Diagnostic Tooling – Be ready when things go wrong in production

Aka, why you should be interested in the Node.js Diagnostics WG

NodeConfEU 2018 – Michael Dawson and Mike Kaufman

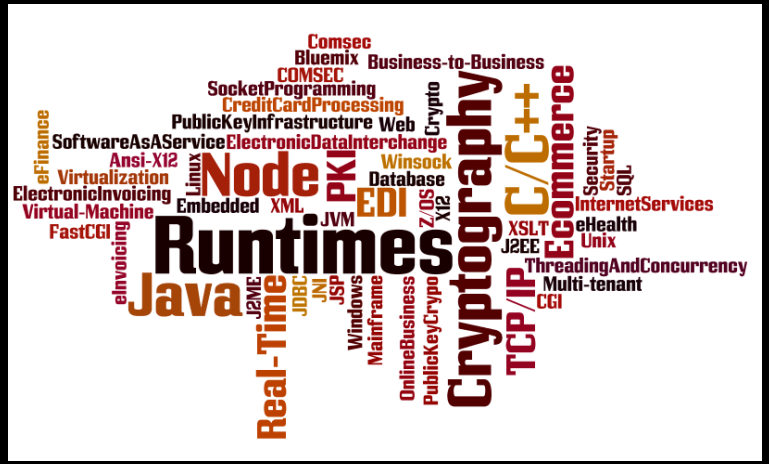
About Michael Dawson

Loves the web and building software (with Node.js!)

Senior Software Developer @ IBM

IBM Node.js Community Lead
Node.js TSC chair and CommComm member

Active in many Node.js project teams, working groups and initiatives



Contact me:

michael_dawson@ca.ibm.com

Github: @mhdawson
Twitter: @mhdawson1
https://www.linkedin.com/in/michael-dawson-6051282

About Mike Kaufman

Hates long plane rides, has a 4-year old

Principal Software Engineer @ Microsoft

Currently:
- Chakra JavaScript Engine
- Edge Browser
- Node.js Diagnostics Working Group

Previously:
- SQL Server, SQL Azure DB
- Visual Studio
- BEA Systems/WebLogic

Contact me:
Mike.Kaufman@microsoft.com

Agenda

- Fail Fast?
- Intro to Diagnostic Working Group
- Key WG Initiatives
- Scenarios/Demos

Fail Fast?

- Fail Fast and Restart?

  https://www.martinfowler.com/ieeeSoftware/failFast.pdf

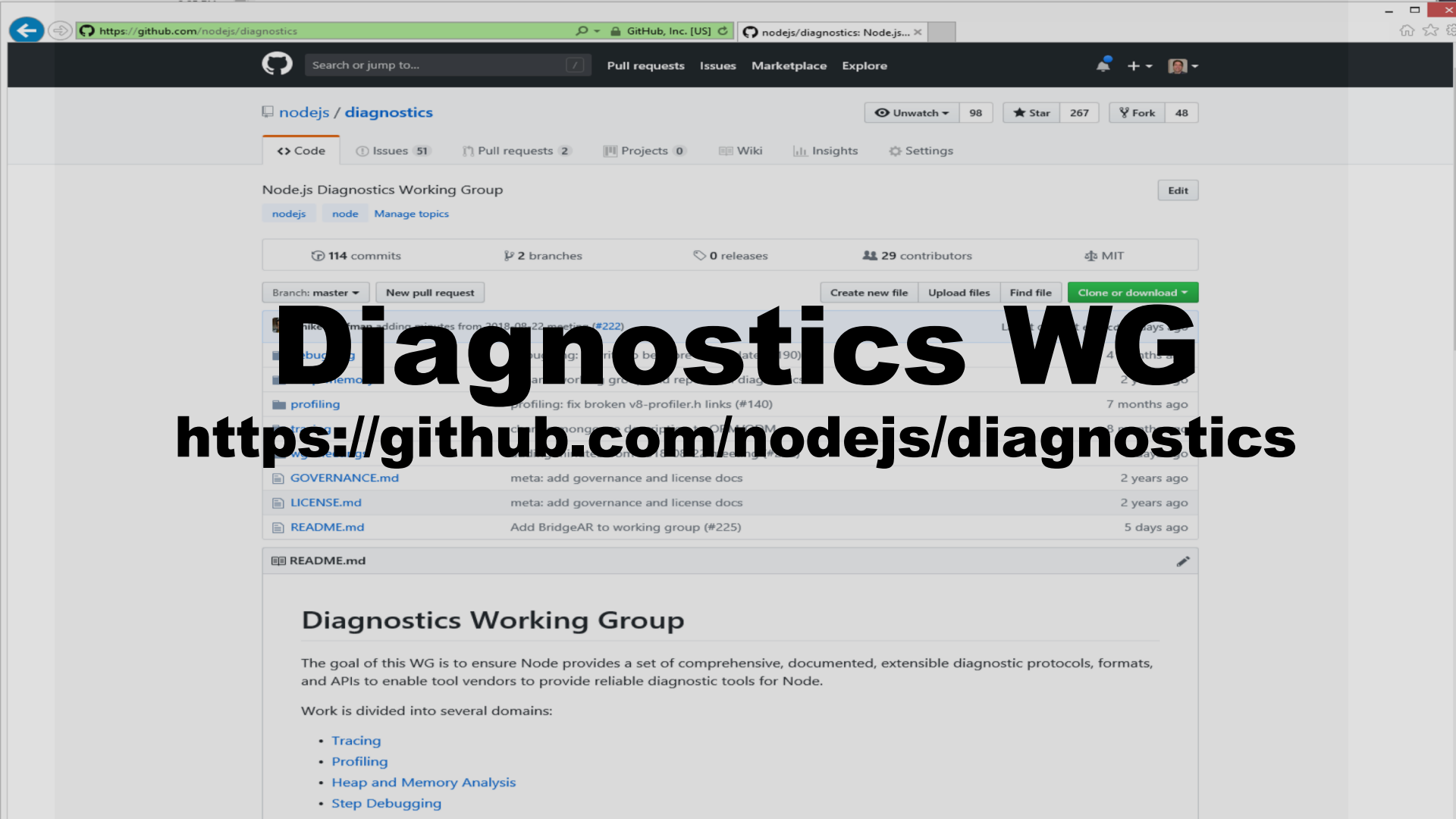
- Reasons to investigate
  - Accumulation
  - Early warning
  - Customer Impacting
  - Performance
  - Resource Usage ($$$)
- Key metrics
  - Mean time to Failure
  - Mean time to Discovery
  - Mean time to Recovery

Scenarios

- Failure
  - Memory Leak
  - Crash
- Performance
  - Slow consistently/periodically
  - Does not scale
- Excessive resource usage
  - CPU
  - Memory
- Monitoring
  - Operational Data Collection
  - Transaction Data Collection

Toolkit

- Logging/Tracing
- Profilers
- Snapshot of execution/system state
  - First Failure Data Capture (FFDC)
  - Heap Dump
  - Core Dump
- APIs



Diagnostics WG

https://github.com/nodejs/diagnostics

Diagnostics WG mission

The goal of this WG is to ensure Node provides a set of comprehensive, documented, extensible diagnostic protocols, formats, and APIs to enable tool vendors to provide reliable diagnostic tools for Node.

Team Members

- @AndreasMadsen - Andreas Madsen
- @BridgeAR - Ruben Bridgewater
- @cjihrig - Colin Ihrig
- @danielkhan - Daniel Khan
- @Flarna - Gerhard Stöbich
- @hashseed - Yang Guo
- @hekike - Peter Marton
- @Hollerberg - Gernot Reisinger
- @jasnell - James M Snell
- @jkrems - Jan Olaf Krems
- @joyeecheung - Joyee Cheung
- @kjin - Kelvin Jin
- @mcollina - Matteo Collina
- @mhdawson - Michael Dawson
- @mike-kaufman - Mike Kaufman
- @mmarchini - Matheus Marchini
- @naugtur - Zbyszek Tenerowicz
- @ofrobots - Ali Ijaz Sheikh
- @othiym23 - Forrest L Norvell
- @Qard - Stephen Belanger
- @sam-github - Sam Roberts
- @watson - Thomas Watson
- @yunong - Yunong Xiao

Key Initiatives

- Documentation/Best Practices
- Tooling/Data Sources
  - Support Tiers
  - Trace Events
  - node-report
  - Post mortem with Promises
  - llnode/JS API (@joyeecheung, @mmarchini)
  - Diagnostic Channel
- APIs
  - Async Hooks

Strategic Initiatives

Initiative	Champion	Stakeholders
Diagnostic Channel	@qard	
Async Hooks	@ofrobots	
Async Context	@mike-kaufman	@kjin
Node-report in core	@mhdawson	@richardlau
Support tiers	@mhdawson	
CPU Profiling	@mmarchini	
Post-mortem WG merge	@mmarchini	

Doc/Best Practices

- Diagnostics of no use UNLESS
  - People know they exist
  - People understand how to use them
- Some existing WG work (@naugtur)
  - https://github.com/nodejs/diagnostics/issues/211
  - https://github.com/naugtur/node-diagnostics-howtos
    - https://github.com/nodejs/nodejs.org/pull/1444

Tooling – Support Tiers

- Tiers
  - 1,2,3,4
  - Uncategorized
- Need help adding tests

Tool Type	Tool/API Name	Regular Testing in Node.js CI	Integrated with Node.js	Target Tier
FFDC	node-report	No	No	1
Memory	mdb_V8	No	No	4
Memory	node-heapdump	No	No	2
Memory	V8 heap profiler	No	Yes	1
Memory	V8 sampling heap profiler	No	Yes	1
AsyncFlow	Async Hooks (API)	?	Yes	1
Debugger	V8 Debug protocol (API)	No	Yes	1
Debugger	Command line Debug Client	?	Yes	1
Debugger	llnode	?	No	2
Debugger	Chrome Dev tools	?	No	3
Debugger	Chakracore - time-travel	No	Data source only	too early
Tracing	trace_events (API)	No	Yes	1
Tracing	DTrace	No	Partial	3
Tracing	LTTng	No	Removed?	N/A

https://github.com/nodejs/node/blob/master/doc/guides/diagnostic-tooling-support-tiers.md

APIs

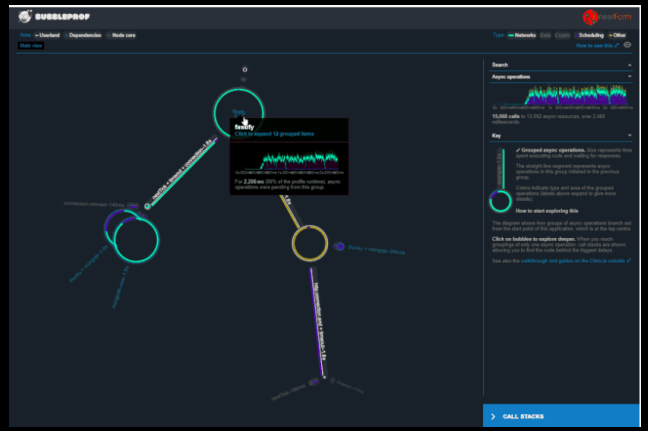
- Async Context
  - Formalization - https://github.com/nodejs/diagnostics/tree/master/async-context/slide-decks
- Async Hooks

```
function init(asyncId, type, triggerAsyncId, resource) { }
function before(asyncId) { }
function after(asyncId) { }
function destroy(asyncId) { }
function promiseResolve(asyncId) { }

const asyncHook =    async_hooks.createHook({ init, before, after, destroy, promiseResolve });

asyncHook.enable();
asyncHook.disable();
```

FSEVENTWRAP, FSREQWRAP, GETADDRINFOREQWRAP, GETNAMEINFOREQWRAP, HTTPPARSER, JSSTREAM, PIPECONNECTWRAP, PIPEWRAP, PROCESSWRAP, QUERYWRAP, SHUTDOWNWRAP, SIGNALWRAP, STATWATCHER, TCPCONNECTWRAP, TCPSERVER, TCPWRAP, TIMERWRAP, TTYWRAP, UDPSENDWRAP, UDPWRAP, WRITEWRAP, ZLIB, SSLCONNECTION, PBKDF2REQUEST, RANDOMBYTESREQUEST, TLSWRAP, Timeout, Immediate, TickObject

Some Examples

- BubbleProf



- Doctrine – HTTP Association
  - https://blog.doctrine.fr/how-doctrine-has-massively-improved-its-node-js-error-analysis-with-async-hooks-2917e4b69d5d?gi=d0820a193d6d

Ok, but how does this help ME?

Back to the Scenarios….

- Failure
  - Memory Leak
  - Crash
- Performance
  - Slow consistently/periodically
  - Does not scale
- Excessive resource usage
  - CPU
  - Memory
- Monitoring
  - Operation Data Collection
  - Transaction Data Collection

Scenario 1 – Node Report

- npm install node-report
- Easy, human readable dump
- node –r node-report test.js


```
export NODEREPORT_EVENTS=exception+fatalerror+signal+apicall
export NODEREPORT_SIGNAL=SIGUSR2|SIGQUIT
export NODEREPORT_FILENAME=stdout|stderr|<filename>
export NODEREPORT_DIRECTORY=<full path>
export NODEREPORT_VERBOSE=yes|no
```

https://github.com/nodejs/node-report

https://developer.ibm.com/node/2016/08/18/nodereport-first-failure-data-capture-for-node-js/

https://developer.ibm.com/node/2017/02/14/announcing-node-report-version-2-1-0-essential-diagnostic-node-js-applications/

https://github.com/RuntimeTools/appmetrics-dash

Scenario 1 – Node Report Not Enough Memory

```
function useMemory() {
  const objectRoot = new Object();
  for (i = 0; i < 100000000; i++) {
    objectRoot[i.toString()] = new Object();
  }
}

useMemory();
```

```
export NODEREPORT_EVENTS=exception+fatalerror+signal+apicall
node --max-old-space-size=128 -r node-report test.js
```

Demo

Scenario 2 – What is my app doing

- Basic tracing

  --trace
  --trace-gc
  --trace-opt

- Trace events
  - Trace engine from V8
  - Unified view across V8/Node
  - chrome://tracing
  - https://nodejs.org/api/tracing.html

https://vimeo.com/287708597 - Gireesh - Follow your code: Node/V8 Tracing

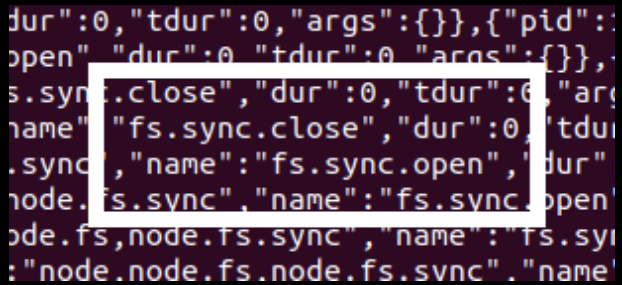
Scenario 2 – What is my app doing

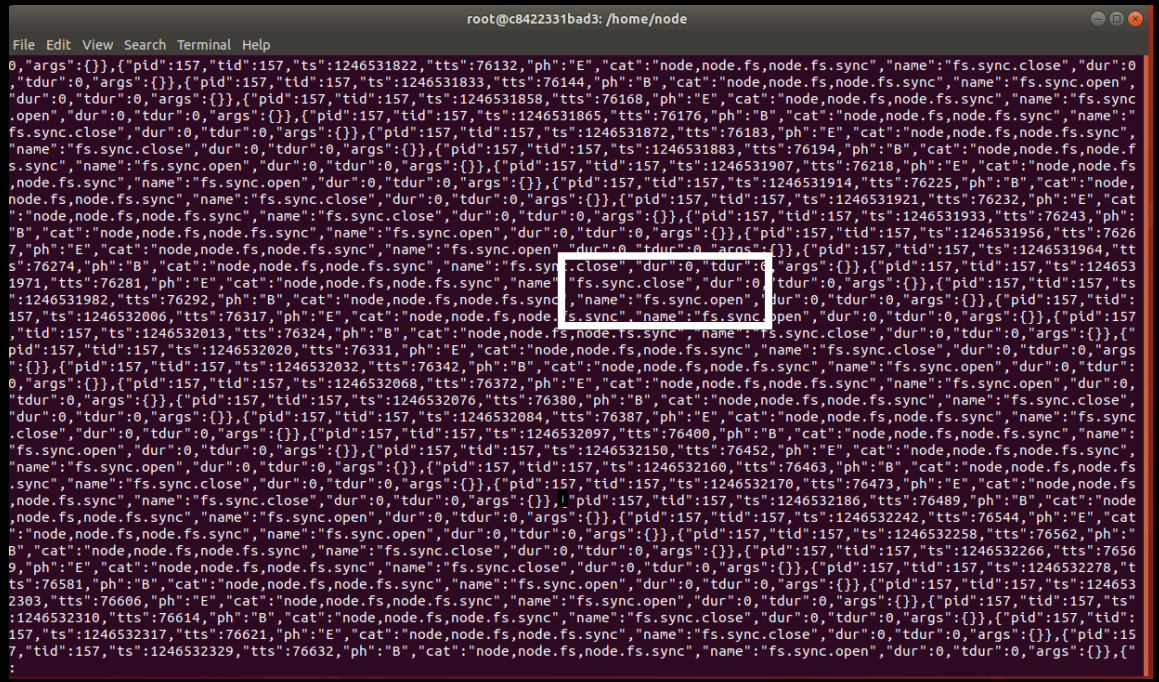
node –trace-events-categories v8,node.fs.sync test-file.js

```
const fs = require('fs');
for (i = 0; i < 10000; i++) {
  const fd = fs.openSync('TestFile' + i, 'a');
  fs.closeSync(fd);
}
```

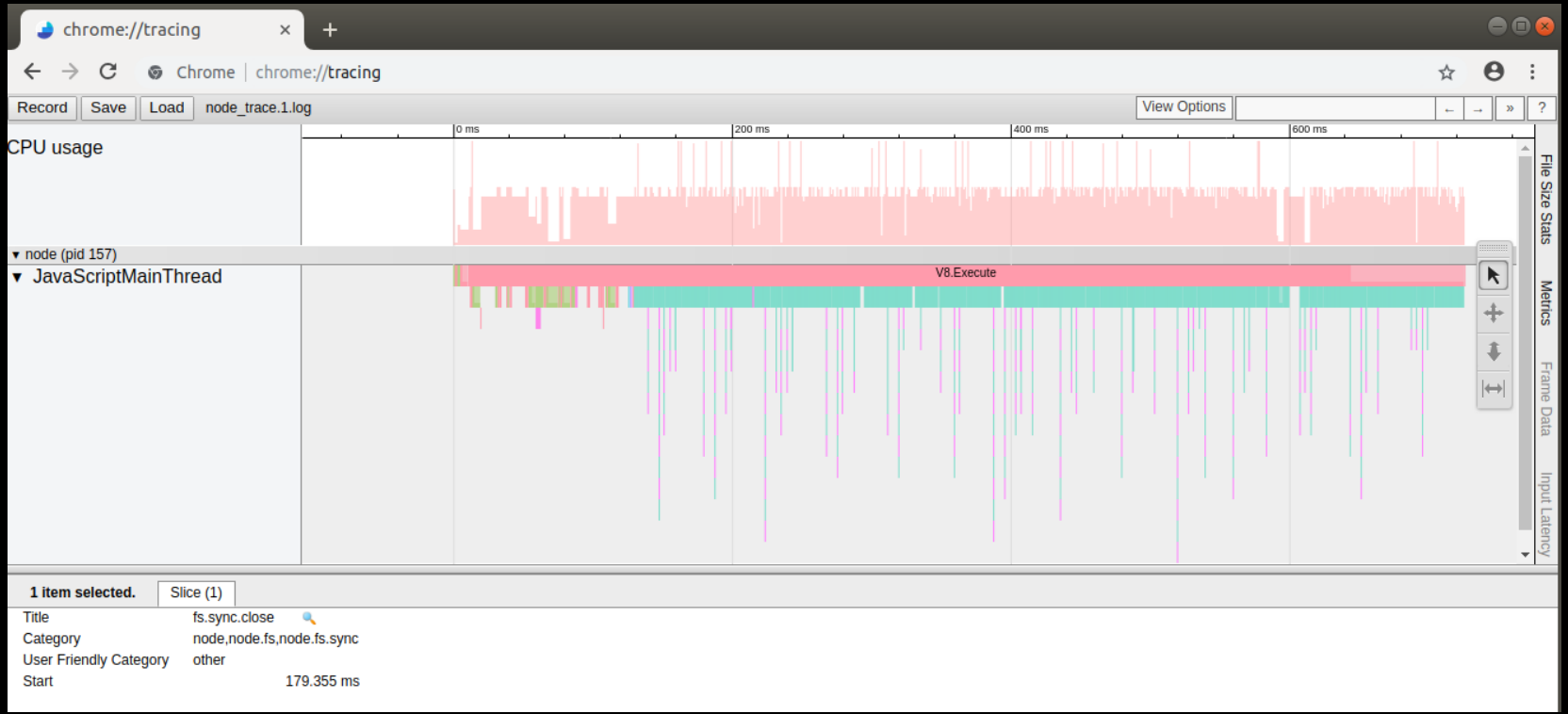
Scenario 2 – What is my app doing

node_trace.1.log





Scenario 2 – What is my app doing



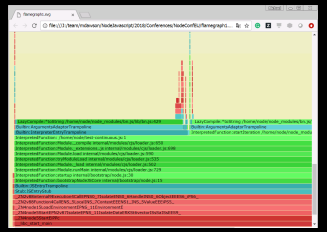
Scenario 3 – Performance Flame Graphs

- Where is my app spending CPU cycles?
- Will be part of best practices, some current info

  https://github.com/nodejs/nodejs.org/pull/1444

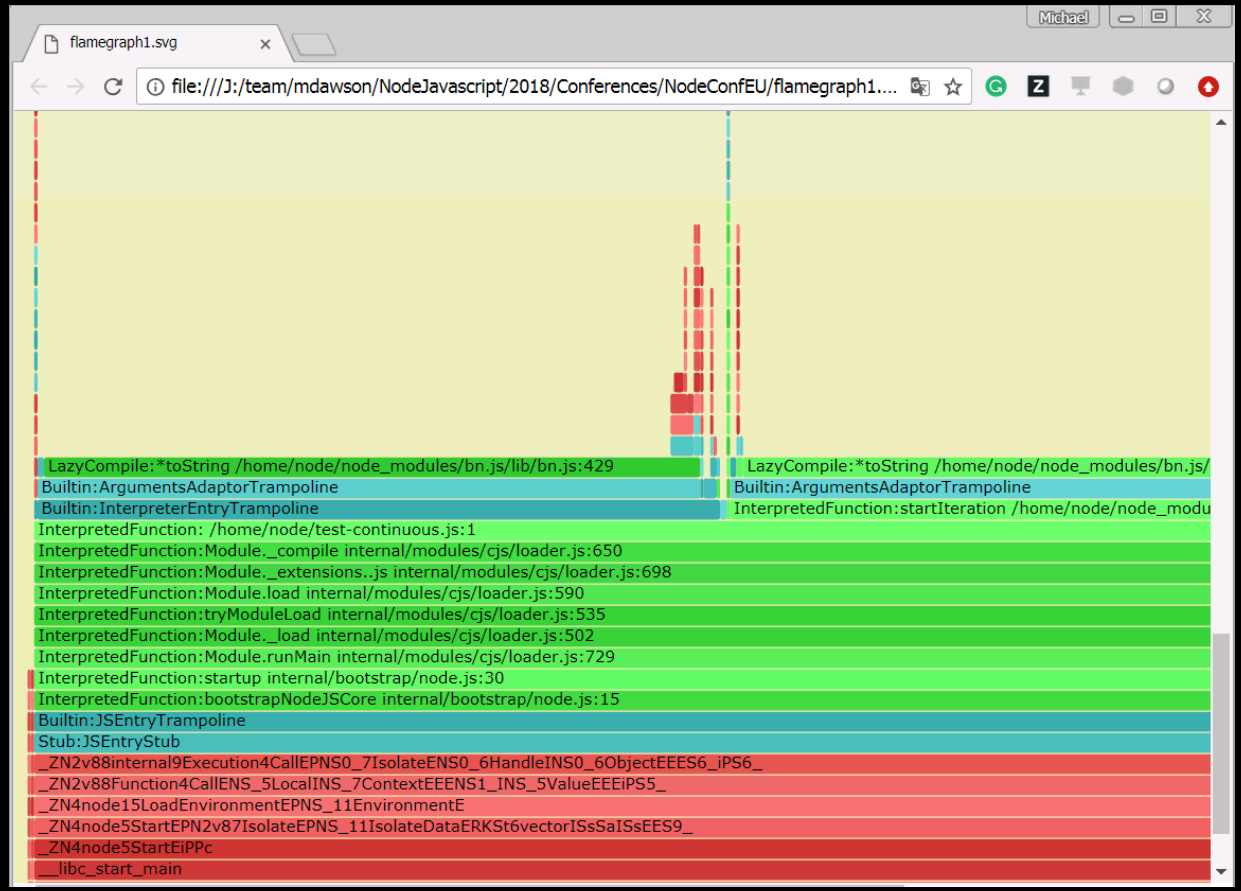
  https://github.com/mmarchini/nodejs-production-diagnostic-tools#linux-perf

- Will demo perf, also V8 profiler and tools that visualize.

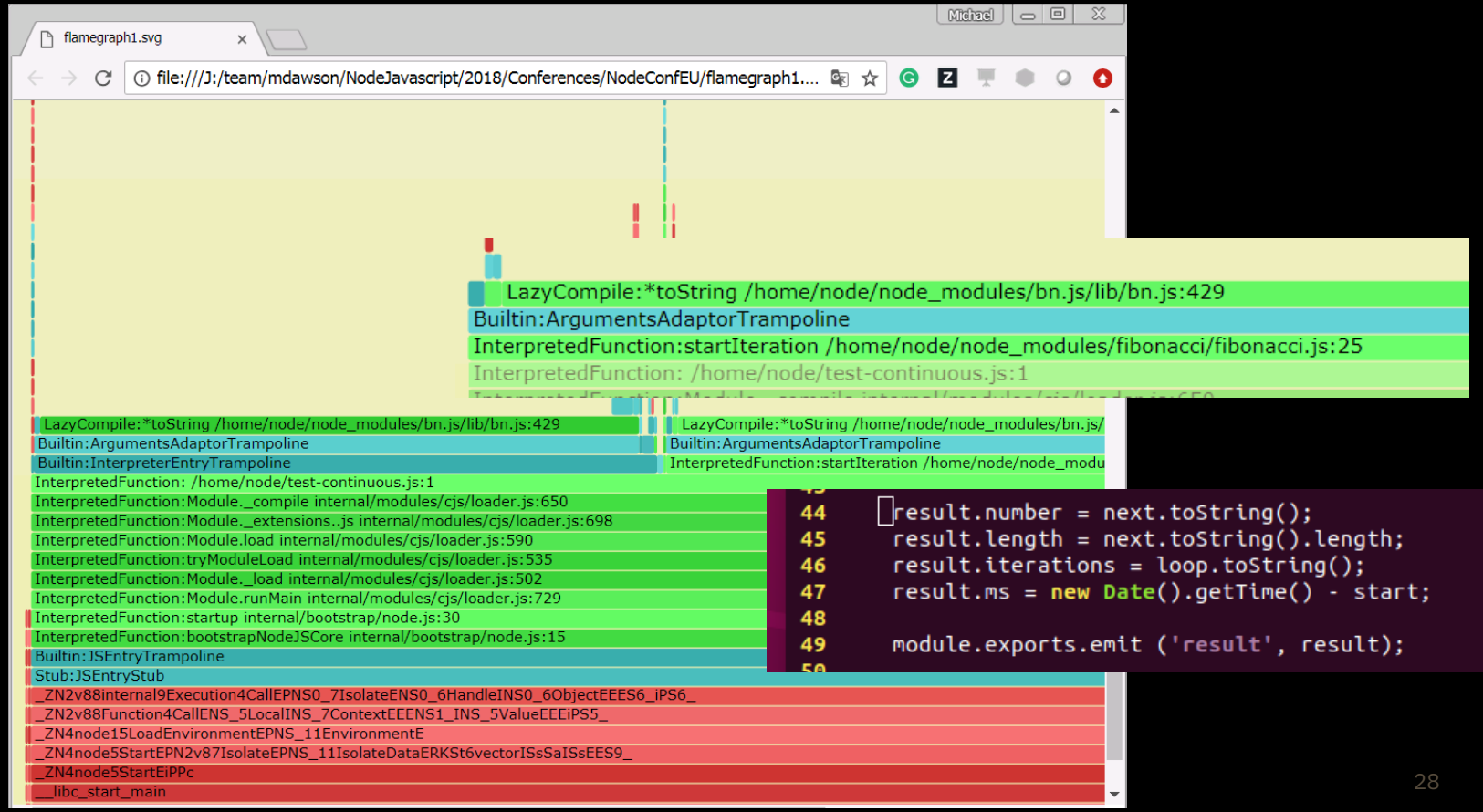


Demo

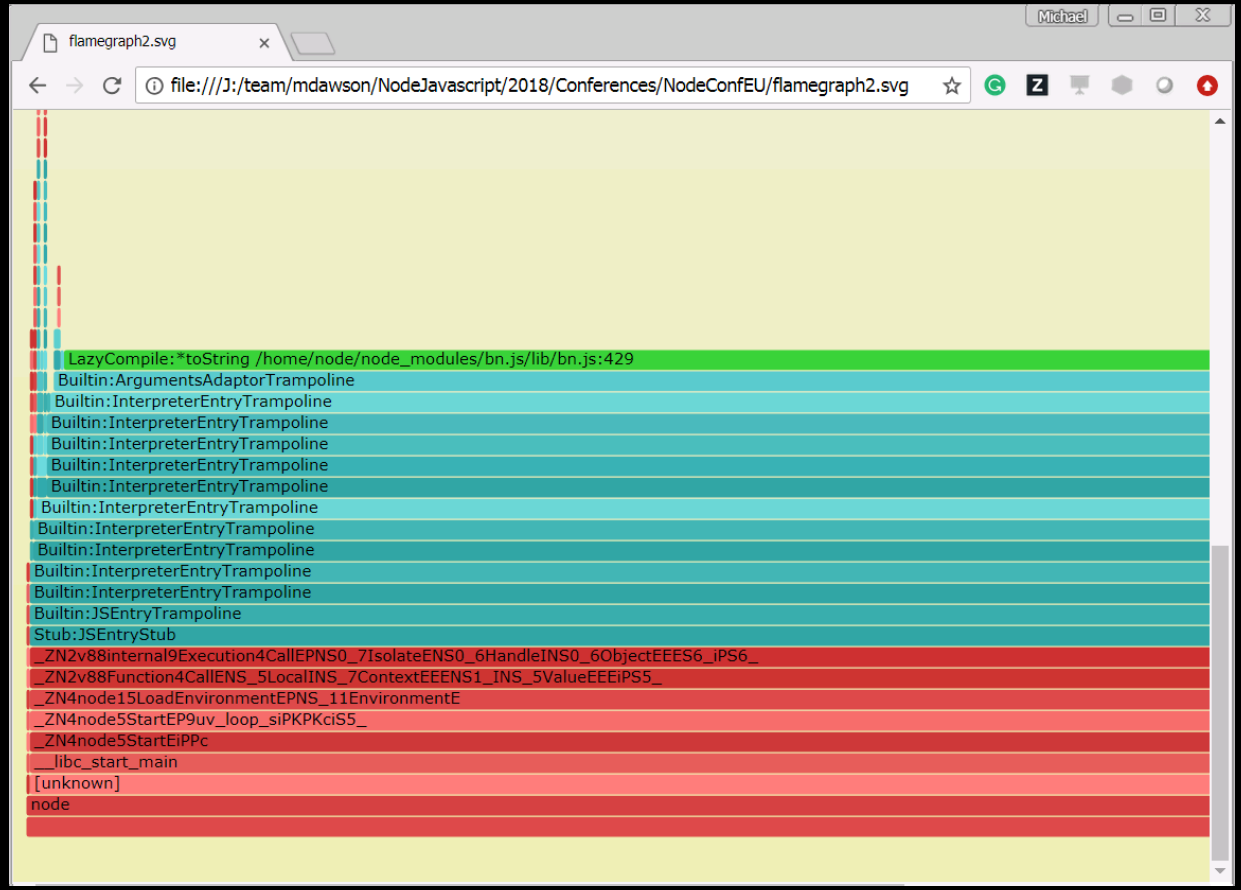
Scenario 3 – Flame Graphs



Scenario 3 – Flame Graphs



Scenario 3 – Flame Graphs



- 8.X missing info ?
- Importance of Tiers/testing

Recap

- Diagnostic Scenarios/Toolbox
- Diagnostics Working Group
- How this helps you

Help Out!

- Find an Initiative
- Make contact
- Contribute

Join us in creating the future of Node.js !

Copyrights and Trademarks