

---

# Node.js: What, Why and How?

Michael Dawson  
IBM Community Lead for Node.js

POWERUp18 Session ID: 180232

Agenda Key: 32CF



# About Michael Dawson

## IBM Community Lead for Node.js



- Active Node.js community member
    - Collaborator
    - Node.js Technical Steering Committee TSC Chair
    - Community Committee member
    - Working group(s) member/leadership



- Twitter: @mhdawson1
  - GitHub: @mhdawson
  - Linkedin: <https://www.linkedin.com/in/michael-dawson-6051282>

# Agenda

---



- Why Node.js ?
- Node.js deep dive
- Positioning versus Java™
- Node.js community
- IBM involvement



# Why Node.js ?

---



- What is it ?
- Ecosystem
- Productivity
- Performance



# Why Node.js – What is it?

---



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

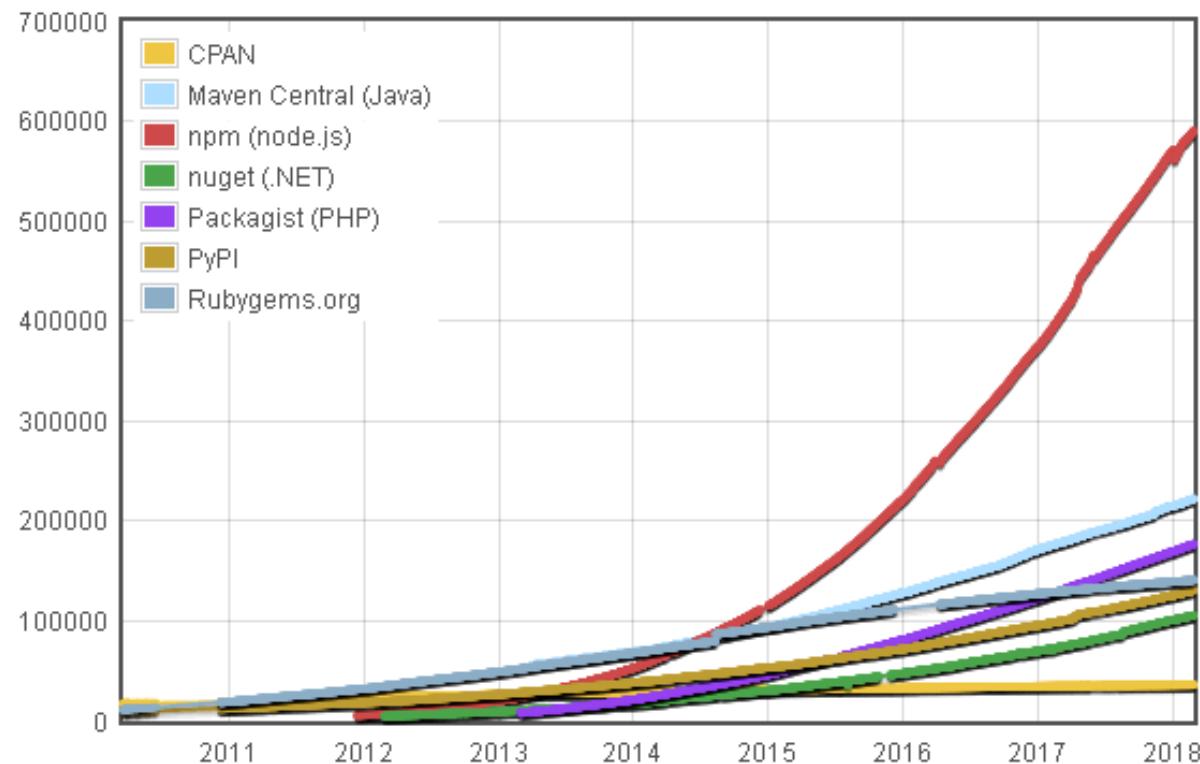


# Why Node.js ? – Ecosystem



- There is a module for that
  - 600K modules
  - #1 on module counts
- #1 on Github (#projects)

## Module Counts



<http://www.modulecounts.com/>

# 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

# 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
- **NextFlix** - <http://www.infoworld.com/article/2610110/javascript/paypal-and-netflix-cozy-up-to-node-js.html>

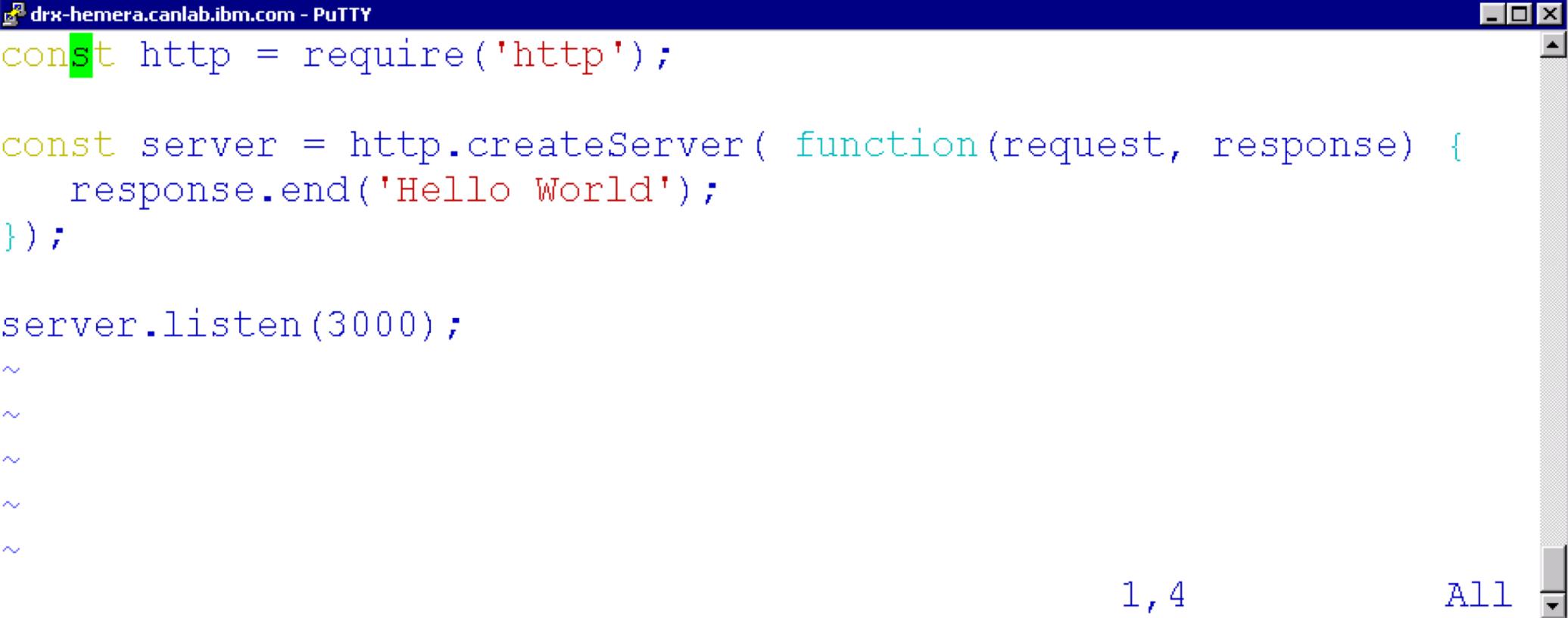
# Why Node.js ? – Productivity

---



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

# Why Node.js ? – Productivity



A screenshot of a PuTTY terminal window titled "drx-hemera.canlab.ibm.com - PuTTY". The window contains the following Node.js code:

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

const server = http.createServer( function(request, response) {
    response.end('Hello World');
});

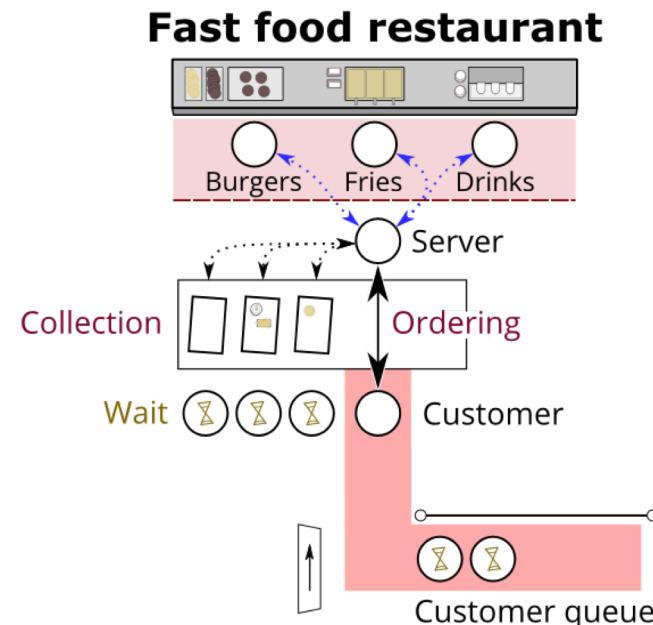
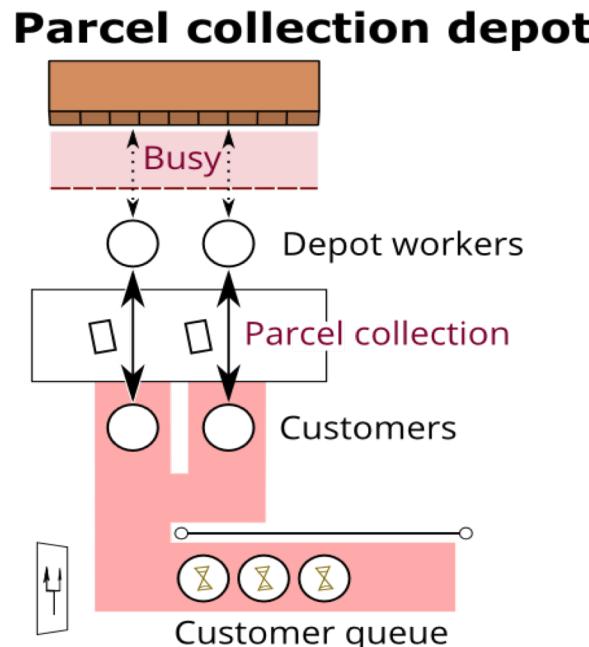
server.listen(3000);
~
```

The word "const" in the first line is highlighted with a green rectangular box. At the bottom right of the terminal window, there are two status indicators: "1, 4" above "All" and a small downward-pointing arrow icon.

# Why Node.js ? - Performance



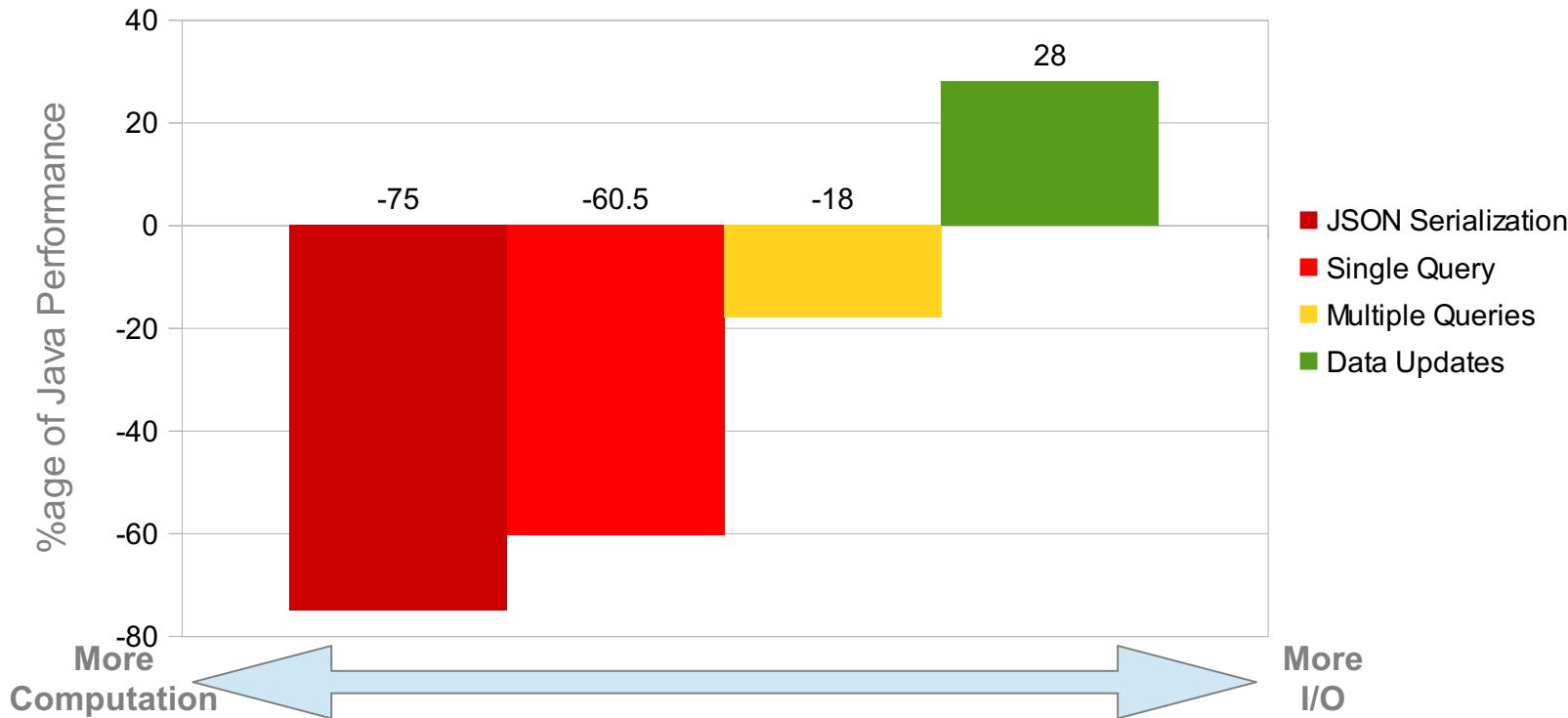
**Event based: perfect fit for asynchronous non-blocking I/O**



# Why Node.js ? - Performance



**Best suited for asynchronous workloads**



# 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 – Deep Dive

---



- Key characteristics
- Components
- Programming model
- Event loop
- Native code
- Common use cases



# Node.js – Deep Dive – Key Characteristics

---

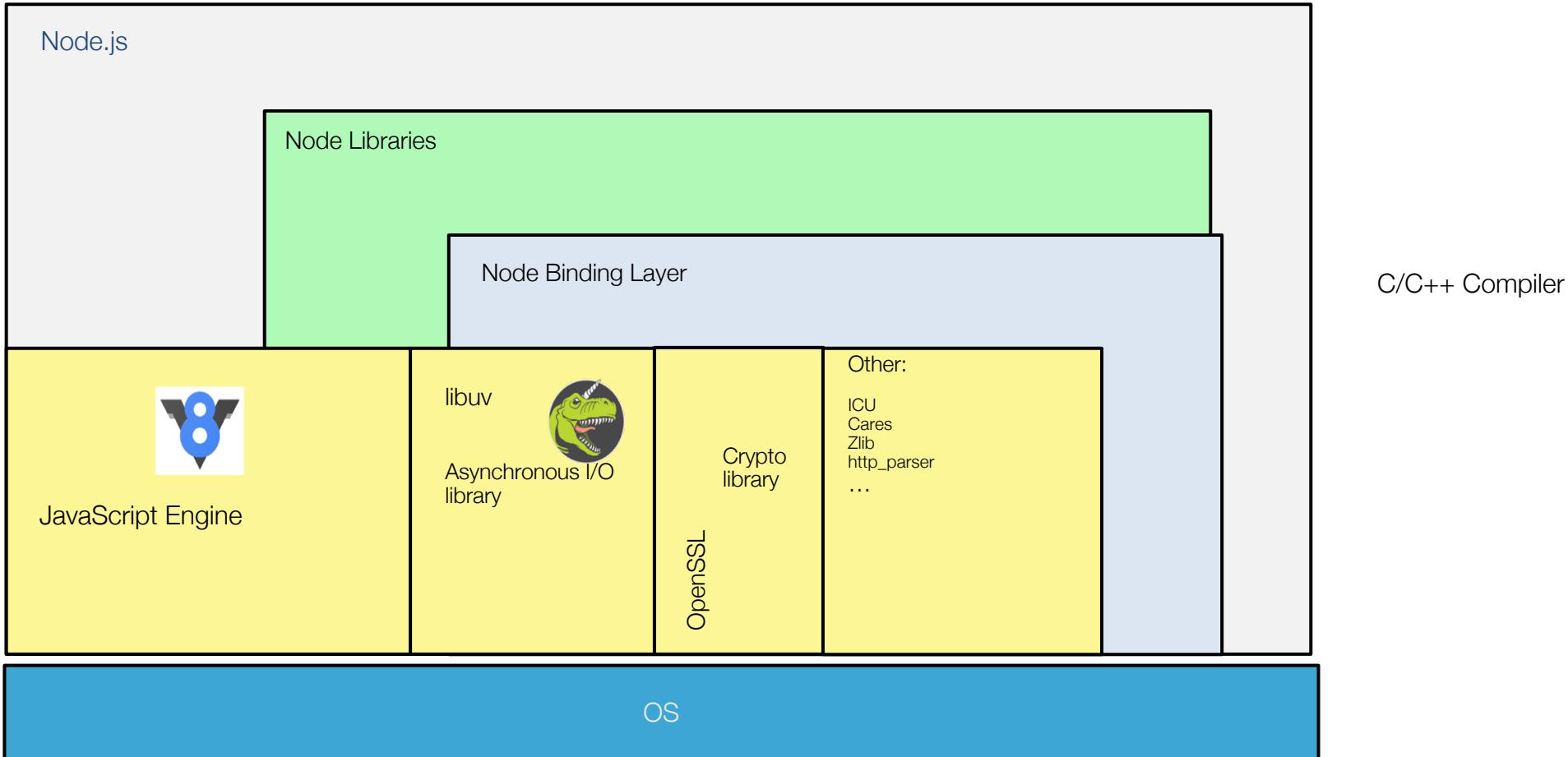


- Small (`linux.tar.xz`)
  - Download **10.8 MB**
  - Uncompressed **69 MB**

<https://nodejs.org/en/download/>
- Fast startup
  - 60 ms

<https://benchmarking.nodejs.org/>
- Small footprint
  - 18 MB

# Node.js – Deep Dive - Components



# Node.js – Deep Dive - Programming Model



- Dynamic
- Functional
- Asynchronous
- Event Based

```
drx-hemera.canlab.ibm.com - PuTTY
-sh-4.2$ cat sample.js

var data = 50;

var myNiftyFunction = function(param, callback) {
  setImmediate(callback.bind(null, param));
}

myNiftyFunction(1000, function(result) {
  console.log('In function:' + (result + data));
});

data = data + 1000000;
console.log('Mainline:' + data);
-sh-4.2$
-sh-4.2$ ./node sample.js
Mainline:1000050
In function:1001050
-sh-4.2$
```

# Node.js – Deep Dive - Programming Model



- Event Based

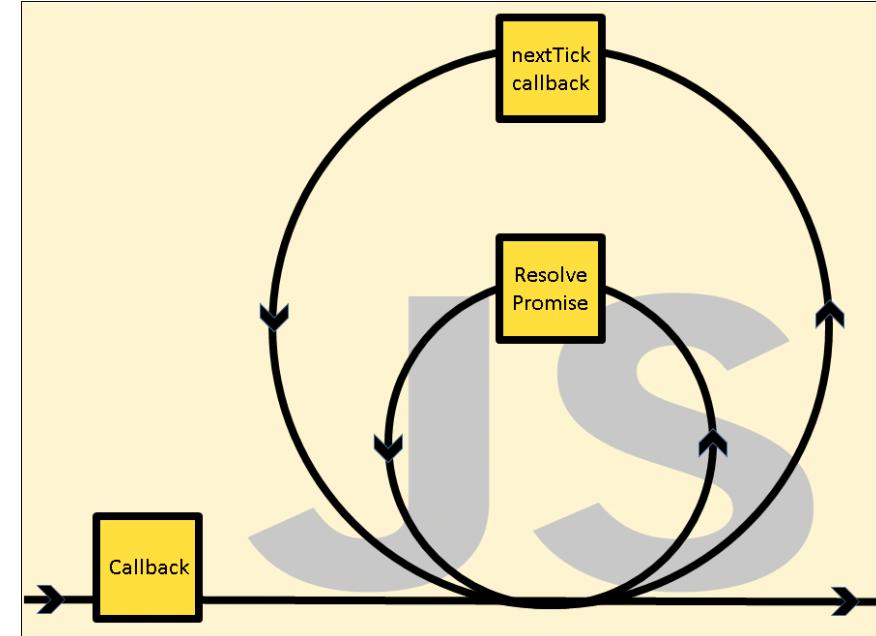
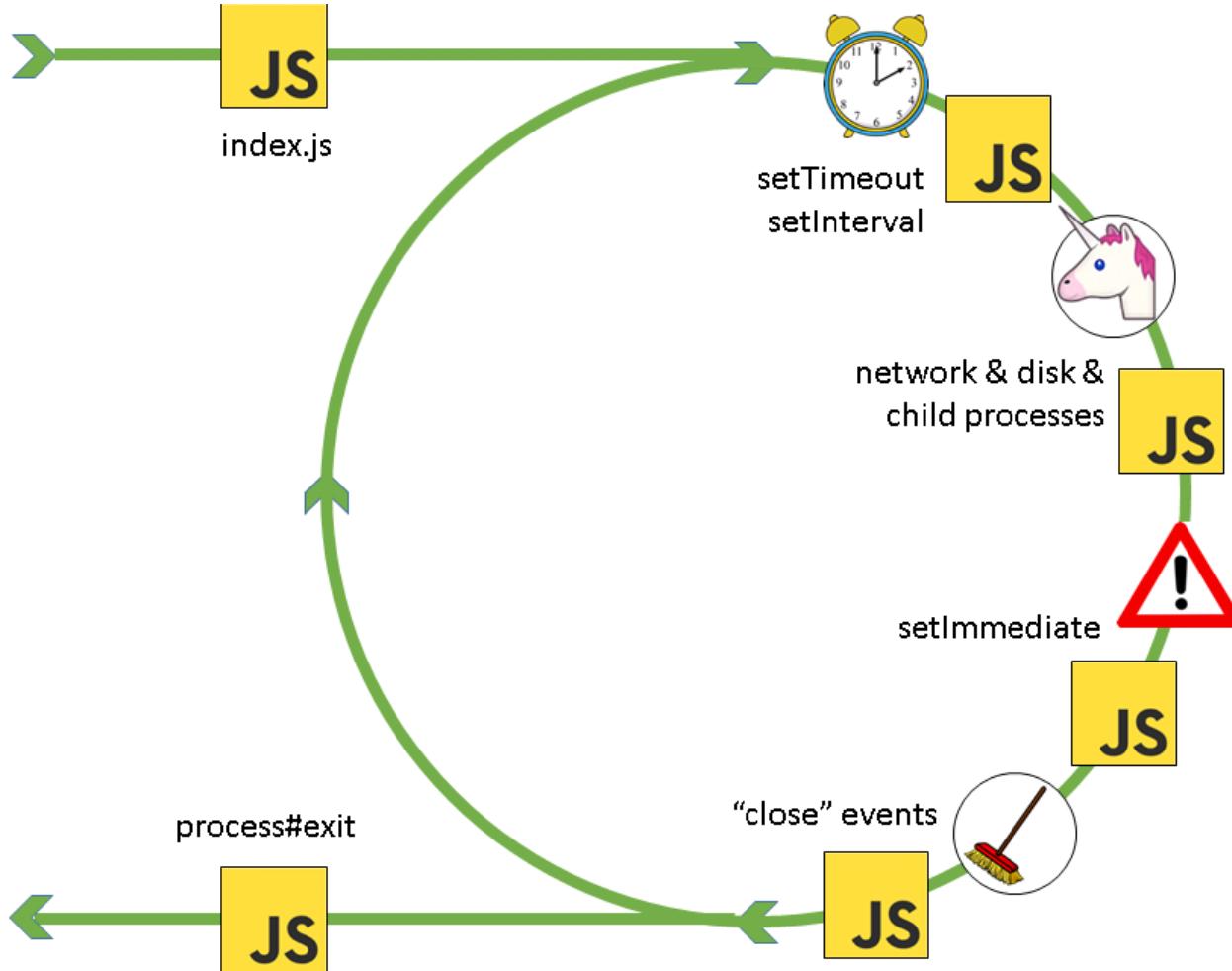
```
var http = require('http');

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 – Native Code

---



```
#include <node.h>

void nativeMethod(const FunctionCallbackInfo<Value> & args) {
  Isolate* is = args.GetIsolate();
  args.GetReturnValue().Set(String::NewFromUtf8(is, "Hi from native"));
}

void init(Local<Object> exports) {
  NODE_SET_METHOD(exports, "callNative", nativeMethod);
}

NODE_MODULE(nativeModule, init);
```

<https://nodejs.org/api/addons.html>

# Node.js – Deep Dive – Native Code

---



```
const nativeModule = require('./build/Release/nativeModule');

console.log(nativeModule.callNative());
```

<https://nodejs.org/api/addons.html>

# Node.js – Deep Dive – Use Cases

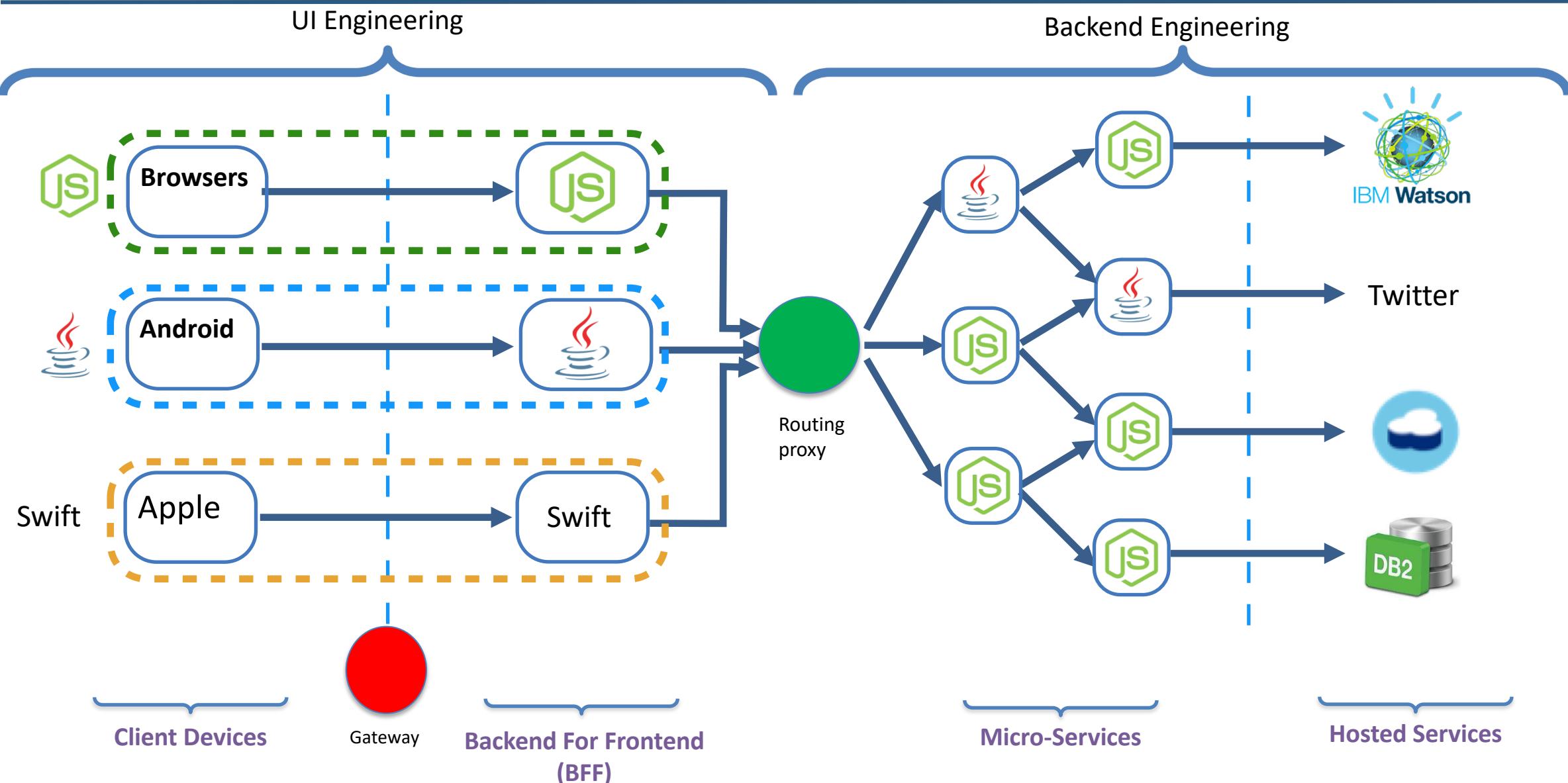
---



[https://github.com/nodejs/benchmarking/blob/master/docs/use\\_cases.md](https://github.com/nodejs/benchmarking/blob/master/docs/use_cases.md)

- Back-end API services
- Service oriented architectures (SOA)
- Microservice-based applications
- Generating/serving dynamic web page content
- SPA applications with bidirectional communication over WebSockets and/or HTTP/2
- Agents and data collectors
- Small scripts

# Node.js With Java



- Strengths and weaknesses
- Choosing the right language
- Hybrid applications

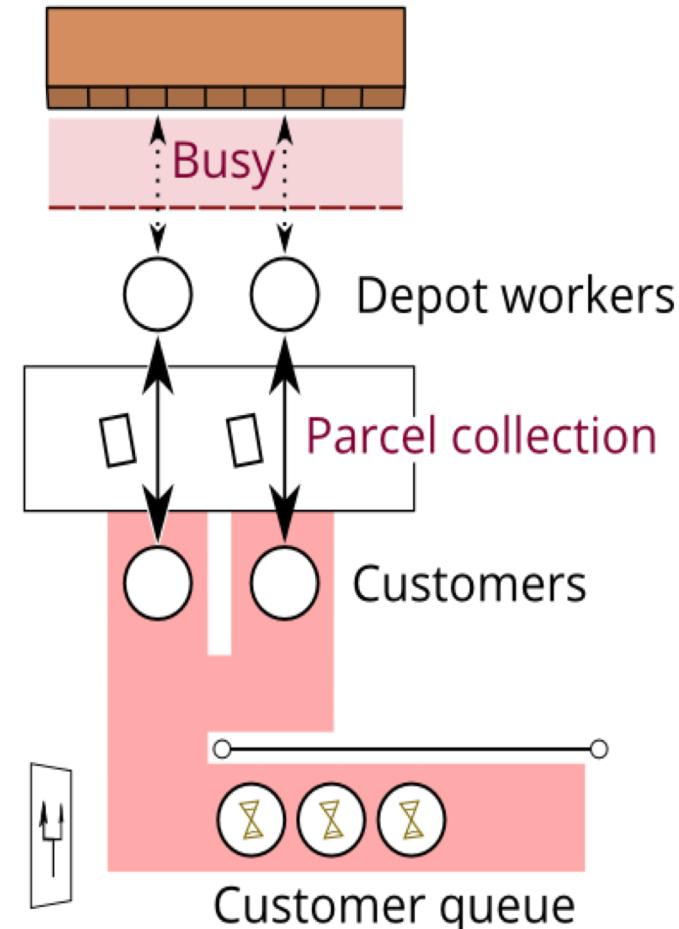


# Node.js With Java – Scaling with Java



- 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

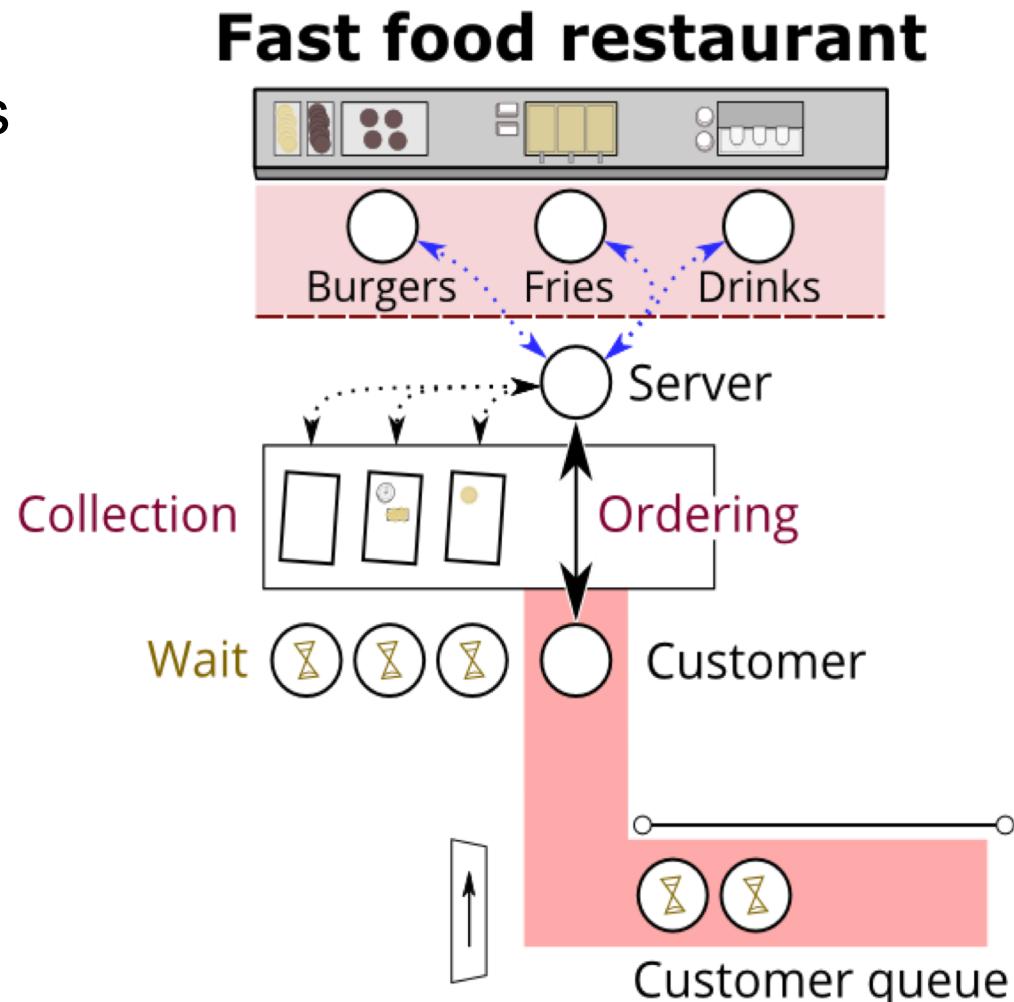
## Parcel collection depot



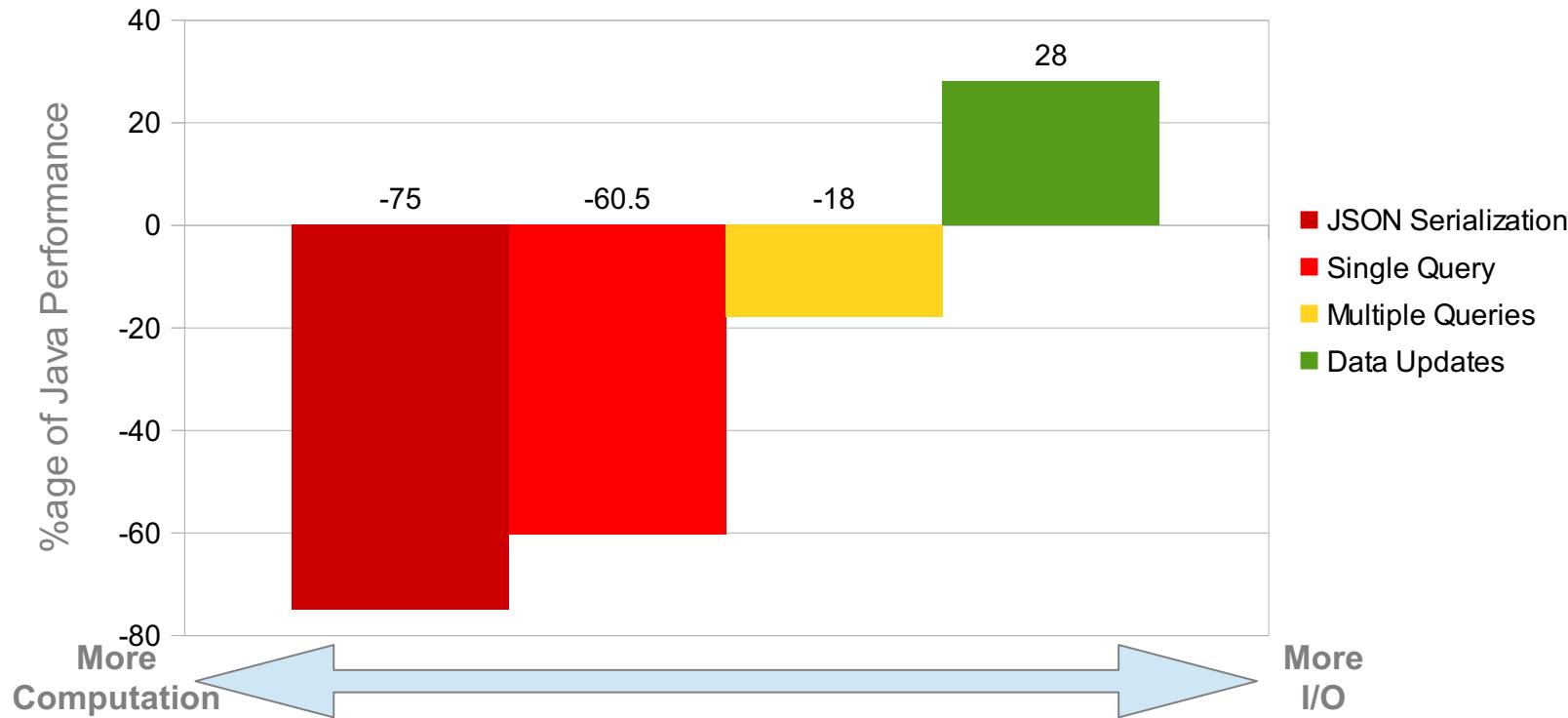
# Node.js versus Java – Scaling with Node.js



- 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



# 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

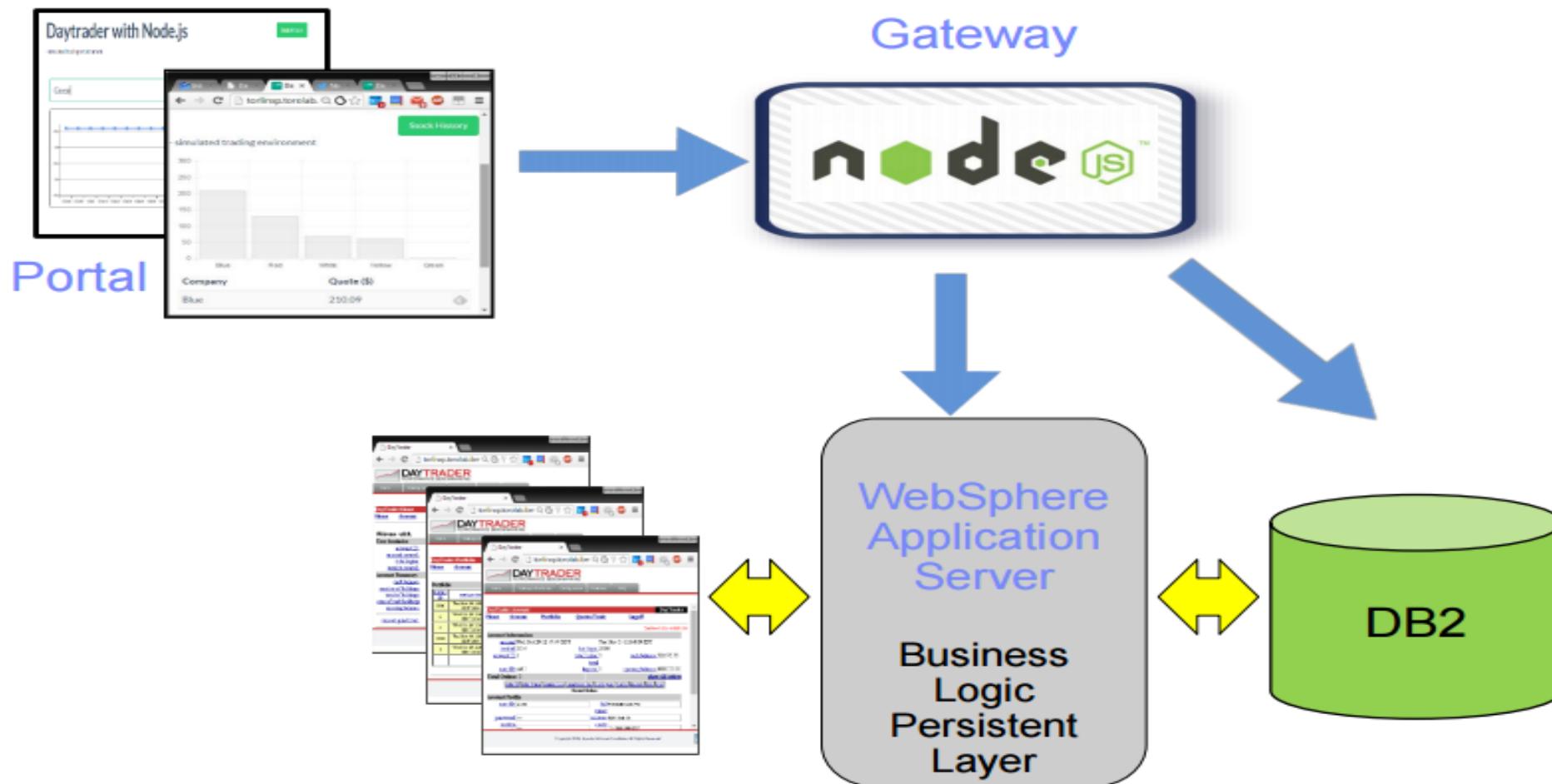
# Node.js With Java— Choosing the Right Language

---



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

# Node.js With Java – Hybrid applications



# Node.js Community

---



- History
- Foundation
- Day to Day



# Node.js Community - History

---



- **2009 – written by Ryan Dahl**
- **Jan 2010 - npm**
- **Sep 2010 – Joyent sponsors Node.js**
- **June 2011 – Windows support**
- **2012 – 2014 – Hand over to Isaac Schlueter, then Timothy J. Fontaine**
- **December 2014 – io.js fork**
- **June 2015 – Node.js Foundation**
- **Oct 2015 – Node.js 4.x unites io.js/node.js 0.12.x lines**
- **Oct 2016 – Node.js 6.x**
- **Oct 2017 – Node.js 8.X**
- **Oct 2018 ....**



# Node.js Community - Foundation

---



- **Mission:**

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.

<https://nodejs.org/en/foundation/>

- **Corporate members**

- 5 platinum(including IBM), 3 gold, 19 Silver

- Individual members

# Node.js Community – Day to Day

---



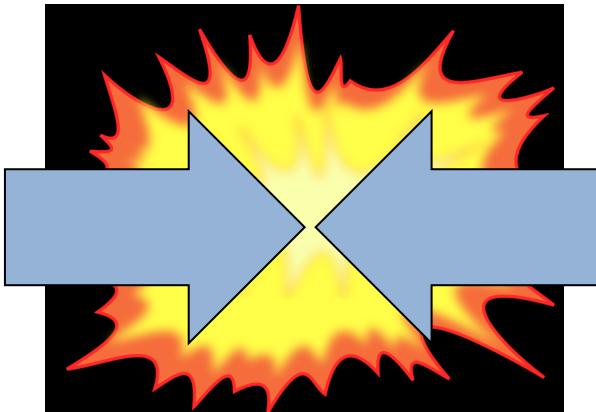
- TSC - Technical Steering Committee <https://github.com/nodejs/TSC/>
- Collaborators (100+) <https://github.com/nodejs/node/>
- Working Groups (Build, LTS, Benchmarking, API etc.)  
[https://github.com/nodejs/node/blob/master/WORKING\\_GROUPS.md](https://github.com/nodejs/node/blob/master/WORKING_GROUPS.md)
- Teams <https://github.com/orgs/nodejs/teams>
- Community Committee
  - Teams (ex user-feedback)
  - Working Groups (to come)

# 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

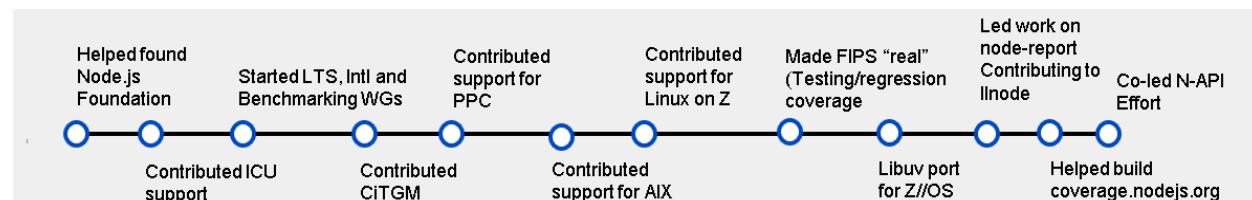
# Enterprise Ready Runtime

- Embrace and Improve Community Runtime

- Engage and lead
- Develop expertise and influence
- Platinum member of Node.js Foundation

- Enterprise Focus

- Stable and Predictable releases
- Platform support
- Security
- Diagnostics
- Performance
- Code quality and safety net
- Key Features



# Stable and Predictable Releases

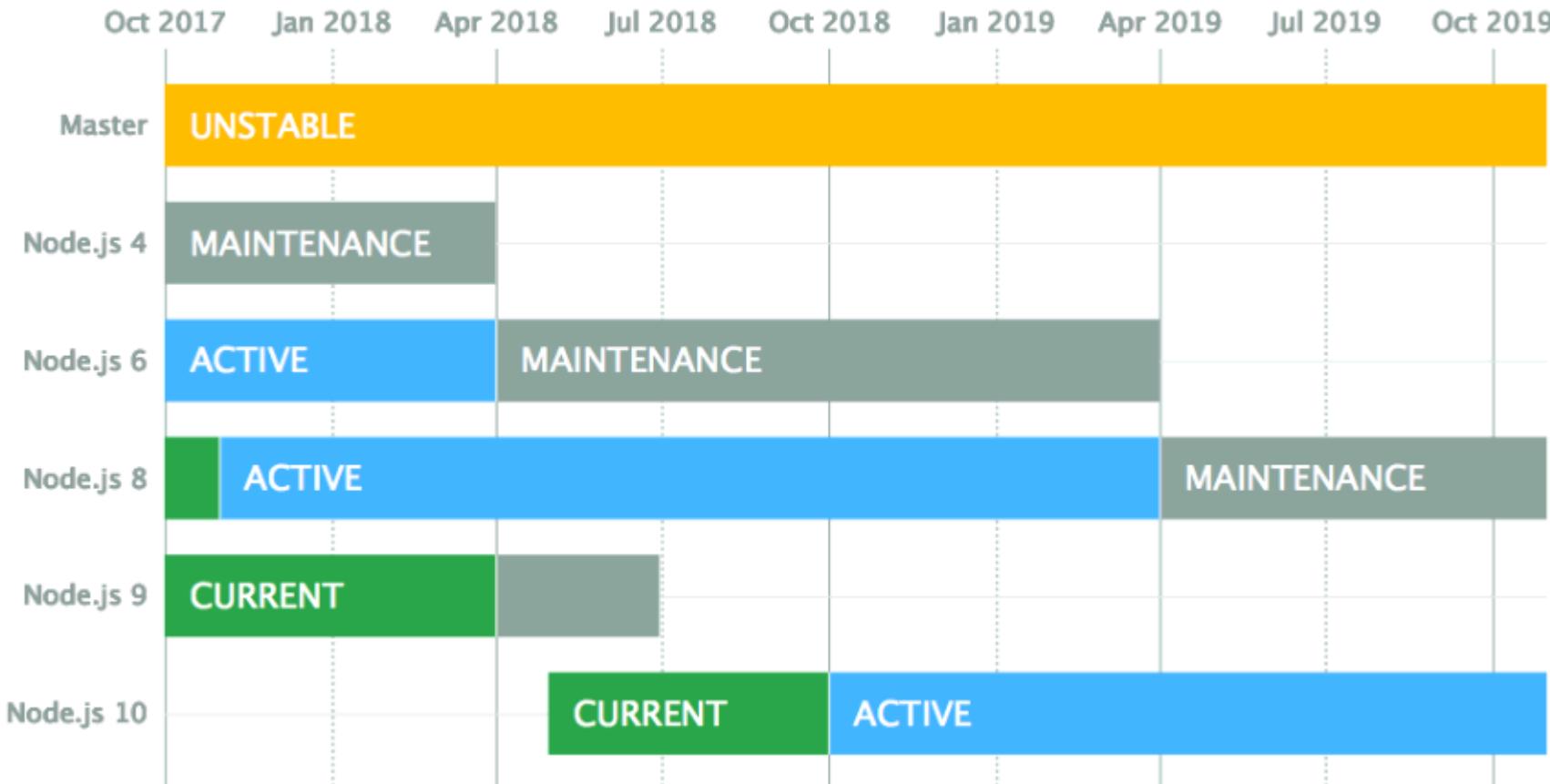
---

- Bleeding Edge
  - Canary
  - Nightlies
- Current
  - Every 6 months
  - Even releases promoted to LTS
- LTS
  - Every 12 months
  - 30 Months support (18 active, 12 maintenance)

# Stable and Predictable Releases - Schedule for 2018



<https://github.com/nodejs/Release>



# Node.js IBM – Tooling - NodeReport

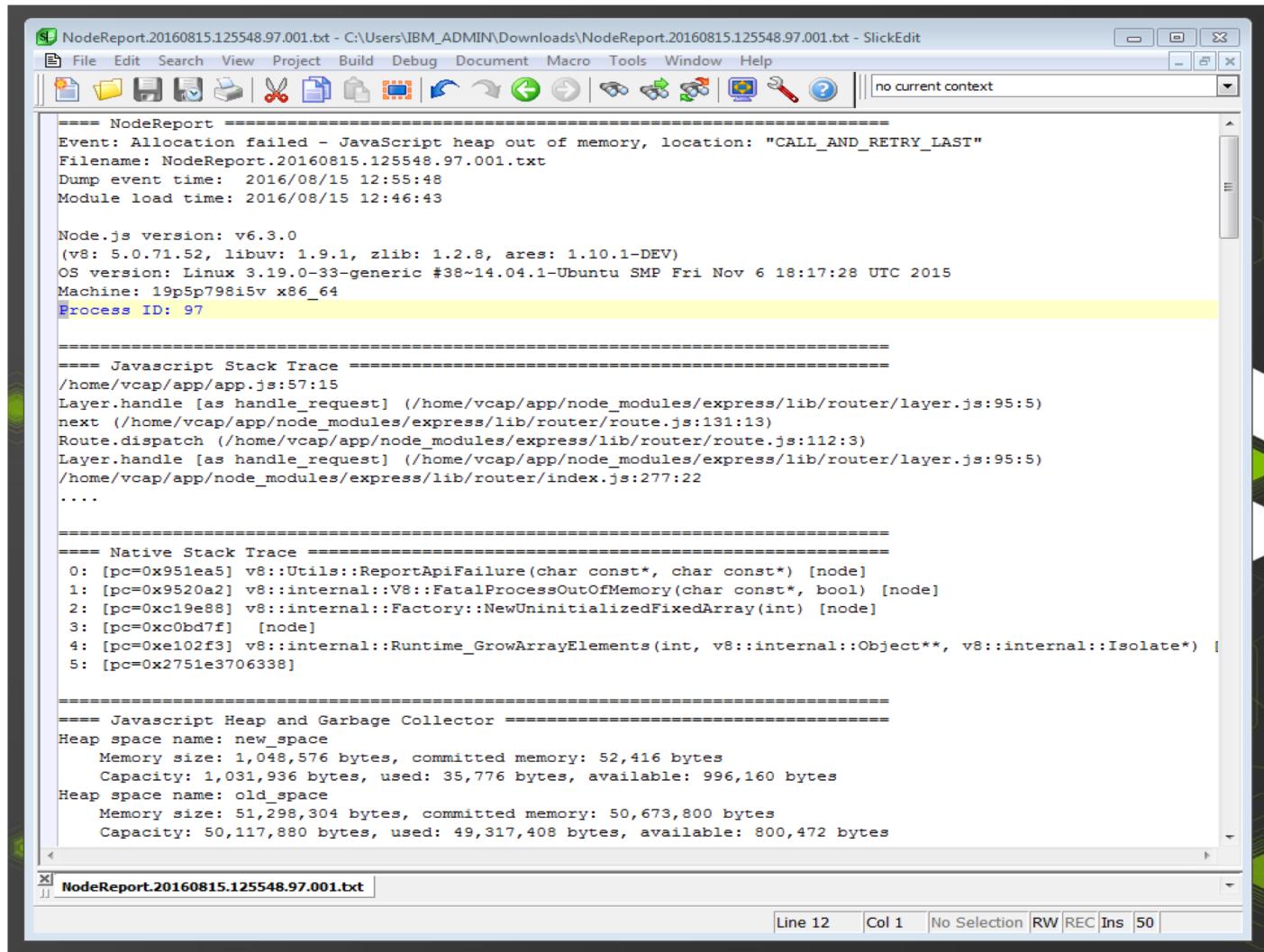


## NodeReport example - heap out of memory error

### 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 ====
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*) [node]
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


```

## 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

---



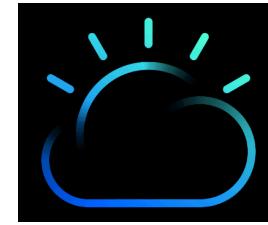
- Deep expertise at V8
- Developed ports to IBM Platforms
- Contribution back to official V8 repositories:

<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 implementation
- ~10-15 commits per week to V8 to maintain PPC/zlinux port
- Internal port for z/OS and IBM i

# Production Enablement

- First Class Cloud Deployment Options



- Freedom of Platform Choice



- Leverage existing Data assets



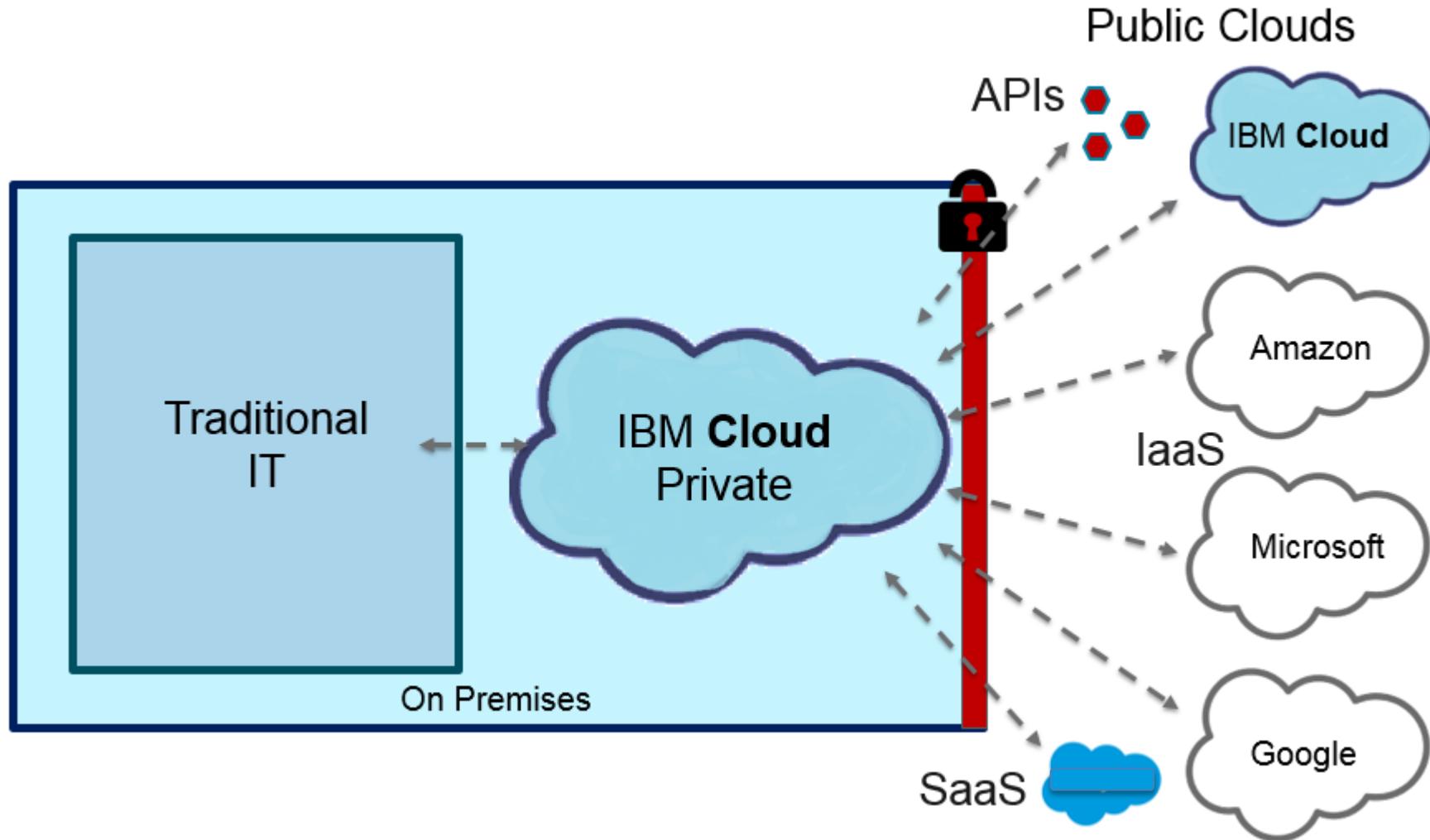
- Tools to Accelerate Development/Deployment

# First Class Cloud Deployment Options

- Public/Private/Hybrid
- Standards Based
  - Docker
  - Kubernetes
- First class Node.js support



# IBM Cloud → The Cloud for Enterprise



- One Cloud Architecture
- Multi-cloud management
- Hybrid workloads
- Hybrid integration
- Multi-cloud DevOps
- Cloud native applications
- Containerized middleware
- Modernized applications
- Data and analytics
- Cognitive solutions
- Blockchain solutions

# Freedom of Platform Choice

- Community Binaries

- Linux on Z
- Linux on P
- AIX



The screenshot shows the Node.js download page at <https://nodejs.org/en/download/>. It features a navigation bar with 'LTS' (Recommended For Most Users) and 'Current' (Latest Features) tabs. Below are sections for Windows, Mac OS X, and Linux installers, and a 'Source Code' section. A large table lists 32-bit and 64-bit versions for each platform. At the bottom, there's a section for 'Additional Platforms' including ARM binaries, SunOS binaries, Docker images, and specific IBM platforms.

Platform	32-bit	64-bit
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	

Platform	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



# Leveraging Existing Data Assets

---

- 68% of the world's production workloads and associated data is hosted in z/OS environments
- Enable Collocation with Data hosted on z/OS
  - Up to 2.5x better throughput,
  - 60% faster response time to DB2 on z/OS\*

Enhance Node.js ecosystem to access z/OS middleware and assets  
CICS, Db2, VSAM, etc.

\* Based on performance test on Linux on Z Z

# Tools to Accelerate Development/Deployment

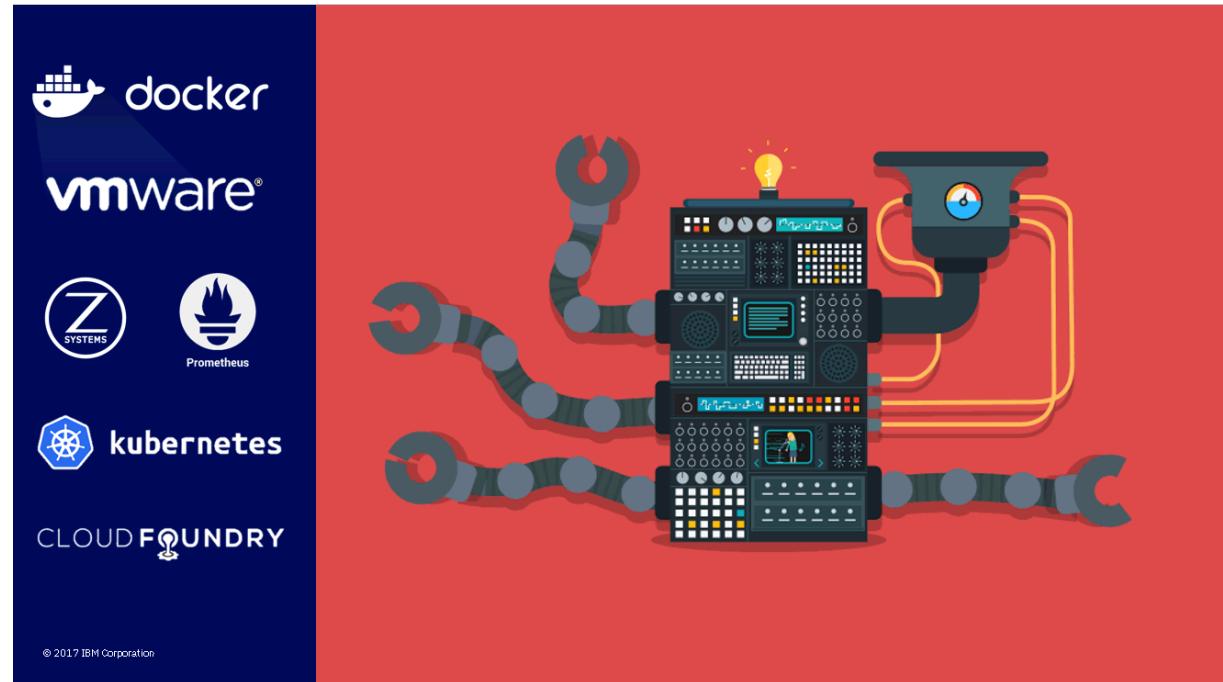
---



- NodeServer
  - IBM Cloud Application Service
  - MicroClimate
  - AppMetrics
  - Loopback
  - Documentation/guidance
- 
- Create
  - Deploy
  - Monitor
- .

# NodeServer – open source generators

- Create Projects pre-wired for monitoring
- Deploy to
  - Docker
  - Kubernetes
  - Cloud Foundry
  - Dev-ops pipeline



<https://www.npmjs.com/package/generator-nodeserver>

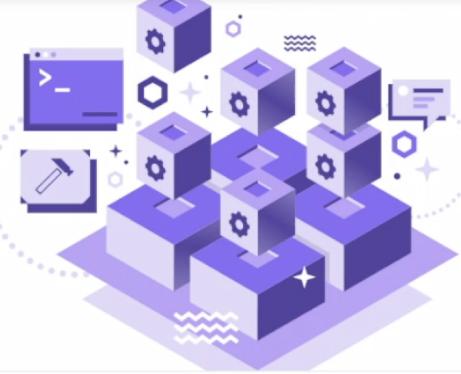
# IBM Cloud Application Service



Overview  
Starter Kits  
Resources  
Projects

Fast on-ramp for building cloud-native apps

## IBM Cloud App Service



Focus on the code

Get started building and deploying Cloud Native apps in minutes with starter-kits pre-integrated with the IBM Cloud.

[Get Starter-Kit](#)

Power your existing apps

Use our developer tools to cloud-enable your server-side apps, and easily deploy to Kubernetes.

[Get CLI Tools](#)

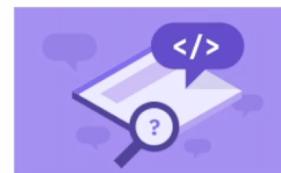
### Featured Resources

Access our guides, documentation, tools and resources

→ [Learn more](#)



**How-to: Deploy to Kubernetes using the CLI**  
Learn how to deploy an application



**Bluemix Blog**  
Check out news, announcements and how-tos about the many services on the IBM Bluemix cloud

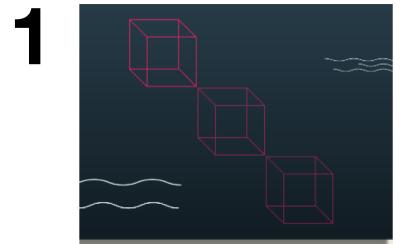


**Tutorial: Introduction to DevOps Continuous Delivery**  
An introduction tutorial on the IBM DevOps Continuous Delivery service

# MicroClimate - IBM Developer Experience



**Microclimate** is an end to end development platform for the creation of cloud native applications and microservices. You can create, edit, build, test and deploy your applications via Continuous Delivery pipelines then run and manage them with IBM Cloud Private



## 1 Containerized Development

Start from scratch using lightweight containers that are easily reproducible to match your production environment locally or on IBM Cloud Private



## 2 Rapid Iteration

Lightning fast round-tripping through edit, build, and run allows real-time performance insights, regardless of what development phase you're in, with an integrated IDE or use your editor of choice with Language Server Protocols



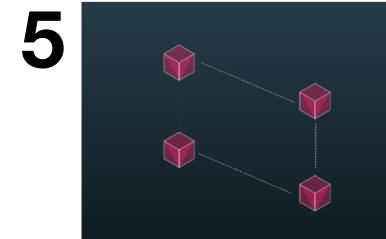
## 3 Intelligent Feedback

Best practices and immediate feedback to help improve your application through your IDE



## 4 Diagnostic Services

Add capability at development time to improve problem determination in production through application metrics.



## 5 Integrated DevOps Pipeline

Get into production fast with a preconfigured DevOps pipeline that can be tailored to your needs

# 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

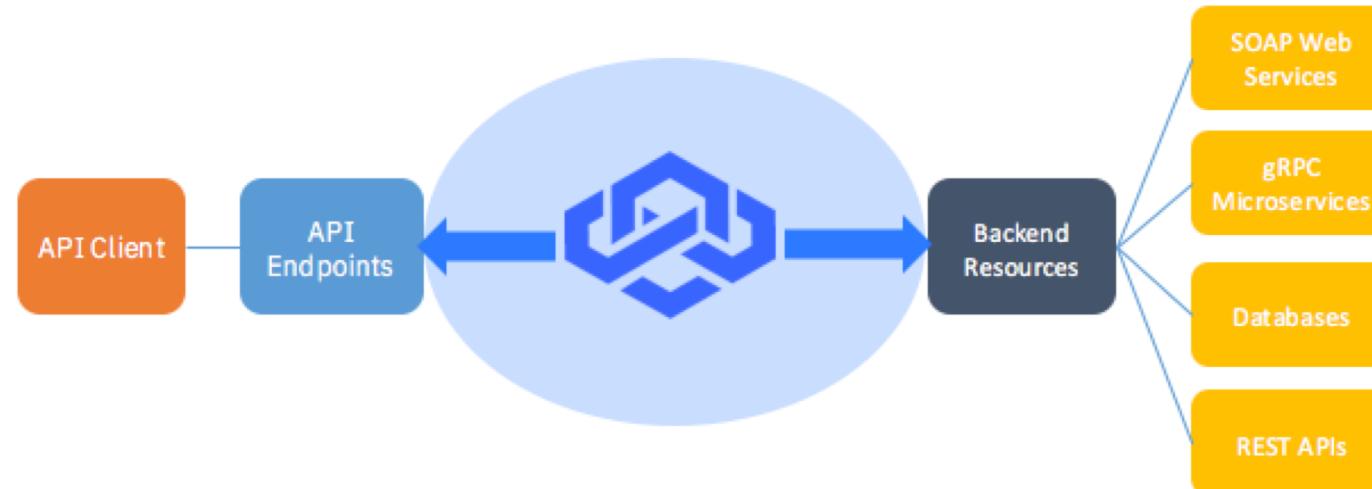
## How to get it?

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

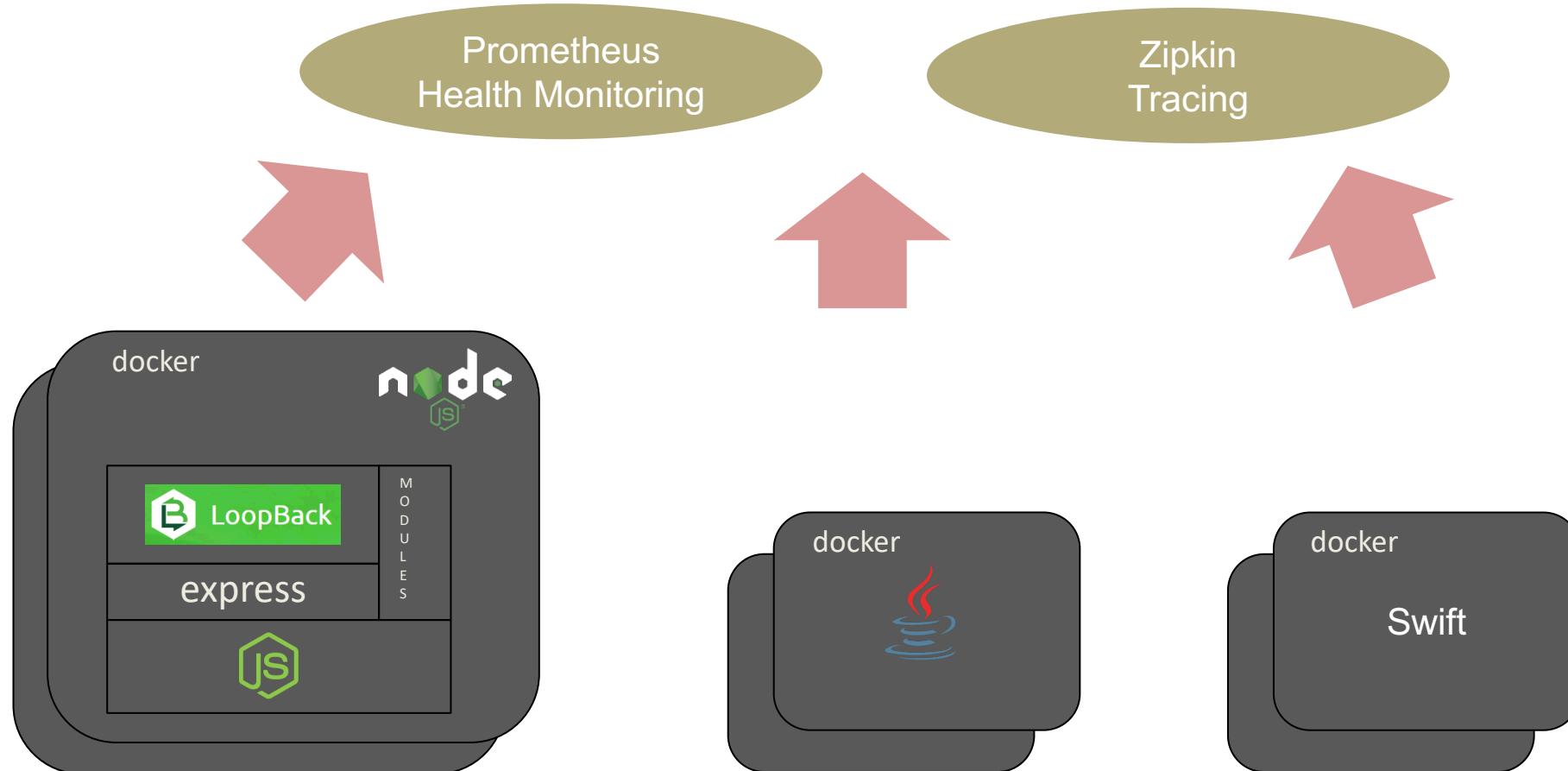


# LoopBack – open-source Node.js framework

- 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)
- LB3 is for production use. LB4 is under active development
- LB4 brings in support for TypeScript



# Tools to Accelerate Development – End Result



# Production Support - IBM Support for Runtimes

---

- Years of experience
- Foundation -Community binaries
- Advanced – Key Modules from the Ecosystem

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

# Questions & Answers