Need to Node: Profiling Node.js Applications

Patrick Mueller

NODESOURCE

January 19, 2016

Questions during the Need to Node webinar?

Post a question to Twitter
with the hashtag:

#needtonode

NodeSource is *the* Enterprise Node.js company offering a suite of products and professional services explicitly focused on the needs of Enterprise users of Node.js.

NODESOURCE ™

Agenda

Introduction to profiling Node.js applications

- Explain what profiling applications means, and what kind of profiling is available for Node.js
- Show what insights are provided when profiling Node.js applications
- Demonstrate using the Node.js profilers

What is profiling?

What is profiling?

Profiling is:

- capturing statistics while a program is running
- displaying those statistics with useful visualizations

Provides deep view into application performance

- no more guessing at what your program is doing

Profiling Node.js applications

- **V8 heap snapshot profiler** - measures memory usage
  - find memory leaks
  - optimize memory use by your program
- **V8 CPU profiler** - measures function execution time
  - find bottlenecks in your application
  - optimize the performance of your program

What is the real value in profiling?

- Run your programs faster
- Run your programs with less RAM
- Find difficult to diagnose problems - leaking memory and code bottlenecks

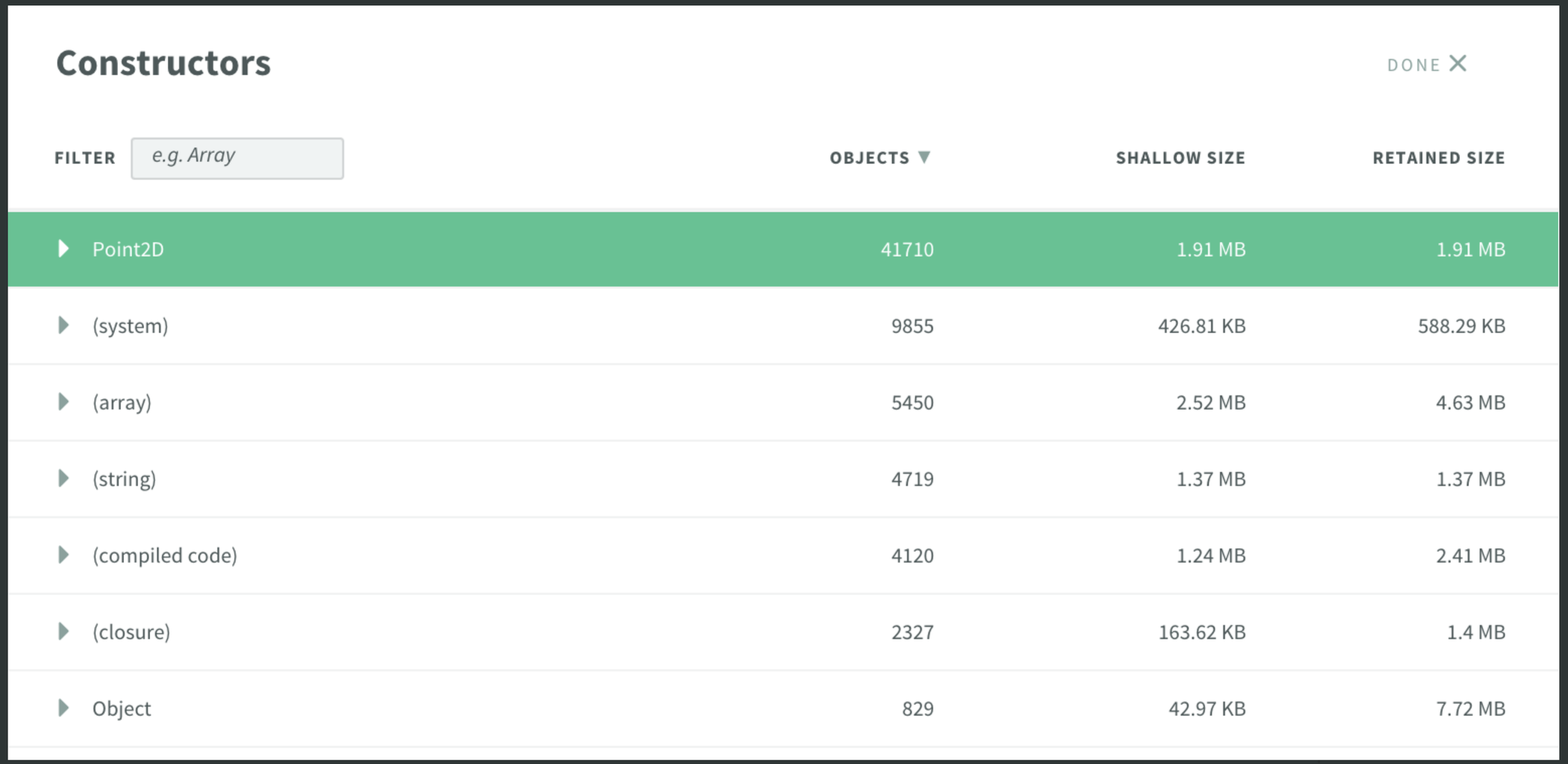
Save $$$!!!

What kind of insights does the heap snapshot profiler provide?

What does the heap snapshot profiler do?

- While your program is running, generates a JSON-able description of all JavaScript objects allocated in your program at a specific point in time, and the references between those objects.
  - This description is quite large; plan on 2x RAM usage of your application.
- Visualization shows object counts/sizes by constructor (class), and references to and from individual objects.

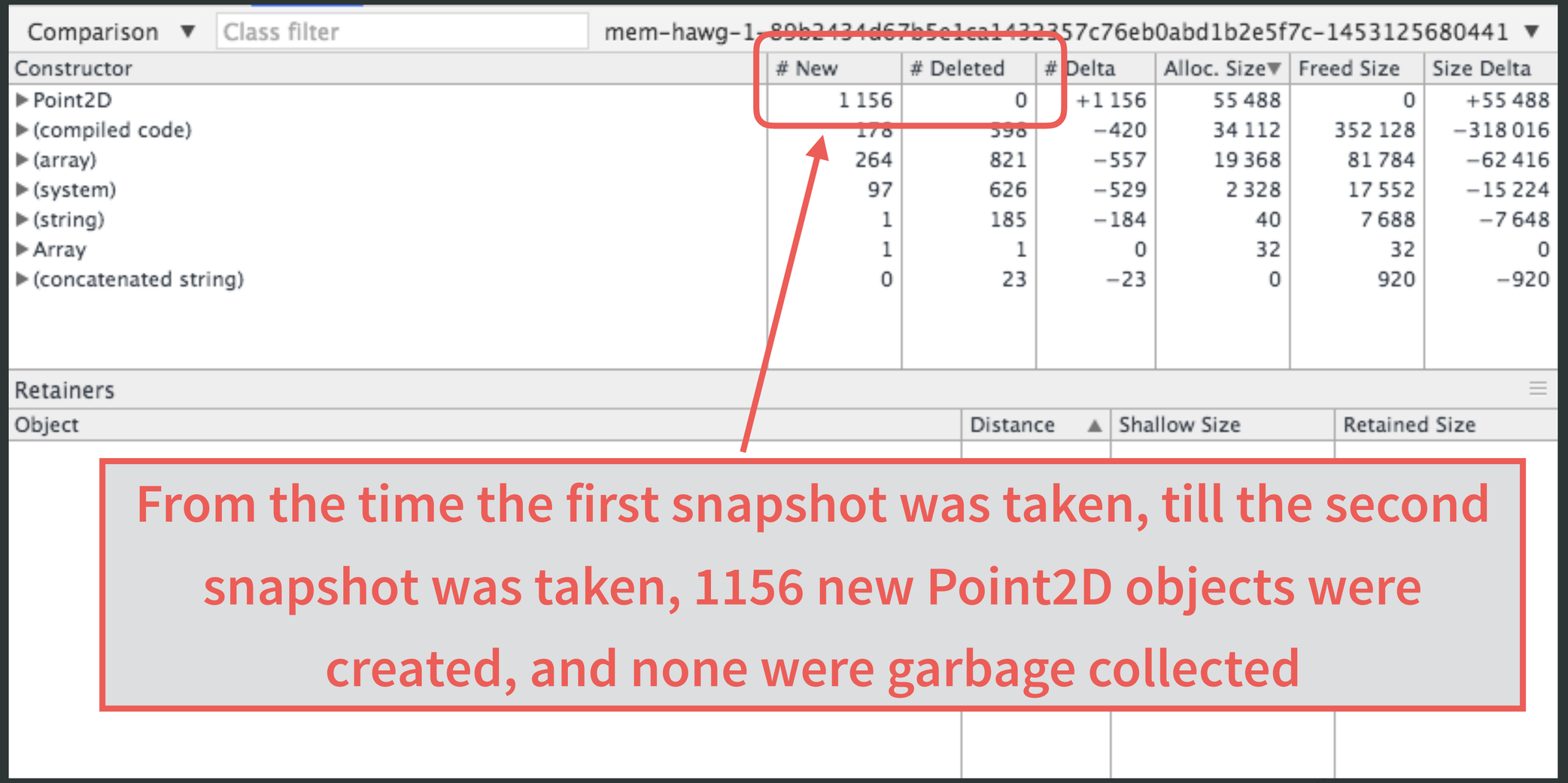
Heap Snapshot insights - counts/sizes per class (N|Solid)



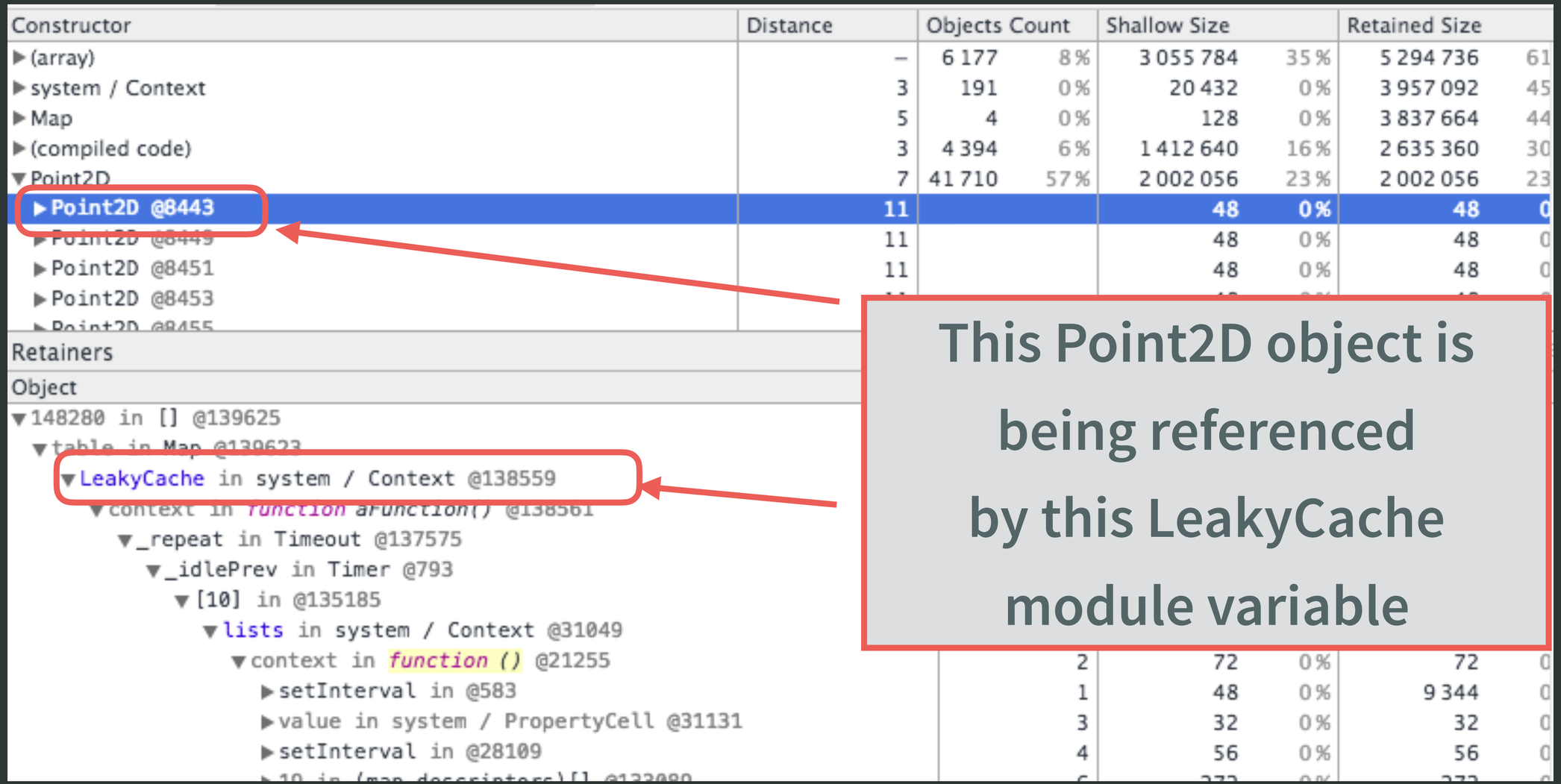
Heap Snapshot insights

- **shallow size** vs **retained size**
  - **shallow size** - amount of memory this object uses just by itself; typically only relevant for Strings and Arrays
  - **retained size** - total amount of memory this object is referencing that would be garbage collected (GC'd) if this object was garbage collected - usually the more interesting number
- Heap Snapshots are typically grouped by constructor name, so literal objects are all lumped into **Object**

Heap Snapshot insights - snapshot diff (Chrome Dev Tools)



Heap Snapshot insights - which objects reference selected object (Chrome Dev Tools)

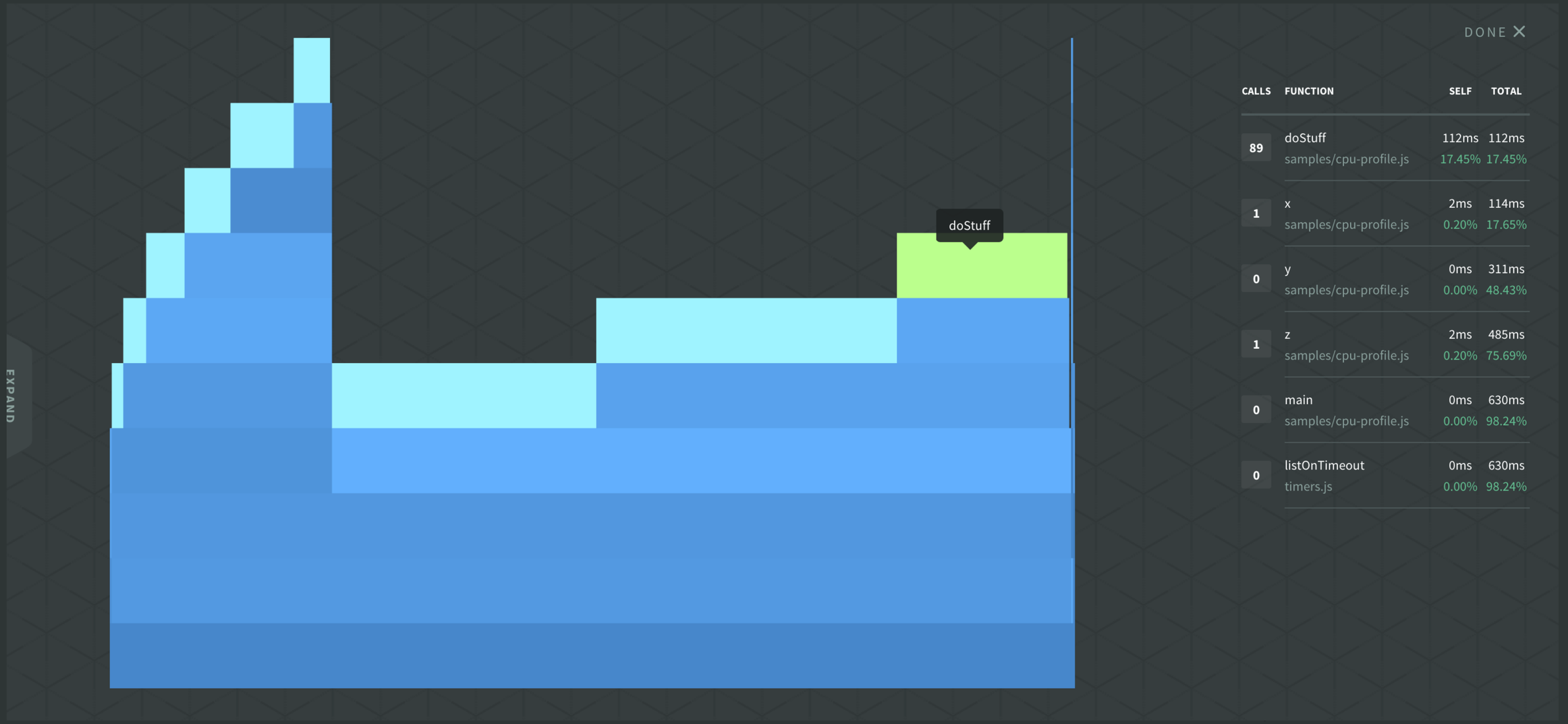


What kind of insights does CPU profiling provide?

What does the CPU profiler do?

- While your program is running, you can start the profiler, let it run for some number of seconds, and then stop the profiler.
  - While the profiler is running, it collects the stack of functions being executed, at a sub-millisecond interval.
  - When the profiler is stopped, that data is aggregated into a JSON-able object.
- Visualizations show object function call stacks, function execution time, and aggregate function call times.

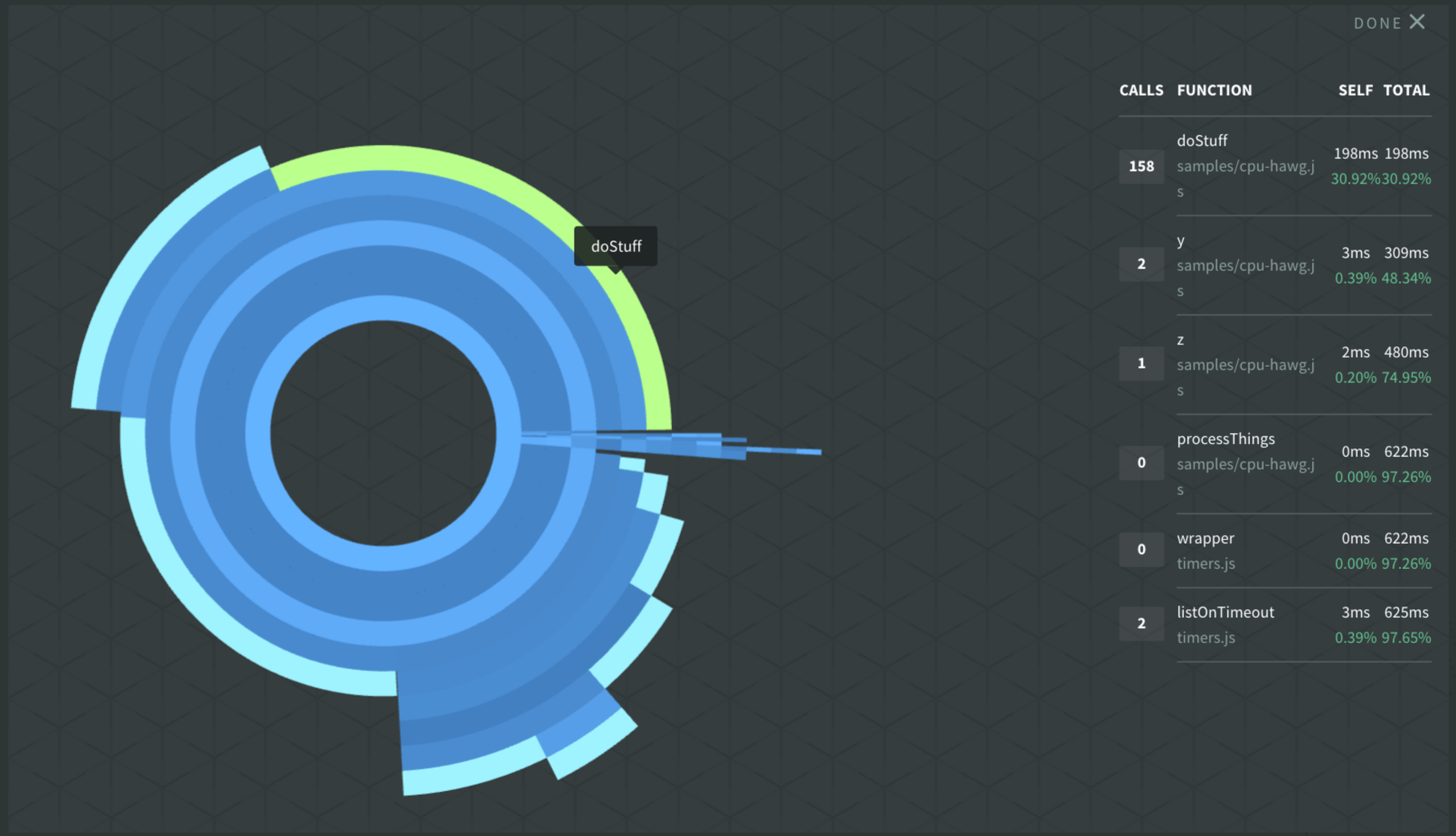
CPU profiling insights - flame graph (N|Solid)



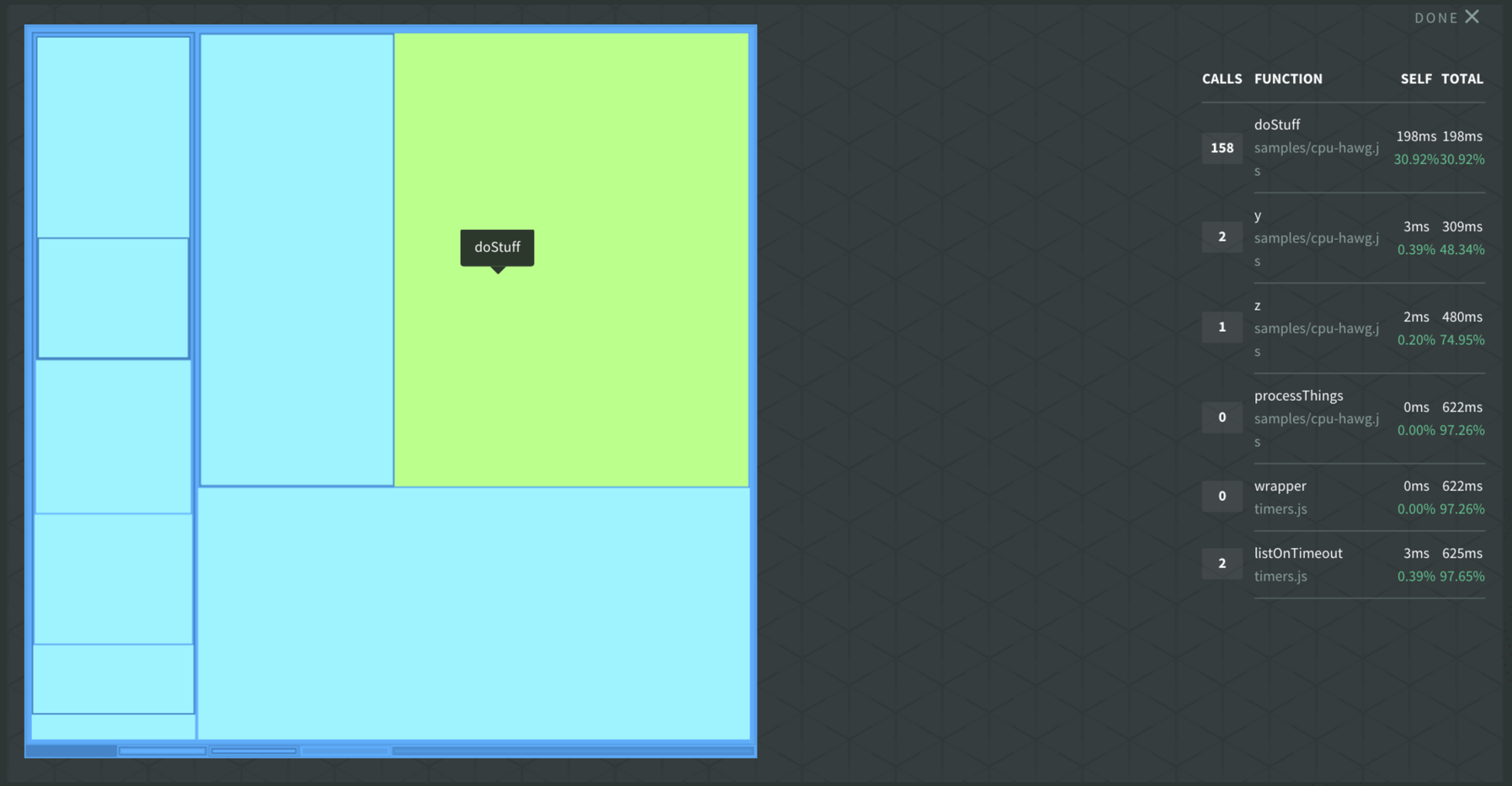
CPU profiling insights

- **total time** vs **self time**
  - **total time** - total elapsed time spent in a function
  - **self time** -total elapsed time spent in a function, minus total time spent in functions called from this function
- **source location** of functions provided in profiling data
- **name your functions**, lest you see lots of functions named “`(anonymous function)`”

CPU profiling insights - sunburst (N|Solid)



CPU profiling insights - treemap (N|Solid)



CPU profiling insights- tablular view of functions (Chrome Dev Tools)

Heavy (Bottom Up)

Self		Total		Function	
4364.6 ms		4364.6 ms		(idle)	(program):-1
0 ms	0 %	630.0 ms	98.24 %	listOnTimeout	timers.js:55
0 ms	0 %	630.0 ms	98.24 %	▶ main	/Users/pmuellr/Projects/slides/2016/01-intro-to-profiling/samples/cpu-profile.js:14
626.2 ms	97.65 %	626.2 ms	97.65 %	▶ doStuff	/Users/pmuellr/Projects/slides/2016/01-intro-to-profiling/samples/cpu-profile.js:30
1.3 ms	0.20 %	485.4 ms	75.69 %	▶ z	/Users/pmuellr/Projects/slides/2016/01-intro-to-profiling/samples/cpu-profile.js:26
0 ms	0 %	310.6 ms	48.43 %	▶ y	/Users/pmuellr/Projects/slides/2016/01-intro-to-profiling/samples/cpu-profile.js:27
0 ms	0 %	144.6 ms	22.55 %	▶ a	/Users/pmuellr/Projects/slides/2016/01-intro-to-profiling/samples/cpu-profile.js:19
0 ms	0 %	137.1 ms	21.37 %	▶ b	/Users/pmuellr/Projects/slides/2016/01-intro-to-profiling/samples/cpu-profile.js:20
0 ms	0 %	122.0 ms	19.02 %	▶ c	/Users/pmuellr/Projects/slides/2016/01-intro-to-profiling/samples/cpu-profile.js:21
1.3 ms	0.20 %	113.2 ms	17.65 %	▶ x	/Users/pmuellr/Projects/slides/2016/01-intro-to-profiling/samples/cpu-profile.js:28
0 ms	0 %	96.8 ms	15.10 %	▶ d	/Users/pmuellr/Projects/slides/2016/01-intro-to-profiling/samples/cpu-profile.js:22
0 ms	0 %	66.6 ms	10.39 %	▶ e	/Users/pmuellr/Projects/slides/2016/01-intro-to-profiling/samples/cpu-profile.js:23
1.3 ms	0.20 %	25.1 ms	3.92 %	▶ f	/Users/pmuellr/Projects/slides/2016/01-intro-to-profiling/samples/cpu-profile.js:24
5.0 ms	0.78 %	5.0 ms	0.78 %	(program)	(program):-1
2.5 ms	0.39 %	2.5 ms	0.39 %	(garbage collector)	(program):-1
0 ms	0 %	1.3 ms	0.20 %	▶ readableAddChunk	_stream_readable.js:122
0 ms	0 %	1.3 ms	0.20 %	processStats	nsolid.js:247
0 ms	0 %	1.3 ms	0.20 %	▶ ChildProcess.spawn	internal/child_process.js:251
0 ms	0 %	1.3 ms	0.20 %	▶ exports.stat	nsolid.js:1165
1.3 ms	0.20 %	1.3 ms	0.20 %	▶ spawn	(program):-1
1.3 ms	0.20 %	1.3 ms	0.20 %	▶ stopProfiling	(program):-1
0 ms	0 %	1.3 ms	0.20 %	▶ exports.spawn	child_process.js:355
1.3 ms	0.20 %	1.3 ms	0.20 %	▶ Readable.read	_stream_readable.js:249
0 ms	0 %	1.3 ms	0.20 %	▶ exports.execFile	child_process.js:117
0 ms	0 %	1.3 ms	0.20 %	▶ exports.exec	child_process.js:108
0 ms	0 %	1.3 ms	0.20 %	▶ Readable.push	_stream_readable.js:98
0 ms	0 %	1.3 ms	0.20 %	start	nsolid.js:105
0 ms	0 %	1.3 ms	0.20 %	▶ profiler.startProfiling	profiler.js:174
0 ms	0 %	1.3 ms	0.20 %	▶ stats.ps	nsolid.js:1315
0 ms	0 %	1.3 ms	0.20 %	▶ (anonymous function)	nsolid.js:1136
0 ms	0 %	1.3 ms	0.20 %	onread	net.js:500

How do you get profiling information from Node.js applications?

Profiling tools

v8-profiler package on npm

- manually instrument your application, load profile data into Chrome Dev Tools

N|Solid from NodeSource

- generate and display profiles at the click of a button

using v8-profiler from npm

- npm install v8-profiler
- add v8-profiler to your package.json dependencies, if not running locally
- add code to your app for triggers for starting/stopping CPU profiles, and to generate heap snapshots
  - the triggers will:
    - call functions in v8-profiler
    - save results of v8-profiler function calls into JSON files
- run your app and trigger the profiles you want to generate
- load JSON files into the Profiles tab of Chrome Dev Tools

what is N|Solid?

- N|Solid is a fully compatible Node.js v4.x LTS runtime that has been enhanced to provide additional runtime diagnostics.
- Provides a web-based console to:
  - monitor applications at scale, in production
  - obtain process- and system- specific statistics for an individual application instance
  - obtain and view CPU profiles and heap snapshots at the click of a button

using N|Solid

- Run your app with the N|Solid runtime
- Open the N|Solid Console in a browser, navigate to your application, press the buttons to generate a CPU profile or heap snapshot
- The results are shown in the N|Solid Console, and the data is available to download for further analysis in Chrome Dev Tools

Demo time!

sample applications being profiled:
https://gist.github.com/pmuellr/2c7e9c7b95352d1b33e0

Get involved in building new profiling tools!

follow the Node.js Tracing Work Group

- https://github.com/nodejs/tracing-wg/issues/38

write your own profiling visualizers

- CPU profiles and heap snapshots are just JSON!

N|Solid references

- **Download N|Solid - free for development**:
  https://nodesource.com/products/nsolid

- **N|Solid documentation**:
  https://docs.nodesource.com/

- **Getting Started with N|Solid**:
  https://nodesource.com/blog/getting-started-with-nsolid-at-the-command-line/

- **Getting Started with the N|Solid Console**:
  https://nodesource.com/blog/getting-started-with-the-nsolid-console/

V8 profiling references

- **Google Developers - “How to Record Heap Snapshots”** - introduction to heap snapshots: https://developers.google.com/web/tools/chrome-devtools/profile/memory-problems/heap-snapshots

- **Google Developers - “Speed Up JavaScript Execution”** - introduction to CPU profiles: https://developers.google.com/web/tools/chrome-devtools/profile/rendering-tools/js-execution

- **v8-profiler package at npm** - open source package exposing V8’s profiling APIs: https://www.npmjs.com/package/v8-profiler

Questions during the Need to Node webcast?

post a question to Twitter

with the hashtag

#needtonode

these slides at pmuellr.github.io/slides

Thank you.

Patrick Mueller

pmuellr@nodesource.com

@pmuellr

NODESOURCE