How to Use the Allocation Profiler Tool

By Meggin Kearney

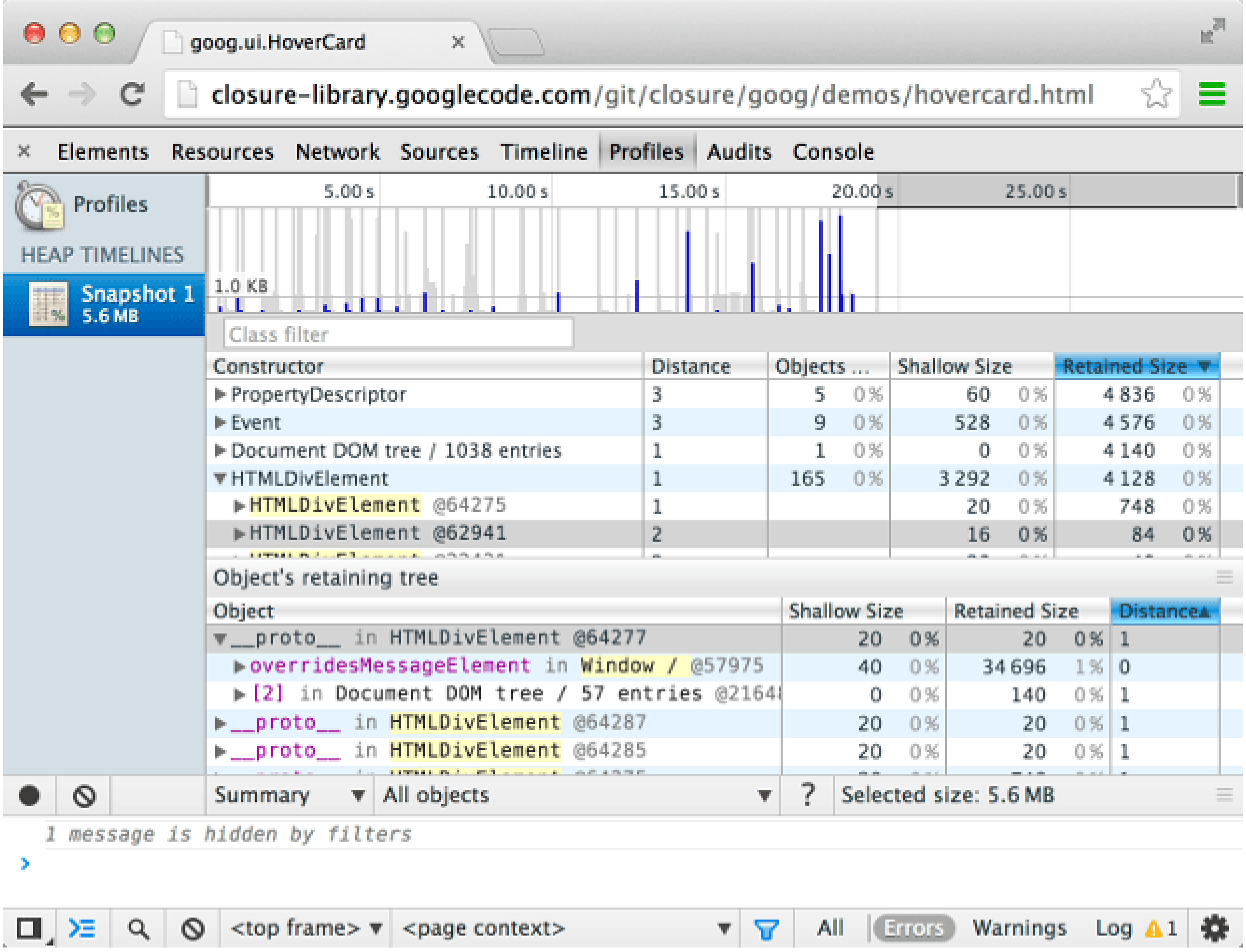
Meggin is a Tech Writer

Use the allocation profiler tool to find objects that aren't being properly garbage collected, and continue to retain memory.

How the tool works

The **allocation profiler** combines the detailed snapshot information of the heap profiler with the incremental updating and tracking of the Timeline panel. Similar to these tools, tracking objects' heap allocation involves starting a recording, performing a sequence of actions, then stop the recording for analysis.

The tool takes heap snapshots periodically throughout the recording (as frequently as every 50 ms!) and one final snapshot at the end of the recording.



Constructor	Distance	Objects ...	Shallow Size	Retained Size ▼
▶ PropertyDescriptor	3	5 0%	60 0%	4 836 0%
▶ Event	3	9 0%	528 0%	4 576 0%
▶ Document DOM tree / 1038 entries	1	1 0%	0 0%	4 140 0%
▼ HTMLDivElement	1	165 0%	3 292 0%	4 128 0%
▶ HTMLDivElement @64275	1		20 0%	748 0%
▶ HTMLDivElement @62941	2		16 0%	84 0%

Object's retaining tree

Object	Shallow Size	Retained Size	Distance▲
▼ __proto__ in HTMLDivElement @64277	20 0%	20 0%	1
▶ overridesMessageElement in Window / @57975	40 0%	34 696 1%	0
▶ [2] in Document DOM tree / 57 entries @2164	0 0%	140 0%	1
▶ __proto__ in HTMLDivElement @64287	20 0%	20 0%	1
▶ __proto__ in HTMLDivElement @64285	20 0%	20 0%	1

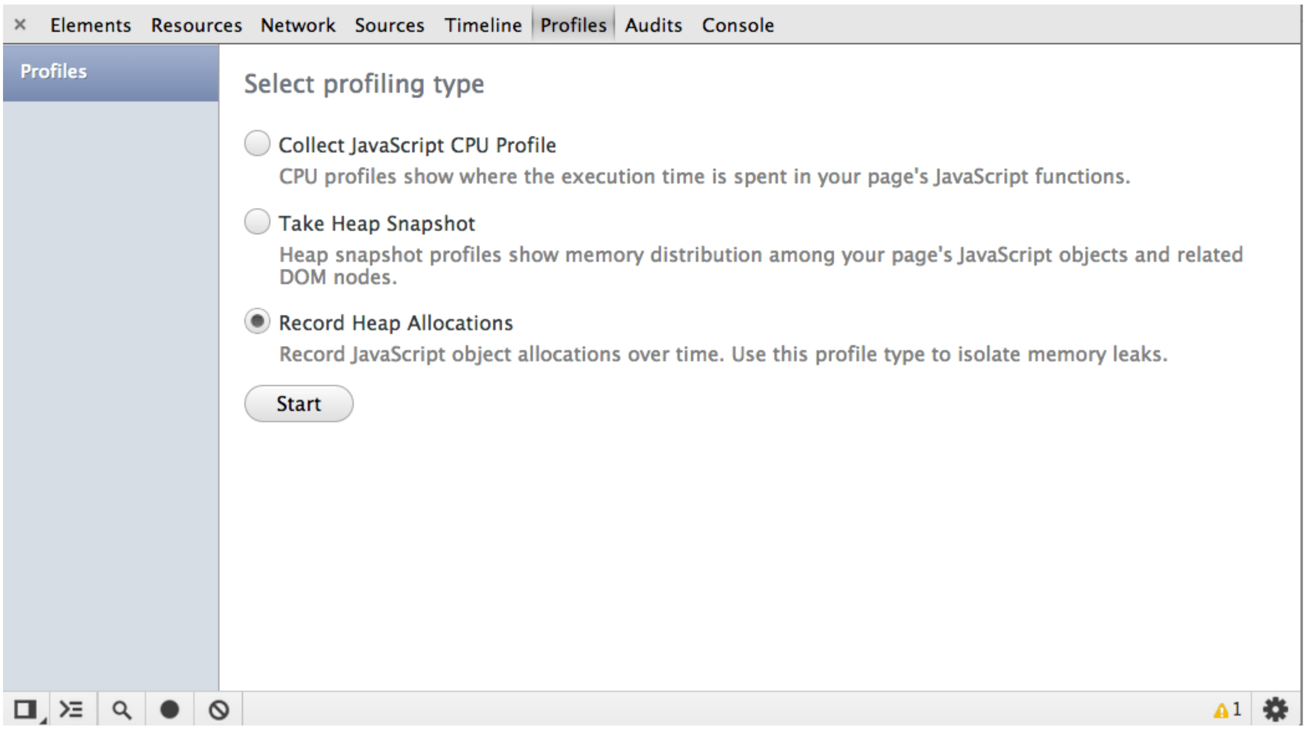
1 message is hidden by filters

Note: The number after the @ is an object ID that persists among multiple snapshots taken. This allows precise comparison between heap states. Displaying an object's address makes no sense, as objects are moved during garbage collections.

Enable allocation profiler

To begin using the allocation profiler:

1. Make sure you have the latest Chrome Canary.
2. Open the Developer Tools and click on the gear icon in the lower right.
3. Now, open the Profiler panel, you should see a profile called "Record Heap Allocations"



Select profiling type

○ Collect JavaScript CPU Profile
CPU profiles show where the execution time is spent in your page's JavaScript functions.

○ Take Heap Snapshot
Heap snapshot profiles show memory distribution among your page's JavaScript objects and related DOM nodes.

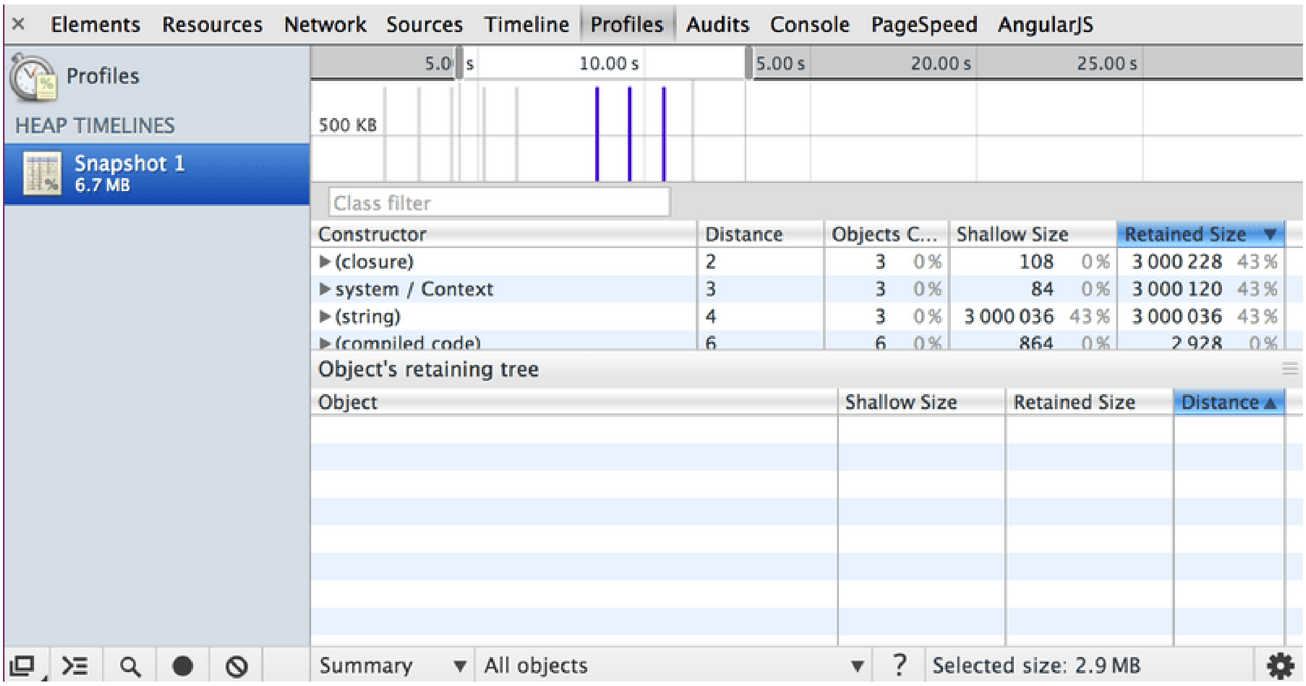
● Record Heap Allocations
Record JavaScript object allocations over time. Use this profile type to isolate memory leaks.

Start

Read a heap allocation profile

The heap allocation profile shows where objects are being created and identifies the retaining path. In the snapshot below, the bars at the top indicate when new objects are found in the heap.

The height of each bar corresponds to the size of the recently allocated objects, and the color of the bars indicate whether or not those objects are still live in the final heap snapshot. Blue bars indicate objects that are still live at the end of the timeline, Gray bars indicate objects that were allocated during the timeline, but have since been garbage collected:



Constructor	Distance	Objects C...	Shallow Size	Retained Size ▼
▶(closure)	2	3 0%	108 0%	3 000 228 43%
▶system / Context	3	3 0%	84 0%	3 000 120 43%
▶(string)	4	3 0%	3 000 036 43%	3 000 036 43%
▶(compiled code)	6	6 0%	864 0%	2 928 0%

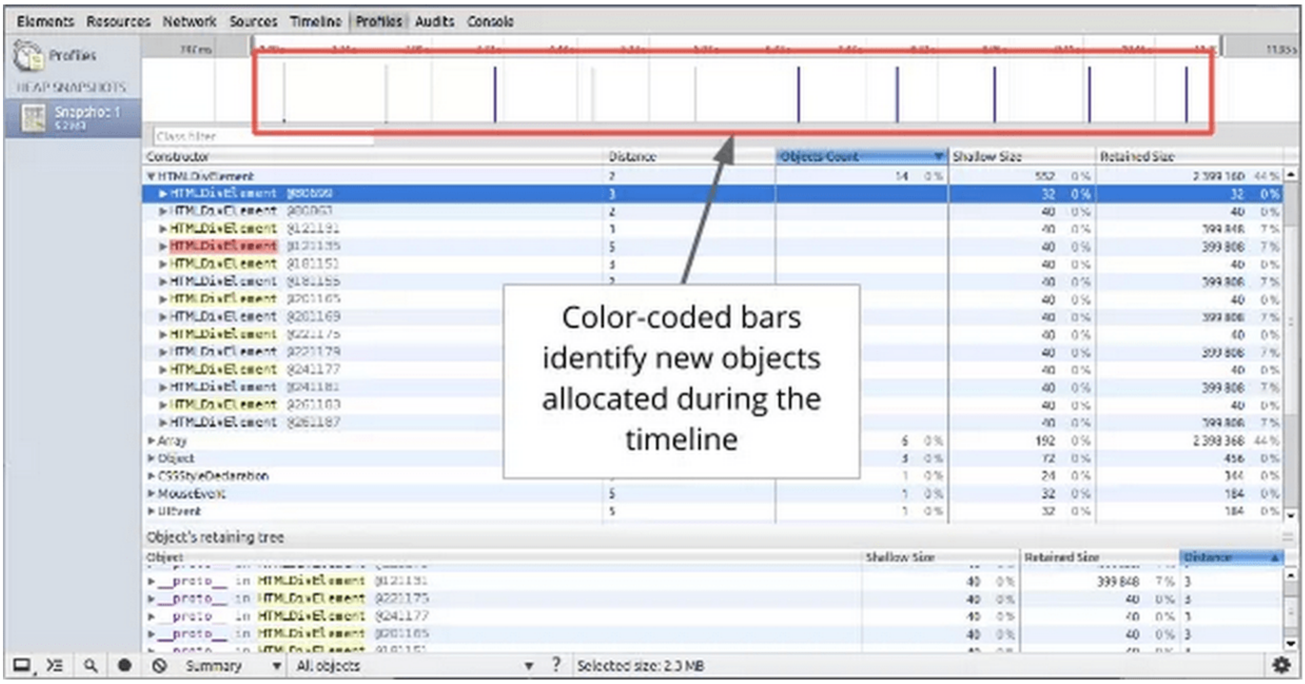
Object's retaining tree

Object	Shallow Size	Retained Size	Distance ▲

Selected size: 2.9 MB

In the snapshot below, an action was performed 10 times. The sample program caches five objects, so the last five blue bars are expected. But the leftmost blue bar indicates a potential problem.

You can then use the sliders in the timeline above to zoom in on that particular snapshot and see the objects that were recently allocated at that point:



Constructor	Distance	Objects Count ▼	Shallow Size	Retained Size
▼HTMLDivElement	2	14 0%	552 0%	2 399 160 44%
▶HTMLDivElement @80699	3		32 0%	32 0%
▶HTMLDivElement @200063	2		40 0%	40 0%
▶HTMLDivElement @121131	1		40 0%	399 848 7%
▶HTMLDivElement @121135	5		40 0%	399 806 7%
▶HTMLDivElement @181151	3		40 0%	40 0%
▶HTMLDivElement @181155	2		40 0%	399 806 7%
▶HTMLDivElement @201165			40 0%	40 0%
▶HTMLDivElement @201169			40 0%	399 806 7%
▶HTMLDivElement @221175			40 0%	40 0%
▶HTMLDivElement @221179			40 0%	399 806 7%
▶HTMLDivElement @241177			40 0%	40 0%
▶HTMLDivElement @241181			40 0%	399 806 7%
▶HTMLDivElement @261183			40 0%	40 0%
▶HTMLDivElement @261187			40 0%	399 806 7%
▶Array		6 0%	192 0%	2 399 368 44%
▶Object		3 0%	72 0%	456 0%
▶CSSStyleDeclaration		1 0%	24 0%	144 0%
▶MouseEvent	5	1 0%	32 0%	184 0%
▶UIEvent	5	1 0%	32 0%	184 0%

Object's retaining tree

Object	Shallow Size	Retained Size	Distance ▲
▶__proto__ in HTMLDivElement @121131	40 0%	399 848 7%	3
▶__proto__ in HTMLDivElement @221175	40 0%	40 0%	3
▶__proto__ in HTMLDivElement @241177	40 0%	40 0%	3
▶__proto__ in HTMLDivElement @201165	40 0%	40 0%	3

Selected size: 2.3 MB

Clicking on a specific object in the heap will show its retaining tree in the bottom portion of the heap snapshot. Examining the retaining path to the object should give you enough information to understand why the object was not collected, and you can make the necessary code changes to remove the unnecessary reference.

View memory allocation by function

You can also view memory allocation by JavaScript function. See Investigate memory allocation by function for more information.

Last updated July 2, 2018.