Chrome Developer Tools

WORKFLOW / DEBUGGING / PROFILING
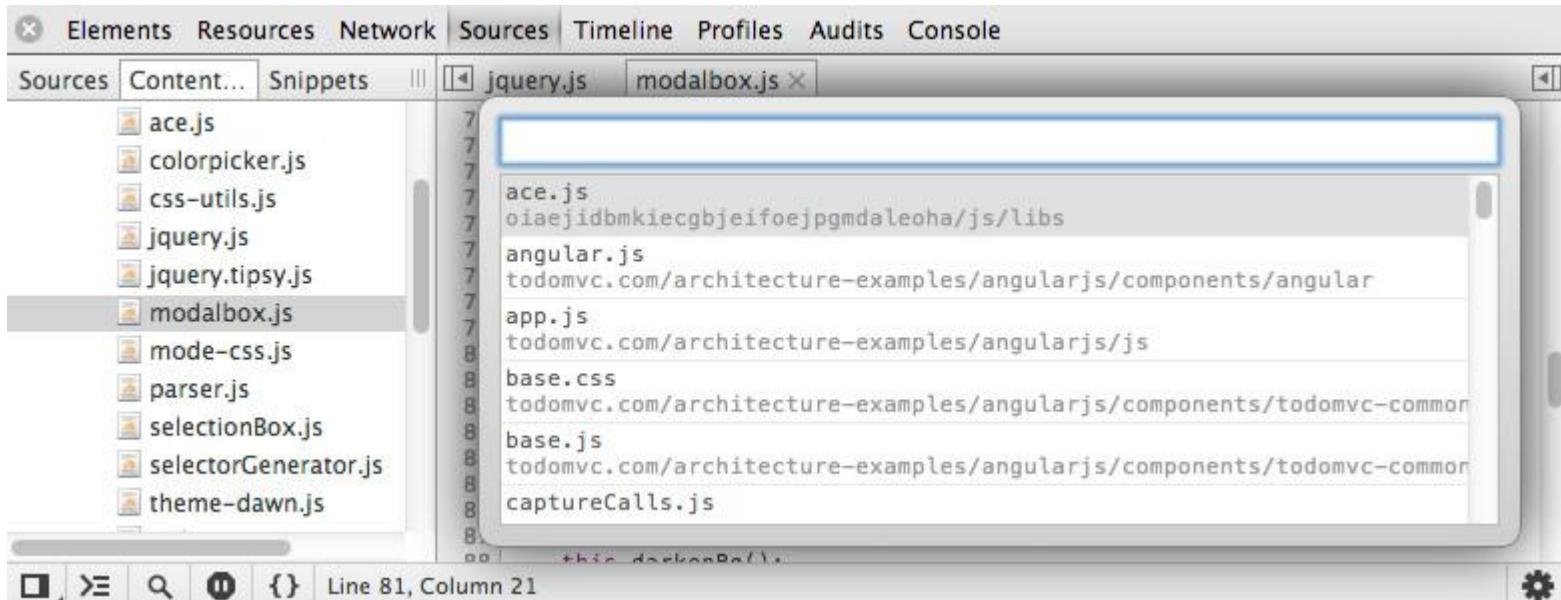
Aurimas Likas
alikas@kayak.com

Overview

- Development workflow
- Elements panel
- Console
- Network
- Mobile (responsive) and remote debugging
- Debugging Javascript
- Performance & Profiling

Search. Navigate. Filter.

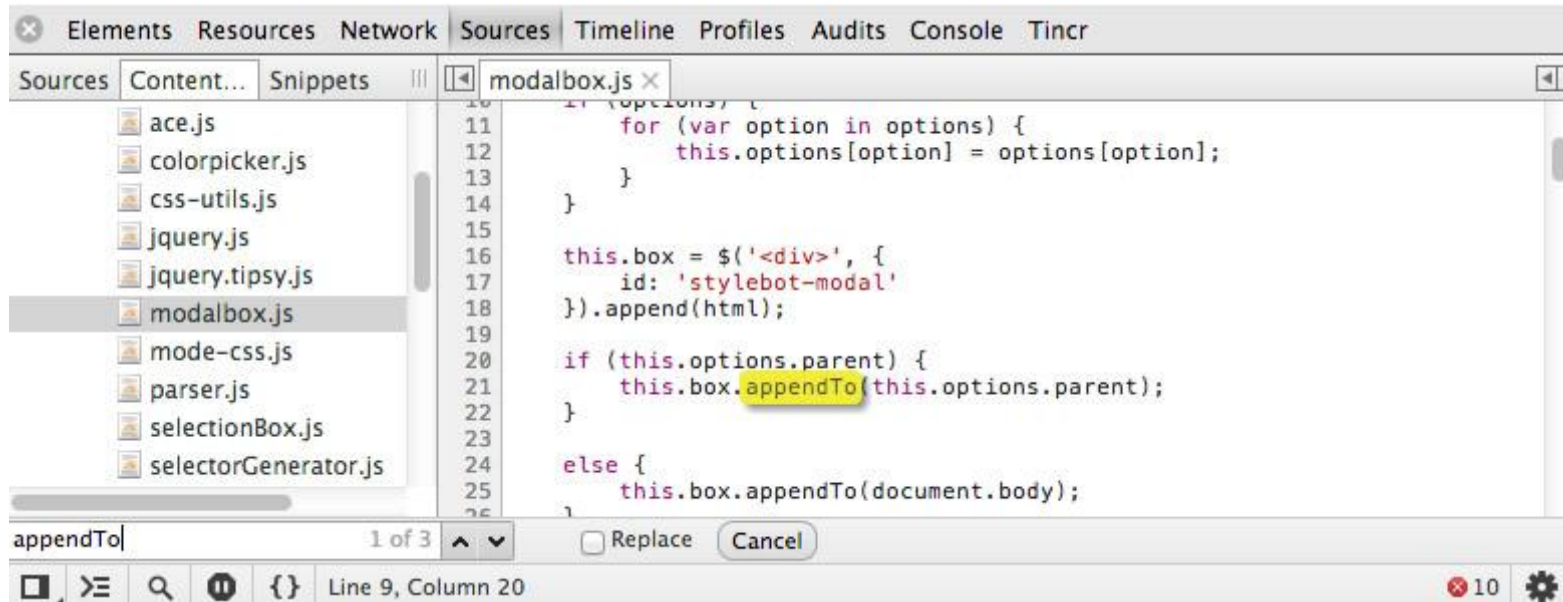# Filter For A Script, Stylesheet Or Snippet By Filename

Ctrl + O (Cmd + O)

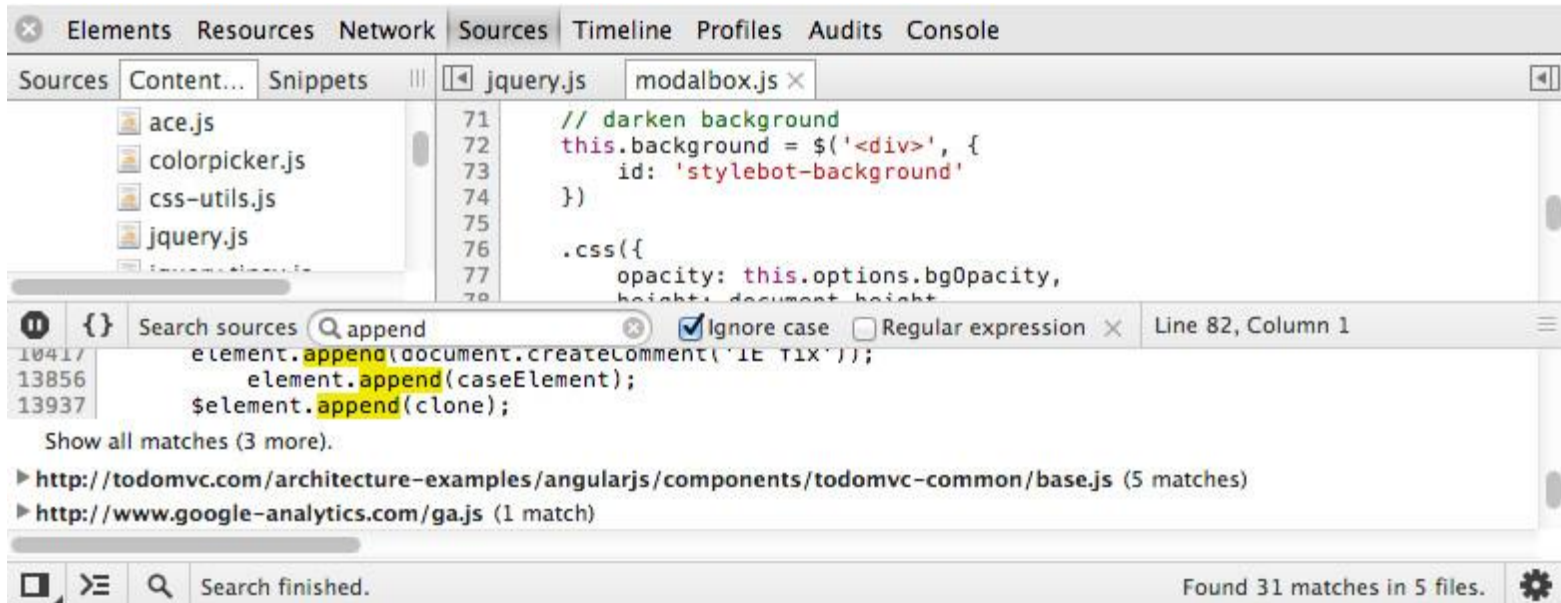# Text Search/Replace Within The Current File

Ctrl + F (Cmd + F)

Enter to jump between matches
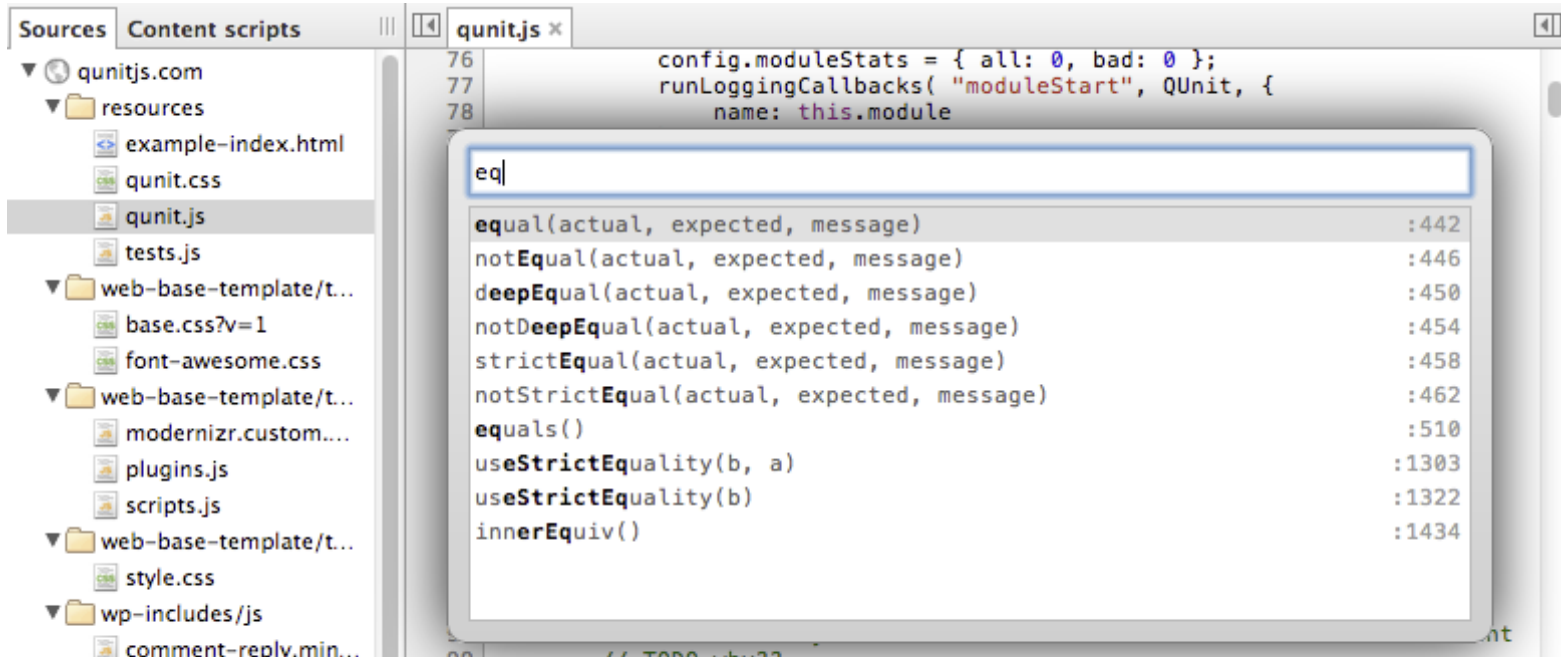
Search Across All Files

Ctrl + Shift + F (Cmd + Opt + F)

Enter to jump between matches

Filter A Function Or Selector Within A File

Ctrl + Shift + O (Cmd + Shift + O)

Live Editing Scripts

1. Click the link to the script (e.g `<script src="app.js"></script>`) in the markup view of the Elements panel
2. Or select the filename of the script from Sources sub-panel under Sources

- *Changes to scripts are only executed at evaluation time*
- *Modifications to code that is not running after the page loads will not have an effect.*
- *Changes to code executed at a later stage, such as mouseover handlers or click-event callbacks can be changed and tested on the fly.*

Live Editing Styles

- Instant updates in a browser window

- "element.style" = properties of the node the style attribute

- Rule origin and the line number

- Matched rules are colored black while others will be gray

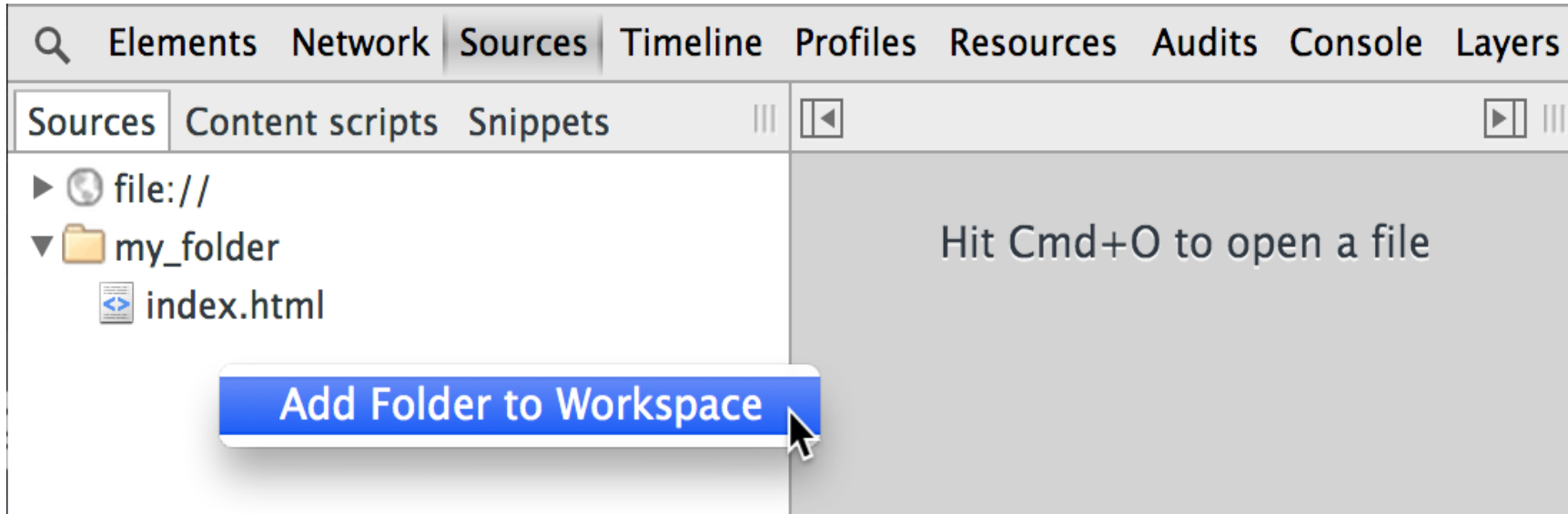- Up/Down (Shift Up/Down) on property value

Local Modifications

A revision history of all changes made to local files (scripts/styles saved before):

- A diff of the changes
- The time change was made
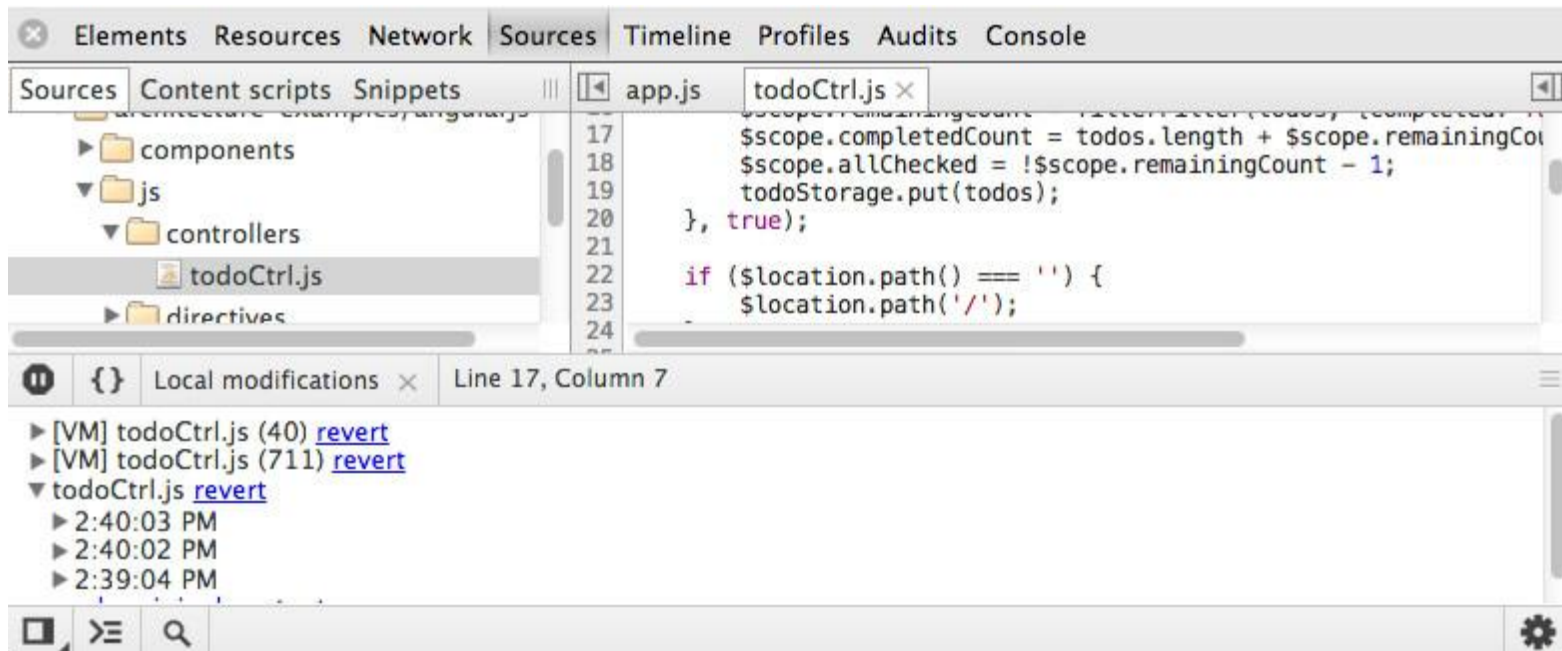
Workspaces

Persist your changes to disk without having to leave the Chrome DevTools

Snippets

- http://bgrins.github.io/devtools-snippets/
- https://github.com/paulirish/devtools-addons/wiki/Snippets
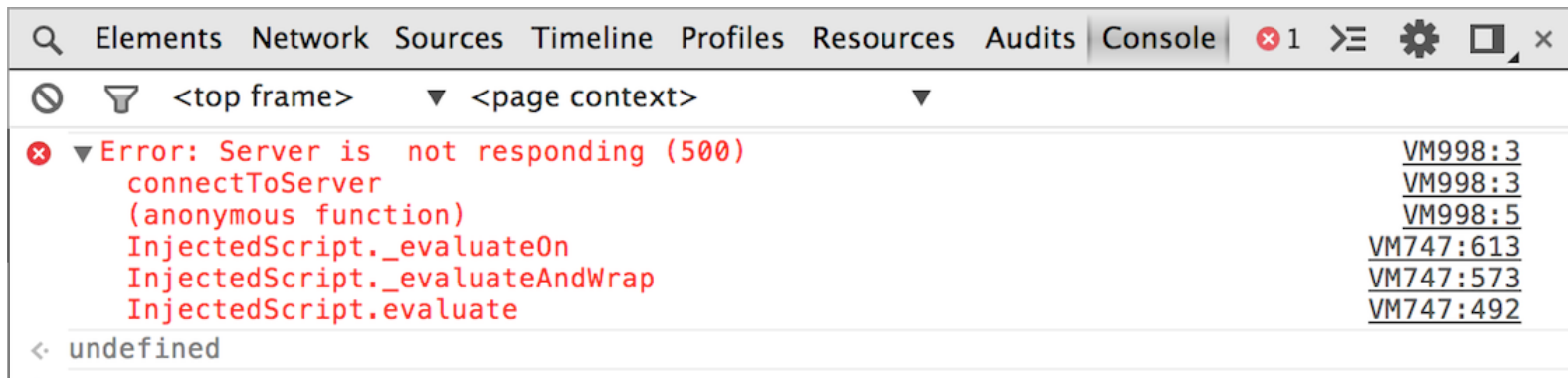
Console.

2 primary functions

- A place to log diagnostic information using methods provided by the [Console API](), such as [console.log()](), or [console.profile()]().
- A shell prompt where you can enter commands and interact with the document and the Chrome DevTools.

console.log()

- Accepts multiple params
- Supports format specifiers

| Format specifier	Description
%s	Formats the value as a string.
%d or %i	Formats the value as an integer.
%f	Formats the object as a floating point value.
%o	Formats the value as an expandable DOM element (as in the Elements panel).
%O	Formats the value as an expandable JavaScript object.
%c	Applies CSS style rules to output string specified by the second parameter.

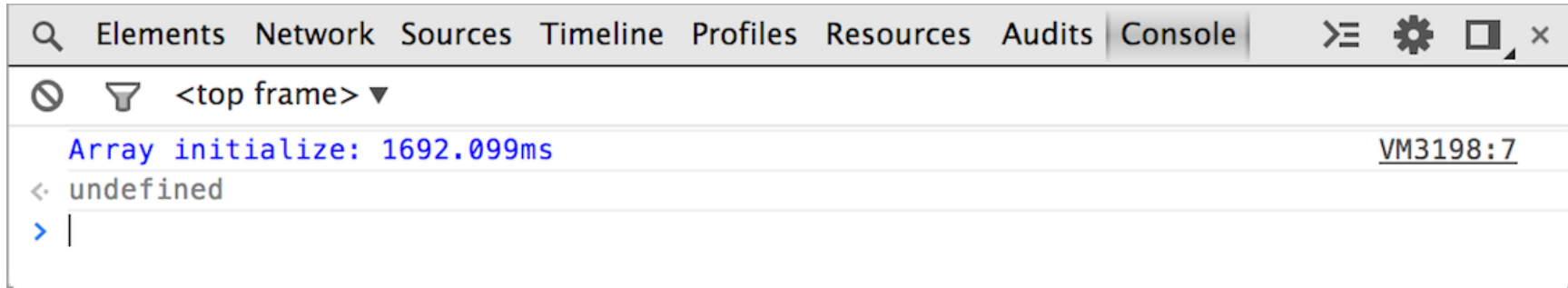# [console.error()](#)

[console.warn()](#)
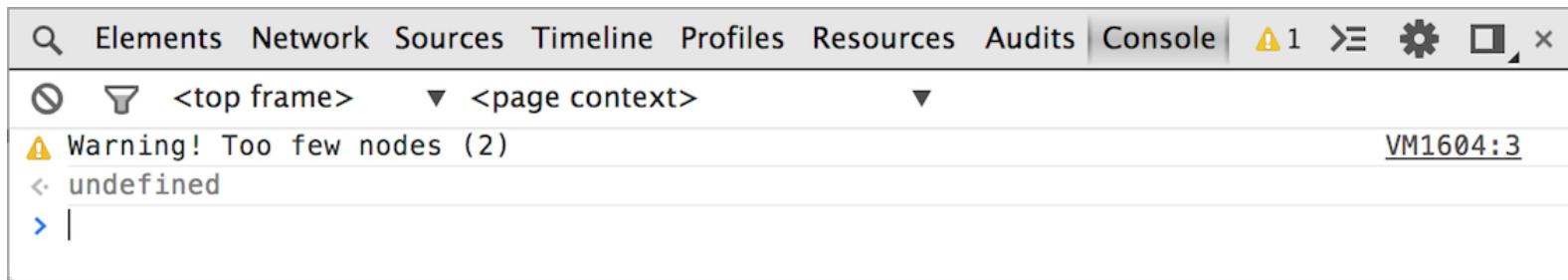
[console.group()](#), [console.groupCollapsed()](#)
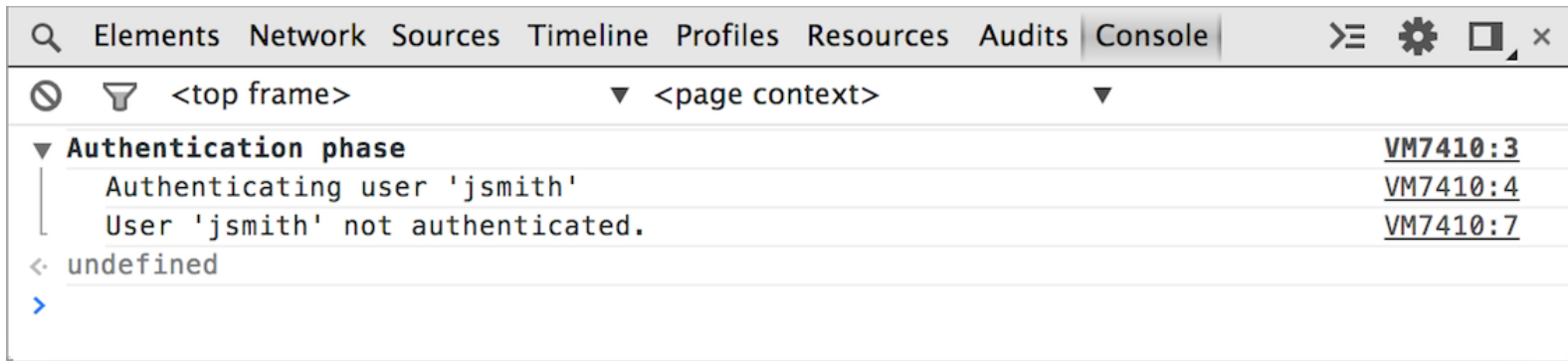
console.time(), console.timeEnd()

Measure how long a function or operation in your code takes to complete

```
console.time("Array initialize");
var array = new Array(1000000);
for (var i = array.length - 1; i >= 0; i--) {
    array[i] = new Object();
};
console.timeEnd("Array initialize");
```
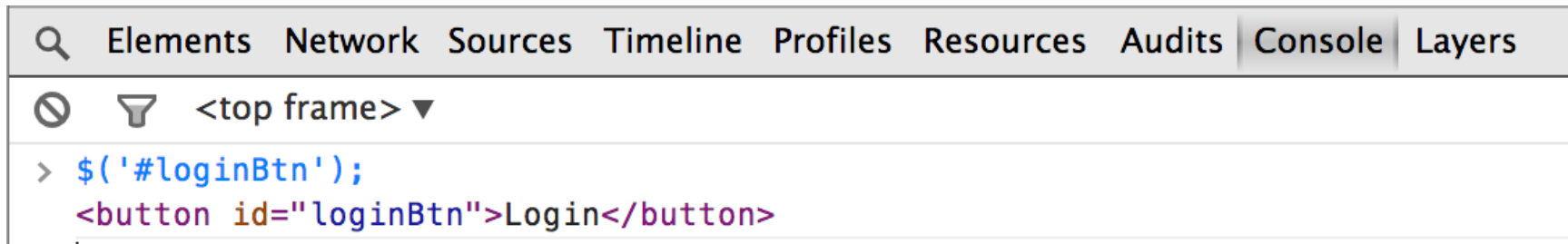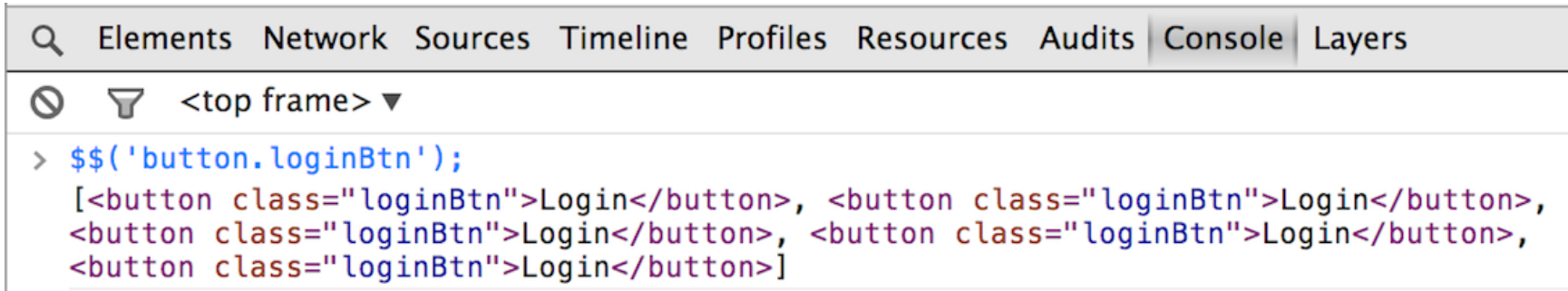
Q Elements Network Sources Timeline Profiles Resources Audits Console >≡ ⚙ ❑ ✕

⊘ ▽ <top frame> ▼

Array initialize: 1692.099ms   VM3198:7
← undefined
> |

Selecting elements

$() - returns the first element that matches the specified CSS selector, just like **document.querySelector()**

```
Q   Elements   Network   Sources   Timeline   Profiles   Resources   Audits | Console | Layers

⊘   ▽   <top frame> ▼

> $('#loginBtn');
  <button id="loginBtn">Login</button>
```

$$() - returns an array of all the elements that match the specified CSS selector, just like **document.querySelectorAll()**

```
Q   Elements   Network   Sources   Timeline   Profiles   Resources   Audits | Console | Layers

⊘   ▽   <top frame> ▼

> $$('button.loginBtn');
  [<button class="loginBtn">Login</button>, <button class="loginBtn">Login</button>,
  <button class="loginBtn">Login</button>, <button class="loginBtn">Login</button>,
  <button class="loginBtn">Login</button>]
```

Selecting elements

x() - takes an XPath path as a parameter and returns an array of all elements that match the specified path

The following returns all the <script> elements that are children of
the <body> tag:

```
$x('/html/body/script ');
```

Accessing recently selected elements and objects

Console remembers the last five elements (or heap objects) you've selected and makes them available as properties named **$0**, **$1**, **$2**, **$3** and **$4**.

| Elements | Network | Sources | Timeline | Profiles | Resources | Audits | Console | Layers |

```
▼ <html>
    <head></head>
    ▼ <body>
```

html   body   **li.devtools**

| Console | Search | Emulation | Rendering |

⊘ ▽ <top frame> ▼

```
> $0
  <li class="devtools">Paul</li>
> $1
  ▶ <div id="chrome_devtools">…</div>
> $2
  ▶ <div class="console_debugging">…</div>
>
```

Monitoring events

```
monitorEvents(window, "resize");
monitorEvents(document.body, [ "mousedown", "mouseup"] );
```

Debugging Javascript

Sources Panel
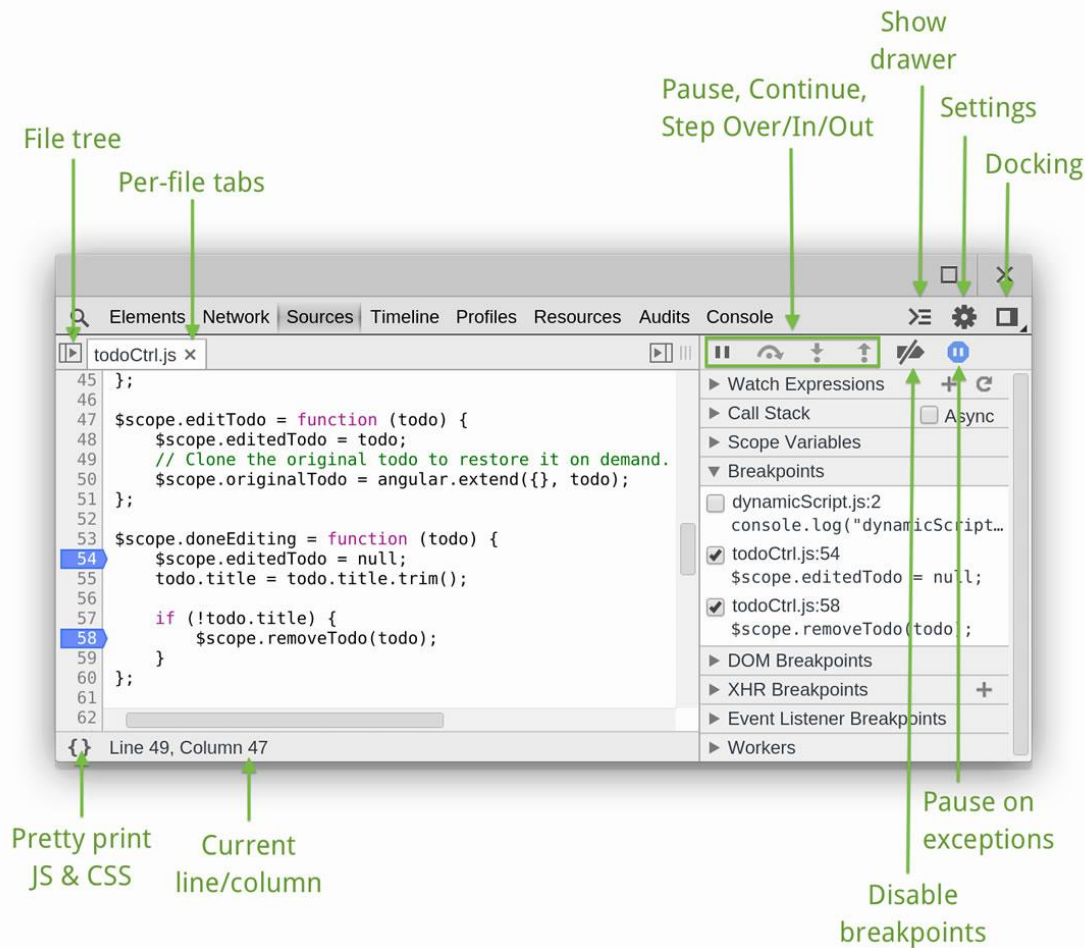
Execution control

Continue (F8): continues code execution until we encounter another breakpoint.

Step over (F10): step through code line-by-line to get insights into how each line affects the variables being updated. Should your code call another function, the debugger won't jump into its code, instead stepping over so that the focus remains on the current function.

Step into (F11): like Step over, however clicking Step into at the function call will cause the debugger to move its execution to the first line in the functions definition.

Step out (Shift+F11): having stepped into a function, clicking this will cause the remainder of the function definition to be run and the debugger will move its execution to the parent function.

Toggle breakpoints: toggles breakpoints on/off while leaving their enabled states intact.

Breakpoints

- Breakpoints (can be conditional)
- Breakpoints on DOM Mutation Events
- Breakpoints on XHR
- Breakpoints on JavaScript Event Listeners
- `debugger;`

https://developer.chrome.com/devtools/docs/javascript-debugging

Tips & Tricks
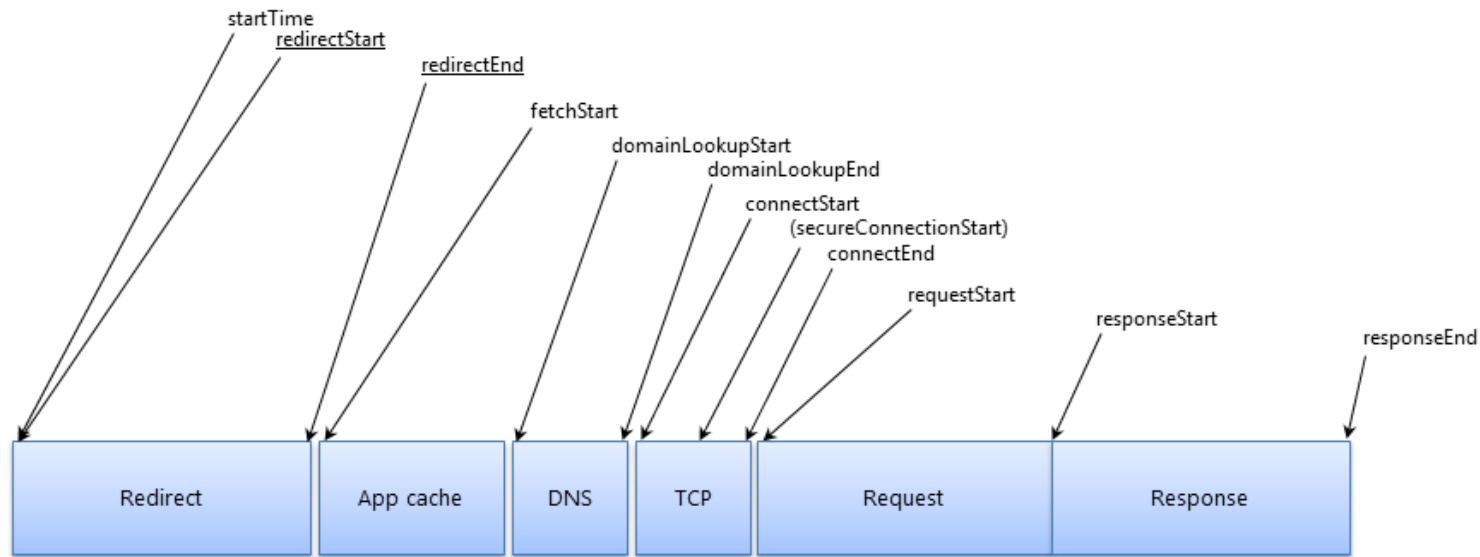
- **Async** checkbox
- Blackbox Javascript files
- Call Stack + Console
- Error.stack / console.trace() / console.assert()
- Pretty Print

Performance & Profiling

Questions answered

- Which resource had the slowest time to first byte?

- Which resources took the longest time to load (duration)?

- Who initiated a particular network request?

- How much time was spent in the various network phases for a particular resource?

Resource Timing API


```
window.performance.getEntries()
```

Network Panel

Waterfall view

- indicates when the the **DOMContentLoaded** and **load** events were fired (blue and and red vertical lines)
- preserving network log upon navigation
- sorting and filtering

Color-coded waterfall

 	Documents
 	Stylesheets
 	Images
 	Scripts
 	XHR
 	Fonts
 	Other

Resource network timing

Stalled/Blocking
   Time the request spent waiting before it could be sent. This time is inclusive of any time spent in proxy negotiation. Additionally, this time will include when the browser is waiting for an already established connection to become available for re-use, obeying Chrome's **maximum six** TCP connection per origin rule.

Proxy Negotiation
   Time spent negotiating with a proxy server connection.

DNS Lookup
   Time spent performing the DNS lookup. Every new domain on a page requires a full roundtrip to do the DNS lookup.

Initial Connection / Connecting
   Time it took to establish a connection, including TCP handshakes/retries and negotiating a SSL.

SSL
   Time spent completing a SSL handshake.

Request Sent / Sending
   Time spent issuing the network request. Typically a fraction of a millisecond.

Waiting (TTFB)
   Time spent waiting for the initial response, also known as the Time To First Byte. This time captures the latency of a round trip to the server in addition to the time spent waiting for the server to deliver the response.

Content Download / Downloading
   Time spent receiving the response data.

Performance profiling with the Timeline

Toggle recording
Clear recording
Toggle filters
Force garbage collection
Glue async events
Capture callstacks

Developer Tools – http://www.html5rocks.com/en/tutorials/internals/h...

Elements   Network   Sources   Timeline   Profiles   Resources   Audits   »

Capture stacks   0 of 0 records shown

Filter   All   ▾   ☑Loading   ☑Scripting   ☑Rendering   ☑Painting

Events   Filters: string, duration, type
Frames   View modes
Memory

RECORDS   Event records

DETAILS

Event details and summary

Profiling JavaScript Performance

Self ▼	Total	Function	
2191....	2194...	▶ montReduce	crypto.js:583
1852....	1852...	(garbage collector)	
1492....	1510...	▼ lin_solve	navier–stokes.js:128
1476....	1493...	▼ project	navier–stokes.js:239
785.2...	792....	▶ runNavierStokes	navier–stokes.js:35
691.4...	701....	▶ vel_step	navier–stokes.js:284
8.9 ms	9.1 ms	▼ diffuse	navier–stokes.js:157
8.9 ms	9.1 ms	▼ dens_step	navier–stokes.js:277
8.9 ms	9.1 ms	▼ FluidField.update	navier–stokes.js:329
8.9 ms	9.1 ms	▼ runNavierStokes	navier–stokes.js:35
8.9 ms	9.1 ms	▶ Measure	base.js:202
7.2 ms	7.5 ms	▼ runNavierStokes	navier–stokes.js:35
7.2 ms	7.5 ms	▶ Measure	base.js:202

Heavy (Bottom Up) ▼   %   👁   ✕

Exclude selected function

Focus selected function

Switch between absolute or percentage times

Select Bottom Up, Top Down or Flame Chart

http://v8.googlecode.com/svn/data/benchmarks/v7/run.html