

Thomas Watson

@wa7son

github.com/watson



Agenda

- Introduction to Post-Mortem Debugging
- What are core dumps
- Introduction to llnode
- Debugging crashes
- Debugging unresponsive processes
- Debugging memory leaks

Node.js Diagnostics Working Group

[github.com / nodejs / diagnostics](https://github.com/nodejs/diagnostics)

Post-Mortem Debugging

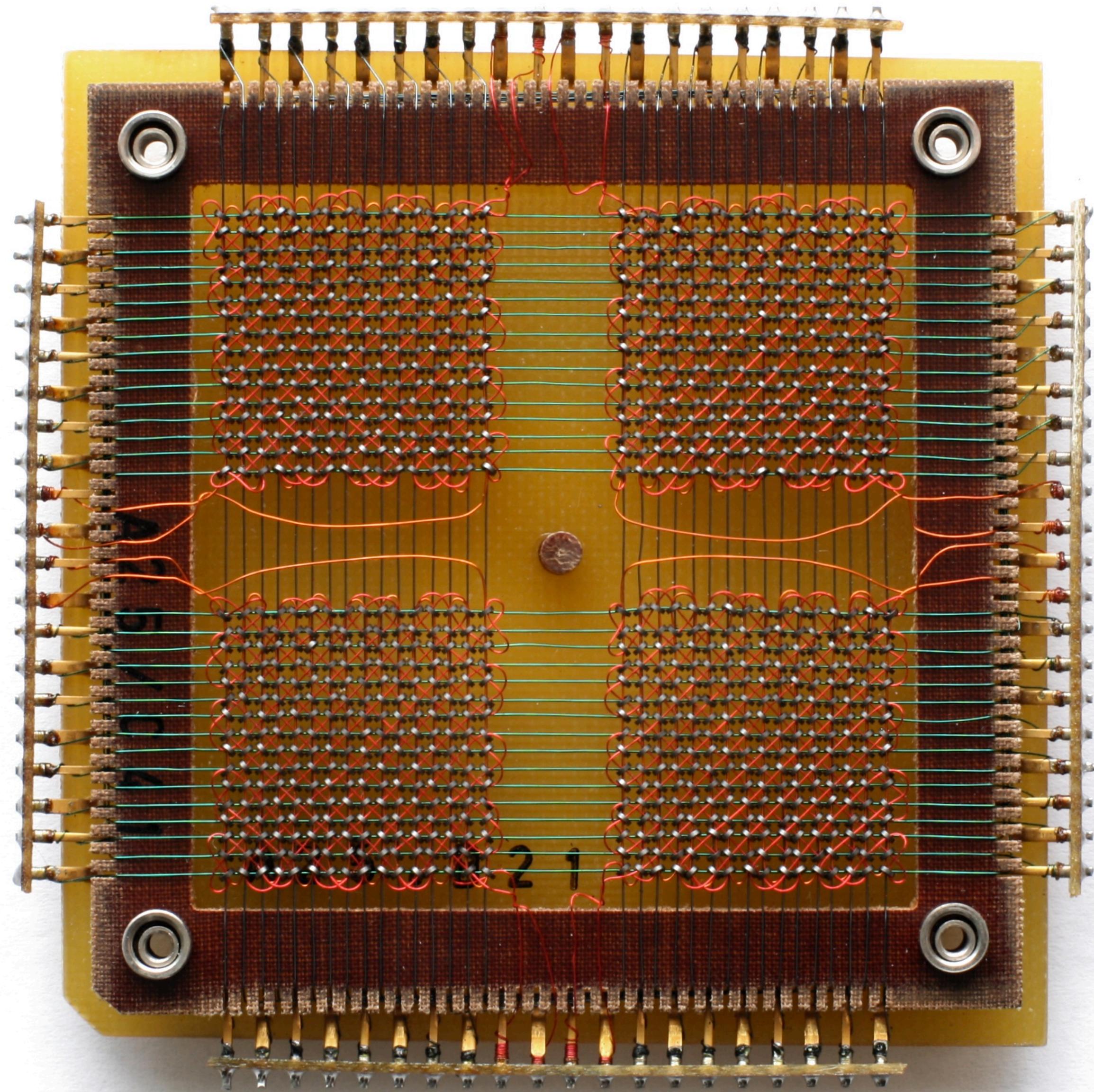
“*Post-mortem debugging* is debugging of the program after it has already crashed.”

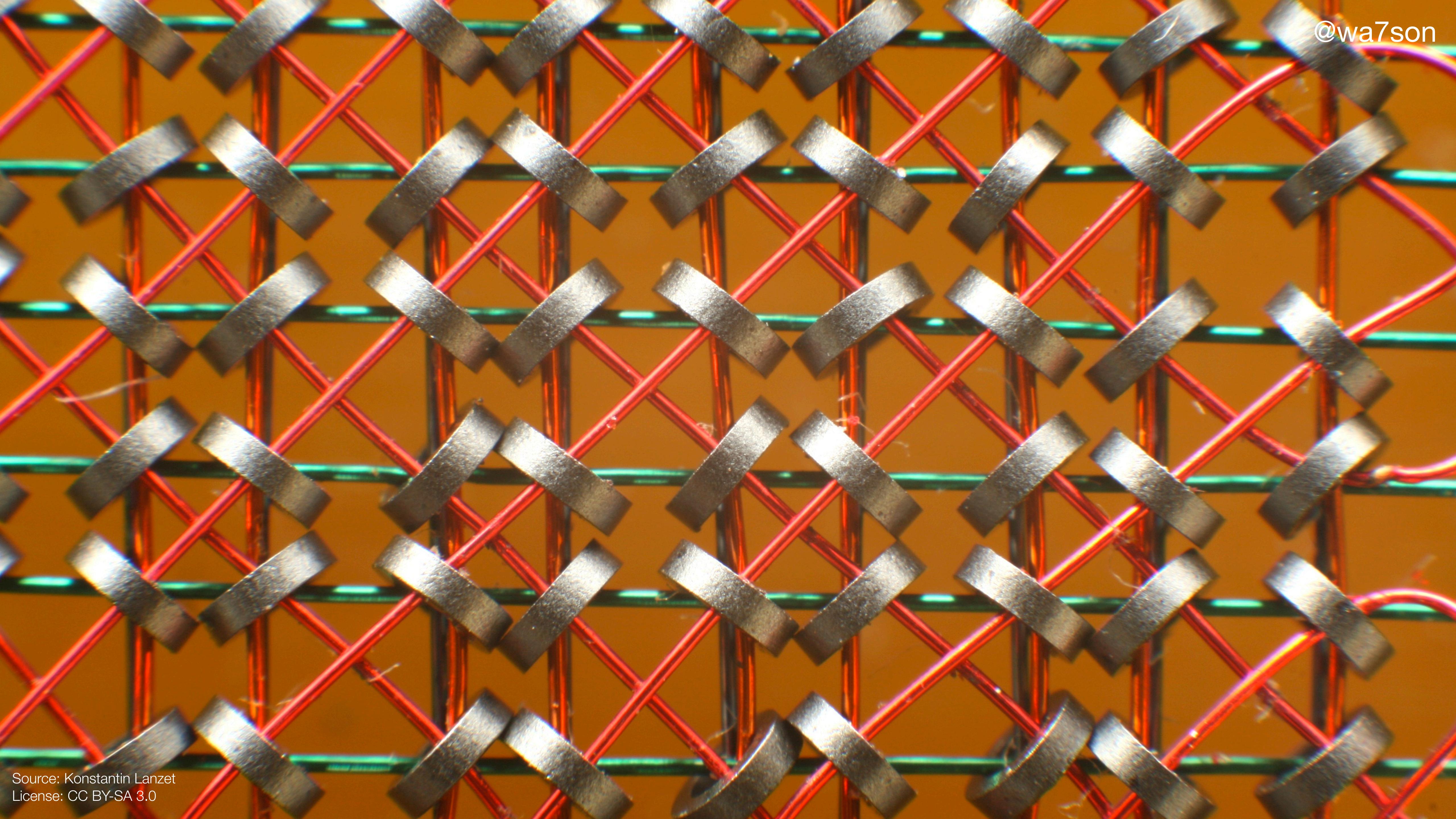
–Wikipedia

Tools

- lldb + llnode
- mdb + mdb_v8
 - autopsy
- node-report => --experimental-report
- node-heapdump => v8.getHeapSnapshot() + v8.writeHeapSnapshot()
- Chrome DevTools

Core Dumps





@wa7son

00000000	0000	0001	0001	1010	0010	0001	0001	0001	0001
00000010	0000	0016	0000	0028	0000	0010	0000	0000	0020
00000020	0000	0001	0004	0000	0000	0000	0000	0000	0000
00000030	0000	0000	0000	0010	0000	0000	0000	0000	0204
00000040	0004	8384	0084	c7c8	00c8	4748	0048	e8e9	
00000050	00e9	6a69	0069	a8a9	00a9	2828	0028	fdfc	
00000060	00fc	1819	0019	9898	0098	d9d8	00d8	5857	
00000070	0057	7b7a	007a	bab9	00b9	3a3c	003c	8888	
00000080	8888	8888	8888	8888	288e	be88	8888	8888	
00000090	3b83	5788	8888	8888	7667	778e	8828	8888	
000000a0	d61f	7abd	8818	8888	467c	585f	8814	8188	
000000b0	8b06	e8f7	88aa	8388	8b3b	88f3	88bd	e988	
000000c0	8a18	880c	e841	c988	b328	6871	688e	958b	
000000d0	a948	5862	5884	7e81	3788	1ab4	5a84	3eec	
000000e0	2d9c	daeb	5abb	9999	9999	9999	9999	9999	

Processor registers

Program counter

System flags

```
0000000 0000 0001 0001 1010 0010 0001 0004 0128  
0000010 0000 0016 0000 0028 0000 0010 0000 0020  
0000020 0000 0001 0004 0000 0000 0000 0000 0000  
0000030 0000 0000 0000 0010 0000 0000 0000 0204  
0000040 0004 8384 0084 c7c8 00c8 4748 0048 e8e9  
0000050 00e9 6a69 0069 a8a9 00a9 2828 0028 fdfc  
0000060 00fc 1819 0019 9898 0098 d9d8 00d8 5857  
0000070 0057 7b7a 007a bab9 00b9 3a3c 003c 8888  
0000080 8888 8888 8888 8888 288e be88 8888 8888  
0000090 3b83 5788 8888 8888 7667 778e 8828 8888  
00000a0 d61f 7abd 8818 8888 467c 585f 8814 8188  
00000b0 8b06 e8f7 88aa 8388 8b3b 88f3 88bd e988  
00000c0 8a18 880c e841 c988 b328 6871 688e 958b  
00000d0 a948 5862 5884 7e81 3788 1ab4 5a84 3eec  
00000e0 3d86 dcbb 5cbb 8888 8888 8888 8888 8888  
00000f0 8888 8888 8888 8888 8888 8888 8888 0000  
0000100 0000 0000 0000 0000 0000 0000 0000 0000
```

Core Dump Formats

a.out	Older versions of Unix
ELF	Modern Linux, System V, Solaris, and BSD systems
Mach-O	macOS, etc

Generating a Core Dump

```
ulimit -c unlimited
```



**Maximum allowed
size of core dump**

Generating a Core Dump

```
node --abort-on-uncaught-exception app.js
```

Generating a Core Dump



**Generate a core dump
from a running process
on Linux**

Generating a Core Dump

```
lldb --attach-pid <PID> -b -o 'process save-core "core.<PID>"'
```

Core dump
filename

Generate a core dump
form a running process
on macOS

Next Step: The Debugger

gdb, lldb, etc

as in DeBugger, not DataBase



MacOS, iOS, Linux, FreeBSD, and Windows



```
brew install --with-lldb --with-toolchain llvm
```



github.com / nodejs / llnode

A screenshot of a web browser displaying the GitHub repository page for `nodejs/llnode`. The repository title is "nodejs / llnode: An lldb plugin for Node.js and V8". The page shows basic repository statistics: 295 commits, 1 branch, 22 releases, and 24 contributors. There are buttons for creating a pull request, uploading files, and cloning or downloading the repository. The latest commit is from 25 days ago.

nodejs / llnode: An lldb plugin for Node.js and V8, which enables inspection of JavaScript states for insights into Node.js processes and their core dumps.

Code Issues 40 Pull requests 5 Projects 0 Wiki Insights

nodejs node

295 commits 1 branch 22 releases 24 contributors View license

Branch: master ▾ New pull request Create new file Upload files Find File Clone or download ▾

mmarchini build: use clang-format from npm ... Latest commit 62ca523 25 days ago

deps/rang src: colorize output for findjsinstances 6 months ago

scripts build: add linter scripts and use it on travis a month ago

```
lldb /path/to/program -c /path/to/core-file
```

```
llnode /path/to/node -c /path/to/core-file
```

A problem has been detected and windows has been shut down to prevent damage to your computer. @wa7son

The problem seems to be caused by the following file: SPCMDCON.SYS

PAGE_FAULT_IN_NONPAGED_AREA

If this is the first time you've seen this Stop error screen, restart your computer. If this screen appears again, follow these steps:

Check to make sure any new hardware or software is properly installed. If this is a new installation, ask your hardware or software manufacturer for any Windows updates you might need.

Crashes

If problems continue, disable or remove any newly installed hardware or software. Disable BIOS memory options such as caching or shadowing. If you need to use Safe Mode to remove or disable components, restart your computer, press F8 to select Advanced Startup Options, and then select Safe Mode.

Technical information:

*** STOP: 0x00000050 (0xFD3094C2,0x00000001,0xFBFE7617,0x00000000)

*** SPCMDCON.SYS - Address FBFE7617 base at FBFE5000, DateStamp 3d6dd67c

```
Error: boom!
    at /app/server.js:4:13
    at /app/server.js:5:7
    at /app/server.js:6:5
    at processTicksAndRejections (internal/process/task_queues.js:79:9)
    at process.runNextTicks [as _tickCallback] (internal/process/task_queues.js:56:3)
    at Function.Module.runMain (internal/modules/cjs/loader.js:871:11)
    at internal/main/run_main_module.js:21:11
```

```
$ node --abort-on-uncaught-exception server.js
```

```
$ node --abort-on-uncaught-exception server.js  
Server is listening on port 3000
```

```
$ node --abort-on-uncaught-exception server.js
Server is listening on port 3000
POST /signup
Uncaught TypeError: Cannot read property 'first' of undefined
FROM
onSave (/app/server.js:16:36)
Timeout._onTimeout (/app/server.js:31:5)
listOnTimeout (internal/timers.js:531:17)
processTimers (internal/timers.js:475:7)
[1]    10904 illegal hardware instruction (core dumped)
node --abort-on-uncaught-exception server.js
```

```
$ node --abort-on-uncaught-exception server.js  
Server is listening on port 3000
```

```
POST /signup
```

```
Uncaught TypeError: Cannot read property 'first' of undefined
```



```
FROM
```

```
onSave (/app/
```

```
Timeout._onTimeout (/app/server.js:31:5,
```

```
listOnTimeout (internal/timers.js:531:17)
```

```
processTimers (internal/timers.js:475:7)
```

```
[1] 10904 illegal hardware instruction (core dumped)
```

```
node --abort-on-uncaught-exception server.js
```

/cores/core.10904

```
$ node --abort-on-uncaught-exception server.js  
Server is listening on port 3000
```

```
POST /signup
```

```
Uncaught TypeError: Cannot read property 'first' of undefined
```

```
FROM  
onSave (/app/server.js:16:36)  
Timeout._onTimeout (/app/server.js:31:5)  
listOnTimeout (internal/timers.js:531:17)  
processTimers (internal/timers.js:475:7)  
[1] 10904 illegal hardware instruction (core dumped)  
node --abort-on-uncaught-exception server.js
```

```
$ node --abort-on-uncaught-exception server.js
Server is listening on port 3000
POST /signup
Uncaught TypeError: Cannot read property 'first' of undefined
FROM
onSave (/app/server.js:16:36)
Timeout._onTimeout (/app/server.js:31:5)
listOnTimeout (internal/timers.js:531:17)
processTimers (internal/timers.js:475:7)
[1]    10904 illegal hardware instruction (core dumped)
node --abort-on-uncaught-exception server.js
```

```
app.post('/signup', function singupRoute (req, res) {  
  save(req.body, function onSave (err, user) {  
    if (err) return res.status(500).send('Error: ' + err.message)  
    res.end('Welcome ' + user.name.first)  
  })  
})
```



Cannot read property 'first' of undefined

Demo Time!



Unresponsive Process



*What is your program
doing right now?*

```
const stringify = require('./lib/stringify')

// Turn this:
//   { foo: 1, bar: 2 }
// Into this:
//   { foo: "1", bar: "2" }
console.log(stringify({ foo: 1, bar: 2 }))
```

Demo Time!

Alternative:

CPU Profiling

Recap 1/2

Production Server

- gcore <pid>
- lldb --attach-pid <PID> -b -o 'process save-core "core.<PID>"'

Recap 2/2

Dev machine

- llnode node -c core

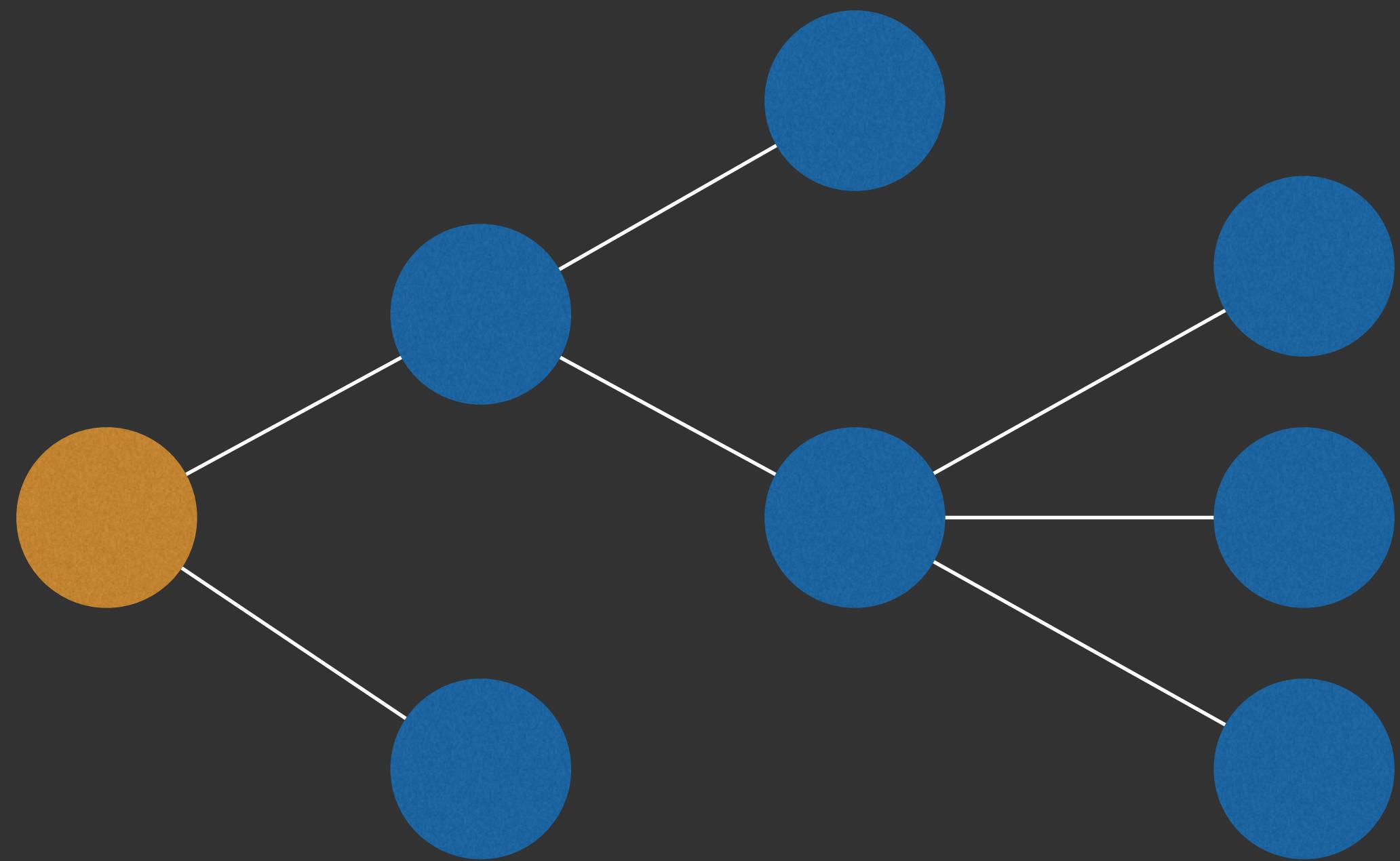
llnode

- v8 help *Get help*
- v8 bt *Get stack trace at crash*
- frame select 3 *Select stack frame #3*
- v8 source list *Show source code at selected stack frame*
- v8 inspect <addr> *Inspect object at address*

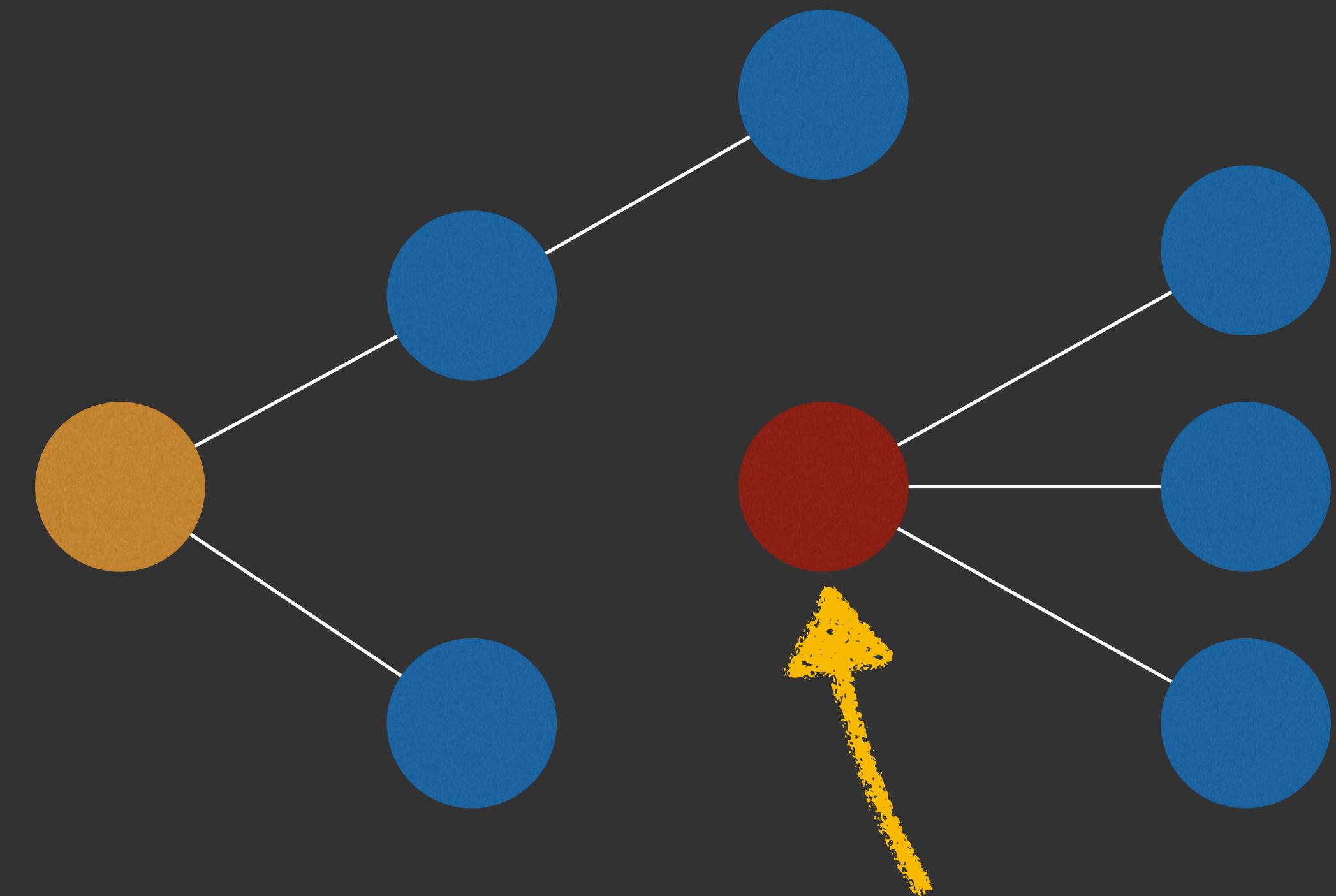
Memory Leaks

FATAL ERROR: CALL_AND_RETRY_2 Allocation failed
- process out of memory Abort trap: 6

What are Memory Leaks in JS?

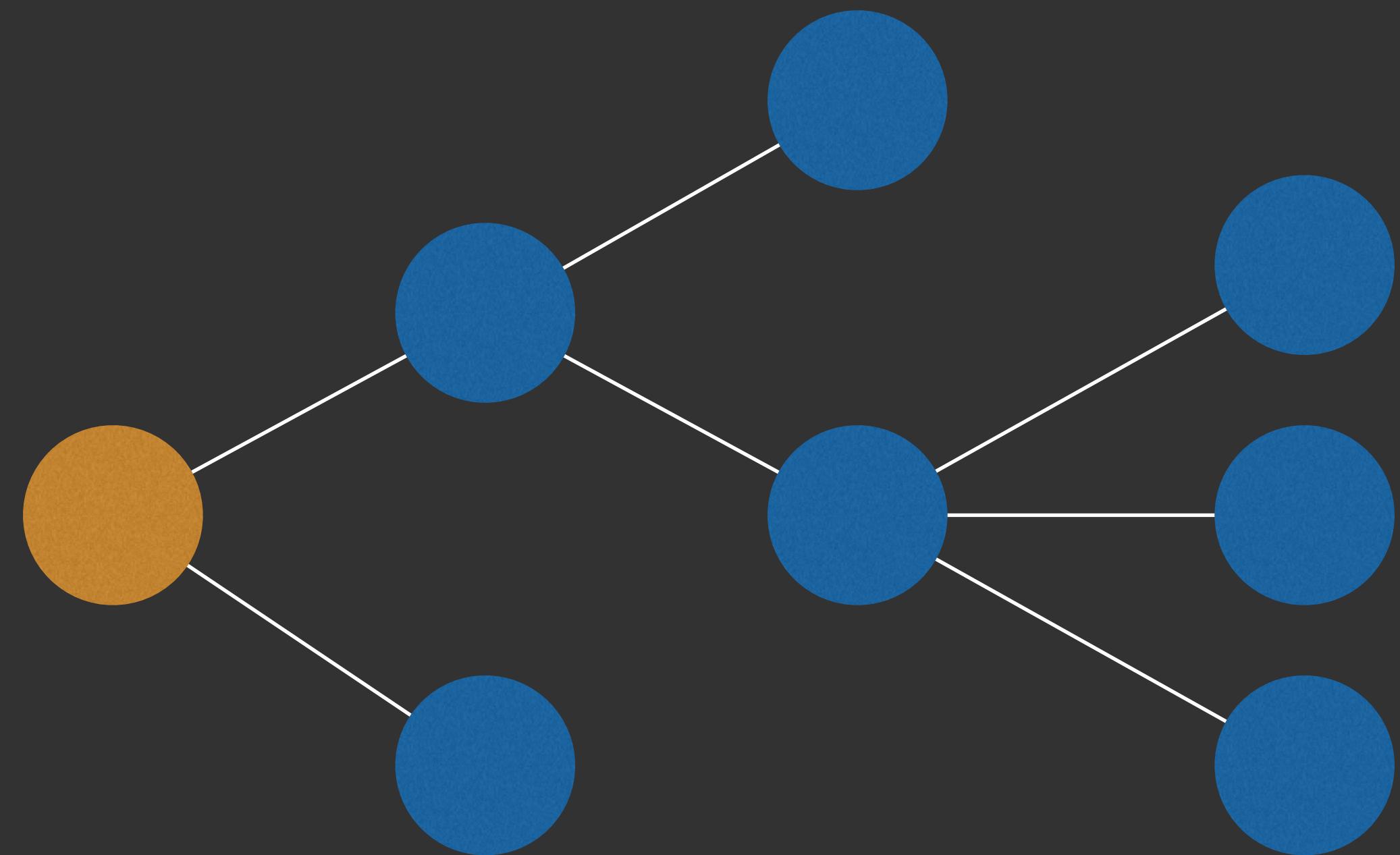


What are Memory Leaks in JS?

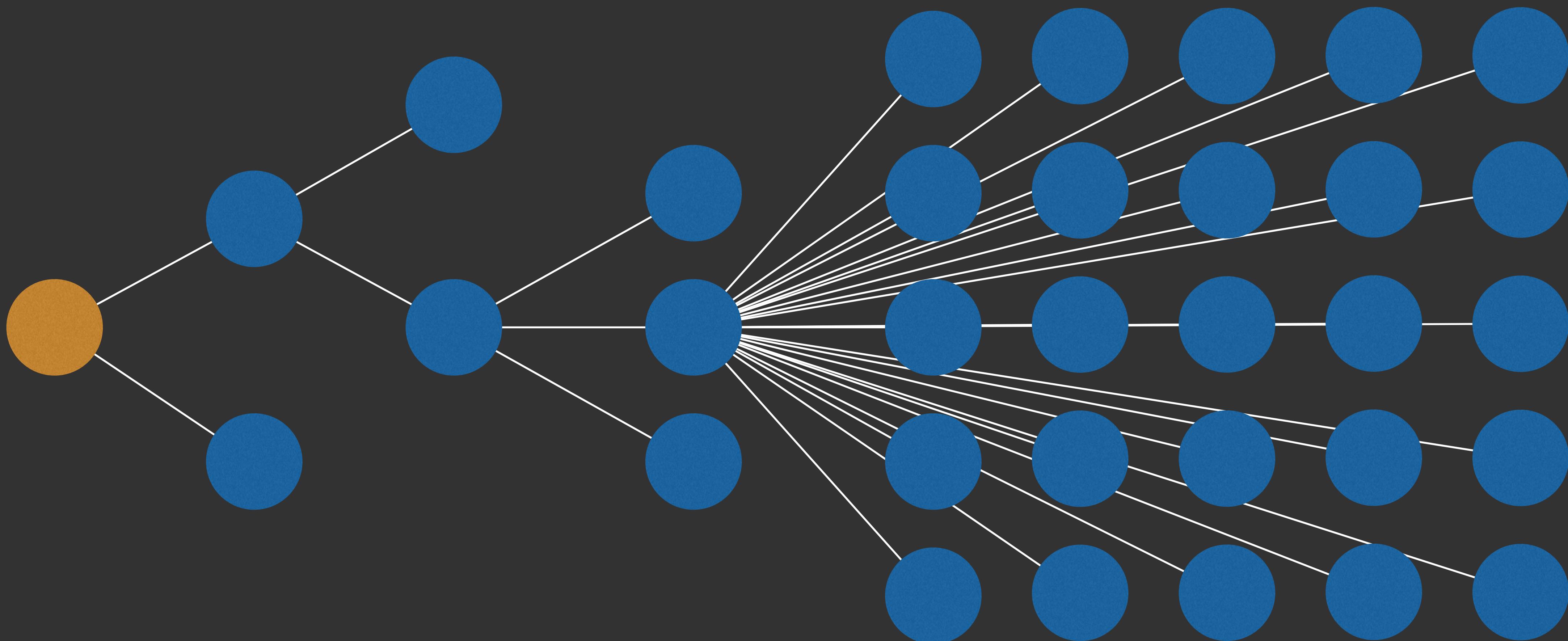


No memory leak
(will be garbage collected)

What are Memory Leaks in JS?



What are Memory Leaks in JS?



(llnode) v8 findjsobjects

```
(lldb) v8 findjsobjects
Instances Total Size Name
-----
1           24  AssertionError
1           24  AsyncResource
1           24  Control
1           24  FastBuffer
1           24  Loader
1           24  ModuleJob
1           24  ModuleMap
1           24  Performance
1           24  PerformanceObserver
1           24  SafeMap
1           24  SafePromise
1           24  SafeSet
1           24  SocketListReceive
1           24  SocketListSend
```

12	384	TCP
12	2688	WritableState
15	1360	(ArrayBufferView)
74	4736	NativeModule
5715	1234440	IncomingMessage
5744	781184	ServerResponse
5747	1103424	ReadableState
5748	275880	BufferList
45980	2942680	TickObject
69344	2219008	(Array)
235515	9420584	Visit
293720	15437744	Object
615411	3750984	(String)

1283140	37182200	

```
(llnode) v8 findjsinstances -d Visit
```

```
(l1node) v8 findjsinstances -d Visit
0x0000176d04402201:<Object: Visit properties {
    .visit_id=<Smi: 82704>,
    .headers=0x0000176d7d99f1c9:<Object: Object}>
0x0000176d04402229:<Object: Visit properties {
    .visit_id=<Smi: 82705>,
    .headers=0x0000176d7d99f191:<Object: Object}>
0x0000176d04402251:<Object: Visit properties {
    .visit_id=<Smi: 82706>,
    .headers=0x0000176d7d99f159:<Object: Object}>
0x0000176d04402279:<Object: Visit properties {
    .visit_id=<Smi: 82707>,
    .headers=0x0000176d7d99f121:<Object: Object}>
0x0000176d044022a1:<Object: Visit properties {
    .visit_id=<Smi: 82708>,
    .headers=0x0000176d7d99f0e9:<Object: Object}>
0x0000176d044022c9:<Object: Visit properties {
```

```
0x0000176d044022c9:<Object: Visit properties { @wa7son
  .visit_id=<Smi: 82709>,
  .headers=0x000176d7d99f0b1:<Object: Object>}>
// A thousand miles later...
0x0000176dffba62d9:<Object: Visit properties {
  .visit_id=<Smi: 156026>,
  .headers=0x000176dffbef559:<Object: Object>}>
0x0000176dffba6301:<Object: Visit properties {
  .visit_id=<Smi: 156027>,
  .headers=0x000176dffbef8a9:<Object: Object>}>
0x0000176dffba6329:<Object: Visit properties {
  .visit_id=<Smi: 156028>,
  .headers=0x000176dffb82481:<Object: Object>}>
```

```
(l1node) v8 inspect -m 0x0000176dffba6329
```

```
(l1node) v8 inspect -m 0x0000176dffba6329
0x0000176dffba6329 (map=0x0000176d689cec29) :<Object:
Visit properties {
  .visit_id=<Smi: 156028>,
  .headers=0x0000176dffb82481:<Object: Object>}>
```

```
(lldb) v8 inspect 0x0000176d689cec29
```

```
(l1node) v8 inspect 0x0000176d689cec29
0x0000176d689cec29:<Map own_descriptors=2
in_object_size=2
instance_size=40
descriptors=0x0000176d7f284569:<FixedArray, len=8
contents={
[0]=<Smi: 2>,
[1]=<Smi: 0>,
[2]=0x0000176dd8566a11:<String: "visit_id">,
[3]=<Smi: 320>,
[4]=<Smi: 1>,
[5]=0x0000176dd8566a31:<String: "headers">,
[6]=<Smi: 1050112>,
[7]=0x0000176d117509f9:<unknown>}>>
```

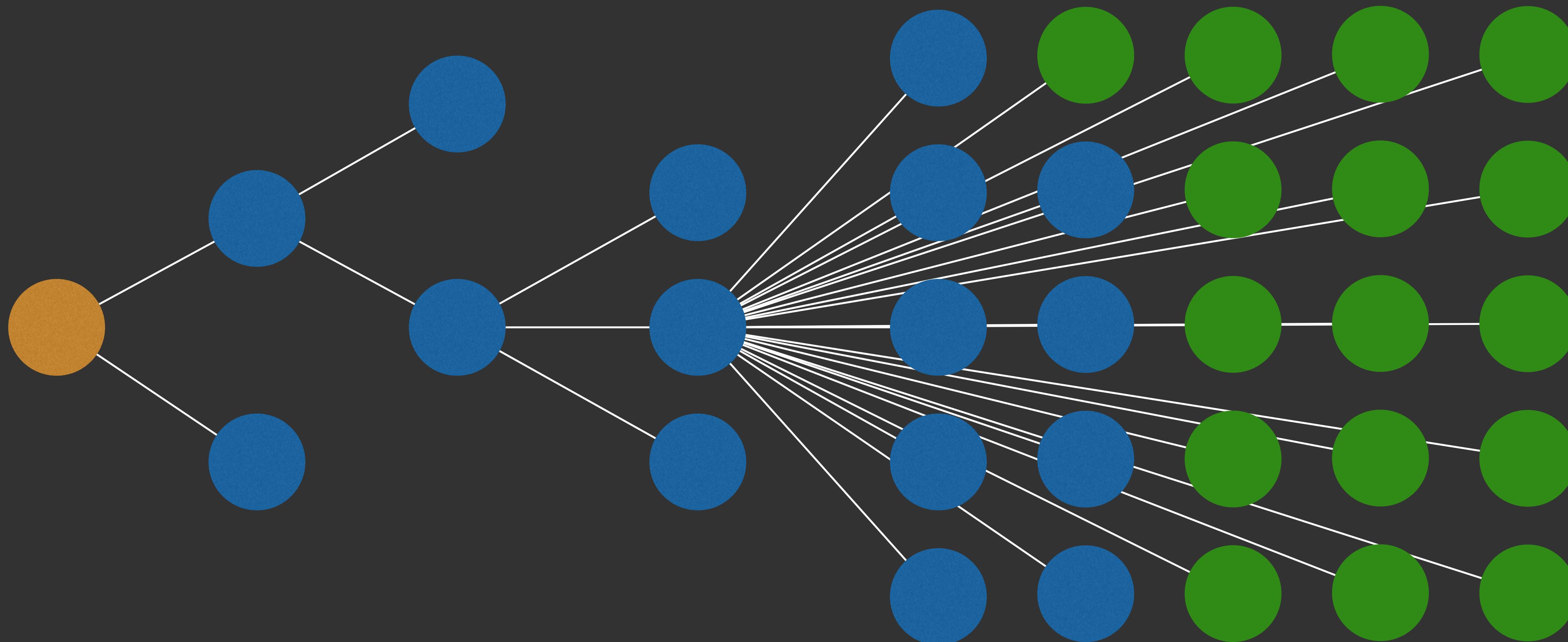
(lldb) v8 findrefs 0x0000176dffba6329

```
(lldb) v8 findrefs 0x0000176dffba6329  
0x176d1f4fac41: (Array) [156027]=0x176dffba6329
```

Can this be improved?

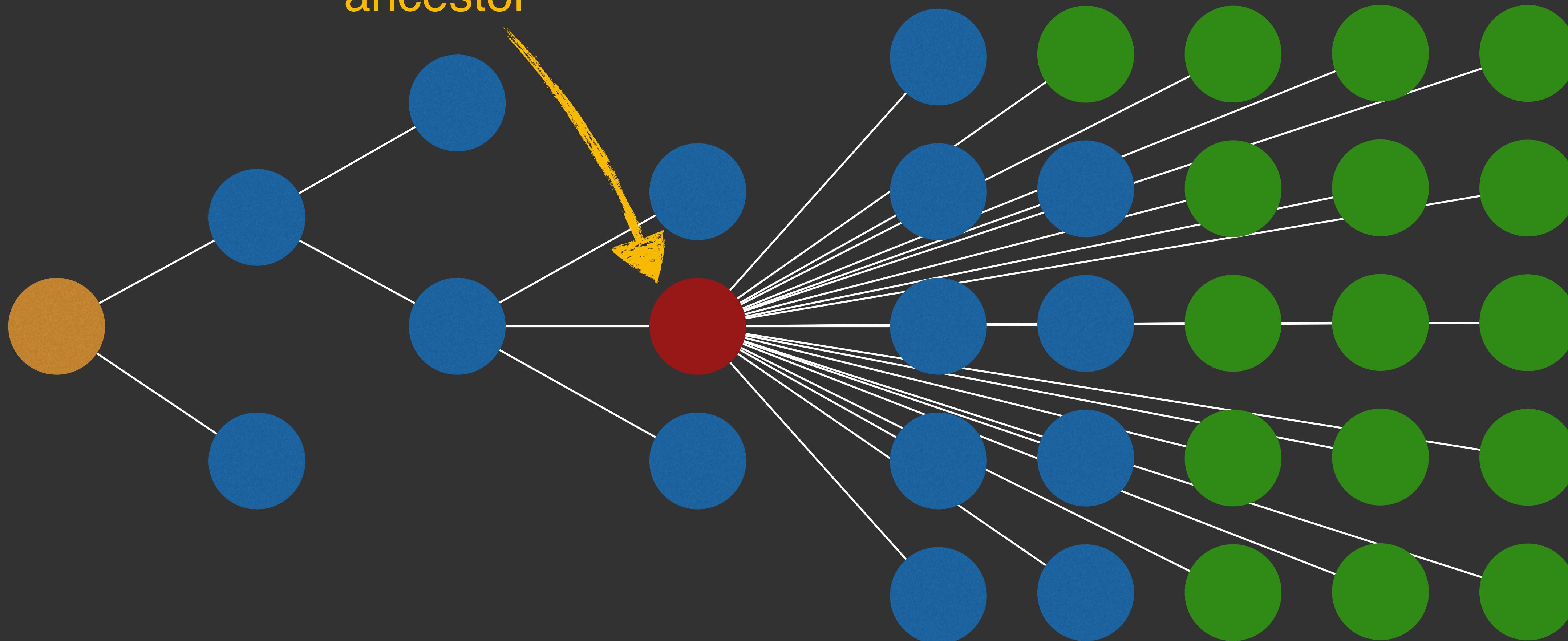
- Convert a core dump to a heap snapshot
- Allow user to trigger a gc + heap snapshot from outside the process

Heap Dump Diffing



Heap Dump Diffing

Find common
ancestor



Writing heap snapshots

```
kill -SIGUSR2 <pid>
```

```
process.on('SIGUSR2', () => {
  const { writeHeapSnapshot } = require('v8')
  console.log('Heap snapshot written:', writeHeapSnapshot())
})
```



(For now, works only on Node.js 11.13.0+)

Make a diff

Load two
heap snapshots

DevTools - github.com/nodejs/diagnostics/issues/180

Memory Application Security Audits

Comparison ▾ Class filter Heap.20190404.070958.75670.0 ▾

Constructor	# New	# Deleted	# Delta ▾	Alloc. Size	Freed Size	Size Delta
► (system)	4 889	598	+4 291	412 640	26 464	+386 17
► (array)	2 793	80	+2 707	284 808	13 024	+271 78
► (string)	787		+783	33 048	152	+32 89
► (closure)	598	2	+596	35 616	120	+35 49
► system / Context	355	1	+354	26 080	80	+26 00
► (compiled code)	261	0	+261	214 496	0	+214 49
► Array	194	0	+194	6 208	0	+6 20
► Object	172	3	+169	8 688	168	+8 52
► (concatenated string)	95	3	+92	3 800	120	+3 68
► HTTPParser	23	0	+23	736	0	+73
► BufferList	22	0	+22	1 056	0	+1 05
► Node / Parser	22	0	+22	38 192	0	+38 19

Retainers

Object	Distance	Shallow Size	Retained Size
▼ line_ends in _http_server.js @63331	6	128	0 %
▼ script_or_debug_info in parserOnIncoming @63603	5	56	0 %
▼ shared in parserOnIncoming() @91367	4	64	0 %
▼ bound_function in native_bind() @120793	3	48	0 %
▼ onIncoming in HTTPParser @117435	2	32	0 %
► wrapped in Node / Parser @88388672 □	1	1 736	0 %
► bound_argument_2 in native_bind() @122385	5	48	0 %
► bound_argument_2 in native_bind() @122349	5	48	0 %
► [2] in Array @91627	6	32	0 %
► 2 in (bound arguments)[] @122405	6	48	0 %
► 2 in (bound arguments)[] @122401	6	48	0 %
► 2 in (object elements)[] @128397	7	360	0 %
► 77 in (Global handles) @29	-	0	0 %
► 8 in system @120795	3	112	0 %
► bound_function in native_bind() @120785	3	48	0 %
► bound_function in native_bind() @120777	3	48	0 %
► bound_function in native_bind() @120769	3	48	0 %
► bound_function in native_bind() @120761	3	48	0 %
► bound_function in native_bind() @120753	3	48	0 %

Object

▼line_ends in _http_server.js @63331

▼script_or_debug_info in parserOnIncoming @63603

 ▼shared in *parserOnIncoming()* @91367

 ▼bound_function in *native_bind()* @120793

 ▼onIncoming in HTTPParser @117435

 ►wrapped in Node / Parser @88388672 □

 ►bound_argument_2 in *native_bind()* @122385

 ►bound_argument_2 in *native_bind()* @122349

 ►[2] in Array @91627

 ►2 in (bound arguments)[] @122405

 ►2 in (bound arguments)[] @122401

 ►2 in (object elements)[] @128397

 ►77 in (Global handles) @29

 ►8 in system @120795

 ►bound_function in *native_bind()* @120785

 ►bound_function in *native_bind()* @120777

Recap

Dev machine

- llnode node -c core

llnode

- v8 help *Get help*
- v8 findjsobjects -d *List all objects sorted by count*
- v8 inspect <addr> *Inspect object at address*
- v8 findrefs <addr> *Find all references to object at address*

A vibrant photograph of a canal scene in Amsterdam. The foreground shows the dark water of the canal, with several boats, including a prominent red houseboat, reflected in it. The background is filled with a dense row of traditional Dutch houses, each with unique architectural details like gabled roofs, decorative facades, and bright colors such as red, yellow, and orange. Some houses have signs indicating they are hotels or restaurants. The sky is overcast with soft, grey clouds.

Thank you

@wa7son

github.com/watson

