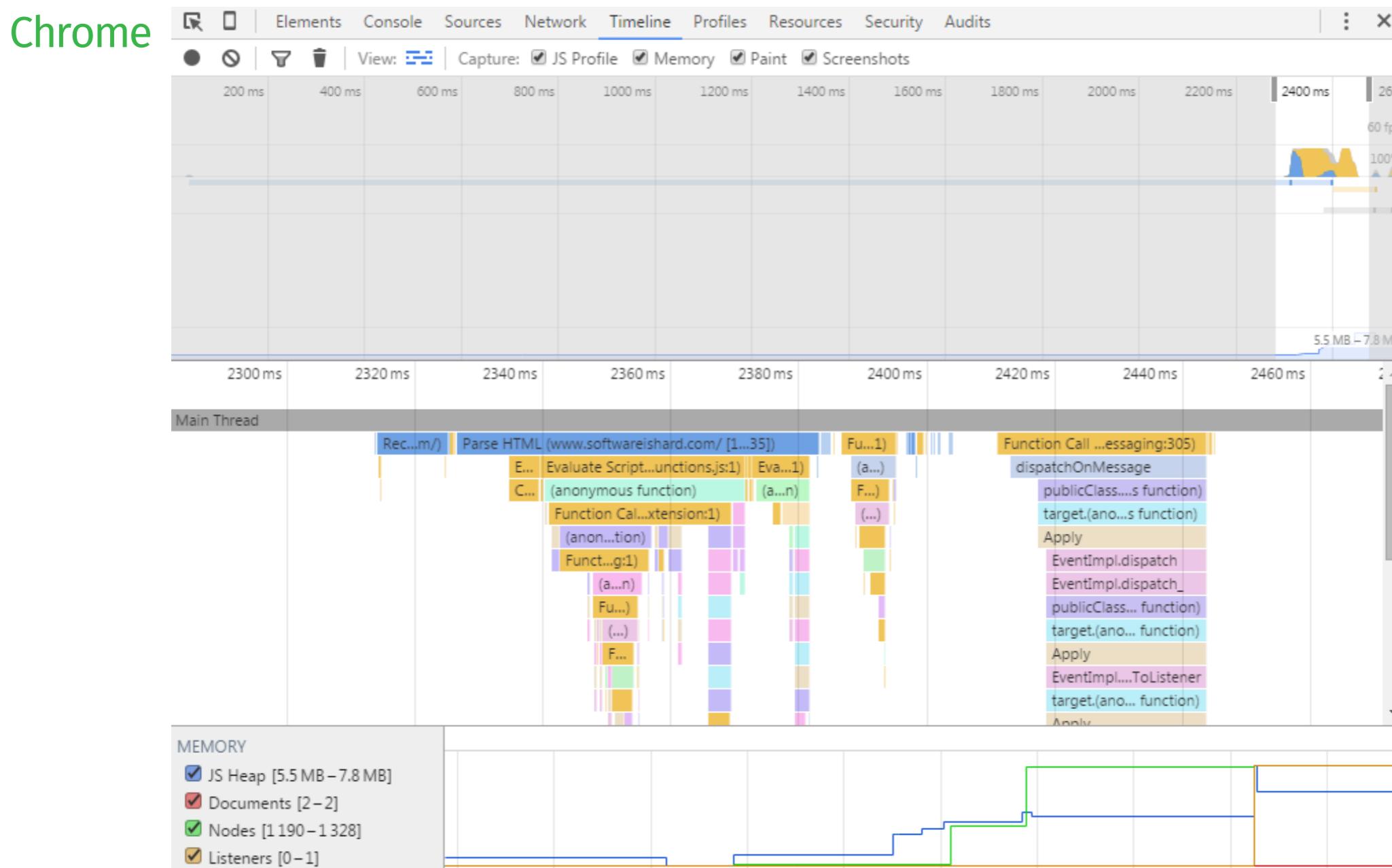
Styling and layout

Actual painting on the screen

Computation, typically JS

# Performance

Every marker in the timeline represents the type of work a browser is doing at a particular point of time



# Network

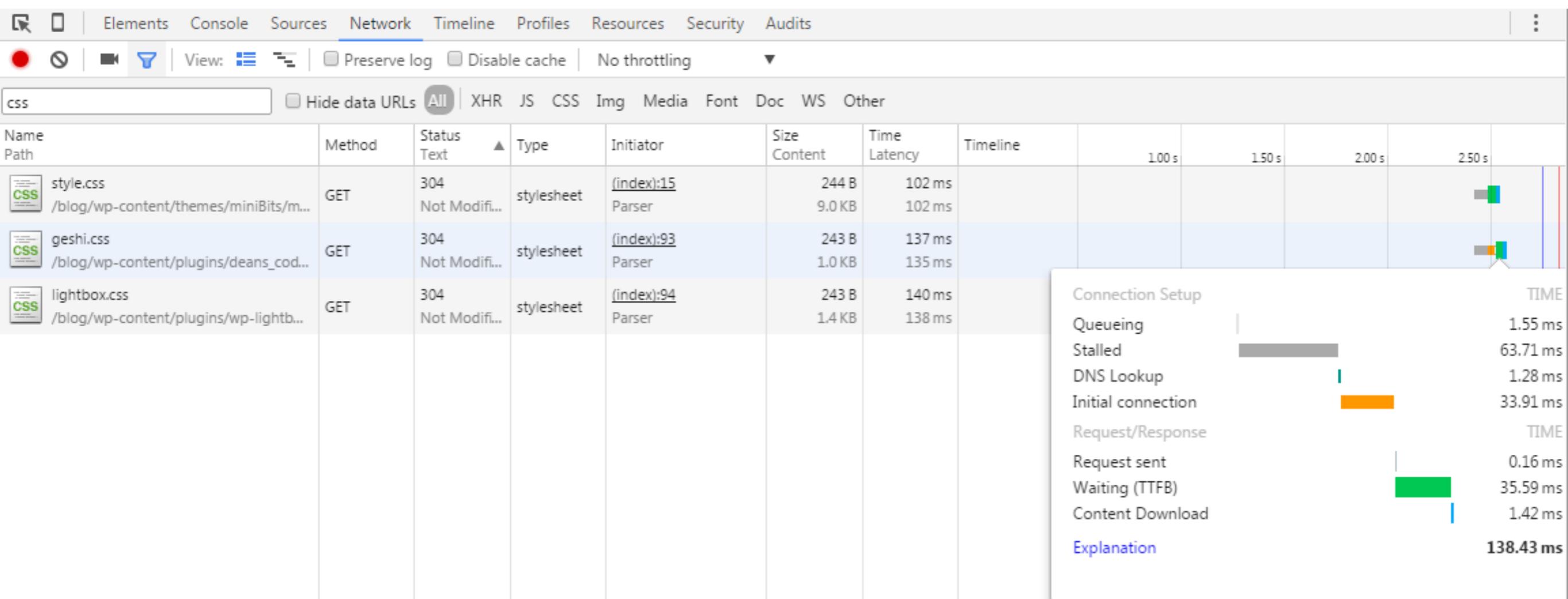
Record + Details View

Check the request method types

Analyse response headers

Filter and search for particular resource and analyze its performance

Chrome



# Network

Record + Details View

Check the request method types

Analyse response headers

Filter and search for particular resource and analyze its performance

## Firefox

Method	File	Domain	Type	Transferred	Size	Time	Headers	Cookies	Params	Response	Tin
200 GET	css?family=Francois+One Open+Sans:40...	fonts.googleapis.com	css	0,38 KB	1,01 KB	→ 90 ms	DNS resolution: → 0 ms				
200 GET	_utm.gif?utmwv=5.6.7&utms=1&ut...	www.google-analytics.c...	gif	0,03 KB	0,05 KB	→ 45 ms	Connecting: → 45 ms				
304 GET	/	html5please.com	html	38,13 KB	164,84 KB	→ 1 ms	Sending: → 0 ms				
304 GET	concat.js	html5please.com	js	8,22 KB	23,46 KB	→ 1 ms	Waiting: → 0 ms				
304 GET	ga.js	www.google-analytics.c...	js	15,65 KB	42,07 KB	→ 3 ms	Receiving: → 0 ms				
304 GET	6cfbd373bb1dd5f26c066c1fc7e68288?...	www.gravatar.com	jpeg	—	2,46 KB	→ 1 ms					
304 GET	21dbcdbf46f8ee3774c1b9b3efadfa9e?...	www.gravatar.com	jpeg	—	2,62 KB	→ 1 ms					
304 GET	56e38d8bcb52c9674dad204a2e8fc804...	www.gravatar.com	jpeg	—	1,90 KB	→ 1 ms					
304 GET	6025224?v=3&s=48	avatars3.githubusercontent...	jpeg	—	1,77 KB	→ 1 ms					
304 GET	denim.png?1436049899	html5please.com	png	—	26,50 KB	→ 1 ms					
304 GET	HTML5-logo.png	html5please.com	png	—	1,14 KB	→ 3 ms					
304 GET	c64a99fb7785e6cf9588a8777ba083?...	www.gravatar.com	png	—	5,36 KB	→ 2 ms					

# Learn more and stay updated

[developer.mozilla.org](https://developer.mozilla.org)

[developer.chrome.com/devtools](https://developer.chrome.com/devtools)

[umaar.com/dev-tips/](http://umaar.com/dev-tips/)

@MozDevNet

@FirefoxDevTools

@googledevs

@umaar

@addyosmani

