Why Node.js?

Michael Dawson
IBM Community Lead for Node.js

Jesse Gorzinski
jgorzins@us.ibm.com
Powerpoint Stealer

Atwood's Law: 2007

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

—Jeff Atwood, Cofounder of StackOverflow

Agenda

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

3

Why Node.js – What is it?

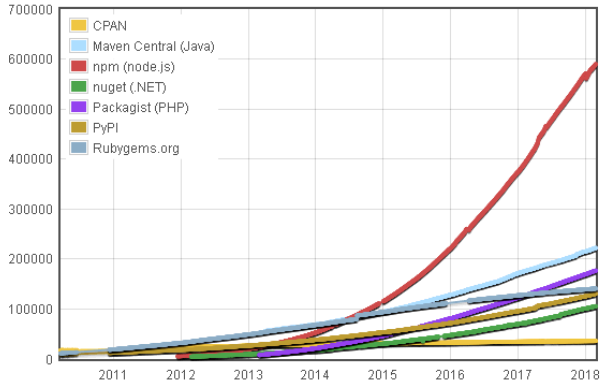
- JavaScript != Java
- Node.js = **Server-side** JavaScript
  - Event-oriented
  - Non-blocking
  - Asynchronous

4

Why Node.js ? – Ecosystem

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

Module Counts

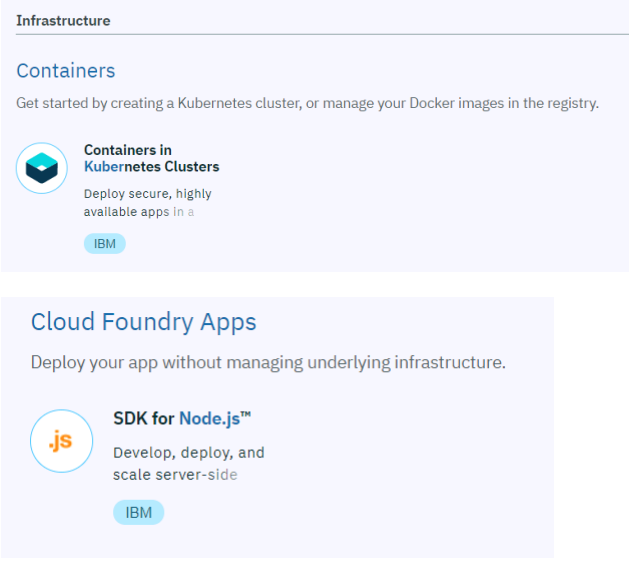


http://www.modulecounts.com/

6

Why Node.js ? – Ecosystem

- Most used runtime in IBM Cloud



7

Why Node.js ? – Productivity

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

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,"**

Who's Using in Production?

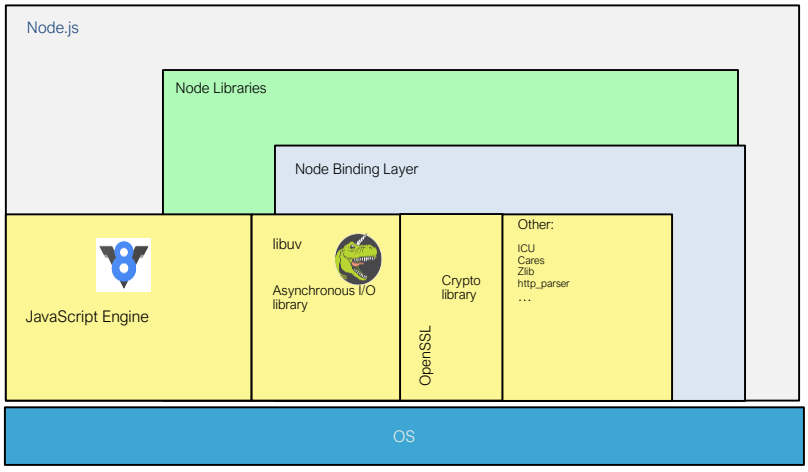
Knee-Deep Dive

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/

Node.js – Deep Dive - Components



C/C++ Compiler

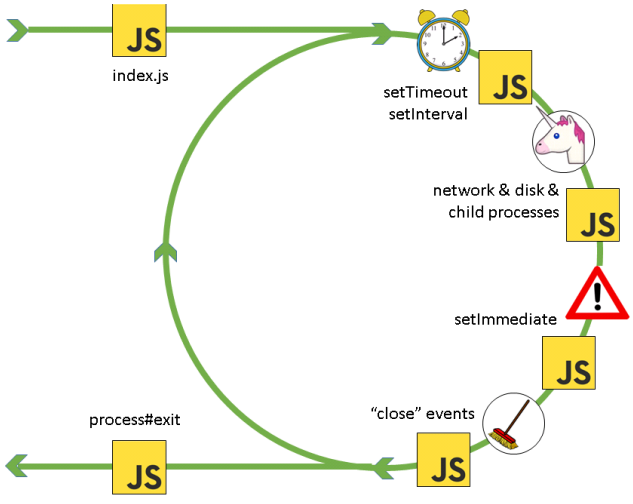
Node.js – Deep Dive - Programming Model

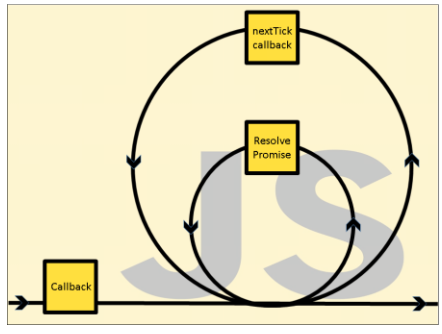
- Event Based

```
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) {});
```

13

Node.js – Deep Dive – Event Loop





19

Node.js – Deep Dive – NPM

- 1,000,000+ 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
```

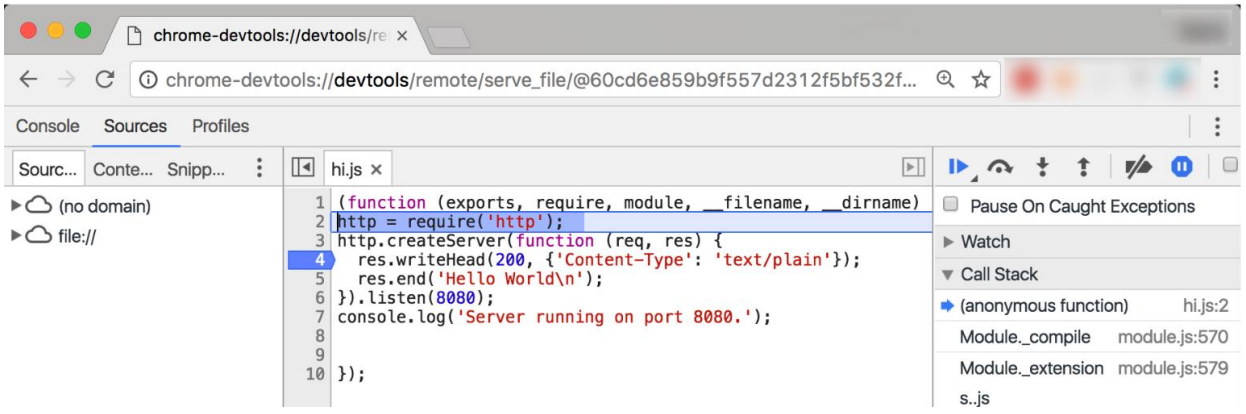
Creates file `package.json`

```
{
  "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

Debugging Node.js in a browser

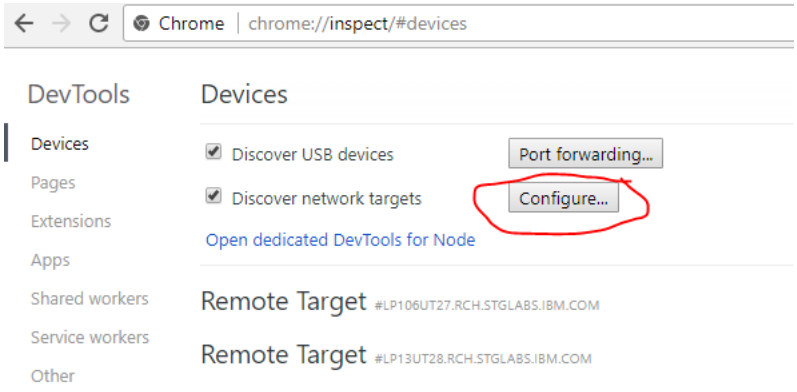


More debugging options: bit.ly/rs-debug-nodejs

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

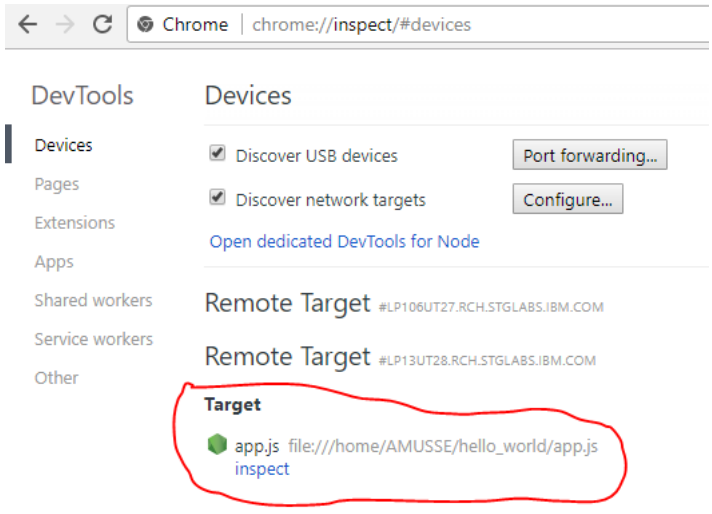
- $ node --inspect=0.0.0.0 hi.js
- Debugger listening on port 9229.

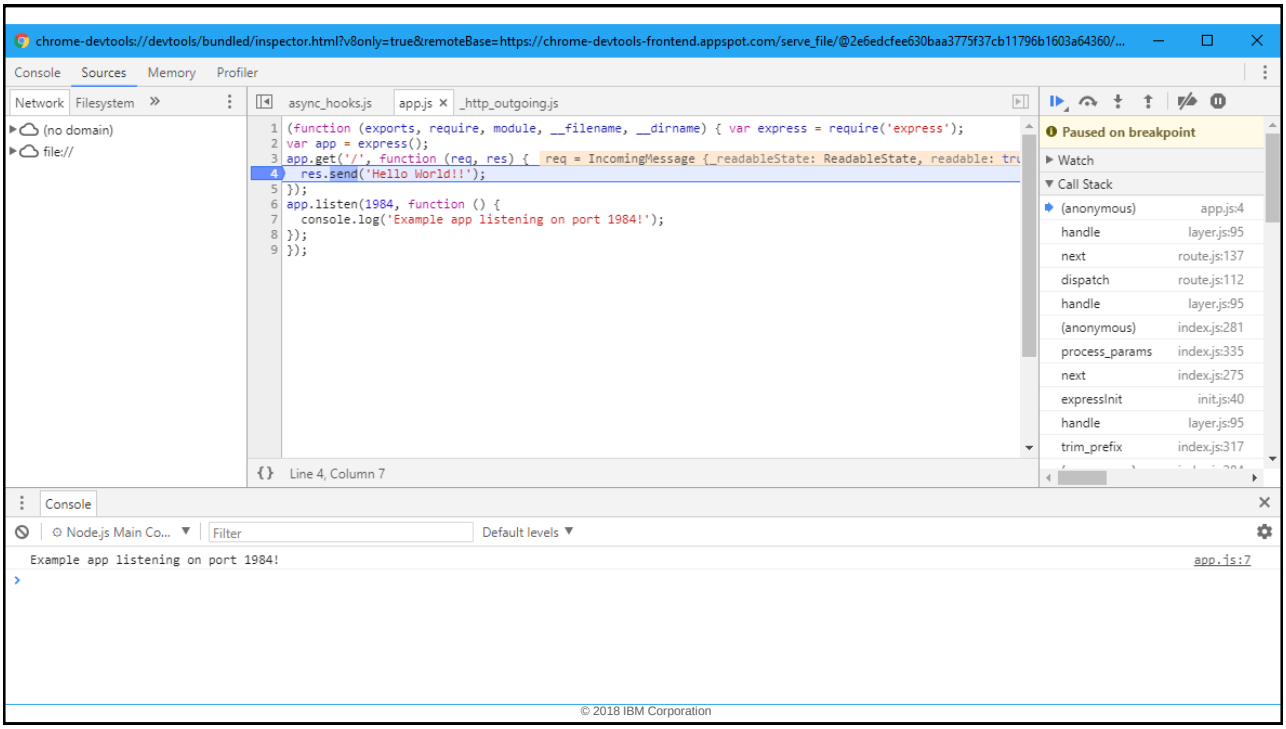
Start node with `--inspect`

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

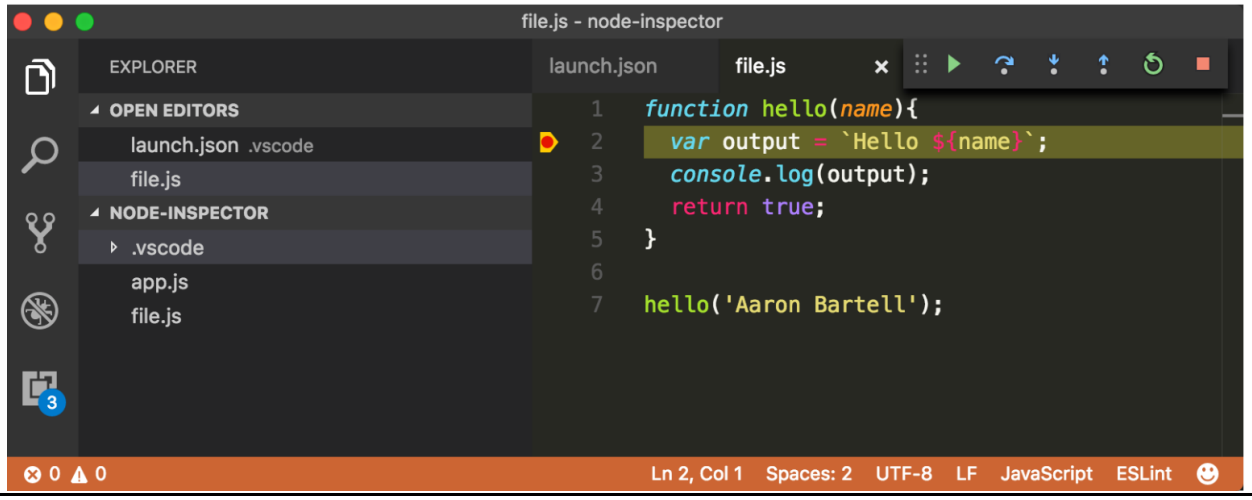
Debugging Node.js in a browser

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



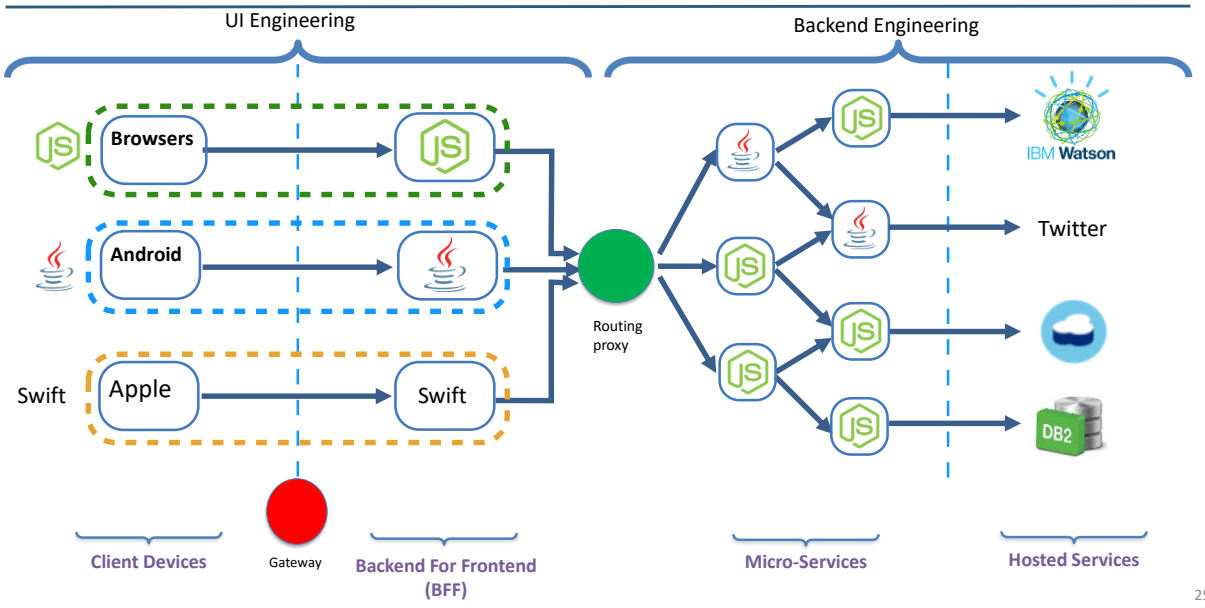


Debugging Node.js in VSCode



Node.js and Java

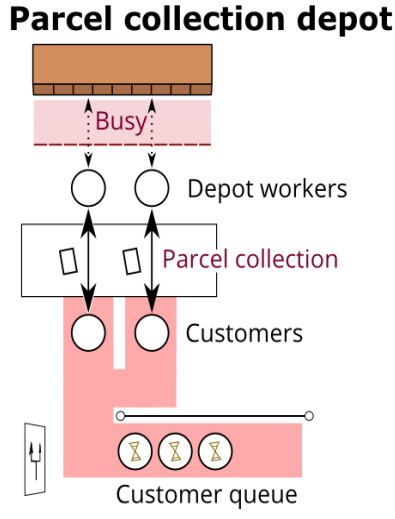
Node.js With Java



Node.js With Java – Scaling with Java

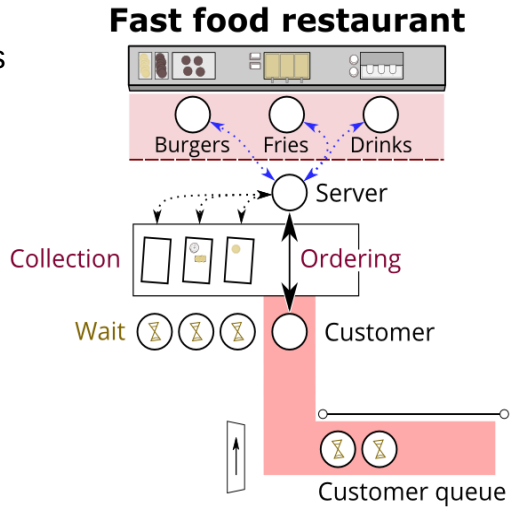
.

- 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



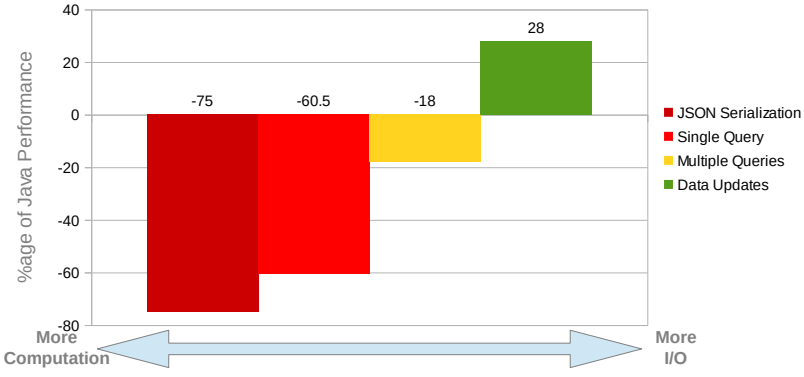
Node.js versus Java – Scaling with Node.js

- 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



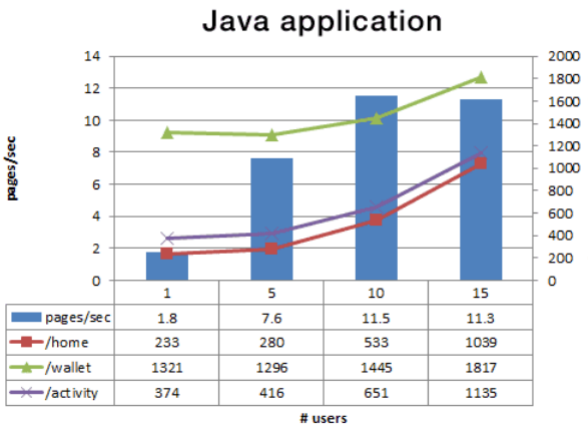
27

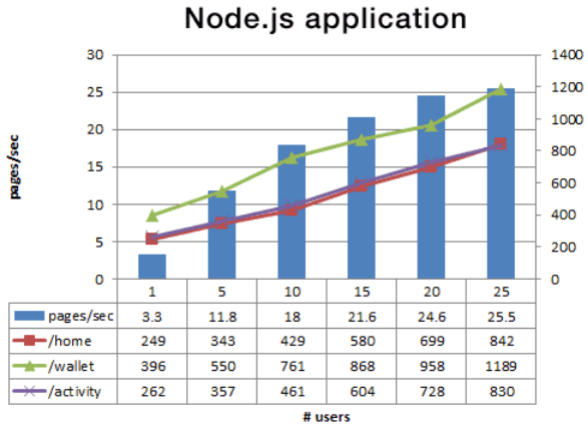
Node.js With Java– Tradeoffs



26

Why Node.js ? Performance





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

29

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%

13

Node.js With Java – Choosing the Right Language

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

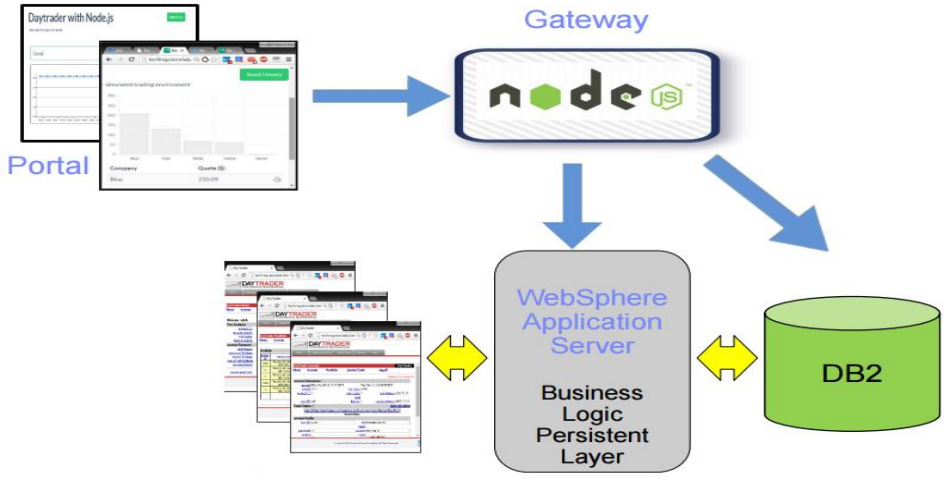
Node.js versus Java – Choosing the Right Language

- 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

Node.js With Java– Hybrid applications



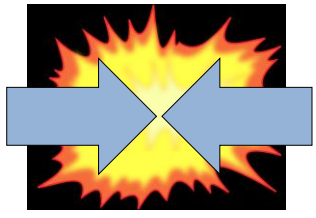
IBM involvement

The Challenge for Every Existing Enterprise:

How to make the old work with the new?

Traditional IT

- On Prem
- Packaged Apps
- SOA / Monolithic
- Relational DB
- Waterfall
- Java / .NET / C# / Other



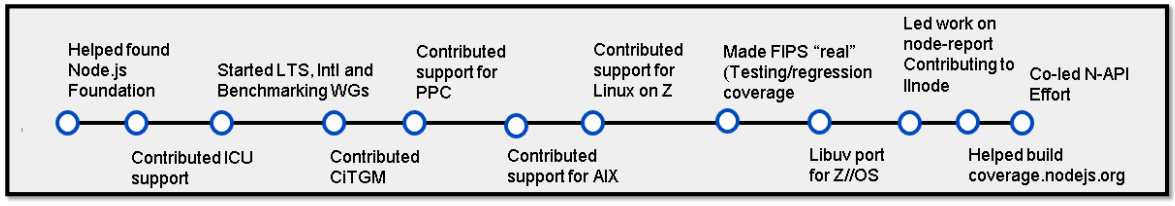
New IT

- Cloud
- SaaS
- Microservices / APIs
- Relational & Non-Relational
- DevOps
- **Node** / SWIFT / Other

IBM Node.js Strategy

- Enterprise Ready Runtime
- Production Enablement
- Production Support

Enterprise Ready Runtime

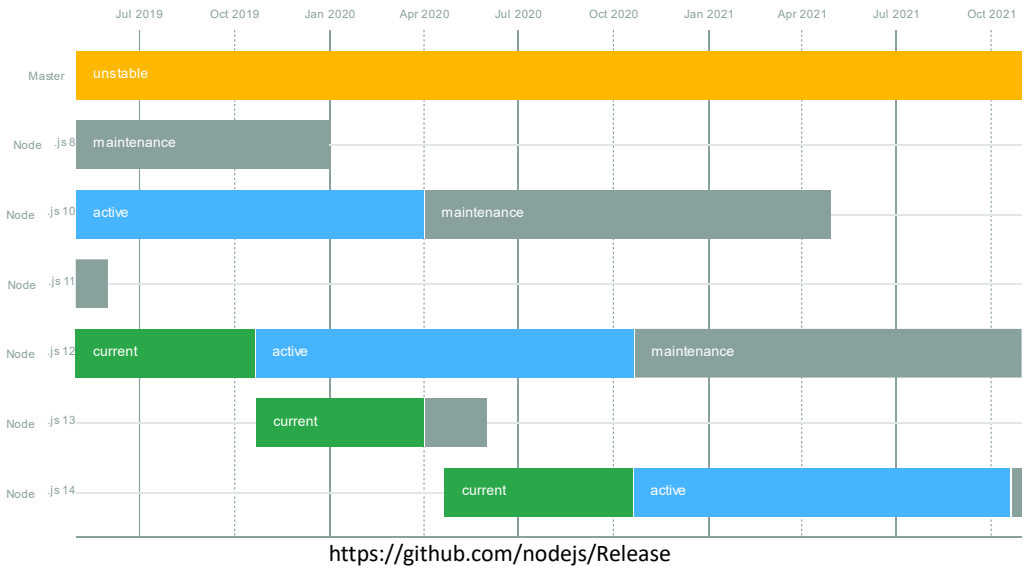


N-API

- 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



https://github.com/nodejs/Release

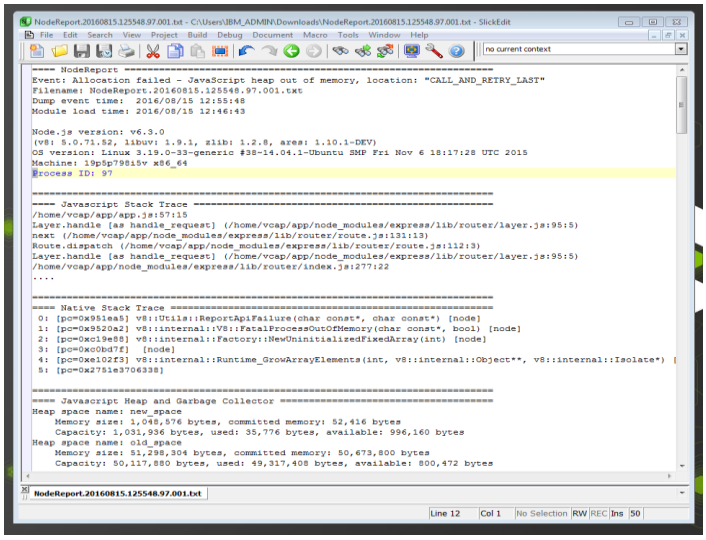
Node.js IBM – Diagnostic Report

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

Now part of Node.js core

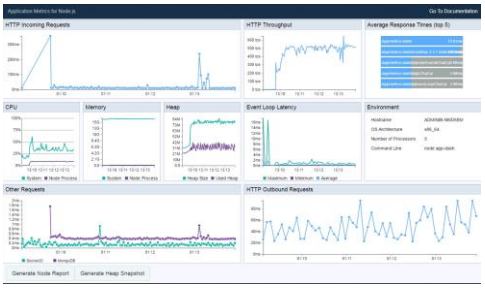


AppMetrics - open-source Node.js monitoring

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

Node.js Community Leadership @ IBM

Michael
Dawson

Gireesh
Punathil

Sam
Roberts

Bethany
Griggs

Ben
Noordhuis

Christopher
Hiller

Sam
Ruby

Richard
Lau

Steven
Loomis

Yi-Hong
Wang

Christian
Clauss

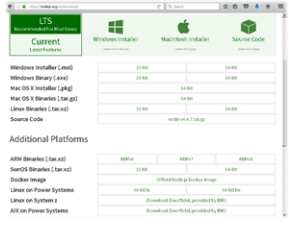
- Four TSC members & eleven total Node.js core collaborators

Node.js IBM – V8 Community Involvement

- 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

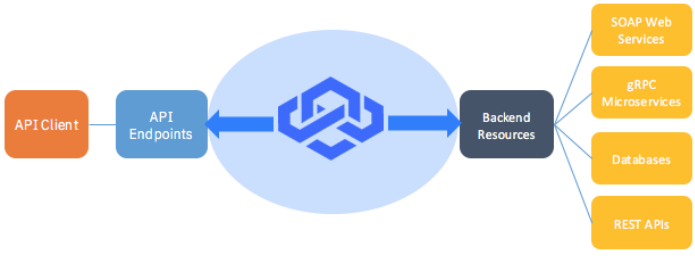
- Community Binaries
  - Linux on Z
  - Linux on P
  - AIX



- IBM Binaries
  - IBM i
  - z/OS

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)



47

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

48

IBM TSS Support for IBM i

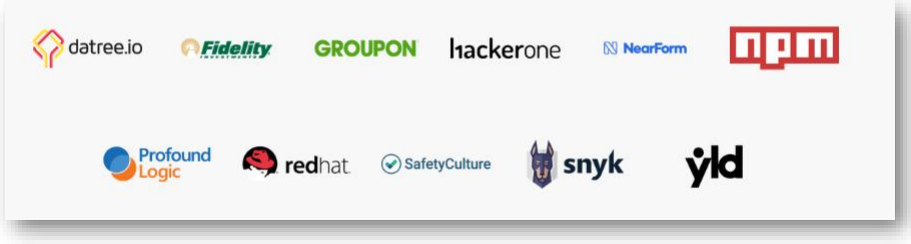
- Git
- Jenkins
- rsync
- Node.js
- Apache Tomcat
- WordPress
- Python

- For more resources, see http://ibm.biz/ibmi-oss-support

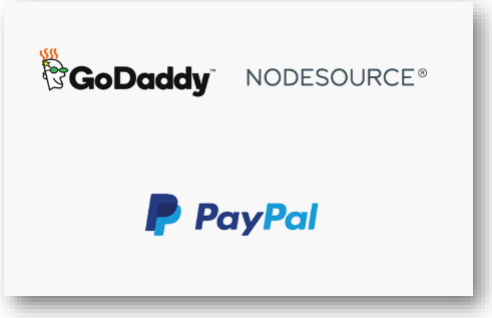
Node.js foundation

- 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



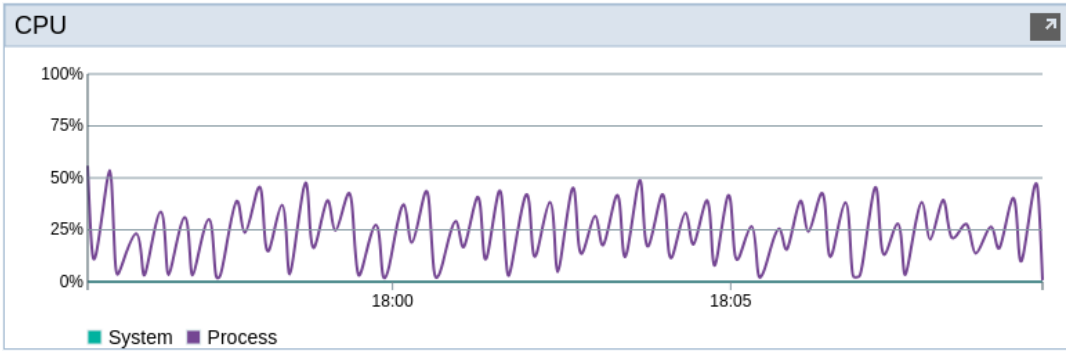
Node.js foundation – Gold Members



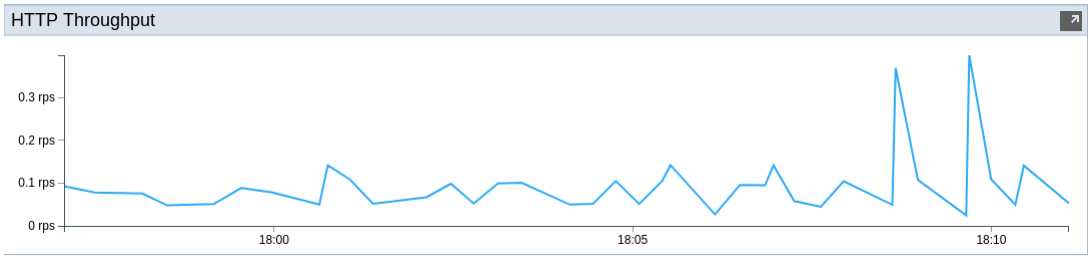
Node.js foundation – Platinum Members

AppMetrics-dash

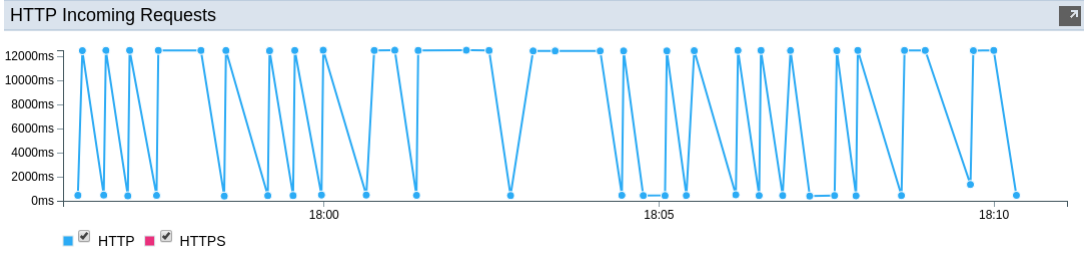
AppMetrics



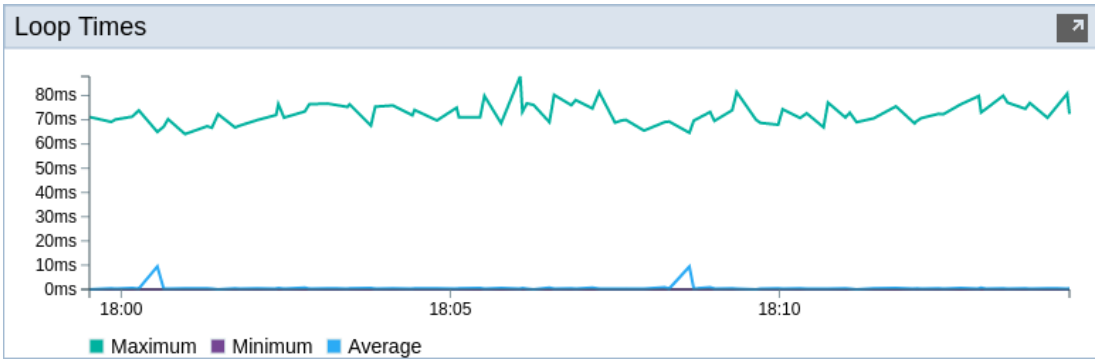
AppMetrics



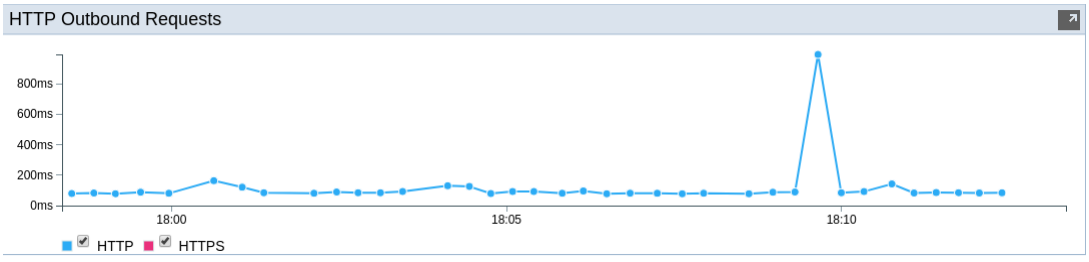
AppMetrics



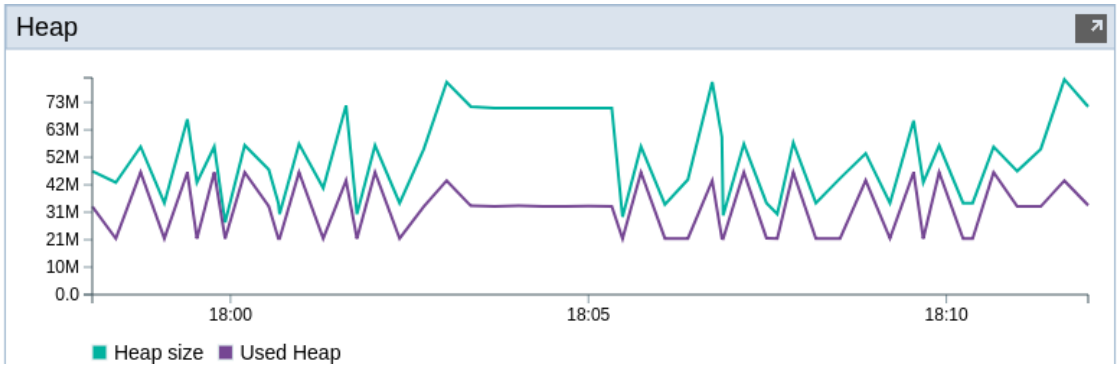
AppMetrics



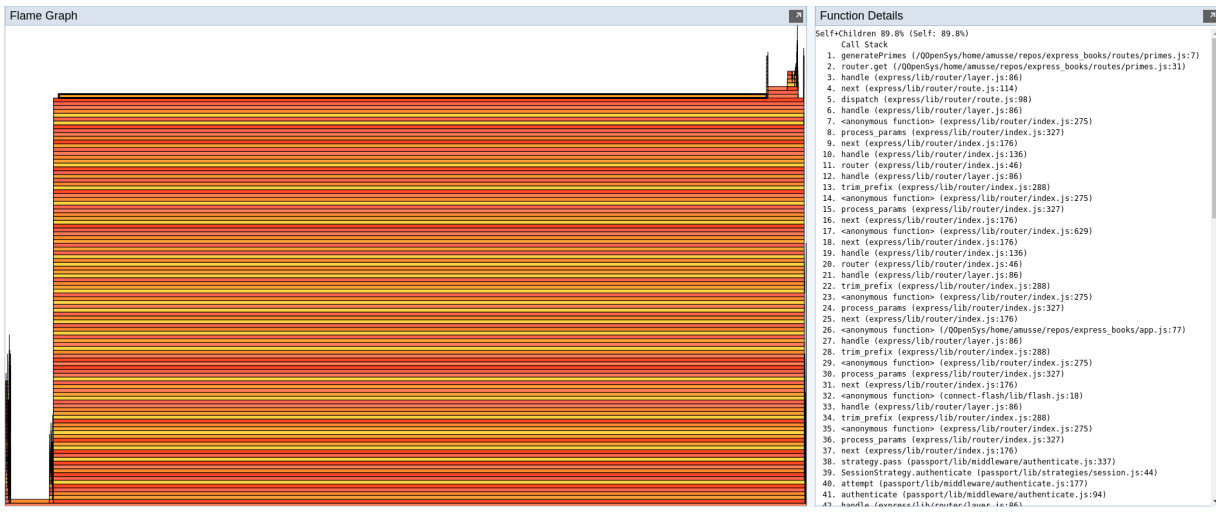
AppMetrics



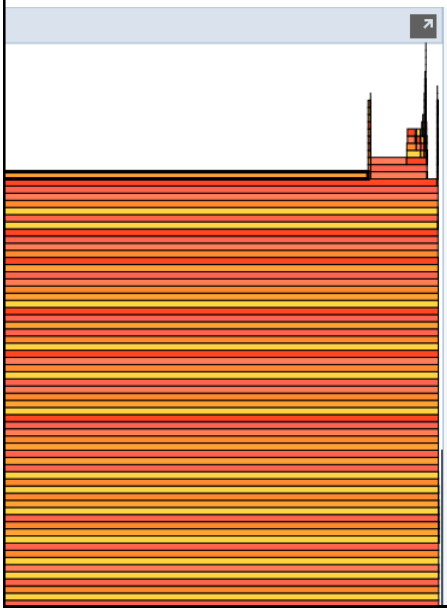
AppMetrics



AppMetrics



AppMetrics



Function Details

```
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)
```

63

IBM i Integration

64

Node.js connectivity libraries - Database

- idb-connector v1.x
  - Direct API access to Db2
  - Standard JavaScript (callback-based) conventions
- idb-pconnector v1.x
  - Built around idb-connector
  - Promises-based API set (aligns with new language standards)
  - Includes connection pooling (avoids many common pitfalls)
- odbc
  - Uses standard ODBC interfaces
  - Provides both traditional and promise-based usage
  - Includes connection pooling
  - https://github.com/IBM/ibmi-oss-examples/blob/master/odbc/odbc.md
  - https://www.ibmsystemsmag.com/Power-Systems/08/2019/ODBC-Driver-for-IBM-i

Node.js connectivity libraries - Data

- itoolkit
  - Allows Node.js to integrate with RPG, Db2, CL, COBOL, etc.
  - **NEW in v1.x** Can use SSH or ODBC to talk to the system
- Sequelize Dialect
  - Db2 via the popular Node.js ORM
- Loopback Connector
  - Connector for connecting to Db2 on IBM i when using LoopBack
  - LoopBack is a framework for creating REST APIs for your data
  - Your data is defined in terms of models, which are objects that mirror database tables

odbc Example

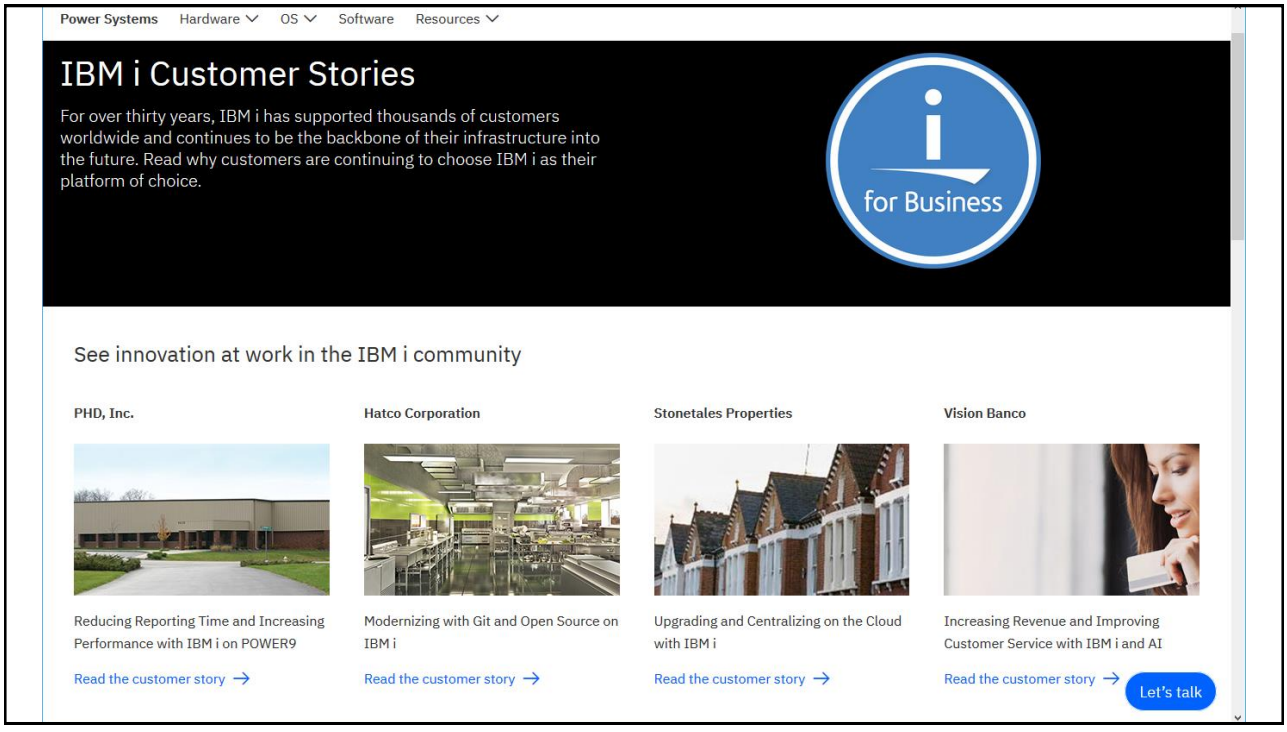
```
$ node app.js
[ { STATE: 'TX' },
  { STATE: 'NY' },
  { STATE: 'VT' },
  { STATE: 'GA' },
  { STATE: 'NY' },
  { STATE: 'CO' },
  { STATE: 'MN' },
  { STATE: 'CA' },
  { STATE: 'WY' },
  { STATE: 'TX' },
  { STATE: 'NY' },
  { STATE: 'MN' },
  statement: 'SELECT STATE FROM QIWS.QCUSTCDT',
  parameters: [],
  return: undefined,
  count: -1,
  columns: [ { name: 'STATE', dataType: 1 } ] ]
$
```

```
$ npm init -y
$ npm install odbc
```

app.js

```
const odbc = require('odbc');
const sql = 'SELECT STATE FROM QIWS.QCUSTCDT'

const connection =
odbc.connect('DSN=IBMISYSTEM');

connection.query(sql, (error1, result) => {
  console.log(result);
  connection.close((error2) => {
    console.log('connection closed');
  });
});
```


Customer Stories

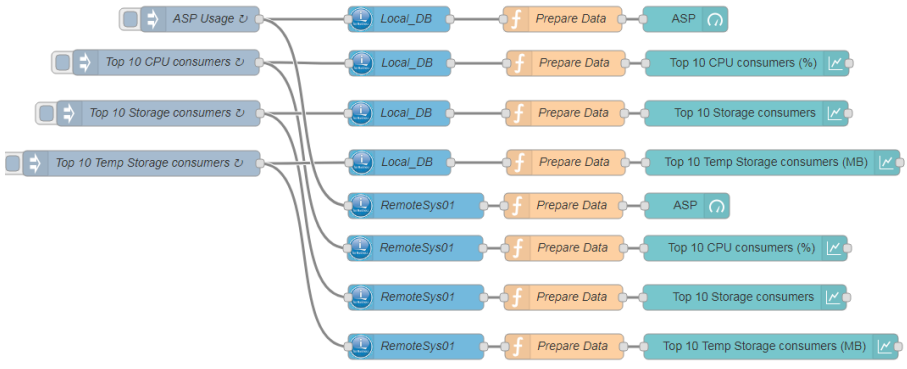


IBM i Microsite Customer Stories using Node.js

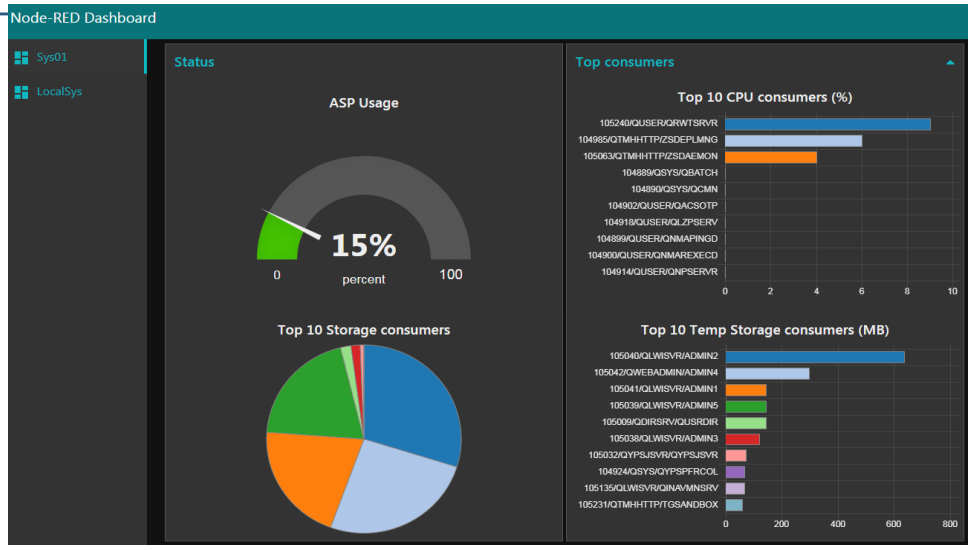
- http://ibm.biz/ibmistories
  - FormaServ
  - Krengeltech
  - Mutual Distributing Company
  - Geodis
  - RPC Superfos
  - HT Bendix A/S
  - Kuehne + Nagel S.à.r.l.

CRAS

Node-RED example



Node-RED example



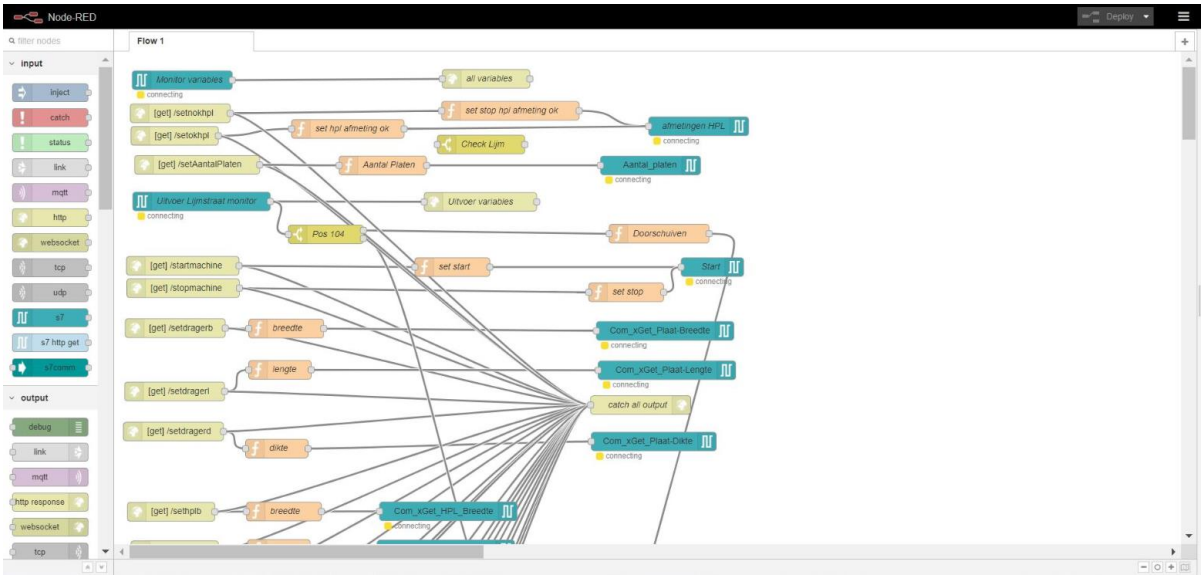
CRAS Woodgroup

- **Who**
  - Cras creates wood products to order, adjusting humidity, temperature and other factors in its plants to meet precise customer requirements.
- **The scenario**
  - Evaluating implementations for transformation effort
  - Trouble linking all the PLCs (problem logic controllers)
- **The Solution**
  - With Node-RED, our BP was able to link all the PLCs directly and easily
- **The Result**
  - IBM i undoubtedly the right place for production

https://www.ibm.com/case-studies/cras-systems-open-source

CRAS WOODSHOPS

Node-RED in production



Stonetales

Stonetales

- **Who**
  - a real estate company involved in housing marketing, cohousing and sales of properties.
- **The scenario**
  - acquired a competitor that was running on Microsoft Windows and x86.
- **The Solution**
  - Node.js on IBM i (new software development providing solutions to improve customer service and streamline processes).
- **The result**
  - Quickly decided to discard the x86 workloads
  - Consolidated to IBM i on POWER hardware.

https://cms.ibm.com/case-studies/stonetales-properties-power-upgrade

A “Happy” Ending

A “Hapi” Ending

Walmart creates a framework!

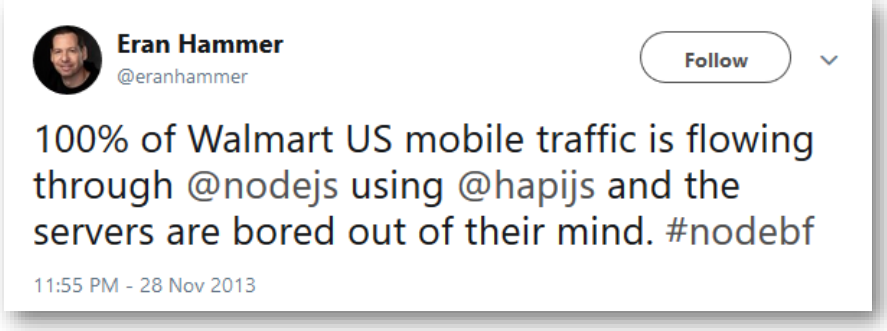
- Express.js appeared in 2009.
- Walmart saw Express.js insufficient for very large projects, but saw the huge potential in Node.js.
- Willing to invest millions of dollars in a new framework.

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

Black Friday 2013

- Full deployment for all mobile shopping!!

- The hardware?
  - 10 CPU cores
  - 28 GB memory



Q&A

The END!

Special notices

Special notices (cont.)