X5 Tech

Снэпшоты памяти не страшно. Снэпшоты полезно.

2024

X5 Tech

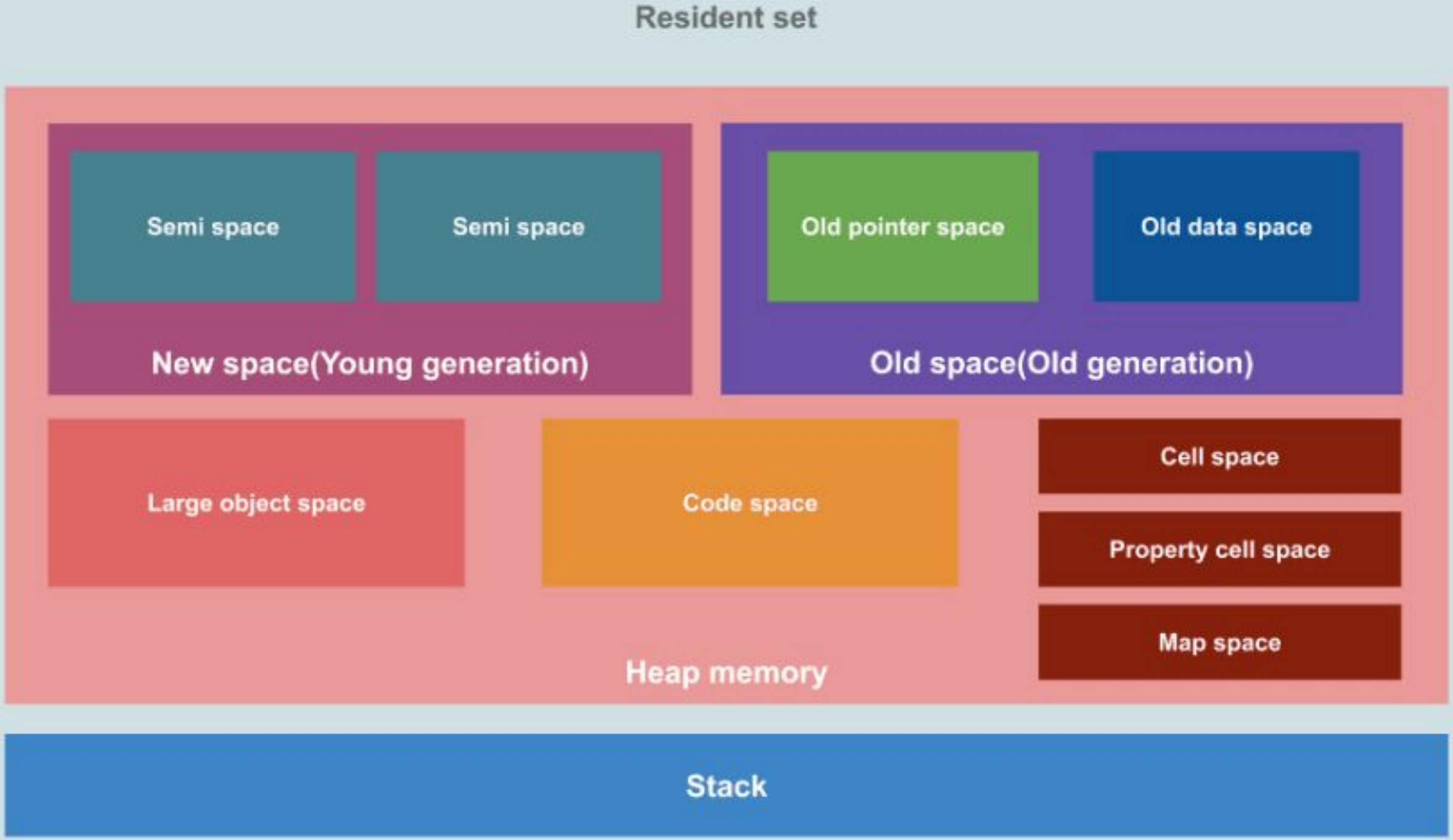
Александр Зайцев,
разработчик

Владислав Молоцило,
разработчик

X5 Tech

О чем мы хотим сегодня поговорить:

- ➔ Как профилировать память V8?
- ➔ Как читать снэпшоты?
- ➔ Как снимать снэпшоты?


Resident set
Semi space
Semi space
New space(Young generation)
Old pointer space
Old data space
Old space(Old generation)
Large object space
Code space
Cell space
Property cell space
Map space
Heap memory
Stack

```
> performance.memory
```

```
MemoryInfo {totalJSHeapSize: 42875906
2, usedJSHeapSize: 403635714, jsHeapS
izeLimit: 4294705152} i
    jsHeapSizeLimit: 4294705152
    totalJSHeapSize: 428759062
    usedJSHeapSize: 403635714
  ▶ [[Prototype]]: MemoryInfo
```

Filter

In this article

Performance: memory property

Deprecated: This feature is no longer recommended. Though some browsers might still support it, it may have already been removed from the relevant web standards, may be in the process of being dropped, or may only be kept for compatibility purposes. Avoid using it, and update existing code if possible; see the compatibility table at the bottom of this page to guide your decision. Be aware that this feature may cease to work at any time.

Non-standard: This feature is non-standard and is not on a standards track. Do not use it on production sites facing the Web: it will not work for every user. There may also be large incompatibilities between implementations and the behavior may change in the future.

Chrome	Edge	Firefox	Opera	Safari	Chrome Android	Firefox for Android	Opera Android	Safari on iOS	Samsung Internet	WebView Android	Deno	Node.js
✓ 7	✓ 79	✗ No	✓ 15	✗ No	✓ 18	✗ No	✓ 14	✗ No	✓ 1.0	✓ 37	✗ No	✗ No

Filter

In this article

Performance: measureUserAgentSpecificMemory() method

Experimental: This is an experimental technology

Check the Browser compatibility table carefully before using this in production.

The `measureUserAgentSpecificMemory()` method is used to estimate the memory usage of a web application including all its iframes and workers.

Chrome	Edge	Firefox	Opera	Safari	Chrome Android	Firefox for Android	Opera Android	Safari on iOS	Samsung Internet	WebView Android	Deno	Node.js
✓ 89	✓ 89	✗ No	✓ 75	✗ No	✓ 89	✗ No	✓ 63	✗ No	✓ 15.0	✓ 89	✗ No	✗ No

```
function runMemoryMeasurements() {
  const interval = -Math.log(Math.random()) * 5 * 60 * 1000;
  console.log(`Next measurement in ${Math.round(interval /
1000)} seconds.`);
  setTimeout(measureMemory, interval);
}

async function measureMemory() {
  const memorySample = await
performance.measureUserAgentSpecificMemory();
  console.log(memorySample);
  runMemoryMeasurements();
}

if (crossOriginIsolated) {
  runMemoryMeasurements();
}
```

```
{
  bytes: 1500000,
  breakdown: [
    {
      bytes: 1000000,
      attribution: [
        {
          url: "https://example.com",
          scope: "Window",
        },
      ],
      types: ["DOM", "JS"],
    },
    {
      bytes: 0,
      attribution: [],
      types: [],
    },
```

```
console.log(process.memoryUsage())
```

```
{
  rss: 64376832,
  heapTotal: 23949312,
  heapUsed: 19027760,
  external: 418148,
  arrayBuffers: 16614
}
```

```
const v8 = require('node:v8')

console.log(v8.getHeapStatistics())
```

```
{
  total_heap_size: 21327872,
  total_heap_size_executable: 524288,
  total_physical_size: 20774912,
  total_available_size: 4326140520,
  used_heap_size: 18765096,
  heap_size_limit: 4345298944,
  malloced_memory: 139368,
  peak_malloced_memory: 1521376,
  does_zap_garbage: 0,
  number_of_native_contexts: 1,
  number_of_detached_contexts: 0,
  total_global_handles_size: 8192,
  used_global_handles_size: 2304,
  external_memory: 406611
}
```

```
node --trace-gc 02_global.js
```

```
[39273:0x7f7e90040000]     4652 ms: Scavenge 11.7 (16.1) -> 11
.4 (27.3) MB, 10.6 / 0.0 ms  (average mu = 1.000, current mu =
 1.000) allocation failure;
[39273:0x7f7e90040000]     8672 ms: Mark-sweep (reduce) 17.3 (
27.3) -> 15.9 (18.6) MB, 11.6 / 0.0 ms  (+ 10.6 ms in 51 steps
 since start of marking, biggest step 4.9 ms, walltime since s
tart of marking 25 ms) (average mu = 1.000, current mu = 1.000
) finalize incremental marking via task; GC in old space reque
sted
```

https://v8.dev/docs/trace

https://nodejs.org/en/learn/diagnostics/memory/using-gc-traces

```
node --trace-gc-object-stats 02_global.js
```

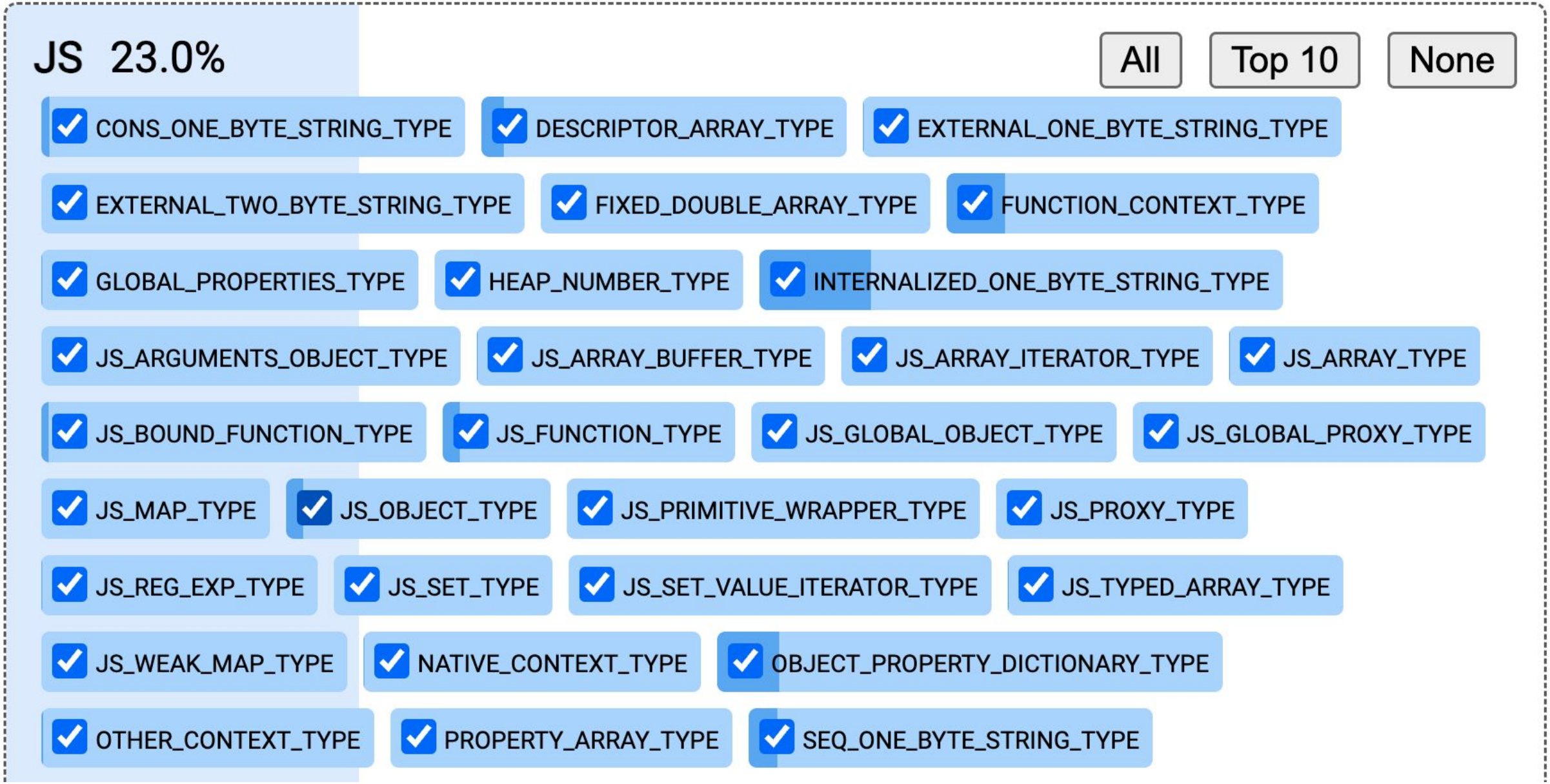

```
{
  "isolate": "0x150008000",
  "id": 1,
  "key": "live",
  "type": "field_data",
  "tagged_fields": 2005792,
  "embedder_fields": 2032,
  "inobject_smi_fields": 4472,
  "boxed_double_fields": 640,
  "string_data": 122312,
  "other_raw_fields": 853984
}
```

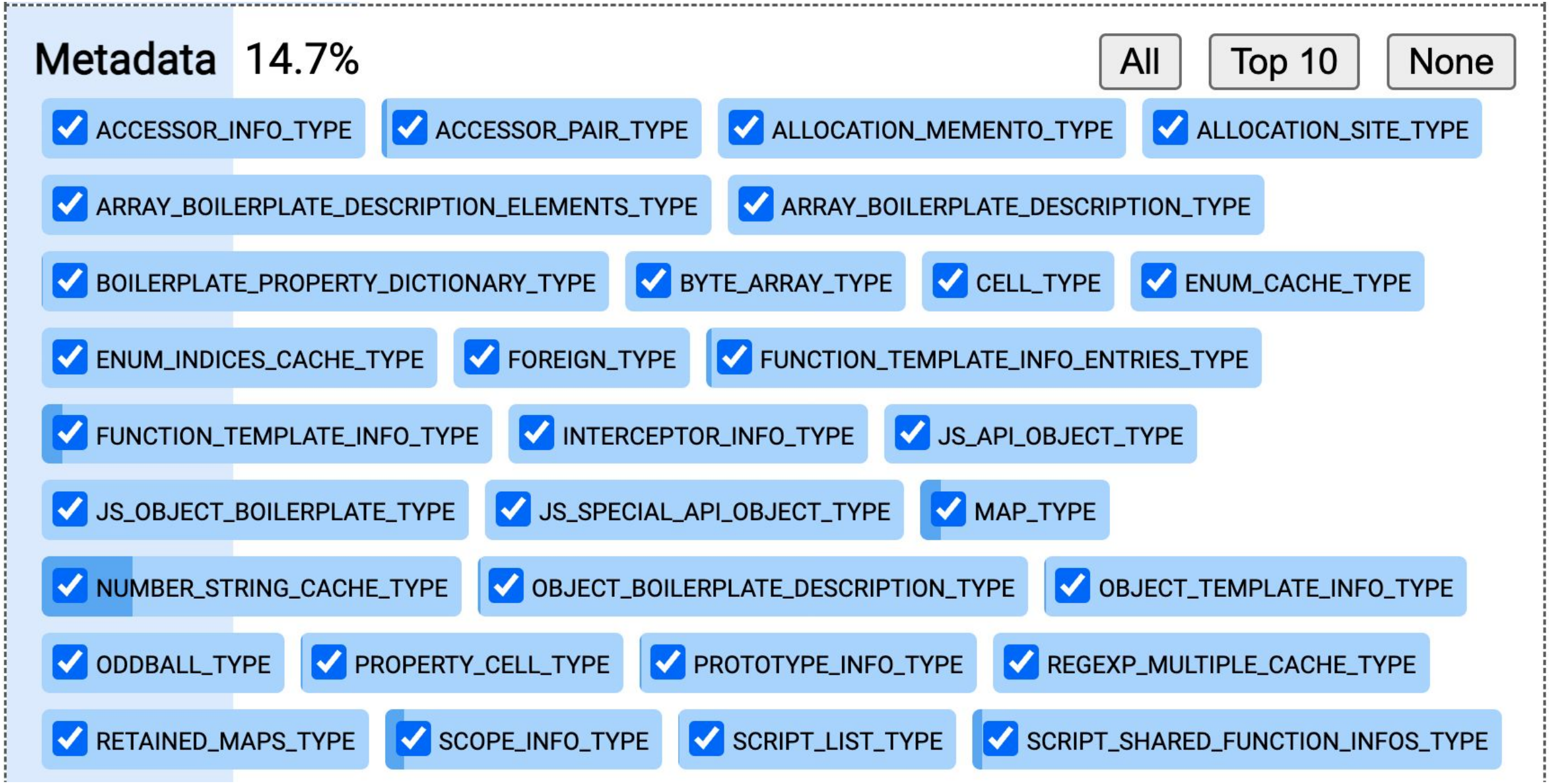
V8 Heap Statistics

Finished loading 'result.json.gz'.

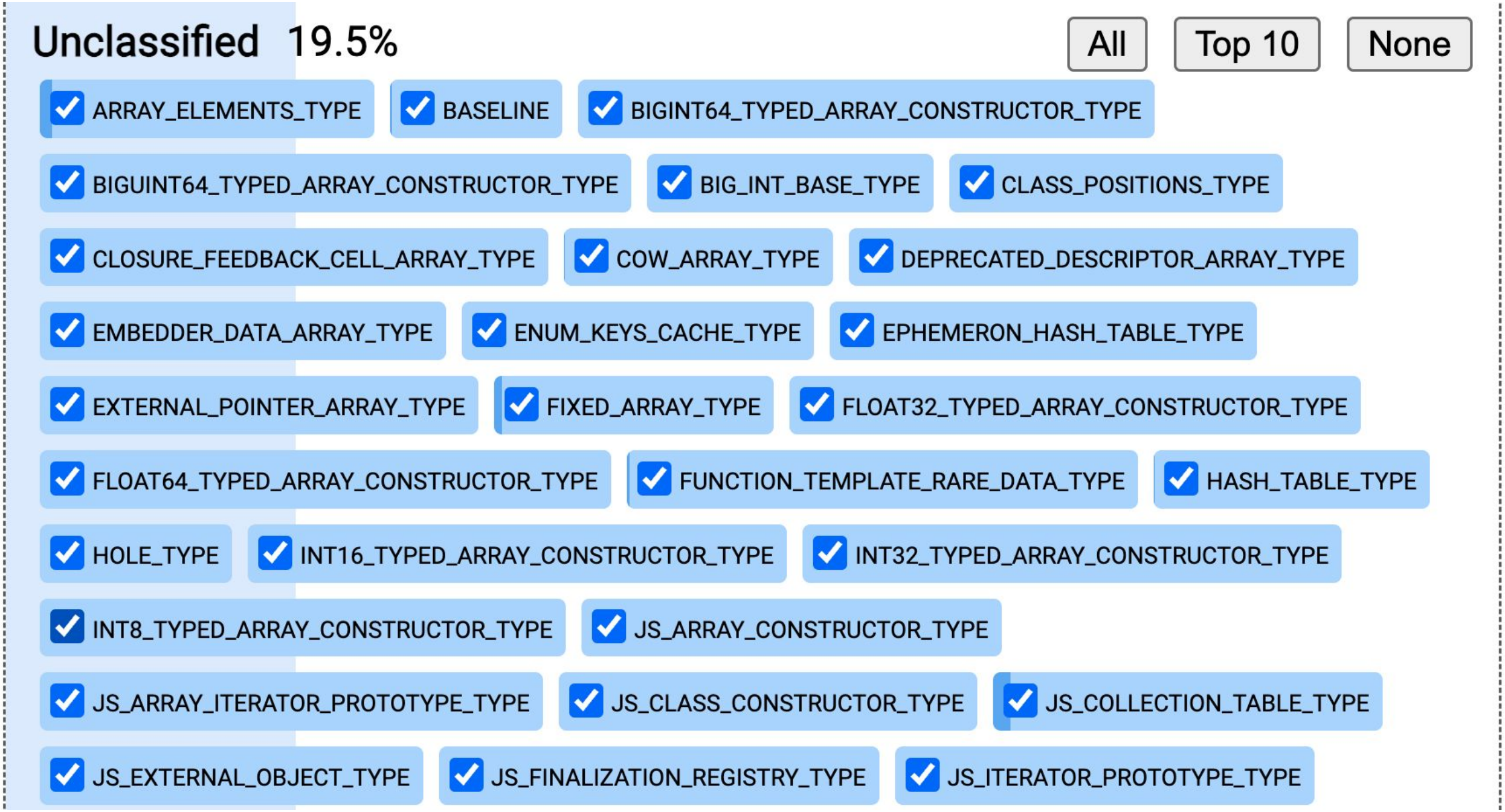
Data selection

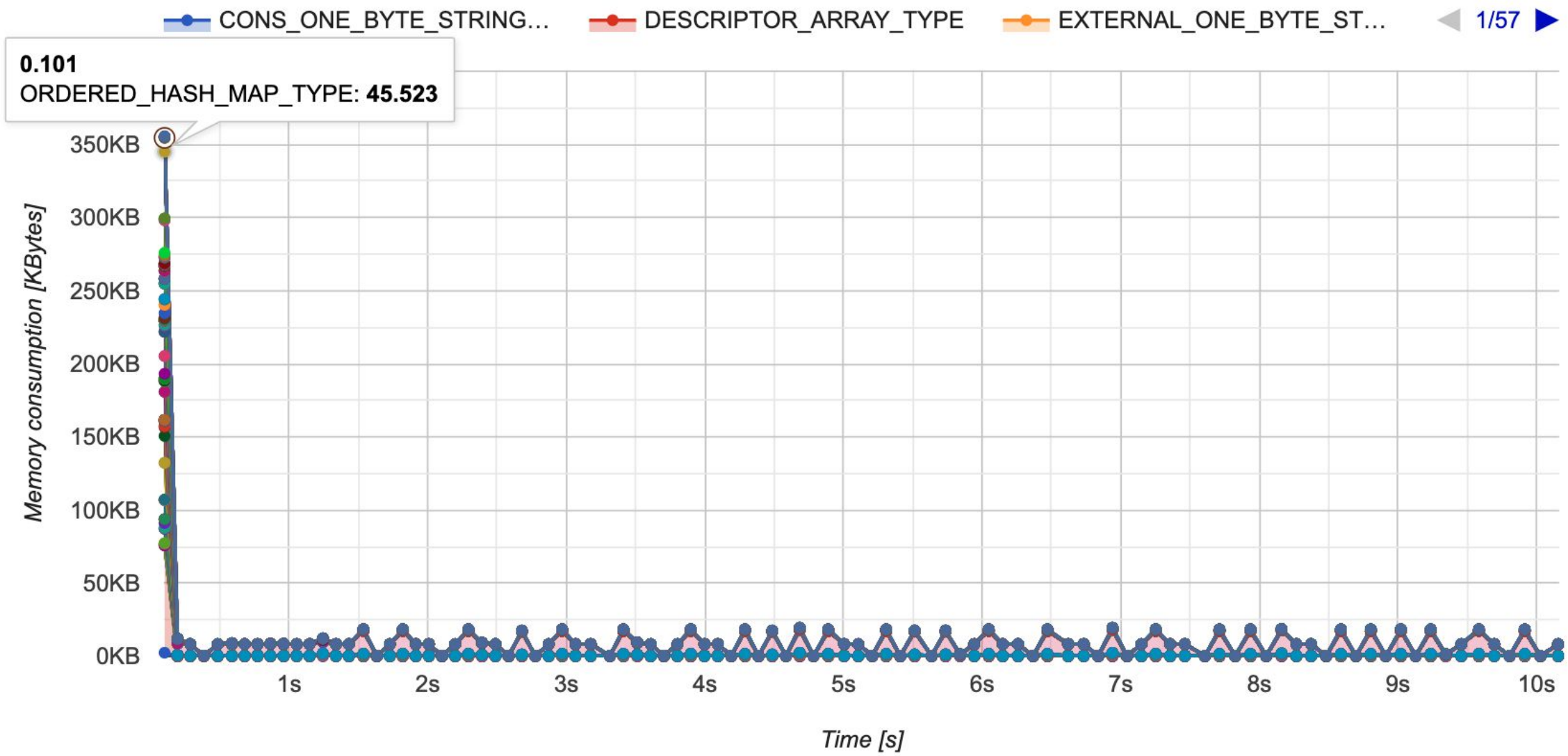
- Isolate 0x150008000: gc=#98 peak=16.76MiB
- Data view Selected instance types
- Data set live
- Garbage collection (at a specific time in ms) 01: 00.10s 004.14MiB
- 0 KB Filter categories with less memory Show top 20 categories only
- Export selection as CSV


JS 23.0%
All
Top 10
None
CONS_ONE_BYTE_STRING_TYPE
DESCRIPTOR_ARRAY_TYPE
EXTERNAL_ONE_BYTE_STRING_TYPE
EXTERNAL_TWO_BYTE_STRING_TYPE
FIXED_DOUBLE_ARRAY_TYPE
FUNCTION_CONTEXT_TYPE
GLOBAL_PROPERTIES_TYPE
HEAP_NUMBER_TYPE
INTERNALIZED_ONE_BYTE_STRING_TYPE
JS_ARGUMENTS_OBJECT_TYPE
JS_ARRAY_BUFFER_TYPE
JS_ARRAY_ITERATOR_TYPE
JS_ARRAY_TYPE
JS_BOUND_FUNCTION_TYPE
JS_FUNCTION_TYPE
JS_GLOBAL_OBJECT_TYPE
JS_GLOBAL_PROXY_TYPE
JS_MAP_TYPE
JS_OBJECT_TYPE
JS_PRIMITIVE_WRAPPER_TYPE
JS_PROXY_TYPE
JS_REG_EXP_TYPE
JS_SET_TYPE
JS_SET_VALUE_ITERATOR_TYPE
JS_TYPED_ARRAY_TYPE
JS_WEAK_MAP_TYPE
NATIVE_CONTEXT_TYPE
OBJECT_PROPERTY_DICTIONARY_TYPE
OTHER_CONTEXT_TYPE
PROPERTY_ARRAY_TYPE
SEQ_ONE_BYTE_STRING_TYPE


Metadata 14.7%
All
Top 10
None
ACCESSOR_INFO_TYPE
ACCESSOR_PAIR_TYPE
ALLOCATION_MEMENTO_TYPE
ALLOCATION_SITE_TYPE
ARRAY_BOILERPLATE_DESCRIPTION_ELEMENTS_TYPE
ARRAY_BOILERPLATE_DESCRIPTION_TYPE
BOILERPLATE_PROPERTY_DICTIONARY_TYPE
BYTE_ARRAY_TYPE
CELL_TYPE
ENUM_CACHE_TYPE
ENUM_INDICES_CACHE_TYPE
FOREIGN_TYPE
FUNCTION_TEMPLATE_INFO_ENTRIES_TYPE
FUNCTION_TEMPLATE_INFO_TYPE
INTERCEPTOR_INFO_TYPE
JS_API_OBJECT_TYPE
JS_OBJECT_BOILERPLATE_TYPE
JS_SPECIAL_API_OBJECT_TYPE
MAP_TYPE
NUMBER_STRING_CACHE_TYPE
OBJECT_BOILERPLATE_DESCRIPTION_TYPE
OBJECT_TEMPLATE_INFO_TYPE
ODDBALL_TYPE
PROPERTY_CELL_TYPE
PROTOTYPE_INFO_TYPE
REGEXP_MULTIPLE_CACHE_TYPE
RETAINED_MAPS_TYPE
SCOPE_INFO_TYPE
SCRIPT_LIST_TYPE
SCRIPT_SHARED_FUNCTION_INFOS_TYPE


Code 42.8%
All
Top 10
None
BYTECODE_ARRAY_CONSTANT_POOL_TYPE
BYTECODE_ARRAY_HANDLER_TABLE_TYPE
BYTECODE_ARRAY_TYPE
CODE_TYPE
DEOPTIMIZATION_DATA_TYPE
EMBEDDED_OBJECT_TYPE
FEEDBACK_CELL_TYPE
FEEDBACK_METADATA_TYPE
FEEDBACK_VECTOR_ENTRY_TYPE
FEEDBACK_VECTOR_HEADER_TYPE
FEEDBACK_VECTOR_SLOT_CALL_TYPE
FEEDBACK_VECTOR_SLOT_CALL_UNUSED_TYPE
FEEDBACK_VECTOR_SLOT_ENUM_TYPE
FEEDBACK_VECTOR_SLOT_LOAD_TYPE
FEEDBACK_VECTOR_SLOT_LOAD_UNUSED_TYPE
FEEDBACK_VECTOR_SLOT_OTHER_TYPE
FEEDBACK_VECTOR_SLOT_STORE_TYPE
FEEDBACK_VECTOR_SLOT_STORE_UNUSED_TYPE
LOAD_HANDLER_TYPE
OPTIMIZED_CODE_LITERALS_TYPE
PREPARSE_DATA_TYPE
RELOC_INFO_TYPE
SCRIPT_SOURCE_EXTERNAL_ONE_BYTE_TYPE
SCRIPT_SOURCE_EXTERNAL_TWO_BYTE_TYPE
SCRIPT_SOURCE_NON_EXTERNAL_ONE_BYTE_TYPE


Unclassified 19.5%
All
Top 10
None
ARRAY_ELEMENTS_TYPE
BASELINE
BIGINT64_TYPED_ARRAY_CONSTRUCTOR_TYPE
BIGUINT64_TYPED_ARRAY_CONSTRUCTOR_TYPE
BIG_INT_BASE_TYPE
CLASS_POSITIONS_TYPE
CLOSURE_FEEDBACK_CELL_ARRAY_TYPE
COW_ARRAY_TYPE
DEPRECATED_DESCRIPTOR_ARRAY_TYPE
EMBEDDER_DATA_ARRAY_TYPE
ENUM_KEYS_CACHE_TYPE
EPHEMERON_HASH_TABLE_TYPE
EXTERNAL_POINTER_ARRAY_TYPE
FIXED_ARRAY_TYPE
FLOAT32_TYPED_ARRAY_CONSTRUCTOR_TYPE
FLOAT64_TYPED_ARRAY_CONSTRUCTOR_TYPE
FUNCTION_TEMPLATE_RARE_DATA_TYPE
HASH_TABLE_TYPE
HOLE_TYPE
INT16_TYPED_ARRAY_CONSTRUCTOR_TYPE
INT32_TYPED_ARRAY_CONSTRUCTOR_TYPE
INT8_TYPED_ARRAY_CONSTRUCTOR_TYPE
JS_ARRAY_CONSTRUCTOR_TYPE
JS_ARRAY_ITERATOR_PROTOTYPE_TYPE
JS_CLASS_CONSTRUCTOR_TYPE
JS_COLLECTION_TABLE_TYPE
JS_EXTERNAL_OBJECT_TYPE
JS_FINALIZATION_REGISTRY_TYPE
JS_ITERATOR_PROTOTYPE_TYPE


CONS_ONE_BYTE_STRING…
DESCRIPTOR_ARRAY_TYPE
EXTERNAL_ONE_BYTE_ST…
1/57
0.101
ORDERED_HASH_MAP_TYPE: 45.523
350KB
300KB
250KB
200KB
150KB
100KB
50KB
0KB
Memory consumption [KBytes]
1s
2s
3s
4s
5s
6s
7s
8s
9s
10s
Time [s]

```
node --allow-natives-syntax
```

```
%DebugPrint(func)
```

```
 - type: JS_FUNCTION_TYPE
 - instance size: 56
 - inobject properties: 0
 - elements kind: HOLEY_ELEMENTS
```

```
> queryObjects(HTMLDivElement)
< undefined
▼ Array(485) i
  ▶ [0 … 99]
  ▶ [100 … 199]
  ▶ [200 … 299]
  ▶ [300 … 399]
  ▶ [400 … 484]
    length: 485
  ▶ [[Prototype]]: Array(0)
>
```

```
▼ [0 … 99]
  ▶ 0: div#share.listbox.listbox_shar
  ▶ 1: div#shareInput.listbox-option_
  ▶ 2: div#shareContent
  ▶ 3: div#interim.textinput.textlaye
  ▶ 4: div#speller.textinput.textlaye
  ▶ 5: div#tooltip.tooltip.state-hidd
  ▶ 6: div#textbox.box.box_src
  ▶ 7: div#textbox2.box.box_dst
  ▶ 8: div#external.boxExternal.state-
  ▶ 9: div#measurer.textinput.textlay
  ▶ 10: div#measurerDst.textinput.tex
  ▶ 11: div#keyboard.keyboard
```

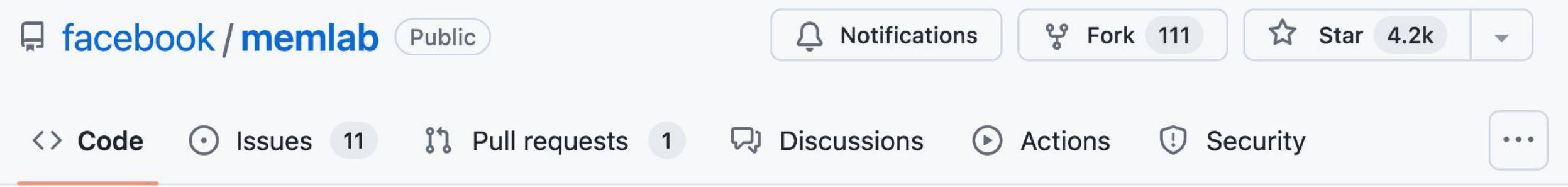
```
it('should collect all AudioBufferSourceNodes', async function () {
    this.timeout(10000);

    // Run the test once because the first run will trigger some memoizations.
    await page.evaluate(run, 1);

    await page.evaluate(() => gc()); // eslint-disable-line no-undef

    const numberOfObjects = await countObjects(page);

    await page.evaluate(run, 1000);

    await page.evaluate(() => gc()); // eslint-disable-line no-undef

    expect(await countObjects(page)).to.equal(numberOfObjects);
});
```

```
const countObjects = async (page) => {
    const prototypeHandle = await page.evaluateHandle(() => Object.prototype);
    const objectsHandle = await page.queryObjects(prototypeHandle);
    const numberOfObjects = await page.evaluate((instances) => instances.length, objectsHandle);

    await Promise.all([prototypeHandle.dispose(), objectsHandle.dispose()]);

    return numberOfObjects;
};
```

https://github.com/chrisguttandin/standardized-audio-context/blob/master/test/memory/

memlab

A framework for finding JavaScript memory leaks and analyzing heap snapshots



https://facebook.github.io/memlab/

```
// initial page load's url
function url() {
  return "https://www.youtube.com";
}

// action where you suspect the memory leak might be happening
async function action(page) {
  await page.click('[id="video-title-link"]');
}

// how to go back to the state before action
async function back(page) {
  await page.click('[id="logo-icon"]');
}

module.exports = { action, back, url };
```

Полезные команды

- ➜ memlab analyze detached-DOM
- ➜ memlab analyze global-variable
- ➜ memlab analyze object
- ➜ memlab heap --snapshot <HEAP_SNAPSHOT_FILE>
- ➜ и многое другое

https://facebook.github.io/memlab/docs/cli/CLI-commands


Referrers of @30067 (press 1 to focus) 531006
[system / Context](object) @2206489 [50.3KB]--p
[system / Context](object) @2206487 [50.3KB]--p
[system / Context](object) @2206485 [50.3KB]--p
[system / Context](object) @2206483 [50.3KB]--p
[system / Context](object) @2206481 [50.3KB]--p
[system / Context](object) @2206479 [50.3KB]--p
[system / Context](object) @2206477 [50.3KB]--p
[system / Context](object) @2206475 [50.3KB]--p
[system / Context](object) @2206473 [50.3KB]--p
[system / Context](object) @2206471 [50.3KB]--p
[system / Context](object) @2206469 [50.3KB]--p
[system / Context](object) @2206467 [50.3KB]--p
[system / Context](object) @2206465 [50.3KB]--p
[system / Context](object) @2206463 [50.3KB]--p
[system / Context](object) @2206461 [50.3KB]--p
[system / Context](object) @2206459 [50.3KB]--p
[system / Context](object) @2206457 [50.3KB]--p
[system / Context](object) @2206455 [50.3KB]--p
Objects (press 2 to focus) 13 items
[50.9MB] [system / Context](object) @30067 [50.
[50.9MB] [](closure) @2078323 [50.9MB]
[50.9MB] [Array](object) @2190947 [50.9MB]
[50.9MB] [system / Context](object) @2190949 [5
[Detached] [Detached Node / BindingData](native
[Detached] [Detached Node / BindingData](native
[Detached] [Detached Node / BindingData](native
[Detached] [Detached Node / BlobBindingData](na
[Detached] [Detached Node / BindingData](native
[Detached] [Detached Node / BindingData](native
[Detached] [Detached Node / BindingData](native
[Detached] [Detached Node / BindingData](native
[Detached] [Detached Node / BindingData](native
Clustered Objects (press 3 to focus) 5 items
[50.9MB] [system / Context](object) @30067 [50.
[50.9MB] [](closure) @2078323 [50.9MB]
[50.9MB] [Array](object) @2190947 [50.9MB]
[50.9MB] [system / Context](object) @2190949 [5
[Cluster] [Detached Node / BindingData](native)
Object: @30067 (press 5 to focus) 8 items
id: @30067
name: system / Context
type: object
self size: 56 bytes
retained size: 50.9MB
# of references: 6
# of referrers: 531006
dominator node: [](synthetic) @1 [54.4MB]
References of @30067 (press 4 to focus) 6
--f(context)---> [](closure) @2078323 [50.9MB]
--previous(internal)---> [system / NativeContex
--timer(context)---> [Timeout](object)
--recursiveClosure(context)---> [recursiveClosu
--scope_info(internal)---> [system / ScopeInfo]
--map(internal)---> [system / Map](object
Retainer Trace of @30067 (press 6 to focus) 9
[](synthetic) @1 [54.4MB]
--2(shortcut)---> [global /](object) @6275 [64
--clearInterval(property)---> [clearInterval](c
--context(internal)---> [system / Context](obje
--timerListMap(context)---> [Object](object)
--1000000(element)---> [TimersList](object)
--_idleNext(property)---> [Timeout](object)
--_onTimeout(property)---> [](closure)
--context(internal)---> [system / Context](obje

(index)	name	type	count	retainedSize
0	''	'closure'	595119	'50.1GB'
1	'groupBy'	'closure'	2	'240 bytes'
2	'transfer'	'closure'	1	'120 bytes'
3	'transferToFixedLength'	'closure'	1	'120 bytes'

```
No leaks found
MemLab found 0 leak(s)
Number of clusters loaded: 0
```

pixtastock.com - 87768435

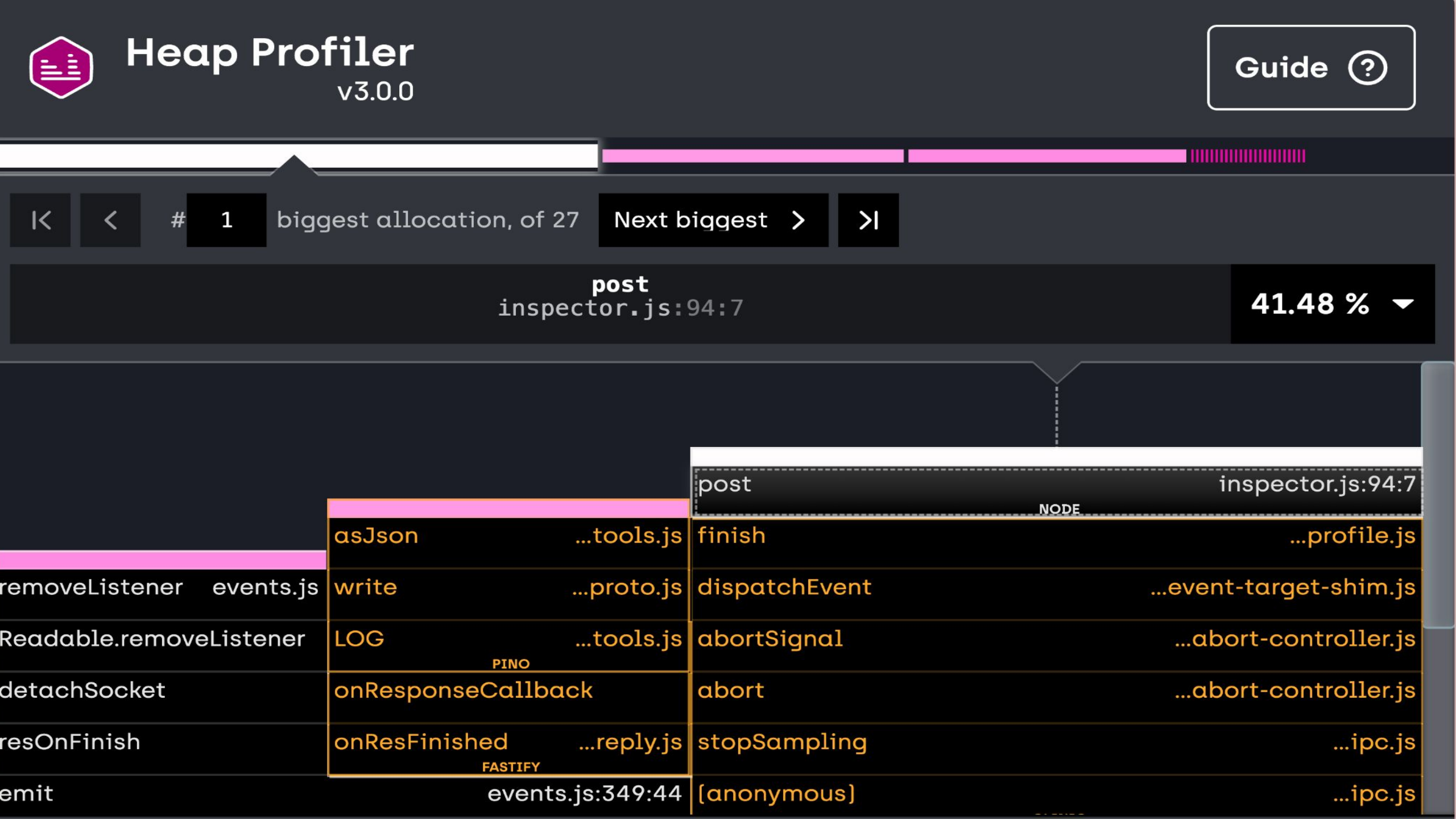
On GitHub

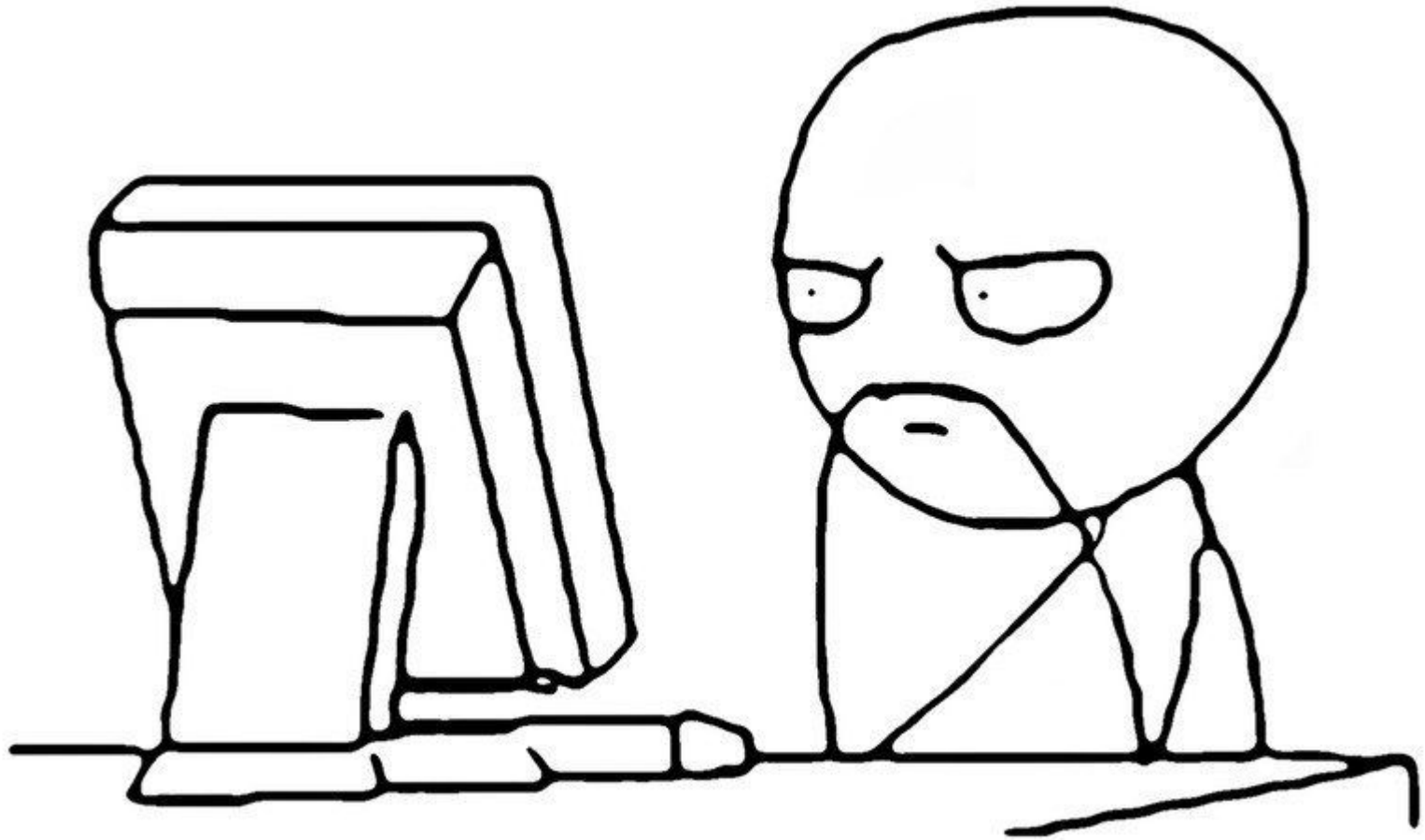
Example

Clinic.js HeapProfiler

Uncovers memory allocations by functions with Flamegraphs.

https://clinicjs.org/heapprofiler/


Heap Profiler
v3.0.0
Guide
1 biggest allocation, of 27
Next biggest
post
inspector.js:94:7
41.48 %
post inspector.js:94:7
NODE
asJson ...tools.js
finish ...profile.js
removeListener events.js
write ...proto.js
dispatchEvent ...event-target-shim.js
Readable.removeListener
LOG ...tools.js
abortSignal ...abort-controller.js
PINO
detachSocket
onResponseCallback
abort ...abort-controller.js
resOnFinish
onResFinished ...reply.js
stopSampling ...ipc.js
FASTIFY
emit events.js:349:44
(anonymous) ...ipc.js



Инструменты могут быть полезными ✅

Могут обнаружить утечку ✅ ❓

Могут указать на причину утечки ❓

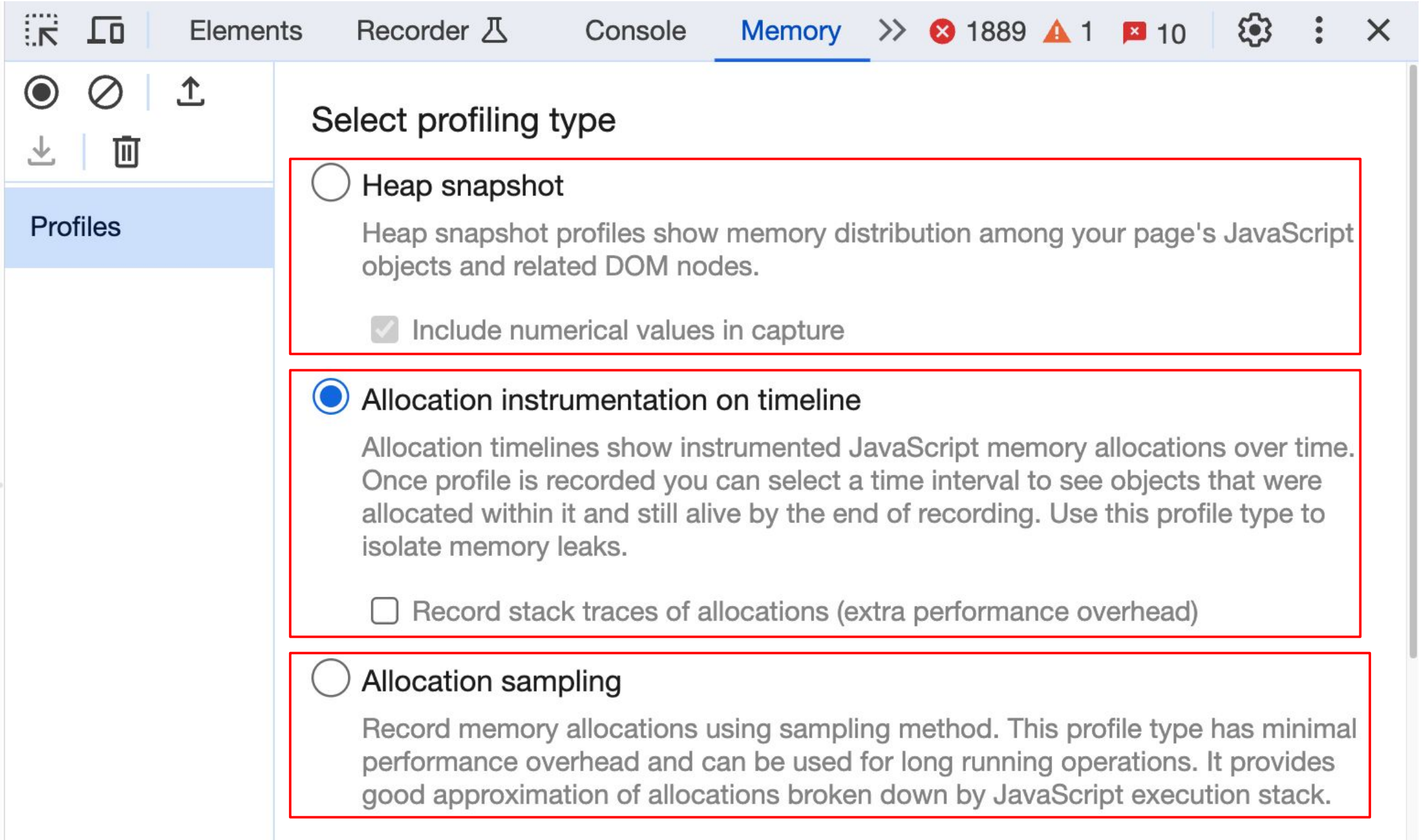
Недостаточно памяти для загрузки страницы

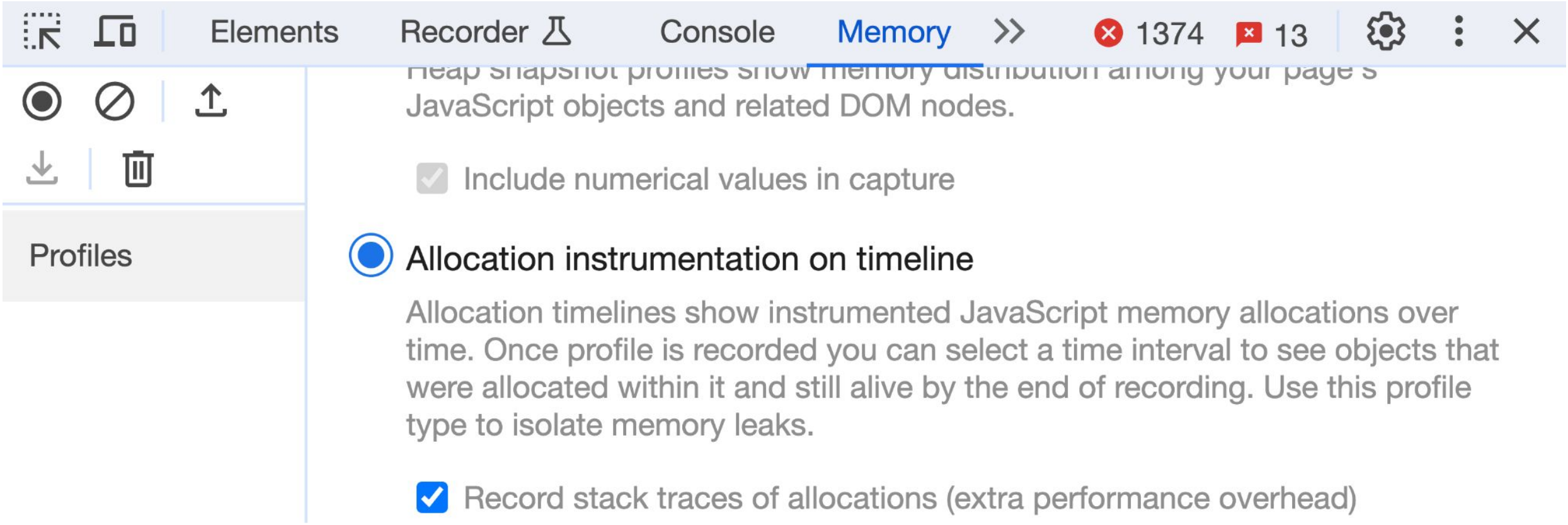
Закройте другие вкладки и программы, чтобы освободить память.

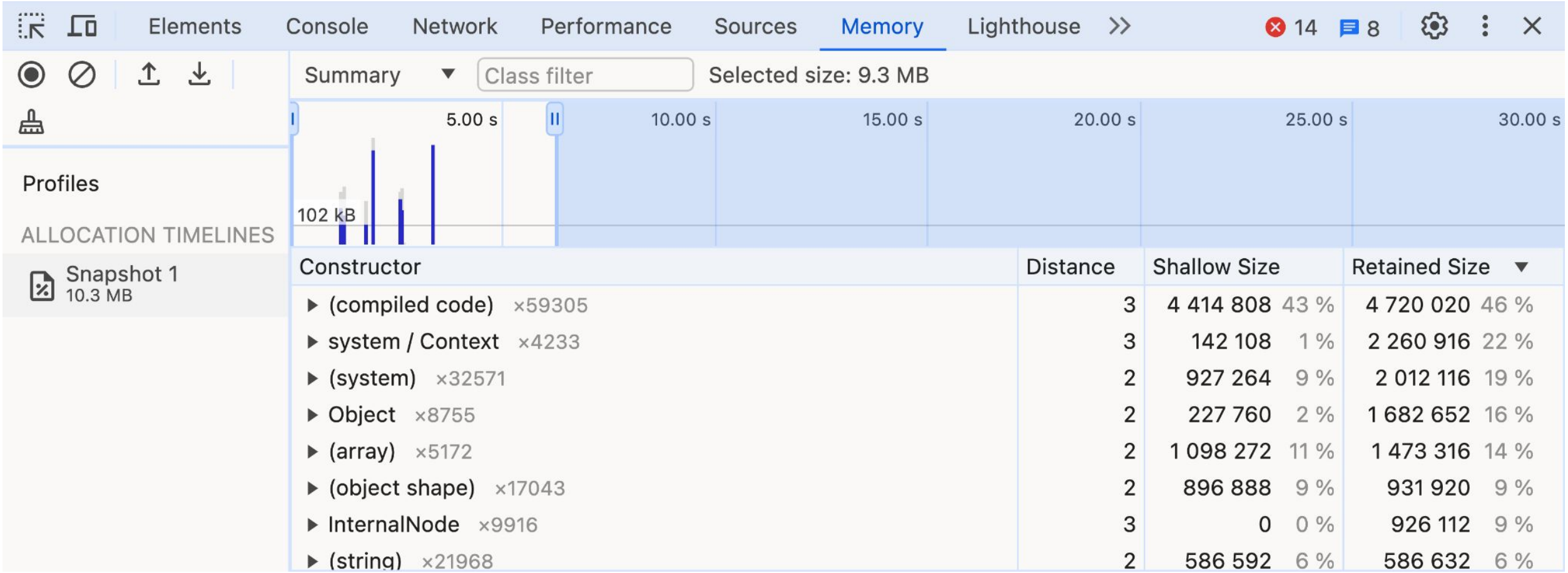
Код ошибки: Out of Memory

Подробнее

Отправить отзыв


Elements
Recorder
Console
Memory
1889
1
10
Profiles
Select profiling type
Heap snapshot
Heap snapshot profiles show memory distribution among your page's JavaScript objects and related DOM nodes.
Include numerical values in capture
Allocation instrumentation on timeline
Allocation timelines show instrumented JavaScript memory allocations over time. Once profile is recorded you can select a time interval to see objects that were allocated within it and still alive by the end of recording. Use this profile type to isolate memory leaks.
Record stack traces of allocations (extra performance overhead)
Allocation sampling
Record memory allocations using sampling method. This profile type has minimal performance overhead and can be used for long running operations. It provides good approximation of allocations broken down by JavaScript execution stack.

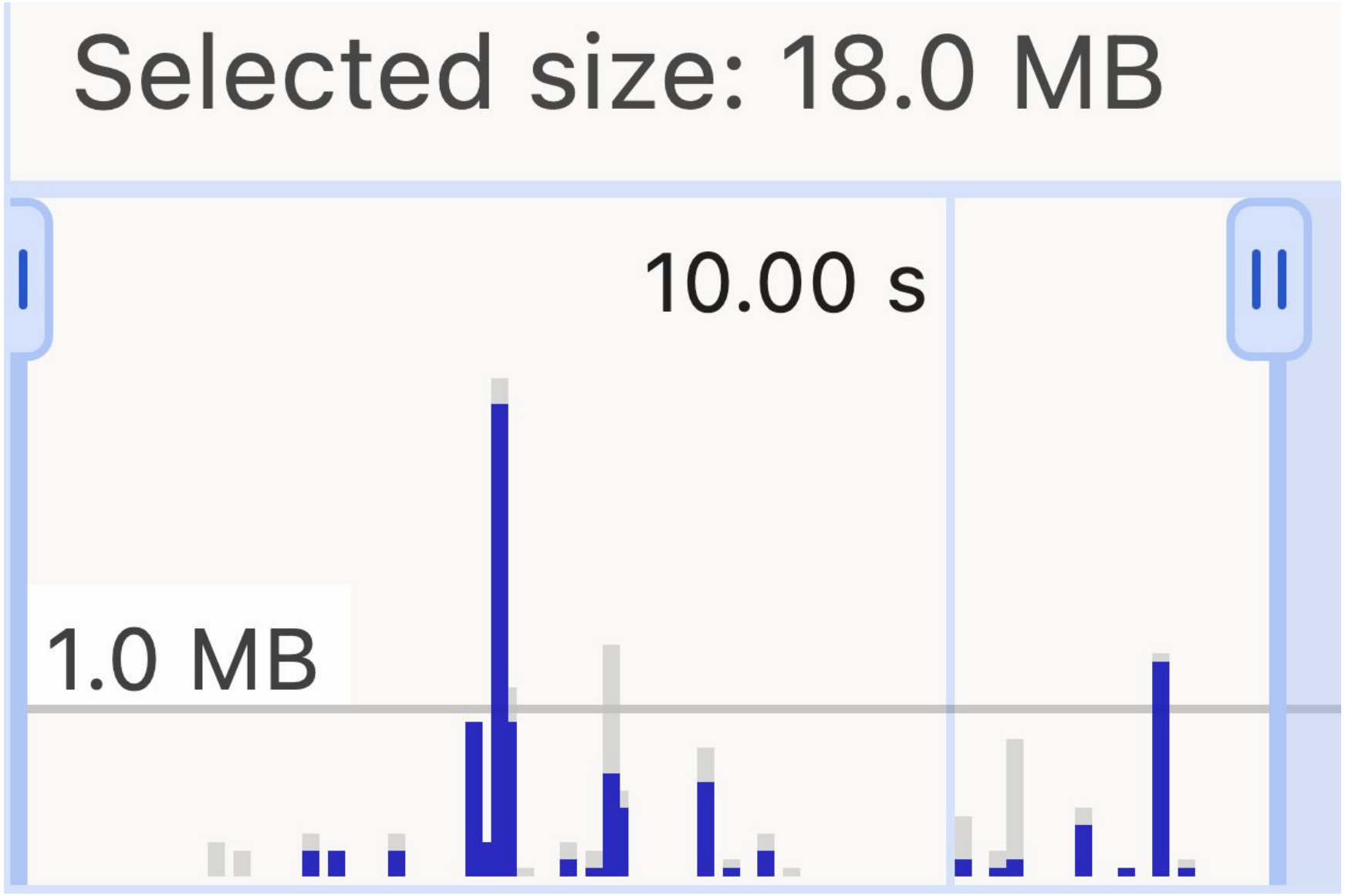

Elements
Recorder
Console
Memory
1374
13
Profiles
JavaScript objects and related DOM nodes.
Include numerical values in capture
Allocation instrumentation on timeline
Allocation timelines show instrumented JavaScript memory allocations over time. Once profile is recorded you can select a time interval to see objects that were allocated within it and still alive by the end of recording. Use this profile type to isolate memory leaks.
Record stack traces of allocations (extra performance overhead)

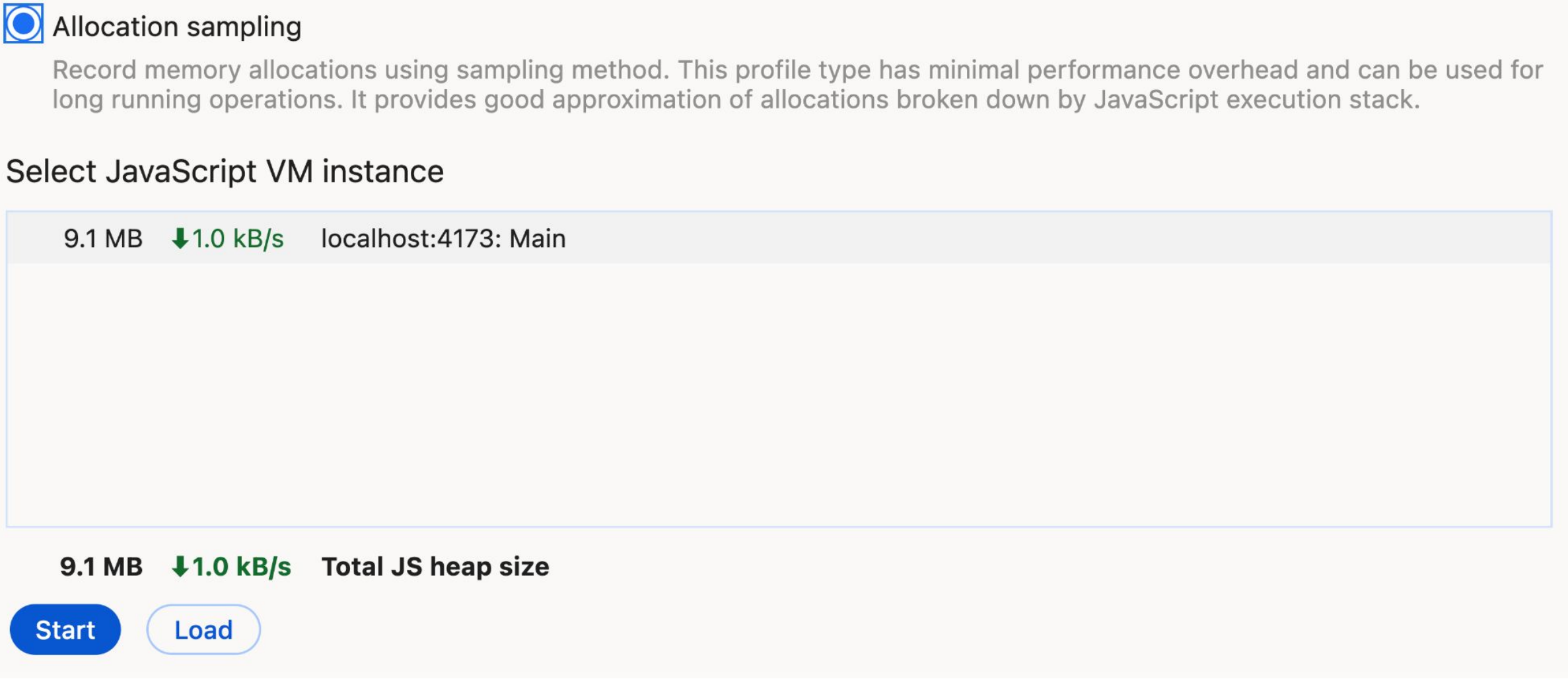

Elements
Console
Network
Performance
Sources
Memory
Lighthouse
14
8
Summary
Class filter
Selected size: 9.3 MB
5.00 s
10.00 s
15.00 s
20.00 s
25.00 s
30.00 s
102 kB
Profiles
ALLOCATION TIMELINES
Snapshot 1
10.3 MB
Constructor
Distance
Shallow Size
Retained Size
(compiled code) ×59305 3 4 414 808 43 % 4 720 020 46 %
system / Context ×4233 3 142 108 1 % 2 260 916 22 %
(system) ×32571 2 927 264 9 % 2 012 116 19 %
Object ×8755 2 227 760 2 % 1 682 652 16 %
(array) ×5172 2 1 098 272 11 % 1 473 316 14 %
(object shape) ×17043 2 896 888 9 % 931 920 9 %
InternalNode ×9916 3 0 0 % 926 112 9 %
(string) ×21968 2 586 592 6 % 586 632 6 %

Constructor	Distance	Shallow Size	Retained Size
▶ Object @411213	20	16 0 %	16 0 %
▼ Object @411229	46	16 0 %	16 0 %
▶ map :: system / Map @80353	7	40 0 %	320 0 %
▶ __proto__ :: Object @20037	3	28 0 %	240 0 %
▶ current :: system / Oddball @67	2	28 0 %	28 0 %
▶ Object @411863	6	16 0 %	16 0 %
▶ Object @411883	6	16 0 %	16 0 %
▶ Object @412219	6	16 0 %	16 0 %

Retainers | Allocation stack

useRef	react-dom.production.min.js:184
ht.useRef	react.production.min.js:25
im	downshift.esm.js:2035
IT	downshift.esm.js:2478
(anonymous)	select.jsx:22
Ev	react-dom.production.min.js:167
ty	react-dom.production.min.js:193
vS	react-dom.production.min.js:290
TO	react-dom.production.min.js:280
Su	react-dom.production.min.js:280
IS	react-dom.production.min.js:268
O	scheduler.production.min.js:13
j	scheduler.production.min.js:14


Selected size: 18.0 MB
10.00 s
1.0 MB


Allocation sampling
Record memory allocations using sampling method. This profile type has minimal performance overhead and can be used for long running operations. It provides good approximation of allocations broken down by JavaScript execution stack.
Select JavaScript VM instance
9.1 MB ⬇1.0 kB/s localhost:4173: Main
9.1 MB ⬇1.0 kB/s Total JS heap size
Start
Load

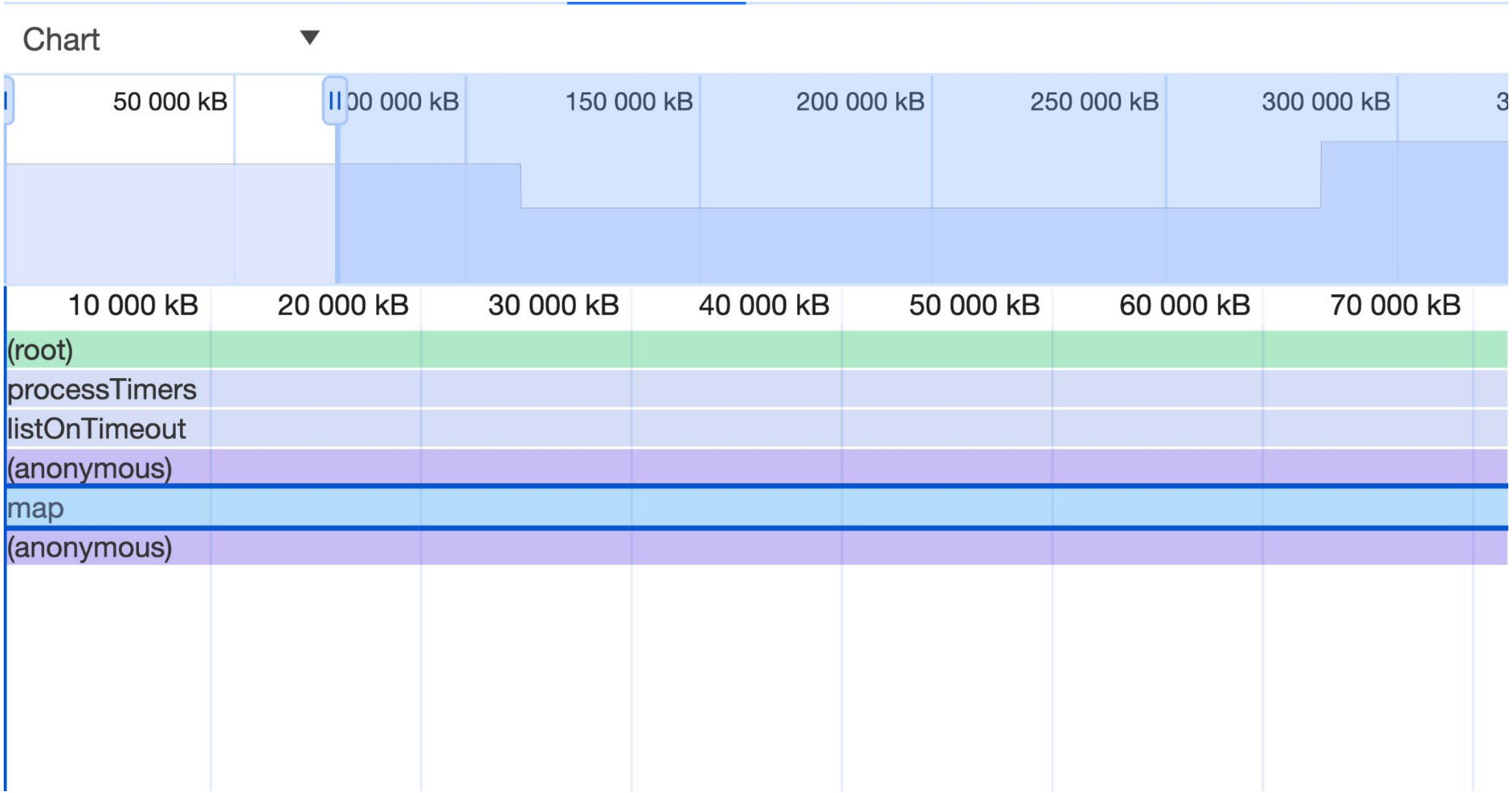
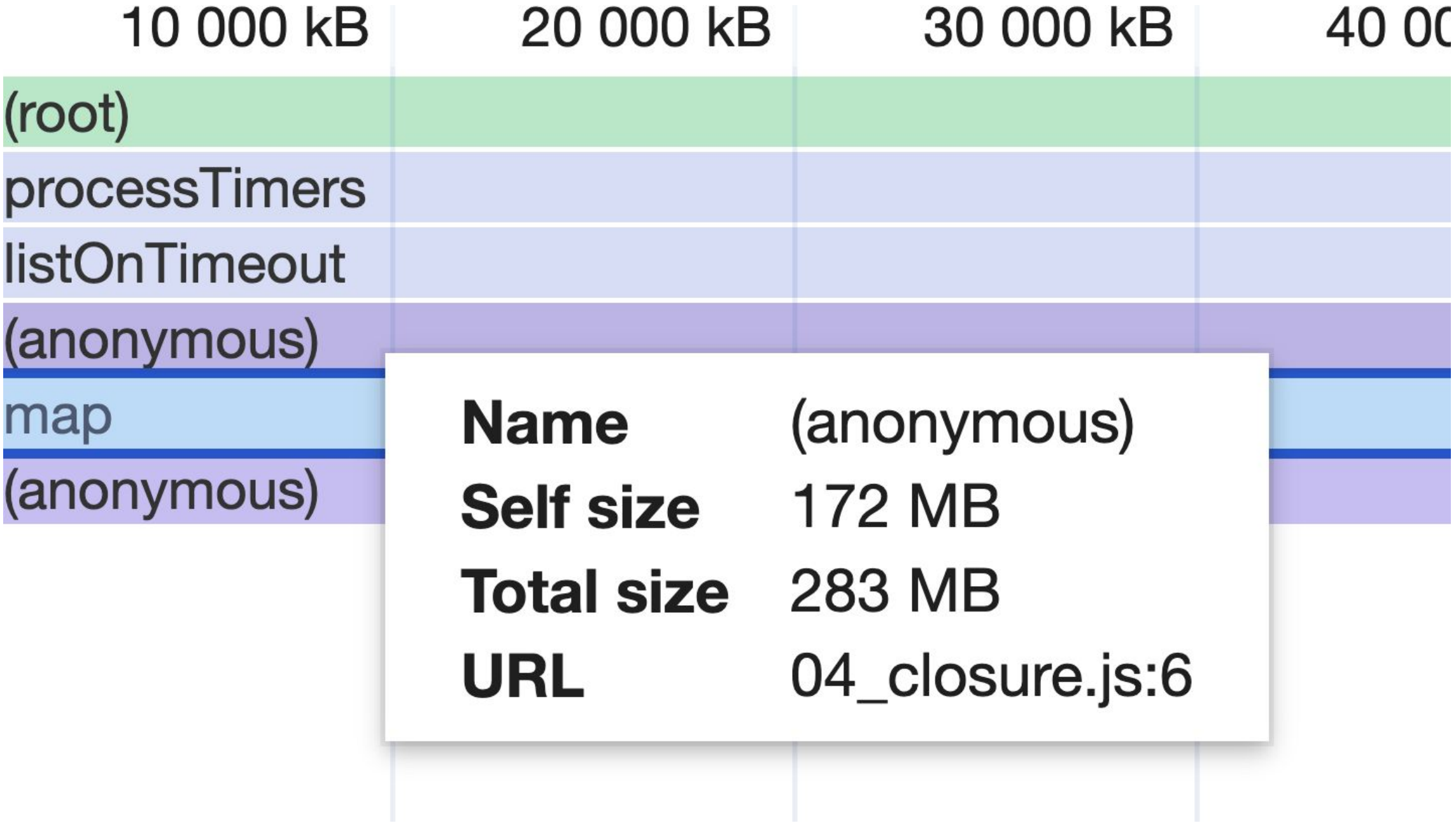


Chart
50 000 kB
00 000 kB
150 000 kB
200 000 kB
250 000 kB
300 000 kB
10 000 kB
20 000 kB
30 000 kB
40 000 kB
50 000 kB
60 000 kB
70 000 kB
(root)
processTimers
listOnTimeout
(anonymous)
map
(anonymous)

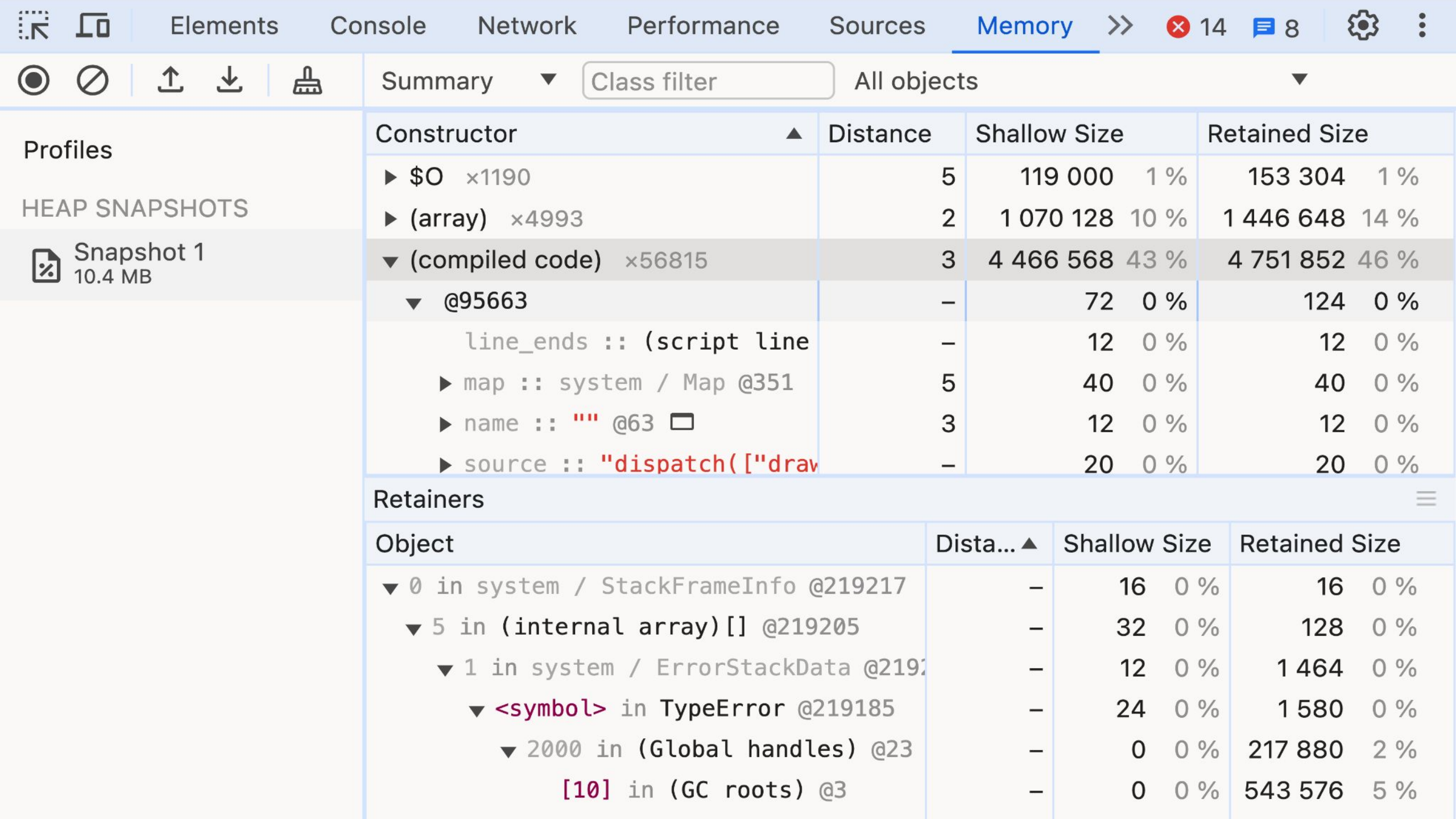

10 000 kB
20 000 kB
30 000 kB
40 00
(root)
processTimers
listOnTimeout
(anonymous)
map
(anonymous)
Name (anonymous)
Self size 172 MB
Total size 283 MB
URL 04_closure.js:6

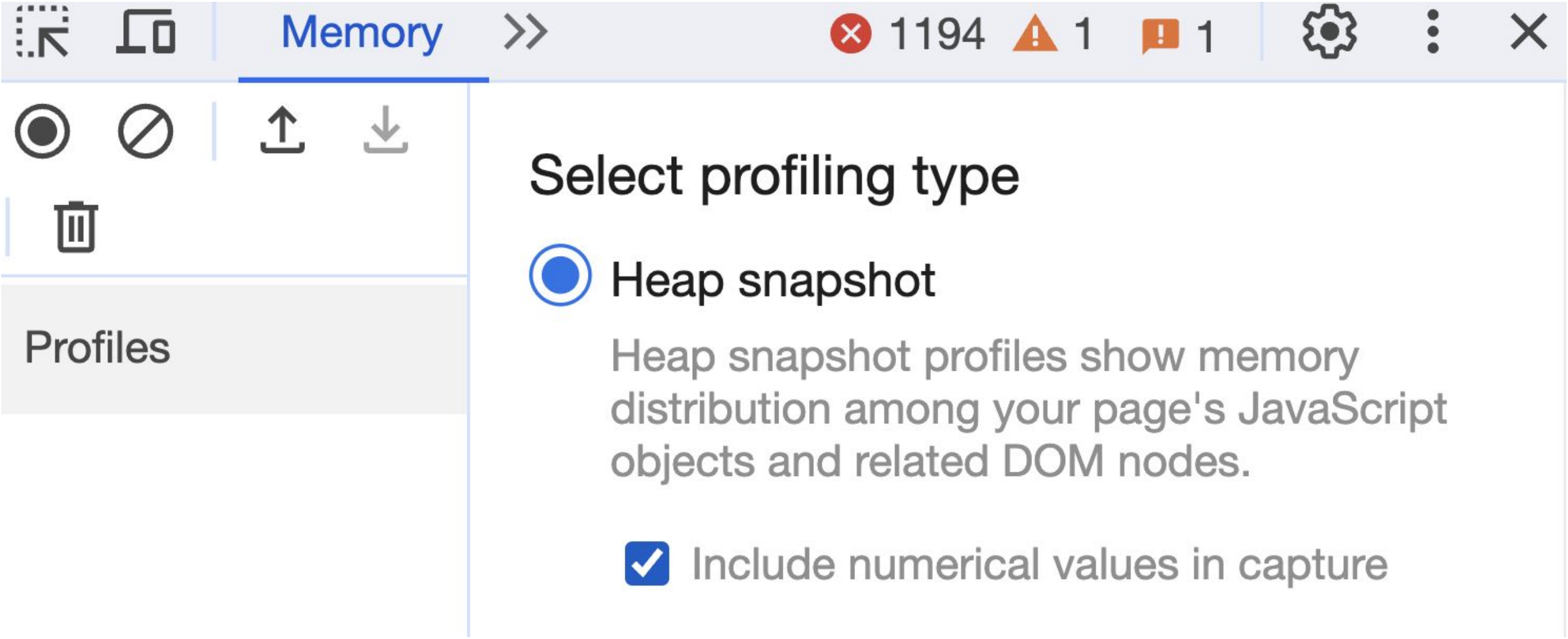
Heavy (Bottom Up) ▼

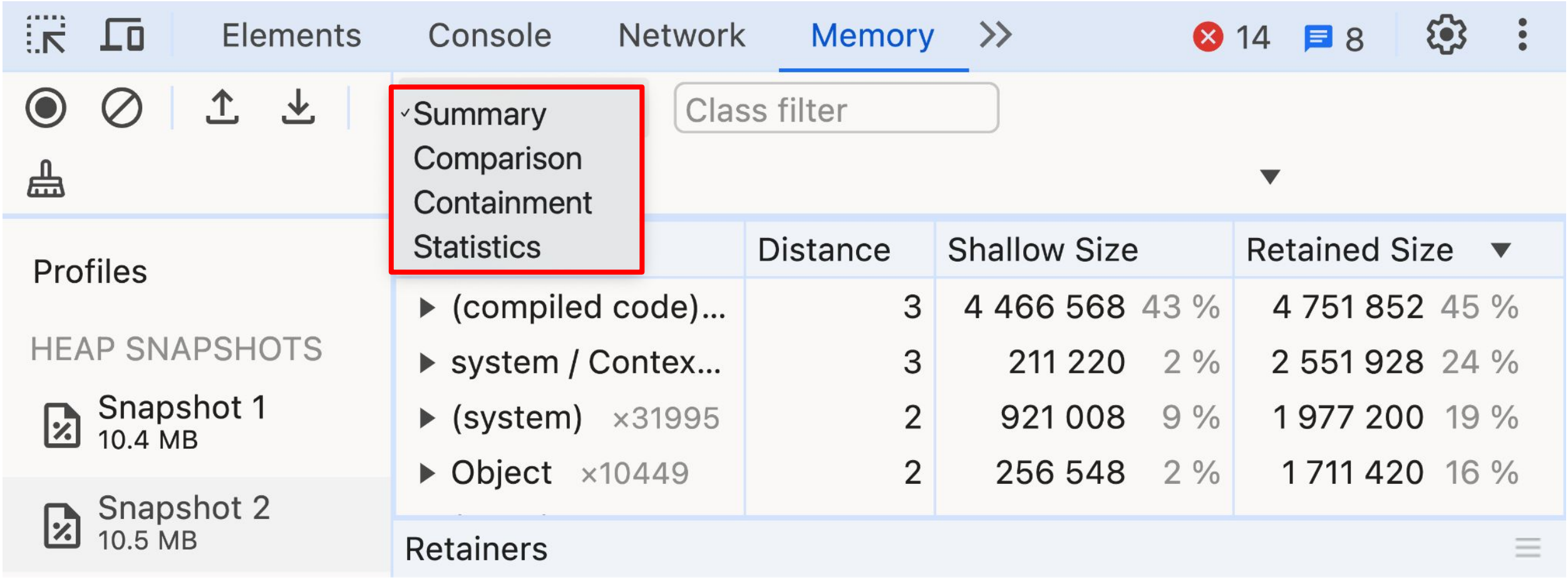
Self Size (bytes)		Total Size (bytes)		Function	
71 961 344	53.11 %	23 776 992	99.99 %	▼ (anonymous)	04_closure.js:6
71 961 344	53.11 %	83 452 768	87.54 %	▶ listOnTimeout	node:internal/timers:517
0	0.00 %	40 324 224	12.45 %	▼ (anonymous)	04_closure.js:9
0	0.00 %	40 324 224	12.45 %	▼ listOnTimeout	node:internal/timers:517
0	0.00 %	40 324 224	12.45 %	processTimers	node:internal/timers:497
51 815 648	46.89 %	51 815 648	46.89 %	▶ (anonymous)	04_closure.js:6
18 368	0.01 %	23 795 360	100.00 %	processTimers	node:internal/timers:497
0	0.00 %	23 776 992	99.99 %	▶ listOnTimeout	node:internal/timers:517
0	0.00 %	40 324 224	12.45 %	▶ (anonymous)	04_closure.js:9
0	0.00 %	51 815 648	46.89 %	▶ map	

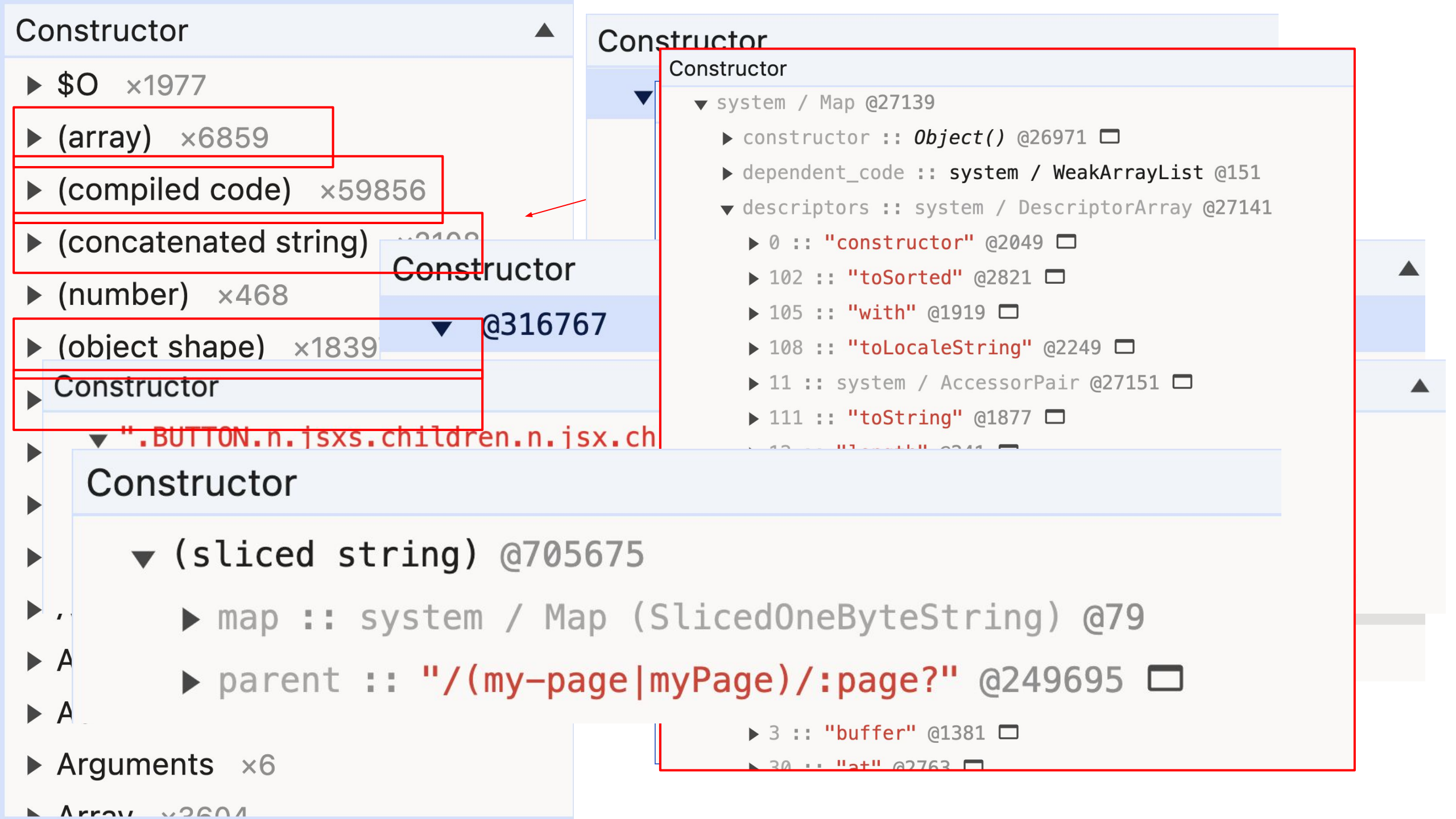
Tree (Top Down)

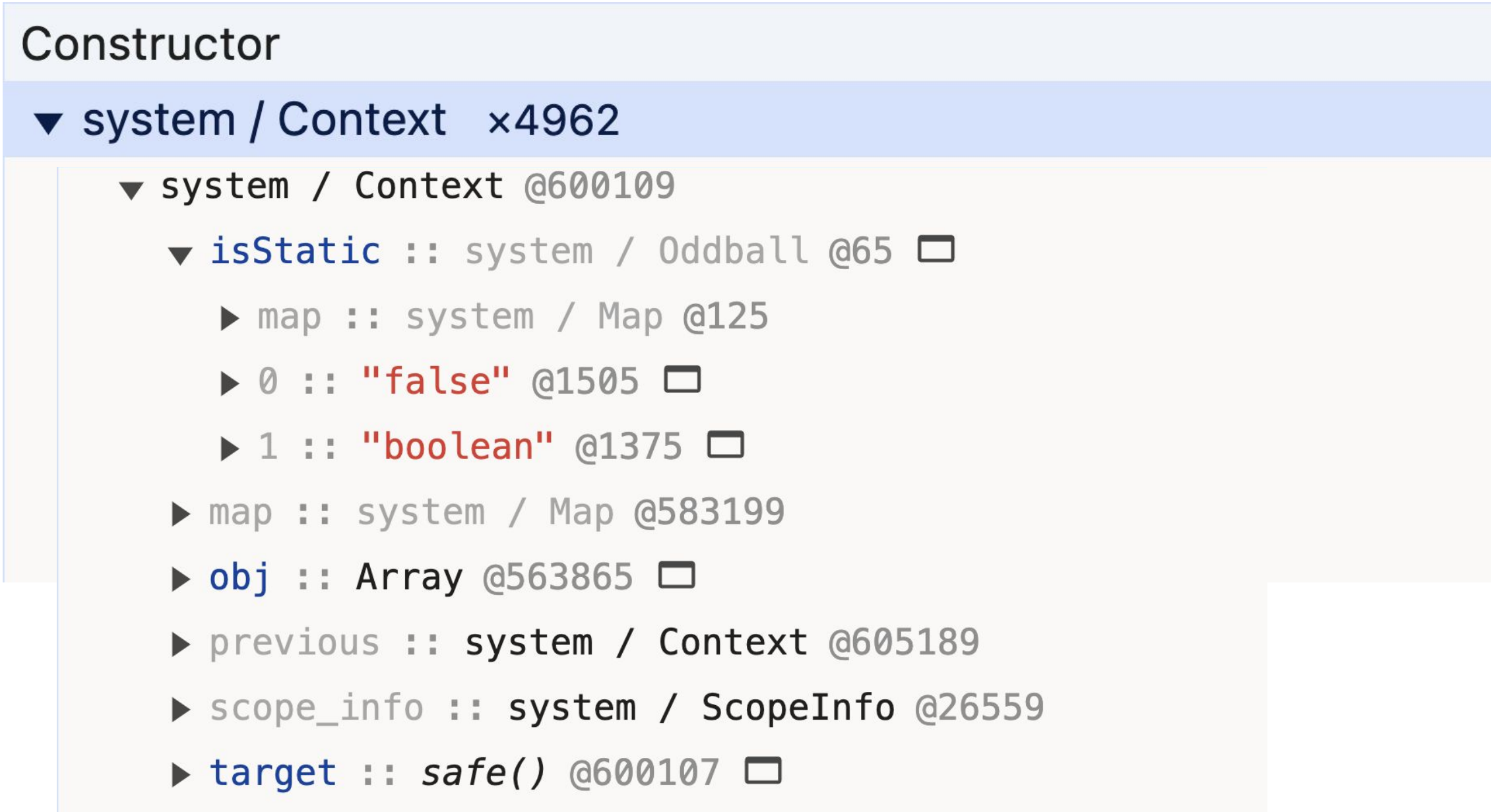
Self Size (bytes)		Total Size (bytes)		Function	
18 368	0.01 %	23 795 360	100.00 %	▼ processTimers	node:internal/timers:497
0	0.00 %	23 776 992	99.99 %	▼ listOnTimeout	node:internal/timers:517
71 961 344	53.11 %	83 452 768	87.54 %	▼ (anonymous)	04_closure.js:6
0	0.00 %	11 491 424	34.43 %	▼ map	
11 491 424	34.43 %	11 491 424	34.43 %	(anonymous)	04_closure.js:6
0	0.00 %	40 324 224	12.45 %	▼ (anonymous)	04_closure.js:9
0	0.00 %	40 324 224	12.45 %	▼ (anonymous)	04_closure.js:6
0	0.00 %	40 324 224	12.45 %	▼ map	
40 324 224	12.45 %	40 324 224	12.45 %	(anonymous)	04_closure.js:6


Elements
Console
Network
Performance
Sources
Memory
14
8
Summary
Class filter
All objects
Profiles
HEAP SNAPSHOTS
Snapshot 1
10.4 MB
Constructor
Distance
Shallow Size
Retained Size
$O ×1190 5 119 000 1 % 153 304 1 %
(array) ×4993 2 1 070 128 10 % 1 446 648 14 %
(compiled code) ×56815 3 4 466 568 43 % 4 751 852 46 %
@95663 – 72 0 % 124 0 %
line_ends :: (script line – 12 0 % 12 0 %
map :: system / Map @351 5 40 0 % 40 0 %
name :: "" @63 3 12 0 % 12 0 %
source :: "dispatch(["drav – 20 0 % 20 0 %
Retainers
Object
Dista...
Shallow Size
Retained Size
0 in system / StackFrameInfo @219217 – 16 0 % 16 0 %
5 in (internal array)[] @219205 – 32 0 % 128 0 %
1 in system / ErrorStackData @219 – 12 0 % 1 464 0 %
<symbol> in TypeError @219185 – 24 0 % 1 580 0 %
2000 in (Global handles) @23 – 0 0 % 217 880 2 %
[10] in (GC roots) @3 – 0 0 % 543 576 5 %


Memory
1194
1
1
Profiles
Select profiling type
Heap snapshot
Heap snapshot profiles show memory distribution among your page's JavaScript objects and related DOM nodes.
Include numerical values in capture


Elements
Console
Network
Memory
14
8
Summary
Comparison
Containment
Statistics
Class filter
Profiles
HEAP SNAPSHOTS
Snapshot 1
10.4 MB
Snapshot 2
10.5 MB
Distance
Shallow Size
Retained Size
(compiled code)...
3
4 466 568 43 %
4 751 852 45 %
system / Contex...
3
211 220 2 %
2 551 928 24 %
(system) ×31995
2
921 008 9 %
1 977 200 19 %
Object ×10449
2
256 548 2 %
1 711 420 16 %
Retainers

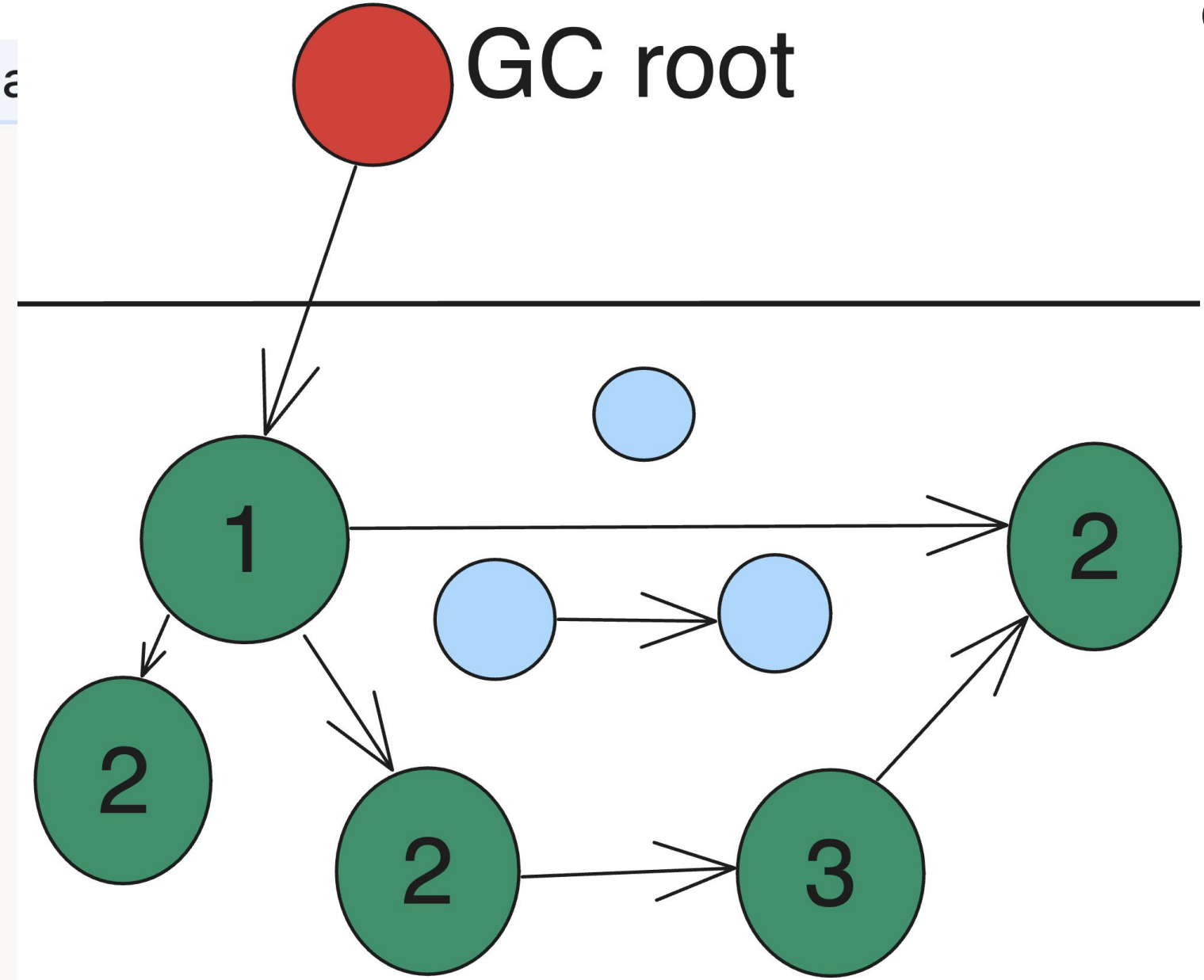

Constructor
$O ×1977
(array) ×6859
(compiled code) ×59856
(concatenated string)
(number) ×468
(object shape) ×1839
Constructor
@316767
Constructor
".BUTTON.n.jsxs.children.n.jsx.ch
Constructor
(sliced string) @705675
map :: system / Map (SlicedOneByteString) @79
parent :: "/(my-page|myPage)/:page?" @249695
Arguments ×6
Constructor
system / Map @27139
constructor :: Object() @26971
dependent_code :: system / WeakArrayList @151
descriptors :: system / DescriptorArray @27141
0 :: "constructor" @2049
102 :: "toSorted" @2821
105 :: "with" @1919
108 :: "toLocaleString" @2249
11 :: system / AccessorPair @27151
111 :: "toString" @1877
3 :: "buffer" @1381


Constructor
system / Context ×4962
system / Context @600109
isStatic :: system / Oddball @65
map :: system / Map @125
0 :: "false" @1505
1 :: "boolean" @1375
map :: system / Map @583199
obj :: Array @563865
previous :: system / Context @605189
scope_info :: system / ScopeInfo @26559
target :: safe() @600107

Constructor

```
▼ Object @602931
  ▶ __DO_NOT_USE__ActionTypes :: Object @603013
  ▶ __esModule :: system / Oddball @67
  ▶ applyMiddleware :: $m() @602951
  ▶ bindActionCreators :: Vm() @602947
  ▶ combineReducers :: Km() @602943
  ▶ compose :: ms() @602949
  ▶ createStore :: Bt() @602939
  ▶ legacy_createStore :: Bt() @602939
  ▶ map :: system / Map @599369
  ▶ properties :: system / PropertyArray @603031
  ▶ __proto__ :: Object @564097
```

Distance	Sha
3	
3	
2	
2	
2	
2	
–	
2	
7	
2	
4	3 616 0 % 408 016 4 %



Summary ▼ Class filter

✓ All objects
Objects allocated before s1
Objects allocated between s1 and s2

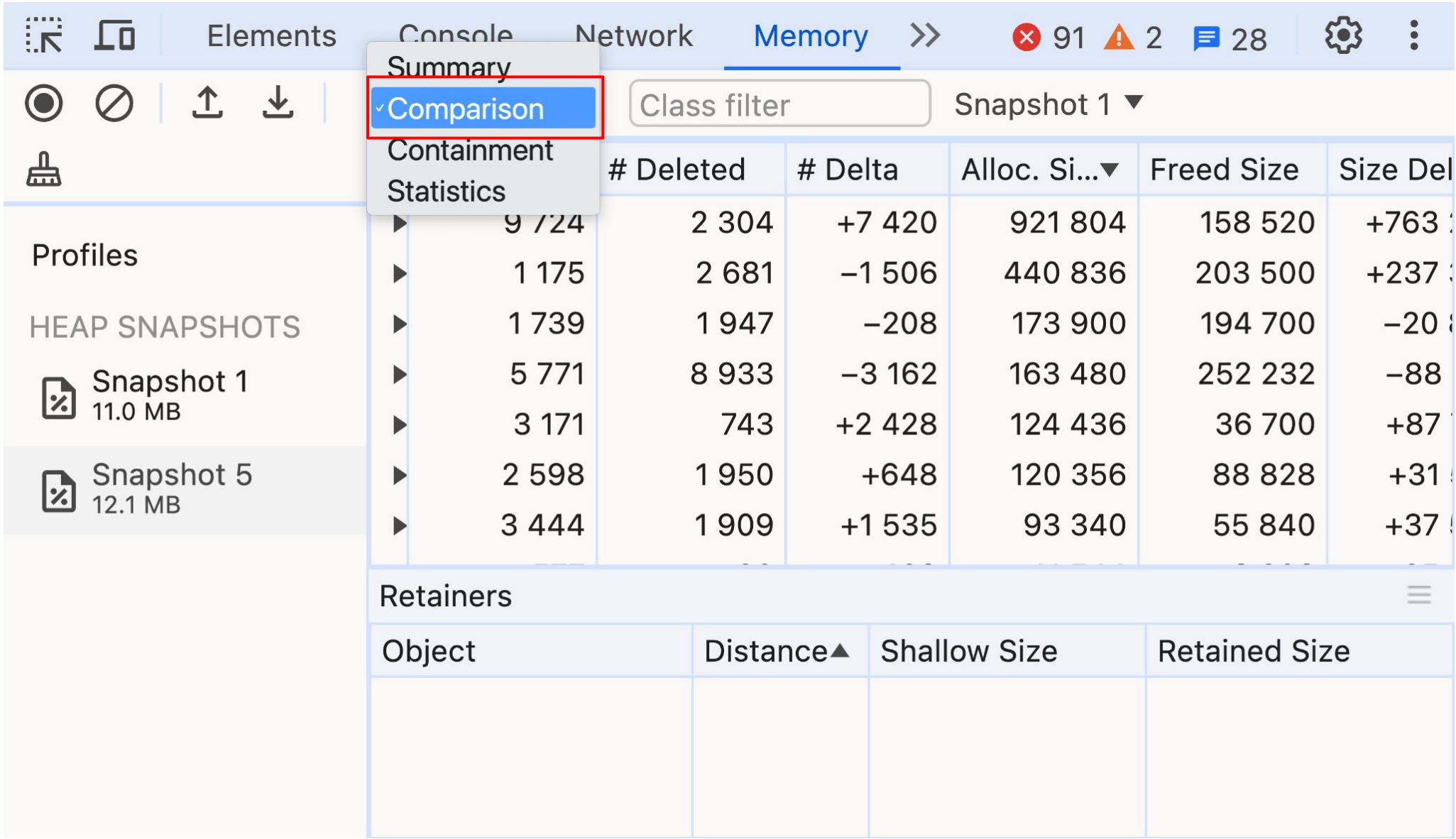
Constructor		allow Size	Retained Size
▶ global /		40 0 %	53 651 031 99 %
▶ (closure) ×536841	2	30 080 024 55 %	52 535 360 97 %
▶ system / Context ×531720	3	21 302 344 39 %	51 446 864 95 %

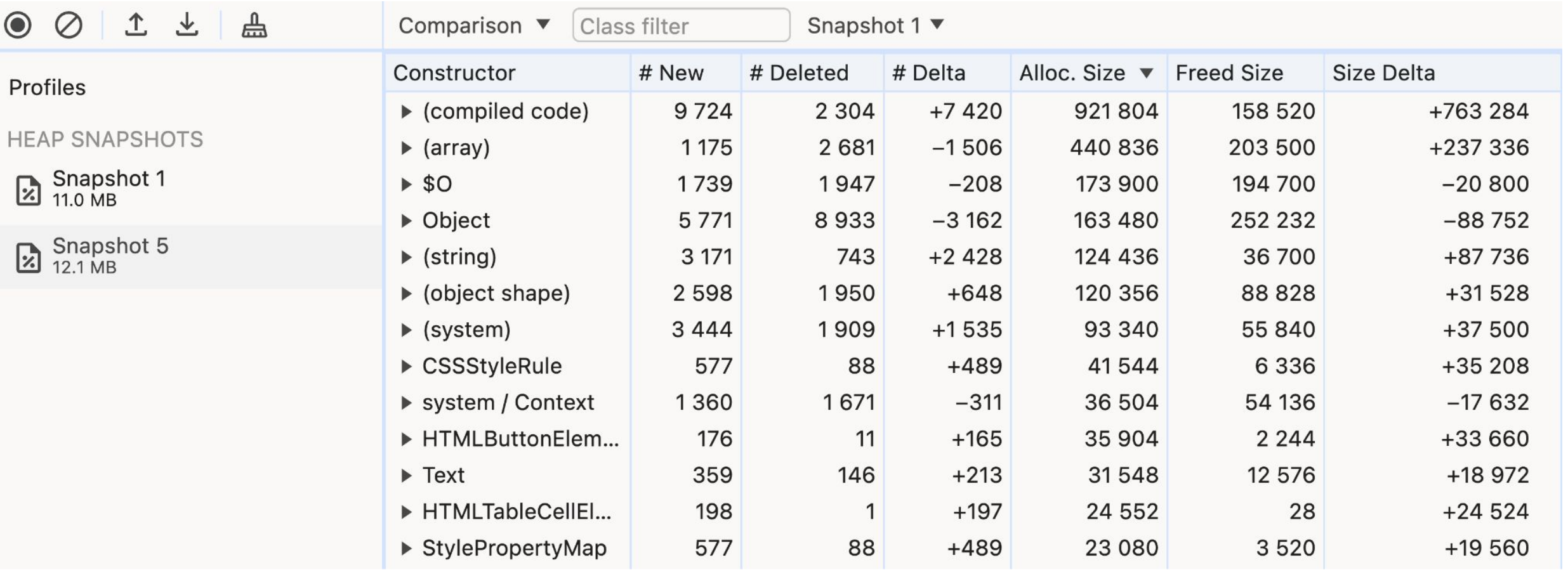
Summary ▼ Class filter Objects allocated between s1 and s2 ▼

Constructor	Distance	Shallow Size	Retained Size
▶ (closure) ×586001	9	32 816 056 30 %	107 240 520 97 %
▶ system / Context ×586001	10	23 440 040 21 %	107 240 464 97 %
▶ Array	11	32 0 %	107 240 424 97 %
▶ (array)	12	8 016 0 %	8 016 0 %

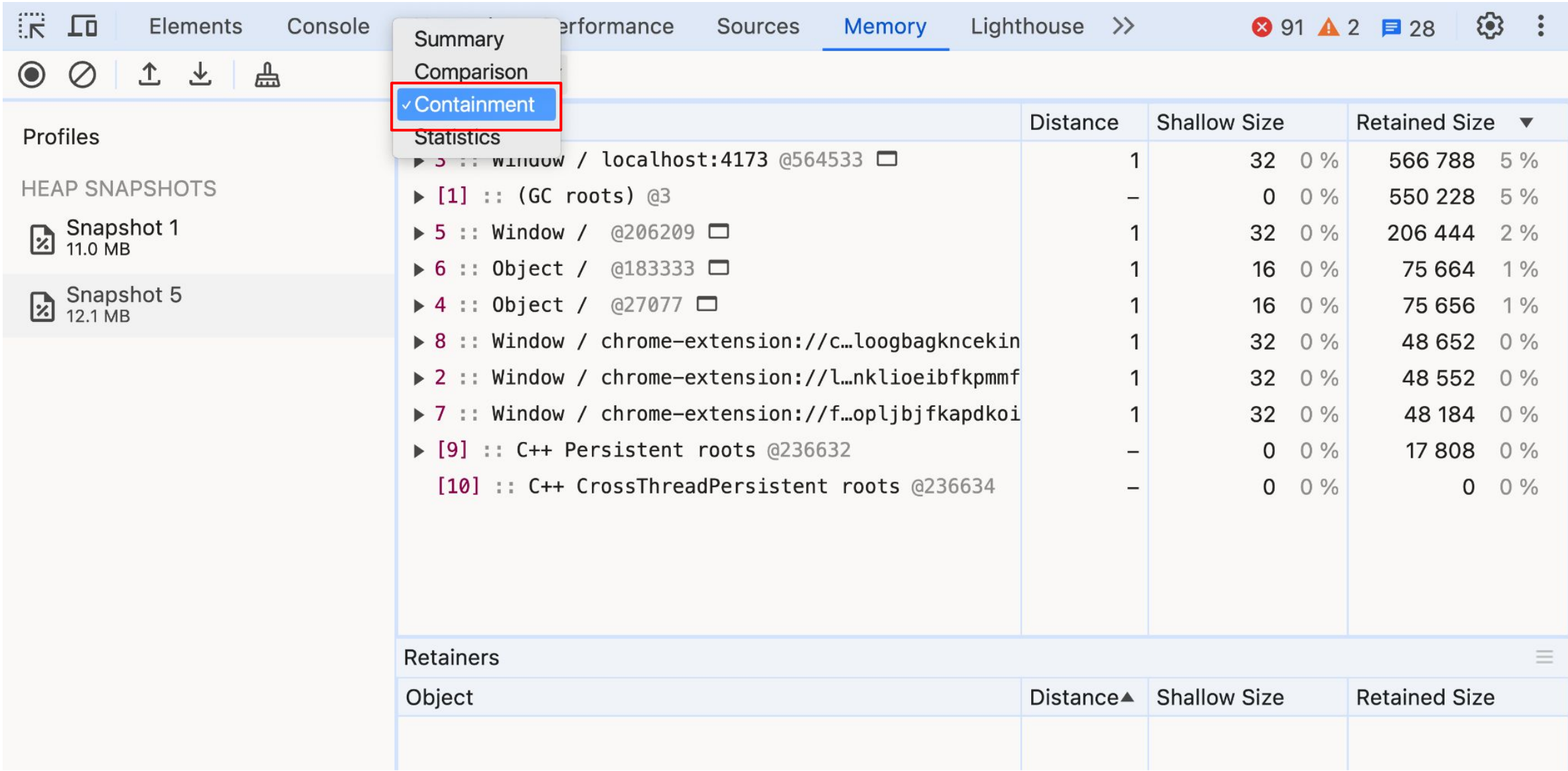
Retainers

Object	Distance	Shallow Size		Retained Size	
▼ legacy_createStore in Object @602931	8	28	0 %	2 016	0 %
▼ t in system / Context @603035	7	20	0 %	2 036	0 %
▼ previous in system / Context @603065	6	20	0 %	20	0 %
▼ context in *get()* @603067	5	28	0 %	48	0 %
▼ get legacy_createStore in Object @603033	4	28	0 %	2 976	0 %
▼ xu in system / Context @603543	3	1 016	0 %	93 992	1 %
▼ context in *Uh()* @603651	2	32	0 %	148	0 %
▶ __REDUX_DEVTOOLS_EXTENSION_COMPOSE__ in Window /	1	32	0 %	54 460	0 %
▶ value in system / PropertyCell @569683	3	20	0 %	20	0 %
▶ value in system / Cell @642383	7	8	0 %	8	0 %
▶ context in *pr()* @601873	2	32	0 %	256	0 %
▶ context in *ur()* @603649	2	32	0 %	880	0 %
▶ context in *Lu()* @603641	3	32	0 %	108	0 %
▶ context in *Nu()* @603643	3	32	0 %	156	0 %
▶ context in *Qe()* @603615	3	32	0 %	108	0 %
▶ context in *Ye()* @603605	3	32	0 %	148	0 %
▶ context in *or()* @603639	3	32	0 %	148	0 %
▶ context in *qe()* @603635	3	32	0 %	108	0 %
▶ context in *Ah()* @603567	4	32	0 %	108	0 %


Elements
Console
Network
Memory
91
2
28
Summary
Comparison
Containment
Statistics
Class filter
Snapshot 1
Profiles
HEAP SNAPSHOTS
Snapshot 1
11.0 MB
Snapshot 5
12.1 MB
Deleted
Delta
Alloc. Si...
Freed Size
Size Del
9 724
2 304
+7 420
921 804
158 520
+763
1 175
2 681
−1 506
440 836
203 500
+237
1 739
1 947
−208
173 900
194 700
−20
5 771
8 933
−3 162
163 480
252 232
−88
3 171
743
+2 428
124 436
36 700
+87
2 598
1 950
+648
120 356
88 828
+31
3 444
1 909
+1 535
93 340
55 840
+37
Retainers
Object
Distance
Shallow Size
Retained Size

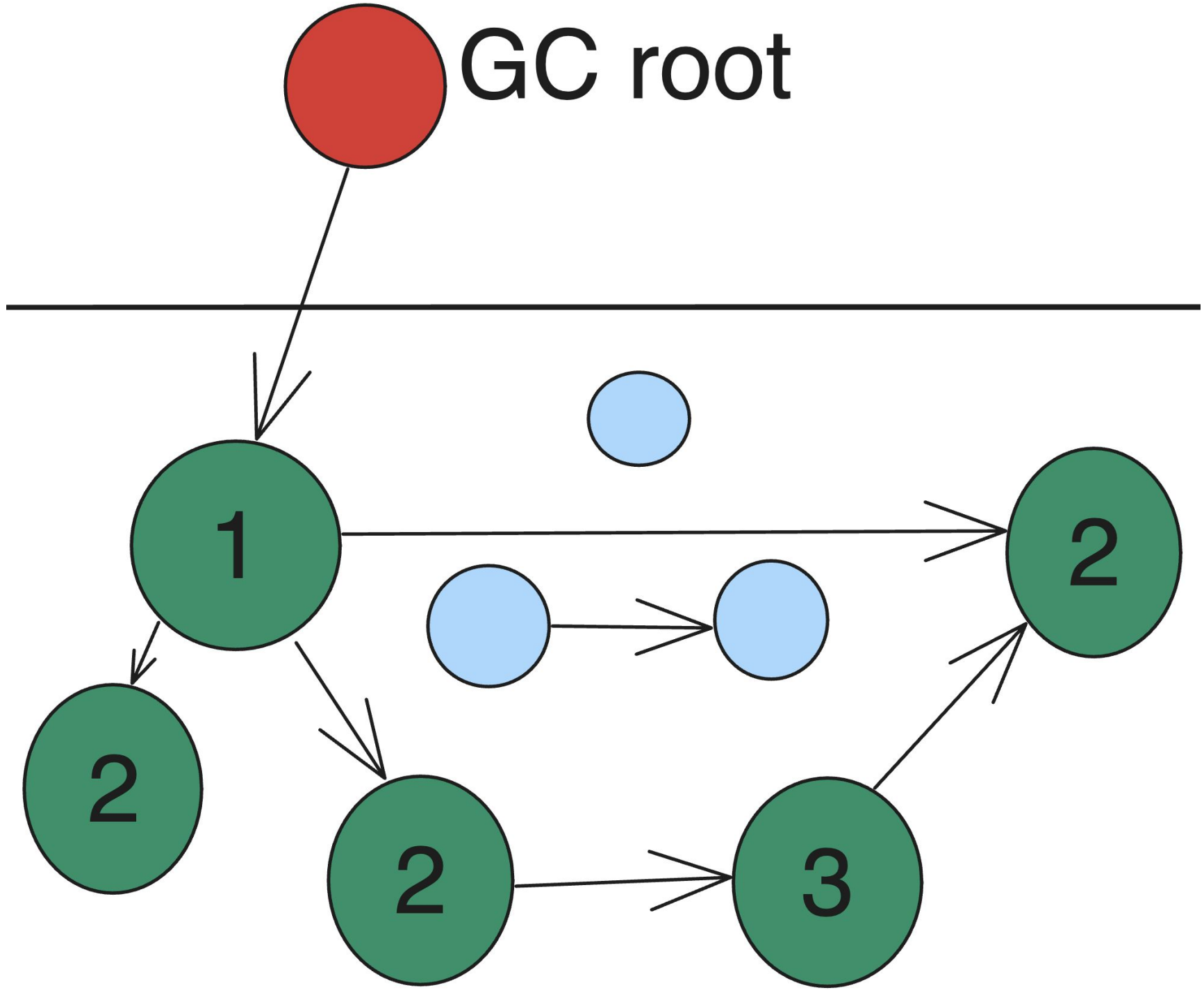


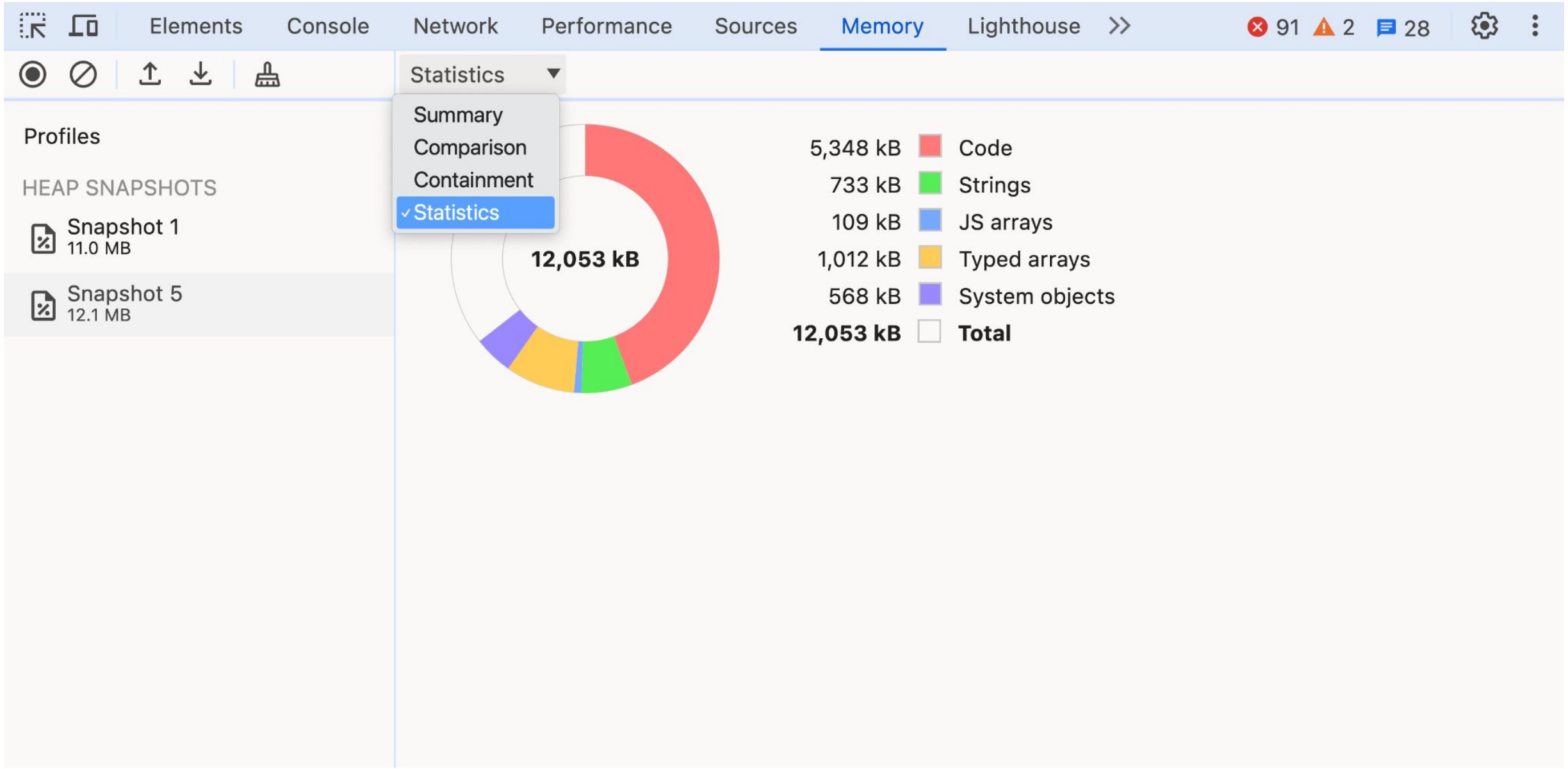
Constructor	# New	# Deleted	# Delta	Alloc. Size	Freed Size	Size Delta
▶ (compiled code)	9 724	2 304	+7 420	921 804	158 520	+763 284
▶ (array)	1 175	2 681	−1 506	440 836	203 500	+237 336
▶ $O	1 739	1 947	−208	173 900	194 700	−20 800
▶ Object	5 771	8 933	−3 162	163 480	252 232	−88 752
▶ (string)	3 171	743	+2 428	124 436	36 700	+87 736
▶ (object shape)	2 598	1 950	+648	120 356	88 828	+31 528
▶ (system)	3 444	1 909	+1 535	93 340	55 840	+37 500
▶ CSSStyleRule	577	88	+489	41 544	6 336	+35 208
▶ system / Context	1 360	1 671	−311	36 504	54 136	−17 632
▶ HTMLButtonElem…	176	11	+165	35 904	2 244	+33 660
▶ Text	359	146	+213	31 548	12 576	+18 972
▶ HTMLTableCellEl…	198	1	+197	24 552	28	+24 524
▶ StylePropertyMap	577	88	+489	23 080	3 520	+19 560


Elements
Console
Performance
Sources
Memory
Lighthouse
91
2
28
Summary
Comparison
Containment
Statistics
Profiles
HEAP SNAPSHOTS
Snapshot 1
11.0 MB
Snapshot 5
12.1 MB
Distance
Shallow Size
Retained Size
3 :: Window / localhost:4173 @564533 1 32 0 % 566 788 5 %
[1] :: (GC roots) @3 – 0 0 % 550 228 5 %
5 :: Window / @206209 1 32 0 % 206 444 2 %
6 :: Object / @183333 1 16 0 % 75 664 1 %
4 :: Object / @27077 1 16 0 % 75 656 1 %
8 :: Window / chrome-extension://c…loogbagkncekin 1 32 0 % 48 652 0 %
2 :: Window / chrome-extension://l…nklioeibfkpmmf 1 32 0 % 48 552 0 %
7 :: Window / chrome-extension://f…opljbjfkapdkoi 1 32 0 % 48 184 0 %
[9] :: C++ Persistent roots @236632 – 0 0 % 17 808 0 %
[10] :: C++ CrossThreadPersistent roots @236634 – 0 0 % 0 0 %
Retainers
Object
Distance
Shallow Size
Retained Size

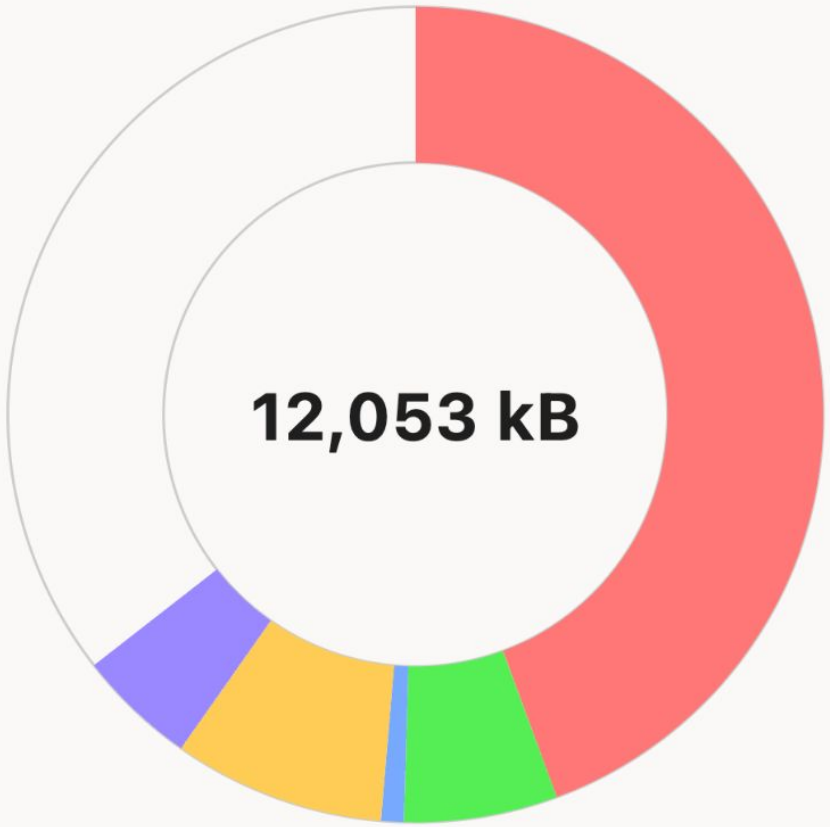
Containment ▼

Object	Distance	Shallow Size	Retained Size
▶ 2 :: global / @6275	1	40 0 %	109 908 519 99 %
▶ [1] :: (GC roots) @3	–	0 0 %	496 152 0 %
▶ 3 :: Object / @6375	1	40 0 %	147 456 0 %
▶ [6] :: Node / Environment @146	–	2 408 0 %	9 218 0 %
[4] :: C++ Persistent roots @142	–	0 0 %	0 0 %
[5] :: C++ CrossThreadPersistent roots @144	–	0 0 %	0 0 %


GC root
1
2
2
2
3


Elements
Console
Network
Performance
Sources
Memory
Lighthouse
91
2
28
Statistics
Summary
Comparison
Containment
Statistics
Profiles
HEAP SNAPSHOTS
Snapshot 1
11.0 MB
Snapshot 5
12.1 MB
12,053 kB
5,348 kB Code
733 kB Strings
109 kB JS arrays
1,012 kB Typed arrays
568 kB System objects
12,053 kB Total

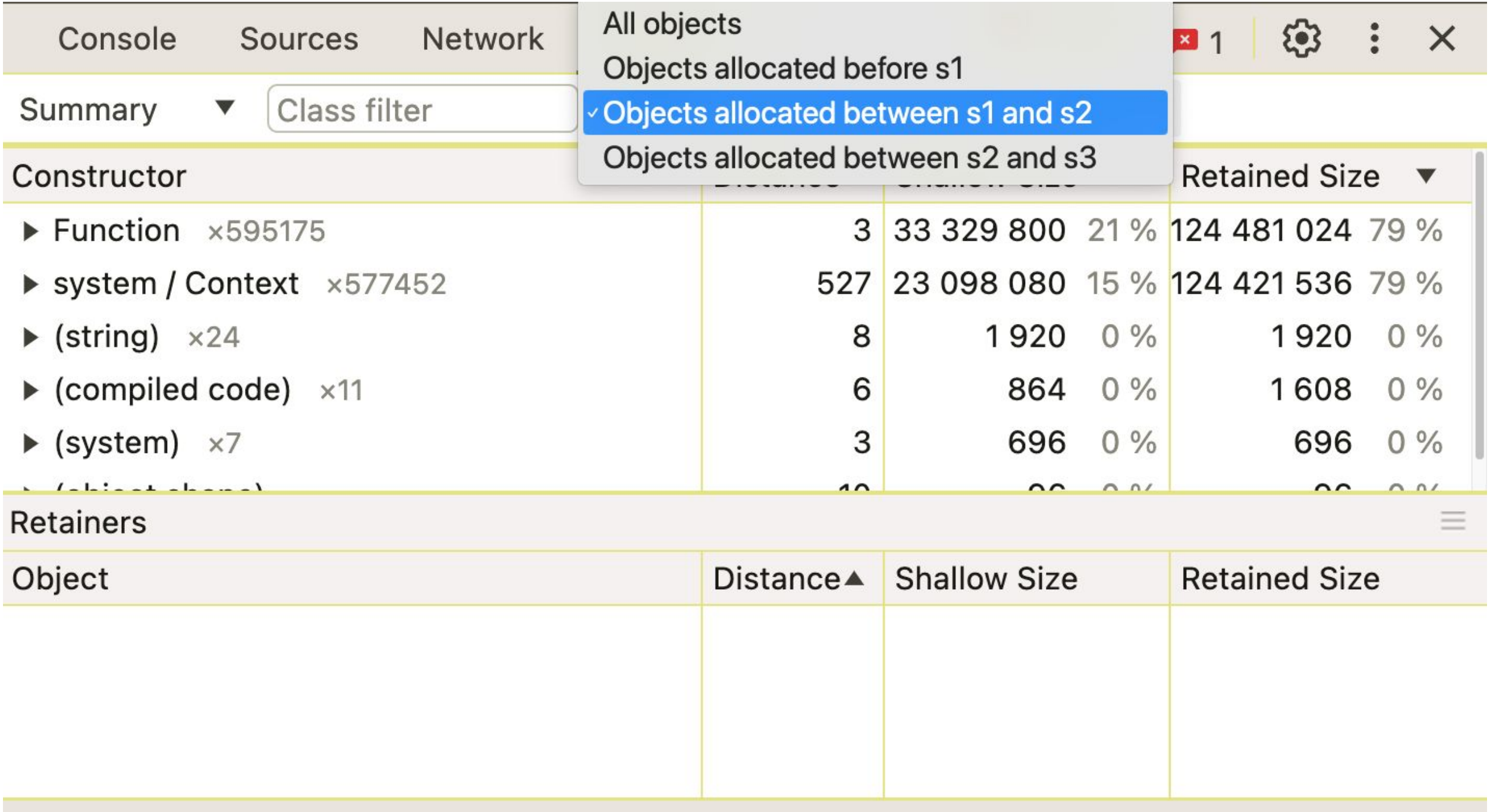
Statistics

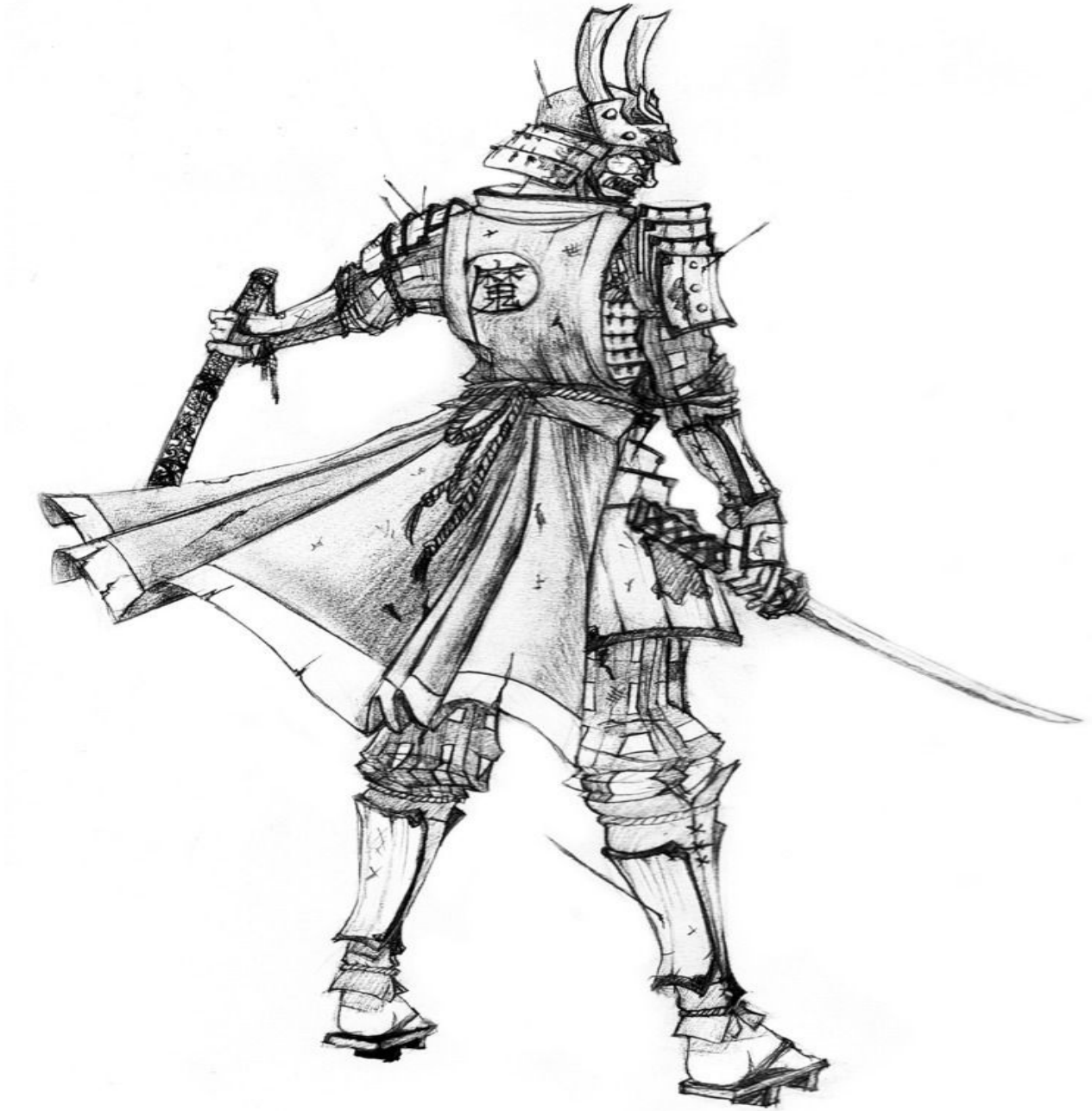


5,348 kB		Code
733 kB		Strings
109 kB		JS arrays
1,012 kB		Typed arrays
568 kB		System objects
12,053 kB		**Total**

Техника трех снэпшотов

- ➜ Прогреваем
- ➜ Делаем первый снэпшот
- ➜ Повторяем несколько раз сценарий
- ➜ Делаем второй снэпшот
- ➜ Повторяем несколько раз сценарий
- ➜ Делаем третий снэпшот
- ➜ Выбираем третий снэпшот в summary и выбираем объекты аллоцированные между первым и вторым снэпшотами


Console
Sources
Network
All objects
Objects allocated before s1
Objects allocated between s1 and s2
Objects allocated between s2 and s3
1
Summary
Class filter
Constructor
Retained Size
Function ×595175 3 33 329 800 21 % 124 481 024 79 %
system / Context ×577452 527 23 098 080 15 % 124 421 536 79 %
(string) ×24 8 1 920 0 % 1 920 0 %
(compiled code) ×11 6 864 0 % 1 608 0 %
(system) ×7 3 696 0 % 696 0 %
Retainers
Object
Distance
Shallow Size
Retained Size



Советы напоследок

- Пробуйте снепшоты
- Мониторьте память
- Кеши - надо чистить)))

Полезные ссылки

1. https://github.com/HowProgrammingWorks/MemoryLeaks
2. https://www.chromium.org/developers/how-tos/trace-event-profiling-tool/ – трассировка приложения
3. https://nodejs.org/en/docs/guides/diagnostics/memory/using-heap-snapshot - node js о снепшоте
4. https://sematext.com/blog/nodejs-memory-leaks/ - хорошая статья, есть еще доп инструменты дополняем
5. https://deepu.tech/memory-management-in-v8/ - визуализация памяти в v8
6. https://developer.chrome.com/docs/devtools/memory-problems/heap-snapshots - chrome о снепшотах и их чтении
7. https://codepen.io/web-dot-dev/live/oNVPRZw - поиграться со снепшотом