IBM

Enterprise/Cloud-ready Node.js

Michael Dawson
IBM Community Lead for Node.js

Jesse Gorzinski
jgorzins@us.ibm.com
Powerpoint Stealer

i for Business

1

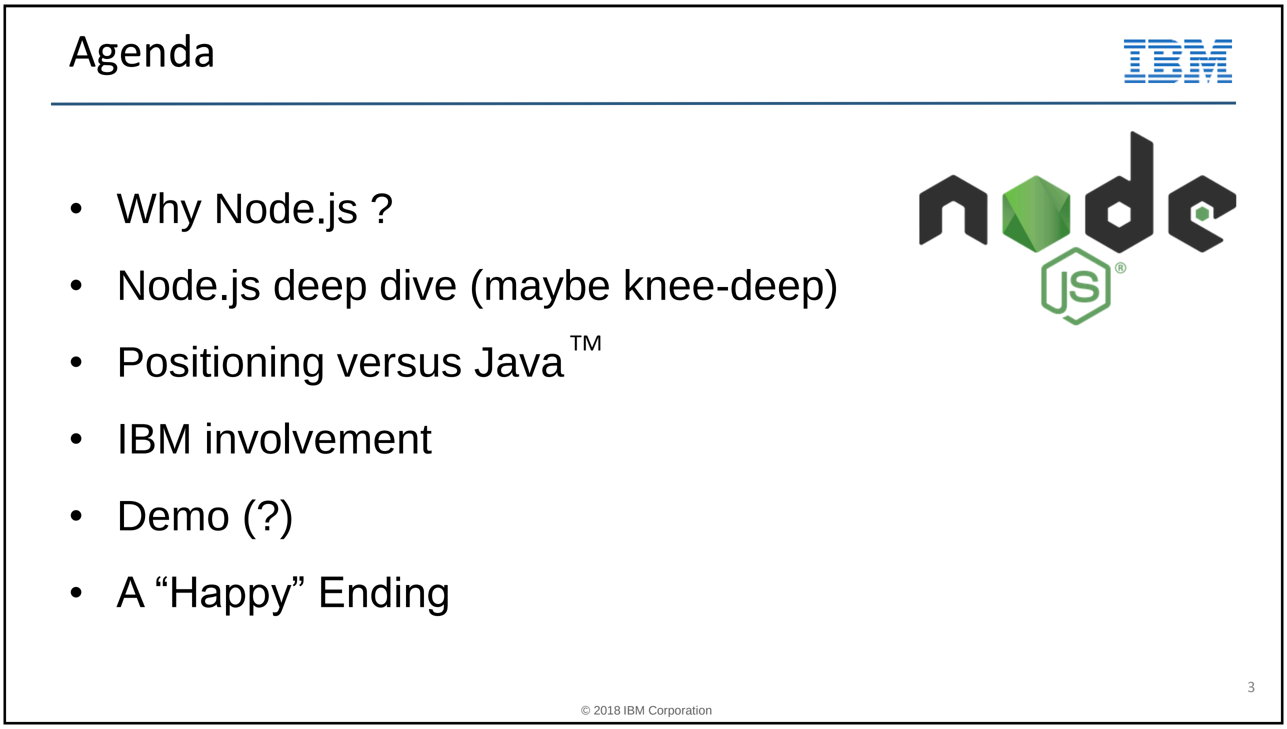
Atwood's Law: 2007

IBM

"Any application that can be written in JavaScript, will eventually be written in JavaScript."

—Jeff Atwood, Cofounder of StackOverflow

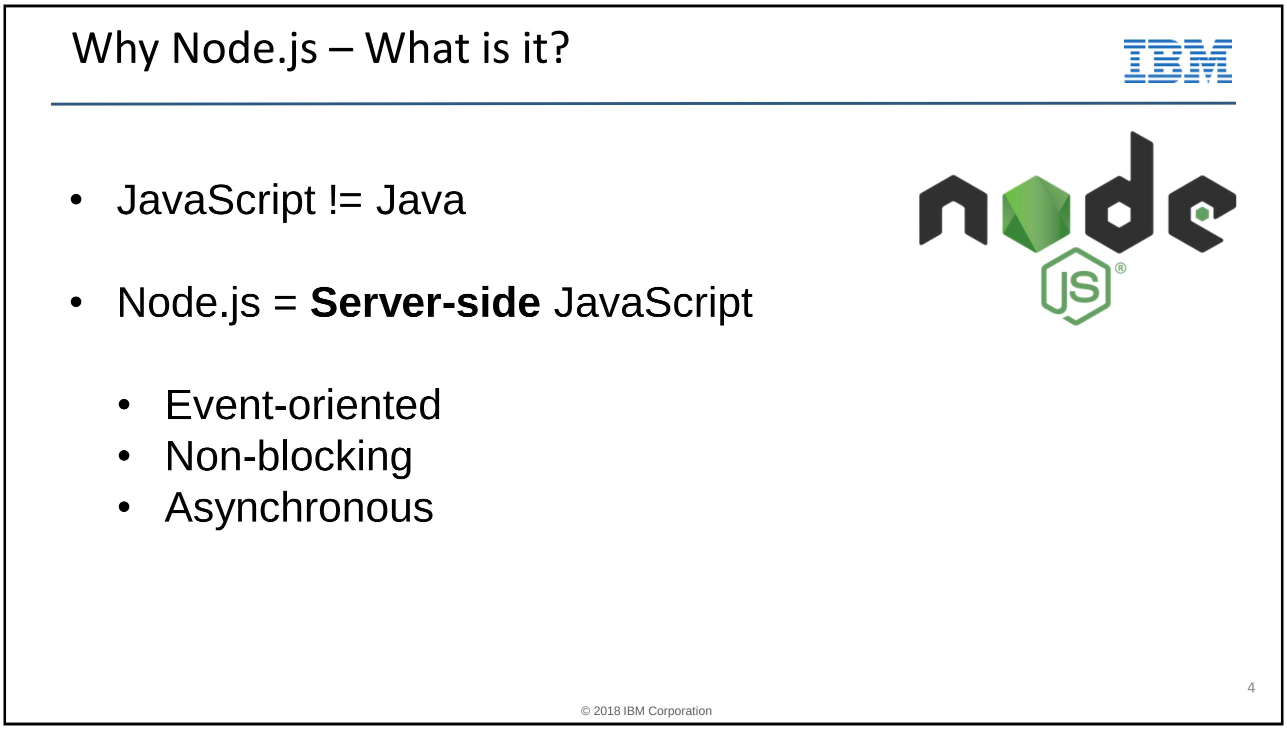
2



Agenda

- Why Node.js ?
- Node.js deep dive (maybe knee-deep)
- Positioning versus Java™
- IBM involvement
- Demo (?)
- A “Happy” Ending

3



Why Node.js – What is it?

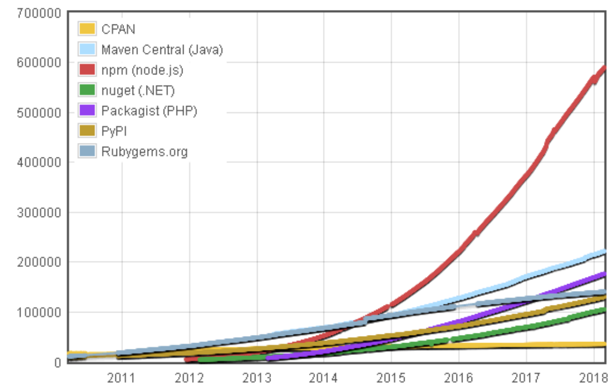
- JavaScript != Java
- Node.js = **Server-side** JavaScript
  - Event-oriented
  - Non-blocking
  - Asynchronous

4

Why Node.js ? – Ecosystem

- There is a module for that
  - 700K modules
  - #1 on module counts
- #1 on Github (#projects)

Module Counts

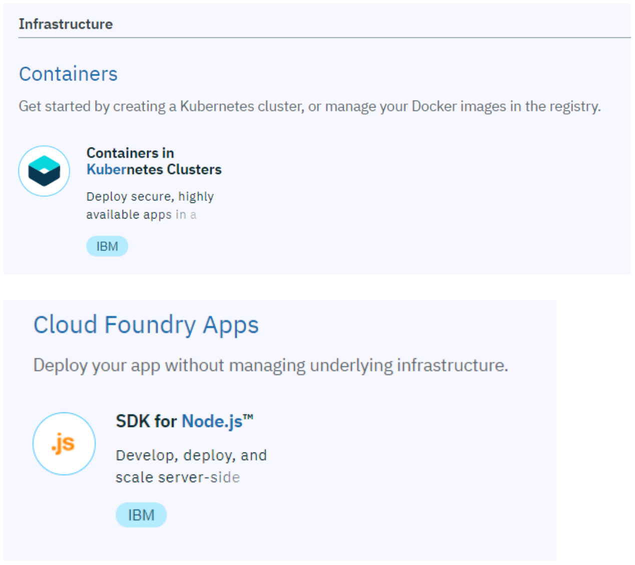


http://www.modulecounts.com/

6

Why Node.js ? – Ecosystem

- Most used runtime in IBM Cloud



Infrastructure

Containers

Get started by creating a Kubernetes cluster, or manage your Docker images in the registry.

Containers in Kubernetes Clusters

Deploy secure, highly available apps in a

IBM

Cloud Foundry Apps

Deploy your app without managing underlying infrastructure.

SDK for Node.js™

Develop, deploy, and scale server-side

IBM

7

Why Node.js ? – Productivity

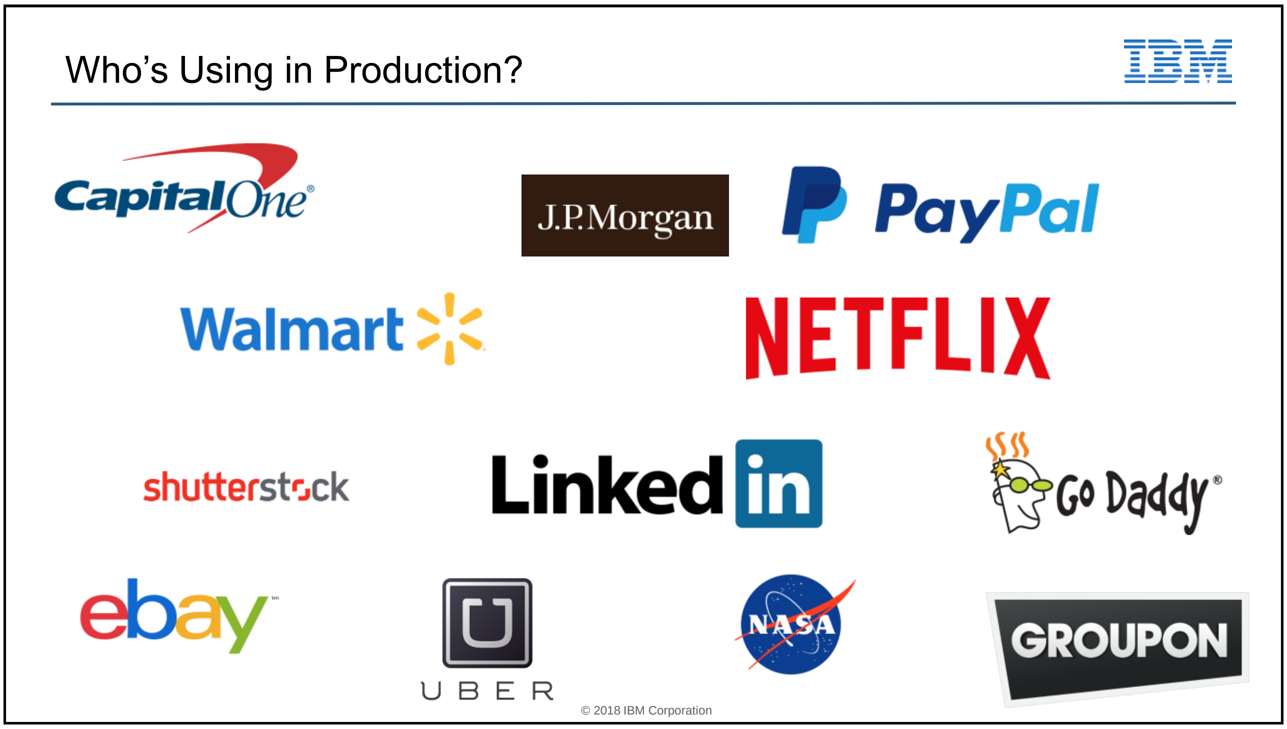
- Reuse of “isomorphic” code components
- Availability of JavaScript talent
- Developer satisfaction

9

Why Node.js ? – Productivity

- **Faster** development **less code**
- **PayPal** - https://www.paypal-engineering.com/2013/11/22/node-js-at-paypal/
  - **Took 1/2 time with less people**
  - **33% fewer lines of code**
  - **40% fewer files**
- **NextFlix** - http://www.infoworld.com/article/2610110/javascript/paypal-and-netflix-cozy-up-to-node-js.html
  - **"We're used to working in JavaScript all day long. Having Node just makes it feel like a very natural extension of our work environment,"**

8



Who's Using in Production?

IBM

Capital One

J.P.Morgan

PayPal

Walmart

NETFLIX

shutterstock

LinkedIn

Go Daddy

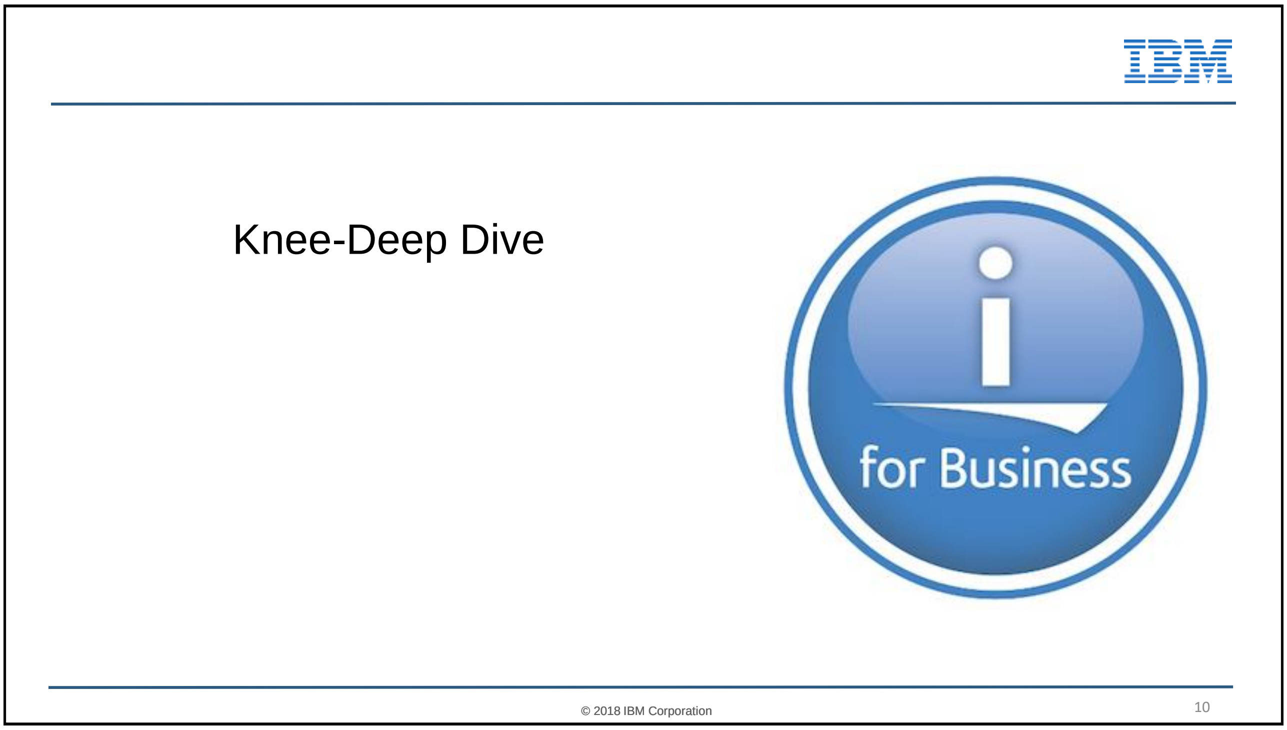
ebay

UBER

NASA

GROUPON

© 2018 IBM Corporation



IBM

Knee-Deep Dive

i for Business

© 2018 IBM Corporation

10

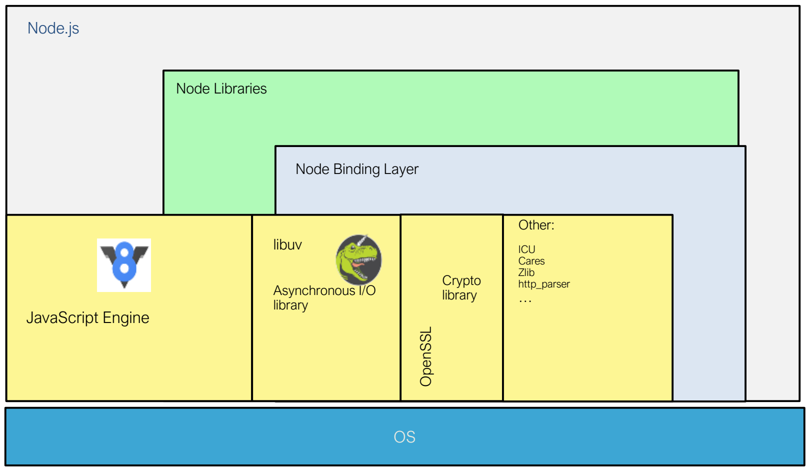
Node.js – Deep Dive – Key Characteristics

- Small (IBM i RPM)
  - Download **20 Mb**
- Fast startup
  - 60 ms
- Small footprint
  - 18 MB

https://benchmarking.nodejs.org/

11

Node.js – Deep Dive - Components



C/C++ Compiler

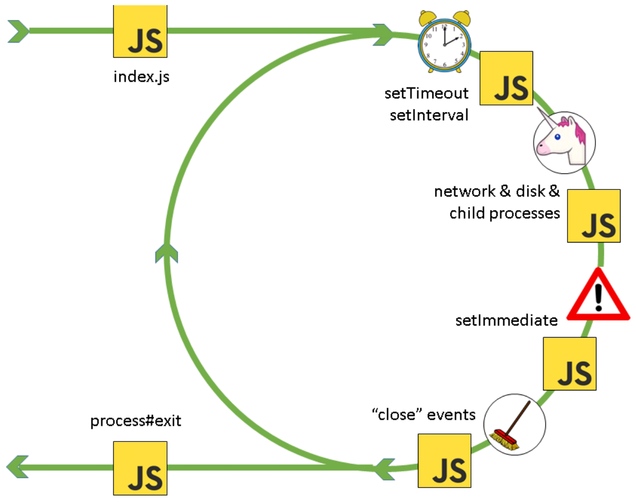
16

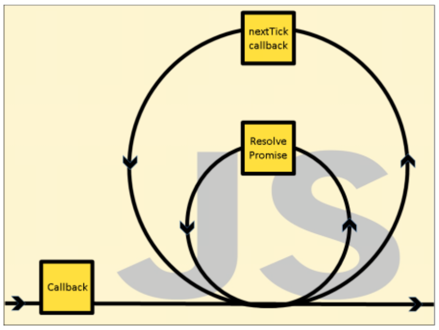
Node.js – Deep Dive - Programming Model

- Event Based

```
var http = require('http');

var server = http.createServer();
server.listen(8080);

server.on('request', function(request, response) {
               response.writeHead(200, {"Content-Type": "text/plain"});
               response.write("Hello World!\n");
               response.end();
});

server.on('connection', function(socket) {});
server.on('close', function() {});
server.on('connect', function(socket) {});
server.on('upgrade', function(request, socket, head) {});
server.on('clientError', function(exception, socket) {});
```

Node.js – Deep Dive – Event Loop





Node.js – Deep Dive – NPM

- 700,000+ modules!!
- Two types of installs:
  - Global: use for command-line utilities
  - Local (default): use for application dependencies
- Fully encapsulates:
  - Dependency list within package.json file
  - Dependencies themselves within node_modules/ directory
- Advantages:
  - Each application can operate independently
  - No global settings (extensions directory, classpaths, etc) to maintain
  - Portable

Node.js – Deep Dive – NPM

```
$ mkdir expressjs_app && cd expressjs_app
$ npm install express
express@4.12.0 node_modules/express
├── utils-merge@1.0.0
├── methods@1.1.1
├── fresh@0.2.4
├── merge-descriptors@0.0.2
├── cookie-signature@1.0.6
├── escape-html@1.0.1
├── range-parser@1.0.2
├── cookie@0.1.2
├── finalhandler@0.3.3
├── vary@1.0.0
├── content-type@1.0.1
├── parseurl@1.3.0
├── content-disposition@0.5.0
├── serve-static@1.9.1
├── path-to-regexp@0.1.3
├── depd@1.0.0
├── on-finished@2.2.0 (ee-first@1.1.0)
├── qs@2.3.3
├── debug@2.1.1 (ms@0.6.2)
├── proxy-addr@1.0.6 (forwarded@0.1.0, ipaddr.js@0.1.8)
├── etag@1.5.1 (crc@3.2.1)
```

Node.js – Deep Dive – NPM

holds meta-data about application

```
$ npm init
```

Creates file `package.json`

```
{
  "name": "expressjs_app",
  "version": "0.0.0",
  "description": "",
  "main": "app.js",
  "dependencies": {
    "express": "^4.12.0"
  },
  "devDependencies": {},
  "author": "Aaron Bartell",
  "license": "ISC"
}
```

Installs these modules when `npm install` is run.

docs.npmjs.com/cli/init - package.json creation
docs.npmjs.com/files/package.json - Docs
browsenpm.org/package.json - Easier docs

Connecting to Db2 / RPG

- Most important task for developing Node.js applications on the IBM i is connecting to Db2 and/or RPG
- All available on NPM
- For RPG, CL, QSH, Db2, etc, use itoolkit

- Some options for Db2:

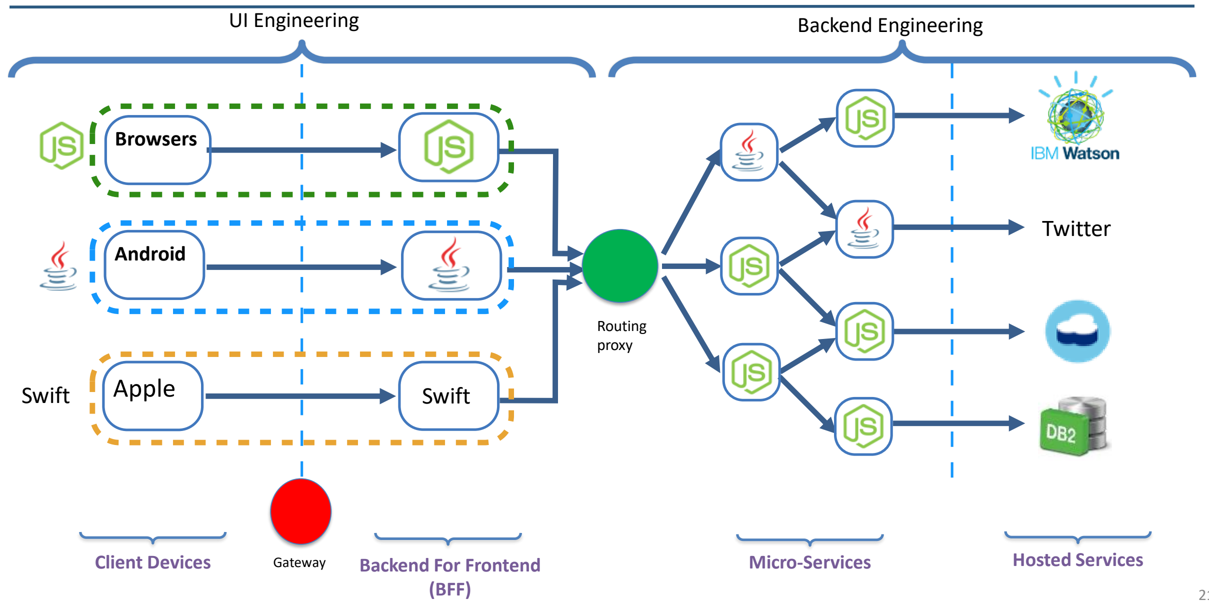
  1. ibm_db
     – LUW license needed
  2. idb-connector
     – Direct Access (traditional)
  3. idb-pconnector
     – Direct Access (Promises-based)
  4. node-odbc
     – Uses an ODBC driver

idb-pconnector example

```
const {Connection} = require('idb-pconnector');

async function execExample() {
  try {
    let statement =  new Connection().connect().getStatement();

    let result = await statement.exec('SELECT * FROM MYSCHEMA.TABLE');

    console.log(`Select results: \n${JSON.stringify(result)}`);

  } catch(error) {
       console.error(`Error was: \n${error.stack}`);
    }
}

execExample();
```

Node.js and Java

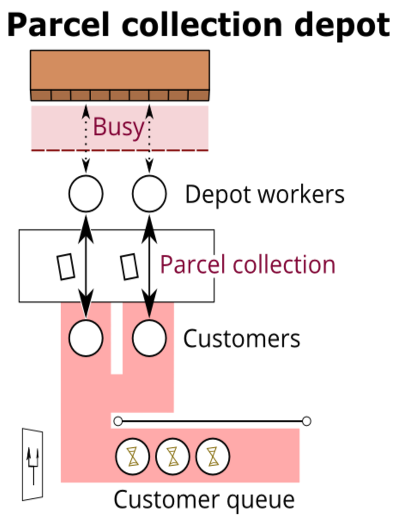
Node.js With Java



Node.js With Java – Scaling with Java

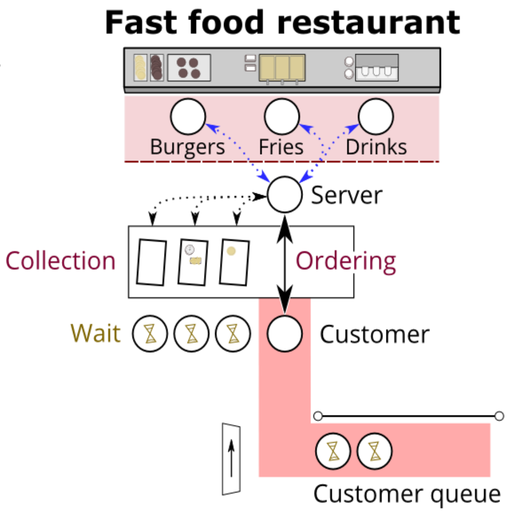
•

- One thread (or process) per connection
  - Each thread waits on a response
  - Scalability determined by number of threads
- Each thread:
  - Consumes memory
  - Is relatively idle
- Concurrency determined by number of depot workers

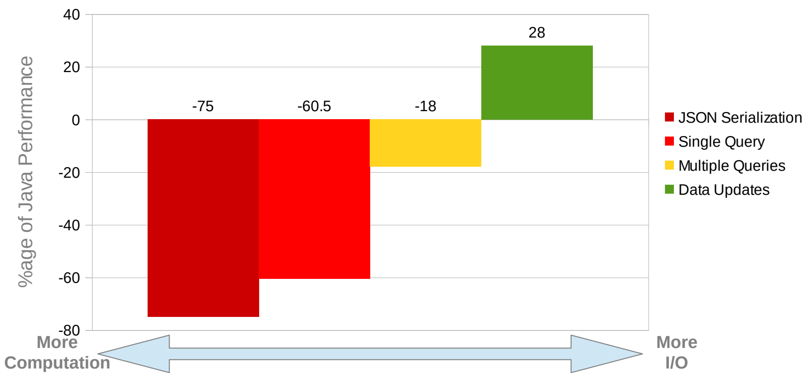


Node.js versus Java – Scaling with Node.js

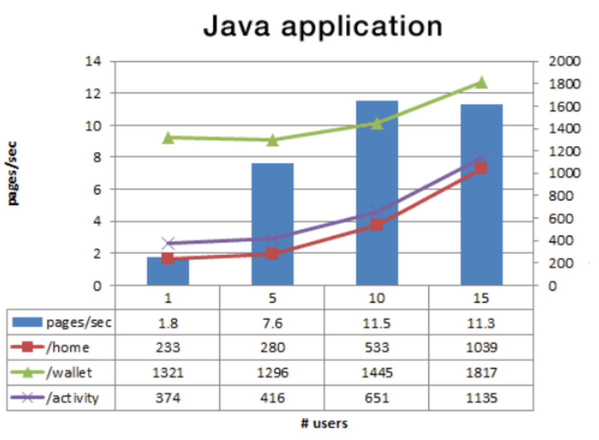
- One thread multiplexes for multiple requests
  - No waiting for a response
  - Handles return from I/O when notified
- Scalability determined by:
  - CPU Usage
  - “Back end” responsiveness
- Concurrency determined by how fast the food server can work



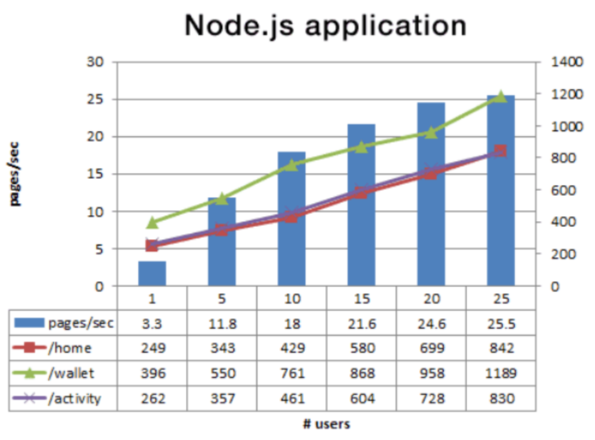
Node.js With Java– Tradeoffs



Why Node.js ? Performance



	1	5	10	15
▬ pages/sec	1.8	7.6	11.5	11.3
■ /home	233	280	533	1039
▲ /wallet	1321	1296	1445	1817
✕ /activity	374	416	651	1135



	1	5	10	15	20	25
▬ pages/sec	3.3	11.8	18	21.6	24.6	25.5
■ /home	249	343	429	580	699	842
▲ /wallet	396	550	761	868	958	1189
✕ /activity	262	357	461	604	728	830

https://www.paypal-engineering.com/2013/11/22/node-js-at-pypal/

Why Node.js ? - Performance

- Thousands of concurrent connections
- PayPal - https://www.paypal-engineering.com/2013/11/22/node-js-at-paypal/
  - **Double** number of requests/sec
  - Response times 35% **lower**
- Groupon – http://www.nearform.com/nodecrunch/node-js-becoming-go-technology-enterprise/
  - Reduced page load times by 50%

Node.js With Java – Choosing the Right Language

- Higher performance for I/O
- Easier async programming
- Fullstack/isomorphic development

Node.js versus Java – Choosing the Right Language

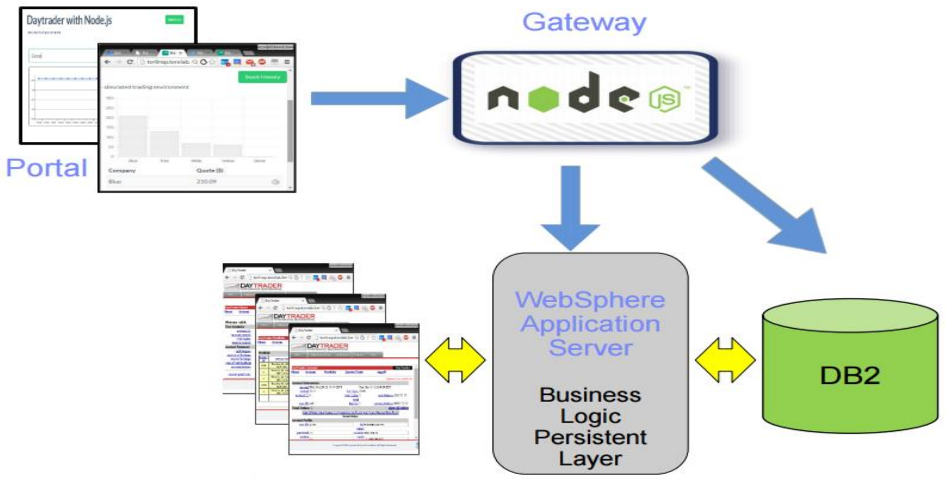
- Higher processing performance
- Type safety for calculations
- Rich processing frameworks

Node.js With Java– Choosing the Right Language

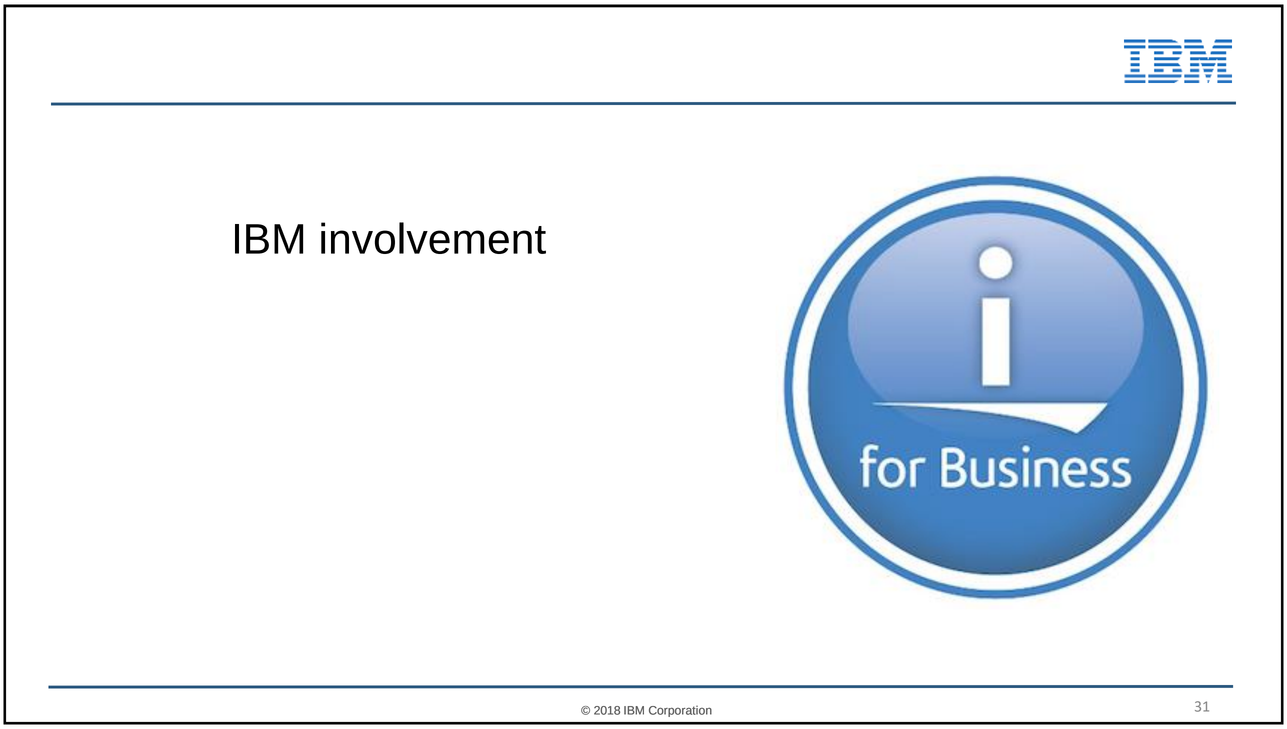
- Highly performant, scalable rich web applications
- Highly performant, reliable transaction processing
- Self-contained micro-service components

29

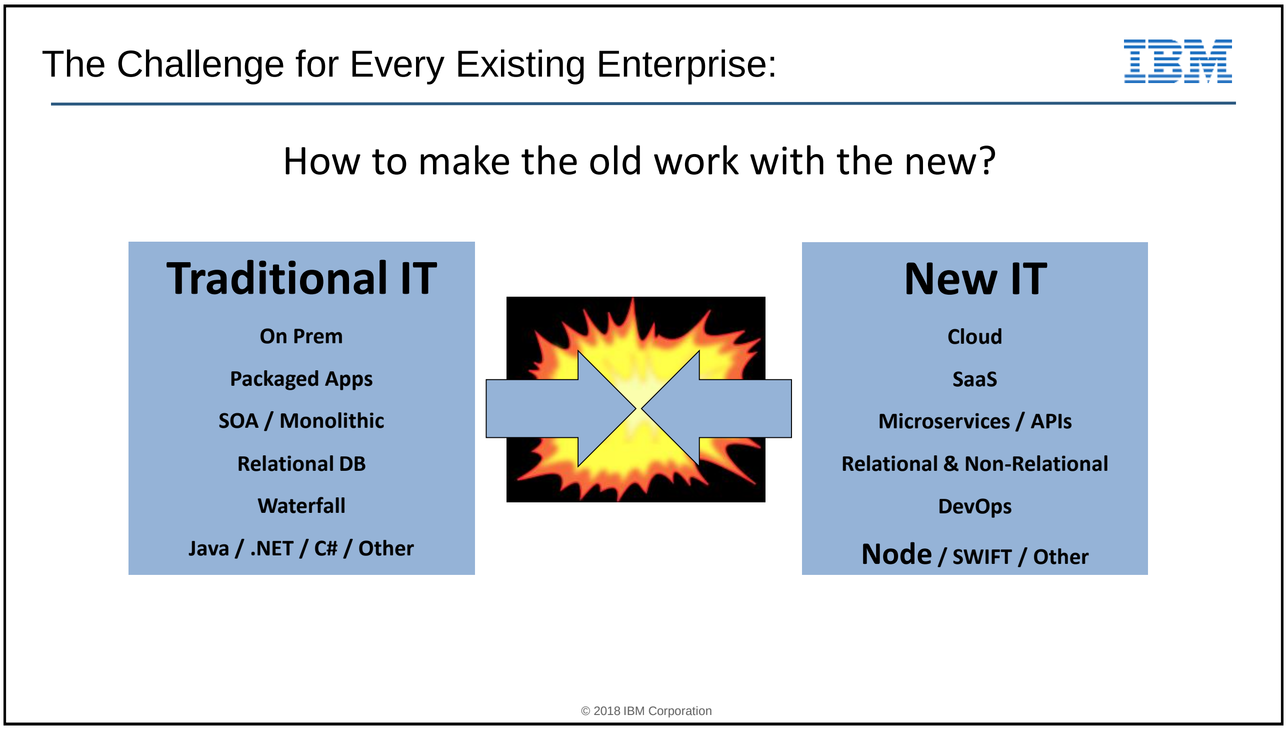
Node.js With Java– Hybrid applications



30



IBM involvement



The Challenge for Every Existing Enterprise:

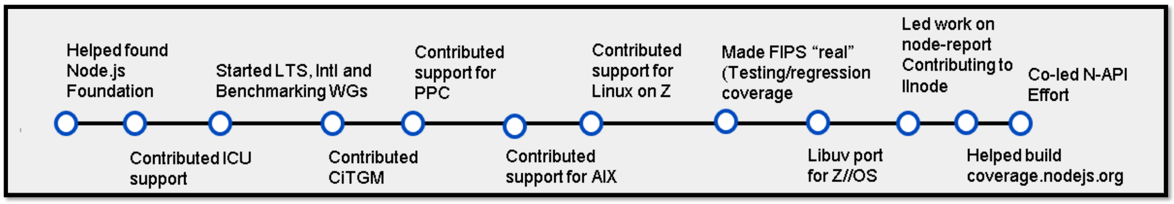
How to make the old work with the new?

Traditional IT	New IT
On Prem	Cloud
Packaged Apps	SaaS
SOA / Monolithic	Microservices / APIs
Relational DB	Relational & Non-Relational
Waterfall	DevOps
Java / .NET / C# / Other	**Node** / SWIFT / Other

IBM Node.js Strategy

- Enterprise Ready Runtime
- Production Enablement
- Production Support

Enterprise Ready Runtime

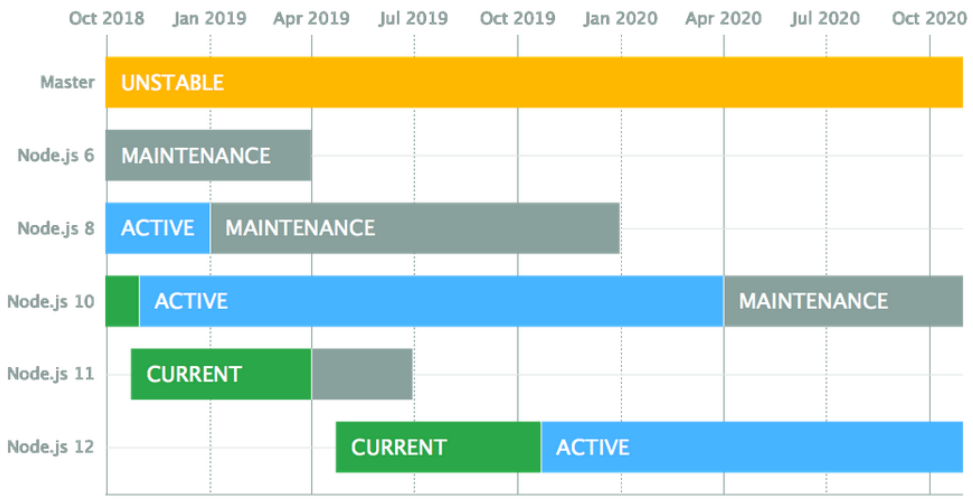

Helped found Node.js Foundation
Contributed ICU support
Started LTS, Intl and Benchmarking WGs
Contributed CiTGM
Contributed support for PPC
Contributed support for AIX
Contributed support for Linux on Z
Made FIPS “real” (Testing/regression coverage
Libuv port for Z//OS
Led work on node-report Contributing to llnode
Helped build coverage.nodejs.org
Co-led N-API Effort

N-API

- Before N-API
  - Native modules coded to V8 API’s
  - Modules needed to be recompiled for each new version of V8
  - Source changes sometimes also needed
  - Code less portable
  - Required non-trivial currency cost for module owners

- After N-API
  - Native modules coded to N-API
  - Need to be only built once for each platform
  - No code changes or recompiles needed to work with future versions of Node.js

Stable and Predictable Releases - Schedule for 2018



https://github.com/nodejs/Release

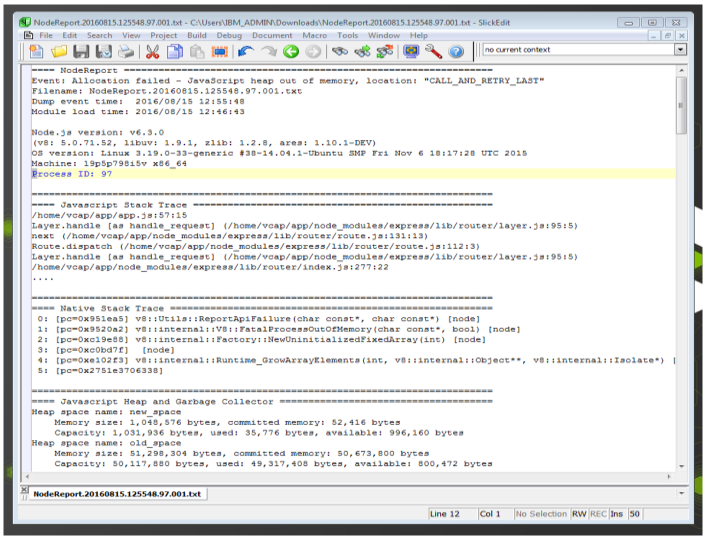
Node.js IBM – Tooling - NodeReport

NodeReport example - heap out of memory error

NodeReport content:

- Event summary
- Node.js and OS versions
- JavaScript stack trace
- Native stack trace
- Heap and GC statistics
- Resource usage
- libuv handle summary
- Environment variables
- OS ulimit settings

https://github.com/nodejs/nodereport


NodeReport.20160815.125548.97.001.txt - C:\Users\IBM_ADMIN\Downloads\NodeReport.20160815.125548.97.001.txt - SlickEdit
File Edit Search View Project Build Debug Document Macro Tools Window Help
no current context

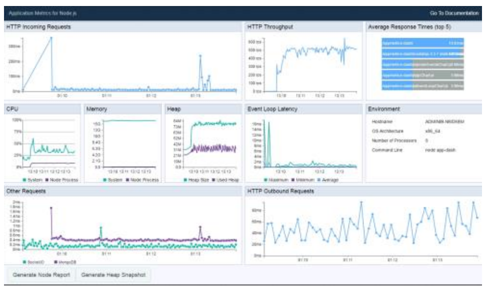
```
==== NodeReport ===============================================================
Event: Allocation failed - JavaScript heap out of memory, location: "CALL_AND_RETRY_LAST"
Filename: NodeReport.20160815.125548.97.001.txt
Dump event time:  2016/08/15 12:55:48
Module load time: 2016/08/15 12:46:43

Node.js version: v6.3.0
(v8: 5.0.71.52, libuv: 1.9.1, zlib: 1.2.8, ares: 1.10.1-DEV)
OS version: Linux 3.19.0-33-generic #38~14.04.1-Ubuntu SMP Fri Nov 6 18:17:28 UTC 2015
Machine: 19p5p798i5v x86_64
Process ID: 97

===============================================================================
==== Javascript Stack Trace ===================================================
/home/vcap/app/app.js:57:15
Layer.handle [as handle_request] (/home/vcap/app/node_modules/express/lib/router/layer.js:95:5)
next (/home/vcap/app/node_modules/express/lib/router/route.js:131:13)
Route.dispatch (/home/vcap/app/node_modules/express/lib/router/route.js:112:3)
Layer.handle [as handle_request] (/home/vcap/app/node_modules/express/lib/router/layer.js:95:5)
/home/vcap/app/node_modules/express/lib/router/index.js:277:22
....

===============================================================================
==== Native Stack Trace =======================================================
 0: [pc=0x951ea5] v8::Utils::ReportApiFailure(char const*, char const*) [node]
 1: [pc=0x9520a2] v8::internal::V8::FatalProcessOutOfMemory(char const*, bool) [node]
 2: [pc=0xc19e88] v8::internal::Factory::NewUninitializedFixedArray(int) [node]
 3: [pc=0xc0bd7f]  [node]
 4: [pc=0xe102f3] v8::internal::Runtime_GrowArrayElements(int, v8::internal::Object**, v8::internal::Isolate*) [
 5: [pc=0x2751e3706338]

===============================================================================
==== Javascript Heap and Garbage Collector ====================================
Heap space name: new_space
    Memory size: 1,048,576 bytes, committed memory: 52,416 bytes
    Capacity: 1,031,936 bytes, used: 35,776 bytes, available: 996,160 bytes
Heap space name: old_space
    Memory size: 51,298,304 bytes, committed memory: 50,673,800 bytes
    Capacity: 50,117,880 bytes, used: 49,317,408 bytes, available: 800,472 bytes
```

AppMetrics - open-source Node.js monitoring

What is it?
An open source module created by IBM for collecting application metrics to diagnose issues while developing your application. Metrics range from HTTP requests, event loop, memory usage, CPU usage, MongoDB connects, and more.

Why use it?
Monitor and diagnose issues while developing your application. App Metrics then connects with IBM Cloud and API Connect for auto-scaling and more detailed availability monitoring

How to get it?
Github at https://github.com/RuntimeTools/appmetrics. Users can view the dashboard by going to /appmetrics-dash or feeding it into their existing dashboard.



IBM Node.js Community Leadership

Participation in Technical Steering Committee

Michael
Dawson

IBM Node.js Community Leadership

9 Core Collaborators

Michel
Dawson

Ben
Noordhuis

Gireesh
Punathil

Bethany
Griggs

Yi-Hong
Wang

Sam
Roberts

Steven
Loomis

Richard
Lau

Ryan
Graham

Node.js IBM – V8 Community Involvement

IBM

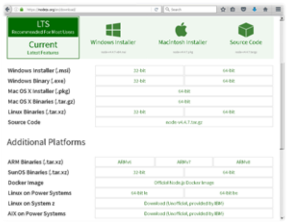
- Deep expertise at V8
- Developed ports to IBM Platforms
- Contribution back to official V8 repositories: https://github.com/v8/v8
  - **PPC**: V8 4.3 and later have full functional PPC implementation
  - **s390**: V8 5.1 and later have full functional implementatio1n
  - ~10-15 commits per week to V8 to maintain PPC/zlinux port
- Internal port for z/OS and IBM i

Freedom of Platform Choice

IBM

- Community Binaries
  - Linux on Z
  - Linux on P
  - AIX

OpenPOWER LNX ONE AIX IBM



Windows Installer (.msi)	32-bit		64-bit
Windows Binary (.exe)	32-bit		64-bit
Mac OS X Installer (.pkg)	64-bit		
Mac OS X Binaries (.tar.gz)	64-bit		
Linux Binaries (.tar.xz)	32-bit		64-bit
Source Code	node-v4.4.7.tar.gz		

Additional Platforms

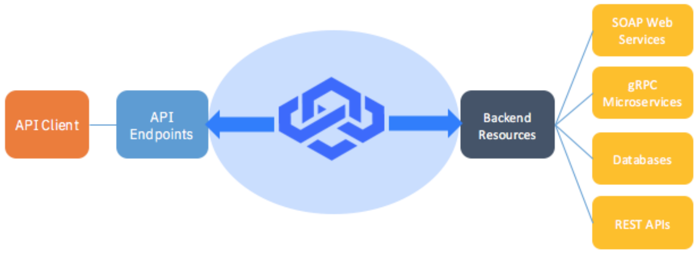
ARM Binaries (.tar.xz)	ARMv6	ARMv7	ARMv8
SunOS Binaries (.tar.xz)	32-bit		64-bit
Docker Image	Official Node.js Docker Image		
Linux on Power Systems	64-bit le		64-bit be
Linux on System z	Download (Unofficial, provided by IBM)		
AIX on Power Systems	Download (Unofficial, provided by IBM)		

- IBM Binaries
  - IBM i
  - z/OS

i for Business

LoopBack – open-source Node.js framework

- Extends Express to accelerate API creation
- Create APIs quickly as microservices from existing services and databases
- Connects the dots between accepting API requests and interacting with backend
- Built for developers by developers (Reached 10k+ GitHub stars)



Production Support - IBM Support for Runtimes

- Years of experience
- Foundation -Community binaries
- Advanced – Key Modules from the Ecosystem
  (Express.js & Loopback)

https://www.ibm.com/uk-en/marketplace/support-for-runtimes/faq

IBM TSS Support for IBM i

- Git
- Jenkins
- rsync
- Node.js
- Apache Tomcat
- WordPress
- Python

- For more resources, see my blog post: http://ibmsystemsmag.com/blogs/open-your-i/december-2018/a-game-changer-for-open-source-support/

Node.js foundation

- https://foundation.nodejs.org/

- The Node.js Foundation's mission is to enable widespread adoption and help accelerate development of Node.js and other related modules through an open governance model that encourages participation, technical contribution, and a framework for long term stewardship by an ecosystem invested in Node.js' success.

Node.js foundation – Silver Members

IBM

datree.io Fidelity GROUPON hackerone NearForm npm

Profound Logic redhat SafetyCulture snyk yld

Node.js foundation – Gold Members

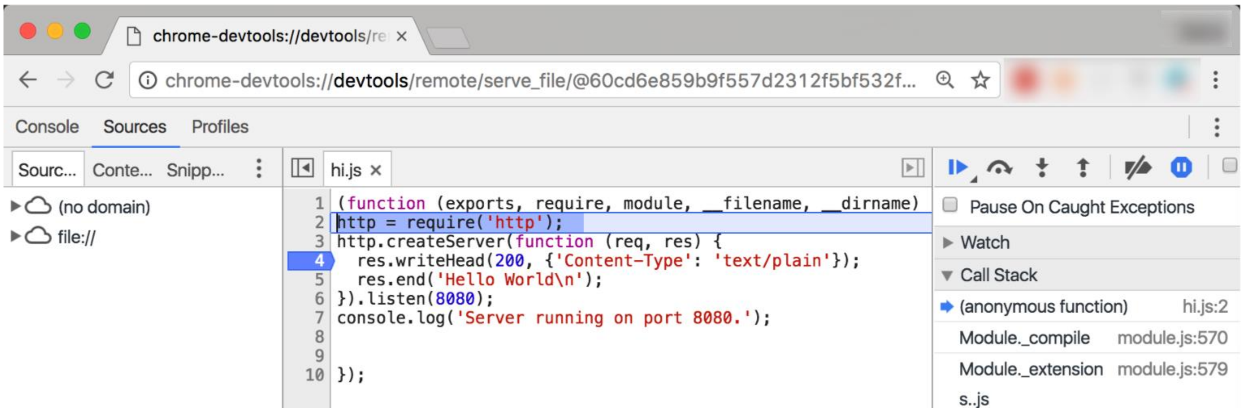
IBM

GoDaddy NODESOURCE®

PayPal

Node.js foundation – Platinum Members

Debugging Node.js in a browser


chrome-devtools://devtools/re
chrome-devtools://devtools/remote/serve_file/@60cd6e859b9f557d2312f5bf532f...
Console
Sources
Profiles
Sourc...
Conte...
Snipp...
(no domain)
file://
hi.js
Pause On Caught Exceptions
Watch
Call Stack

```
(function (exports, require, module, __filename, __dirname)
http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World\n');
}).listen(8080);
console.log('Server running on port 8080.');


});
```

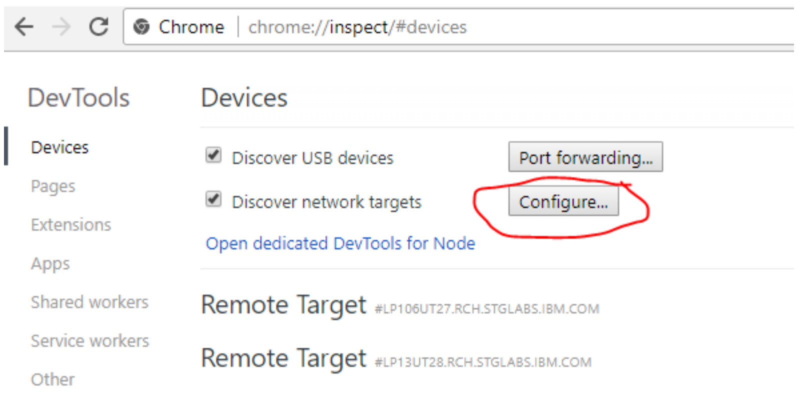
(anonymous function)	hi.js:2
Module._compile	module.js:570
Module._extension	module.js:579
s..js	

More debugging options: bit.ly/rs-debug-nodejs

Debugging Node.js in a browser

Visit `chrome://inspect` in chrome

Configure your hostname and port as a “network target” (port 9229 is default port)



Debugging Node.js in a browser

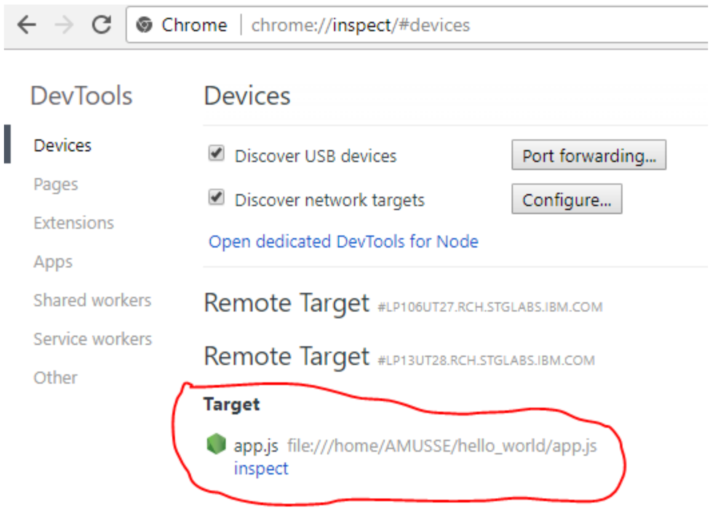
- $ node --inspect=0.0.0.0 hi.js
- Debugger listening on port 9229.

Start node with `--inspect`

NOTE: IP Address ‘0.0.0.0’ is important! Port will default to 9229 if not specified

Debugging Node.js in a browser

VOILA!! You will now see the remote target and can launch debug!



Devices

☑ Discover USB devices

☑ Discover network targets

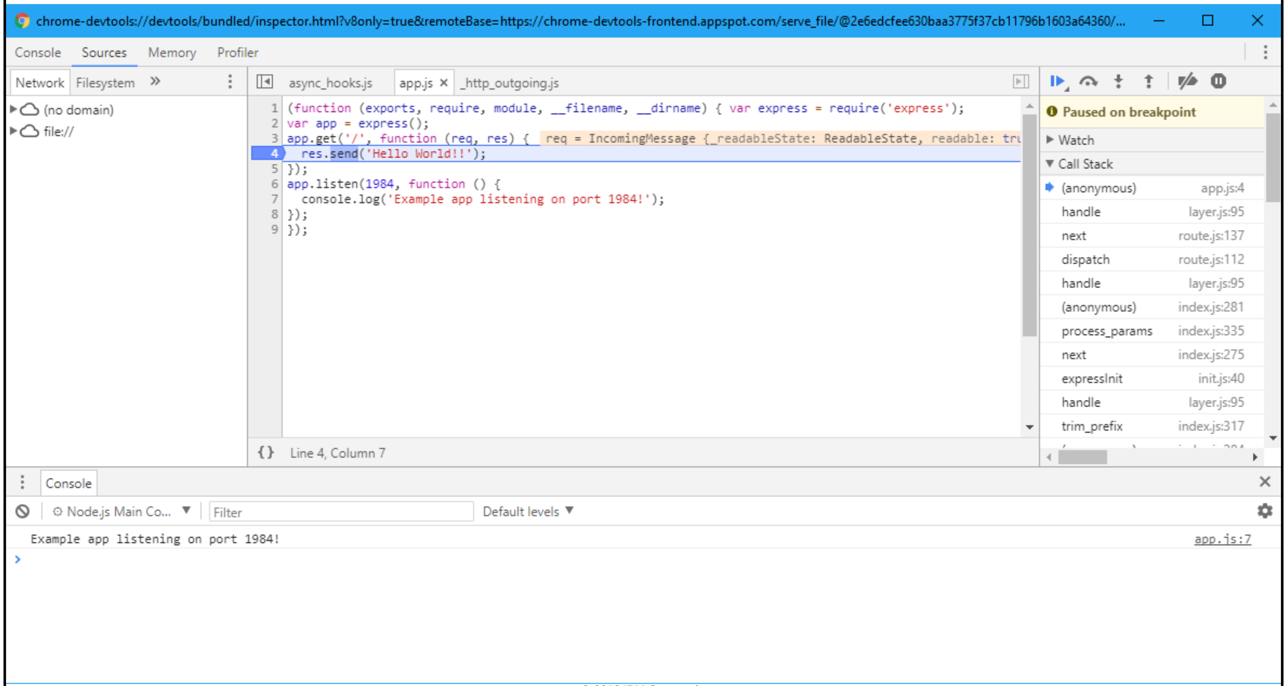
Open dedicated DevTools for Node

Remote Target #LP106UT27.RCH.STGLABS.IBM.COM

Remote Target #LP13UT28.RCH.STGLABS.IBM.COM

Target

app.js file:///home/AMUSSE/hello_world/app.js
inspect


chrome-devtools://devtools/bundled/inspector.html?v8only=true&remoteBase=https://chrome-devtools-frontend.appspot.com/serve_file/@2e6edcfee630baa3775f37cb11796b1603a64360/...
Console
Sources
Memory
Profiler
Network
Filesystem
(no domain)
file://
async_hooks.js
app.js
_http_outgoing.js
Watch
Call Stack
Console
Node.js Main Co...
Filter
Default levels

```
(function (exports, require, module, __filename, __dirname) { var express = require('express');
var app = express();
app.get('/', function (req, res) {  req = IncomingMessage {_readableState: ReadableState, readable: tru
  res.send('Hello World!!');
});
app.listen(1984, function () {
  console.log('Example app listening on port 1984!');
});
});
```

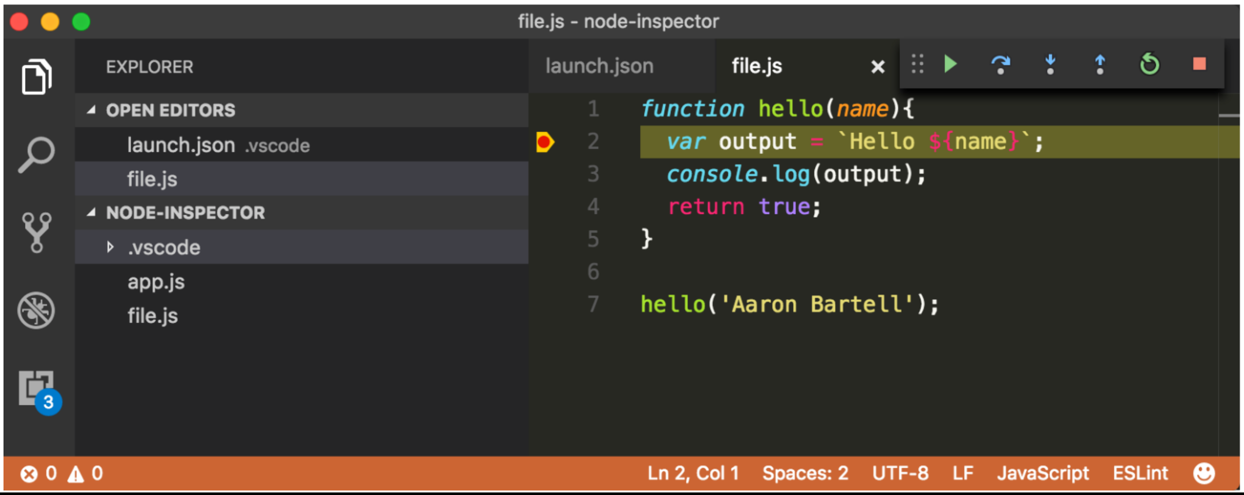
Line 4, Column 7

Paused on breakpoint

(anonymous)	app.js:4
handle	layer.js:95
next	route.js:137
dispatch	route.js:112
handle	layer.js:95
(anonymous)	index.js:281
process_params	index.js:335
next	index.js:275
expressInit	init.js:40
handle	layer.js:95
trim_prefix	index.js:317

```
Example app listening on port 1984!                                    app.js:7
```

Debugging Node.js in VSCode



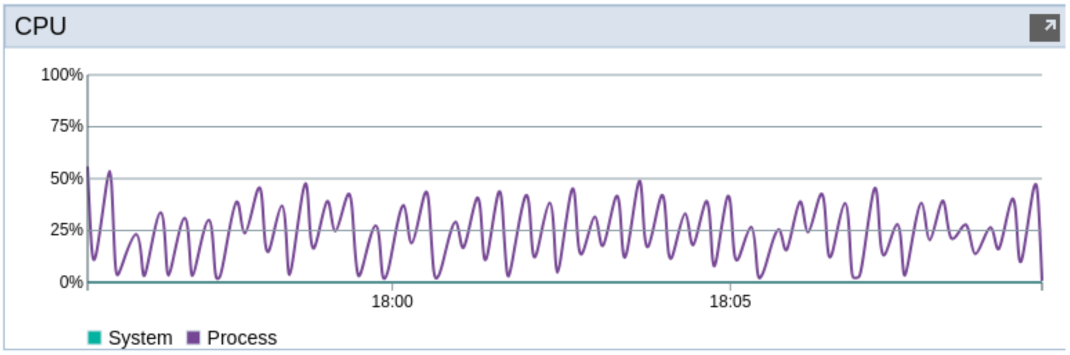
```
function hello(name){
  var output = `Hello ${name}`;
  console.log(output);
  return true;
}

hello('Aaron Bartell');
```

⊗ 0 ⚠ 0 Ln 2, Col 1 Spaces: 2 UTF-8 LF JavaScript ESLint

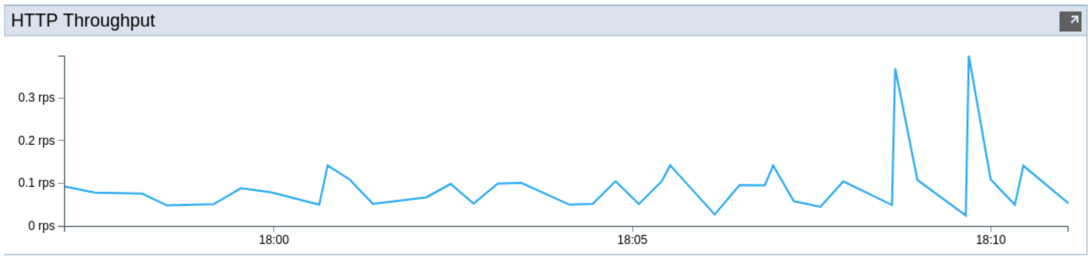
Demo!

56

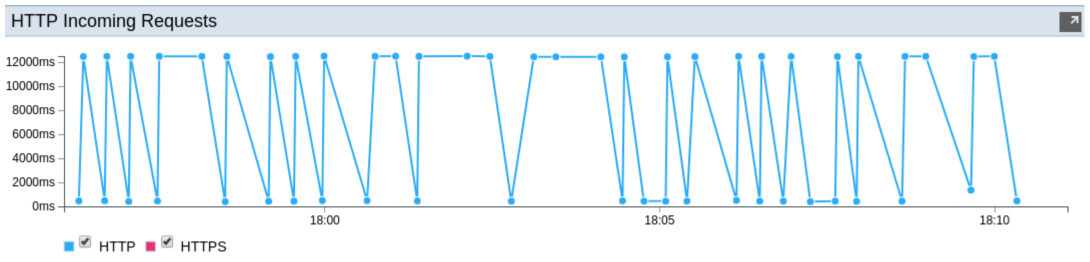
AppMetrics



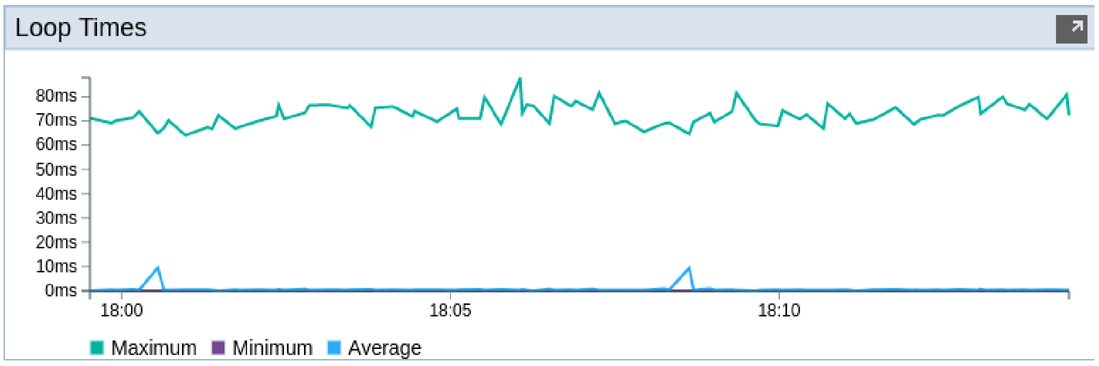
AppMetrics



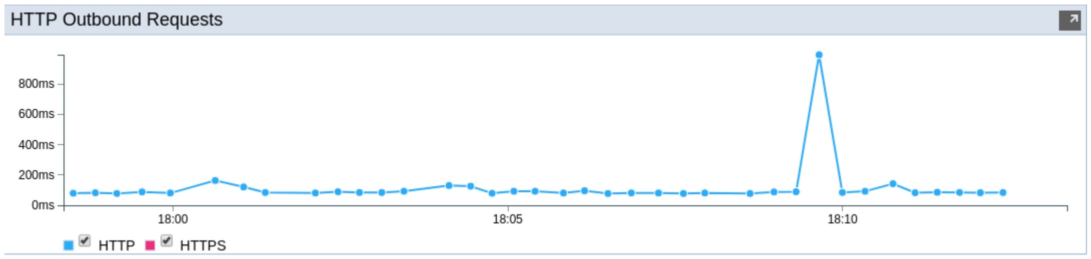
AppMetrics



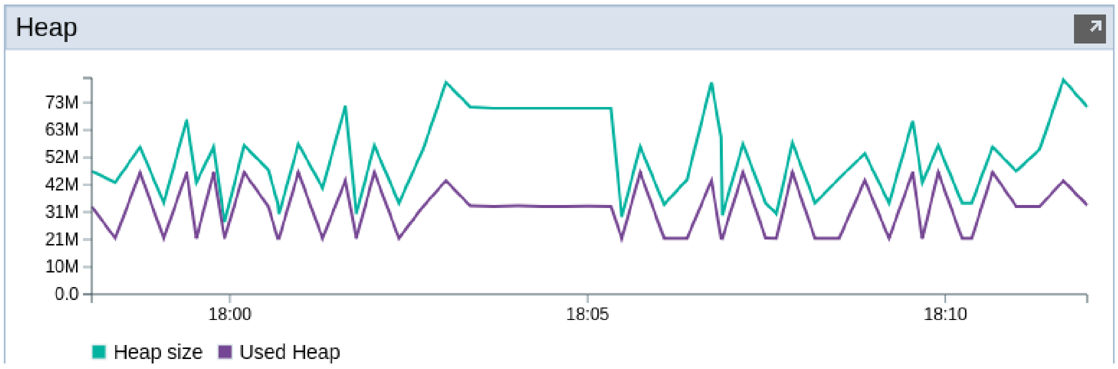
AppMetrics



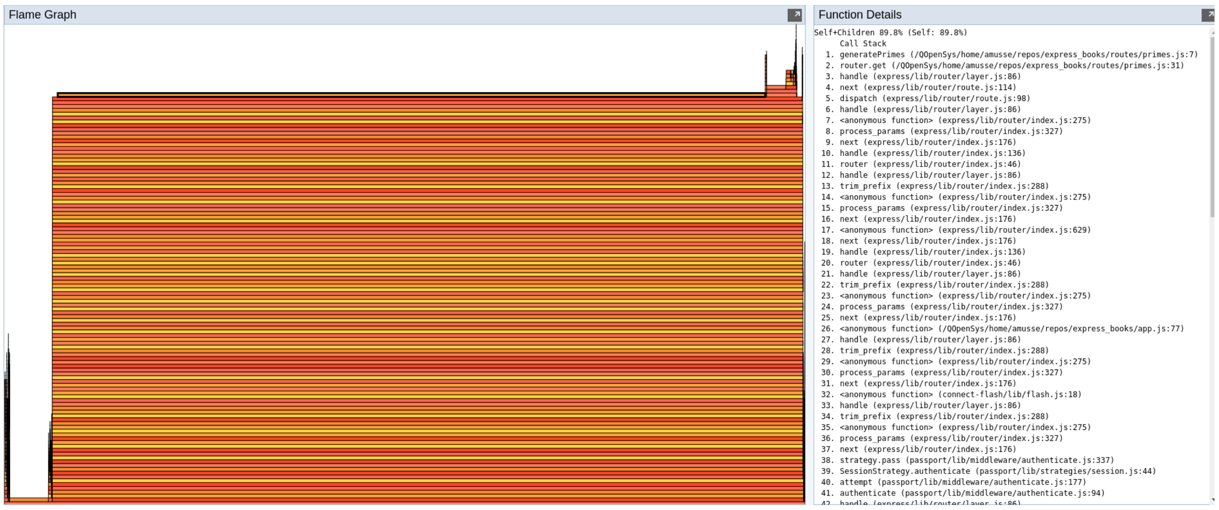
AppMetrics


HTTP Outbound Requests

AppMetrics

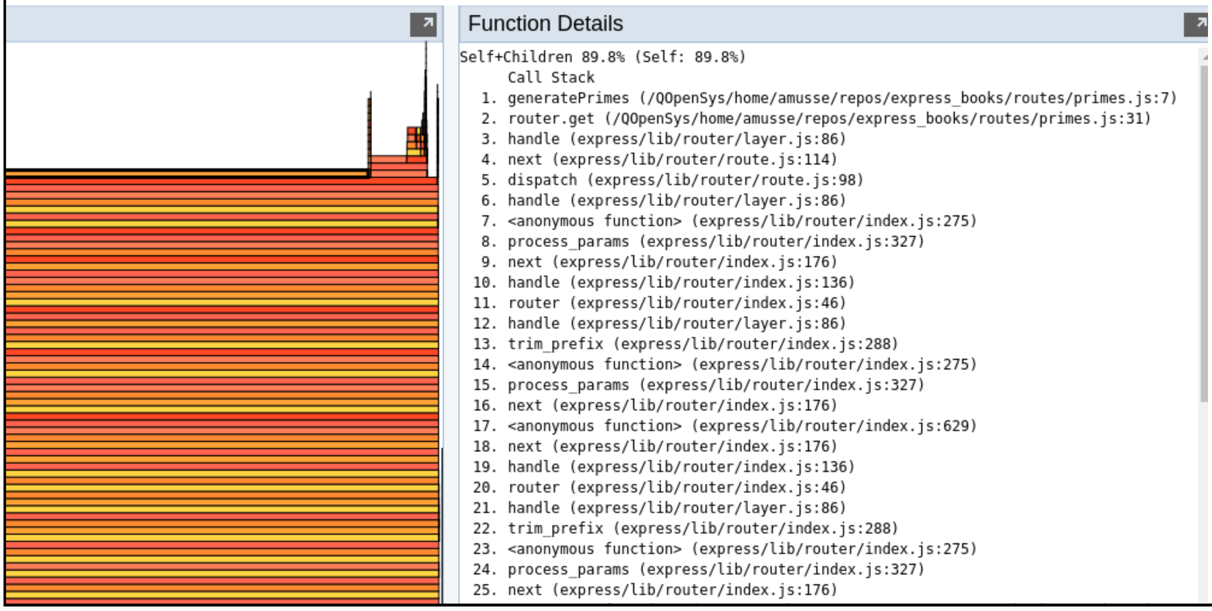

Heap

AppMetrics



```
Self+Children 89.8% (Self: 89.8%)
     Call Stack
  1. generatePrimes (/QOpenSys/home/amusse/repos/express_books/routes/primes.js:7)
  2. router.get (/QOpenSys/home/amusse/repos/express_books/routes/primes.js:31)
  3. handle (express/lib/router/layer.js:86)
  4. next (express/lib/router/route.js:114)
  5. dispatch (express/lib/router/route.js:98)
  6. handle (express/lib/router/layer.js:86)
  7. <anonymous function> (express/lib/router/index.js:275)
  8. process_params (express/lib/router/index.js:327)
  9. next (express/lib/router/index.js:176)
 10. handle (express/lib/router/index.js:136)
 11. router (express/lib/router/index.js:46)
 12. handle (express/lib/router/layer.js:86)
 13. trim_prefix (express/lib/router/index.js:288)
 14. <anonymous function> (express/lib/router/index.js:275)
 15. process_params (express/lib/router/index.js:327)
 16. next (express/lib/router/index.js:176)
 17. <anonymous function> (express/lib/router/index.js:629)
 18. next (express/lib/router/index.js:176)
 19. handle (express/lib/router/index.js:136)
 20. router (express/lib/router/index.js:46)
 21. handle (express/lib/router/layer.js:86)
 22. trim_prefix (express/lib/router/index.js:288)
 23. <anonymous function> (express/lib/router/index.js:275)
 24. process_params (express/lib/router/index.js:327)
 25. next (express/lib/router/index.js:176)
 26. <anonymous function> (/QOpenSys/home/amusse/repos/express_books/app.js:77)
 27. handle (express/lib/router/layer.js:86)
 28. trim_prefix (express/lib/router/index.js:288)
 29. <anonymous function> (express/lib/router/index.js:275)
 30. process_params (express/lib/router/index.js:327)
 31. next (express/lib/router/index.js:176)
 32. <anonymous function> (connect-flash/lib/flash.js:18)
 33. handle (express/lib/router/layer.js:86)
 34. trim_prefix (express/lib/router/index.js:288)
 35. <anonymous function> (express/lib/router/index.js:275)
 36. process_params (express/lib/router/index.js:327)
 37. next (express/lib/router/index.js:176)
 38. strategy.pass (passport/lib/middleware/authenticate.js:337)
 39. SessionStrategy.authenticate (passport/lib/strategies/session.js:44)
 40. attempt (passport/lib/middleware/authenticate.js:177)
 41. authenticate (passport/lib/middleware/authenticate.js:94)
 42. handle (express/lib/router/layer.js:86)
```

AppMetrics



```
Self+Children 89.8% (Self: 89.8%)
     Call Stack
  1. generatePrimes (/QOpenSys/home/amusse/repos/express_books/routes/primes.js:7)
  2. router.get (/QOpenSys/home/amusse/repos/express_books/routes/primes.js:31)
  3. handle (express/lib/router/layer.js:86)
  4. next (express/lib/router/route.js:114)
  5. dispatch (express/lib/router/route.js:98)
  6. handle (express/lib/router/layer.js:86)
  7. <anonymous function> (express/lib/router/index.js:275)
  8. process_params (express/lib/router/index.js:327)
  9. next (express/lib/router/index.js:176)
 10. handle (express/lib/router/index.js:136)
 11. router (express/lib/router/index.js:46)
 12. handle (express/lib/router/layer.js:86)
 13. trim_prefix (express/lib/router/index.js:288)
 14. <anonymous function> (express/lib/router/index.js:275)
 15. process_params (express/lib/router/index.js:327)
 16. next (express/lib/router/index.js:176)
 17. <anonymous function> (express/lib/router/index.js:629)
 18. next (express/lib/router/index.js:176)
 19. handle (express/lib/router/index.js:136)
 20. router (express/lib/router/index.js:46)
 21. handle (express/lib/router/layer.js:86)
 22. trim_prefix (express/lib/router/index.js:288)
 23. <anonymous function> (express/lib/router/index.js:275)
 24. process_params (express/lib/router/index.js:327)
 25. next (express/lib/router/index.js:176)
```

A “Happy” Ending

A “Hapi” Ending

67

Walmart creates a framework!

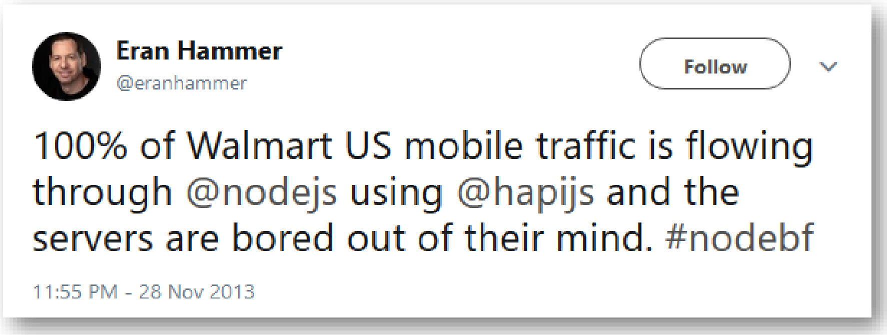
- Express.js appeared in 2009.
- Walmart saw Express.js insufficient for very large projects, but saw the huge potential in Node.js.
- Willing to invest millions of dollars in a new framework.

- https://garage.socialisten.at/2016/12/enterprise-level-backend-framework-from-walmart

68

Black Friday 2013

- Full deployment for all mobile shopping!!
- The hardware?
  - 10 CPU cores
  - 28 GB memory



Eran Hammer
@eranhammer

Follow

100% of Walmart US mobile traffic is flowing through @nodejs using @hapijs and the servers are bored out of their mind. #nodebf

11:55 PM - 28 Nov 2013

69

Q&A

The END!

70

Special notices

Special notices (cont.)