Your Node.js Questions Answered

Michael Dawson
IBM Community Lead for Node.js

Jesse Gorzinski
Business Architect of open source technologies, IBM

POWERUp18 Session ID: 180230

Agenda Key: 27CK

Agenda

- Introductions
  - Who we are
  - Audience
- Common Questions
- Q/A

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

About **Jesse**

- Business Architect of Open Source on i
  - Collaborator
  - Leader of development teams
  - Owns IBM i OSS strategy

- Twitter: @IBMJesseG
- GitHub: @ThePrez
- Linkedin: https://www.linkedin.com/in/ibmjesseg

About the Audience

- Have you written anything in Node.js?
- Have you pushed a module to npm?
- Are you running Node.js in production?
- Is it outward facing?
- Is your company planning to use Node.js?

Common Questions

- What/Why?
- Project Organization
- Production Concerns
- Technical
- Security

Common Questions – What/Why?

- WW1 - What is Node.js
- WW2 - Why are people interested
- WW3 - What are the key components
- WW4 - What platforms are supported
- WW5 - What are the common use cases
- WW6 - When should I use Node.js versus Java

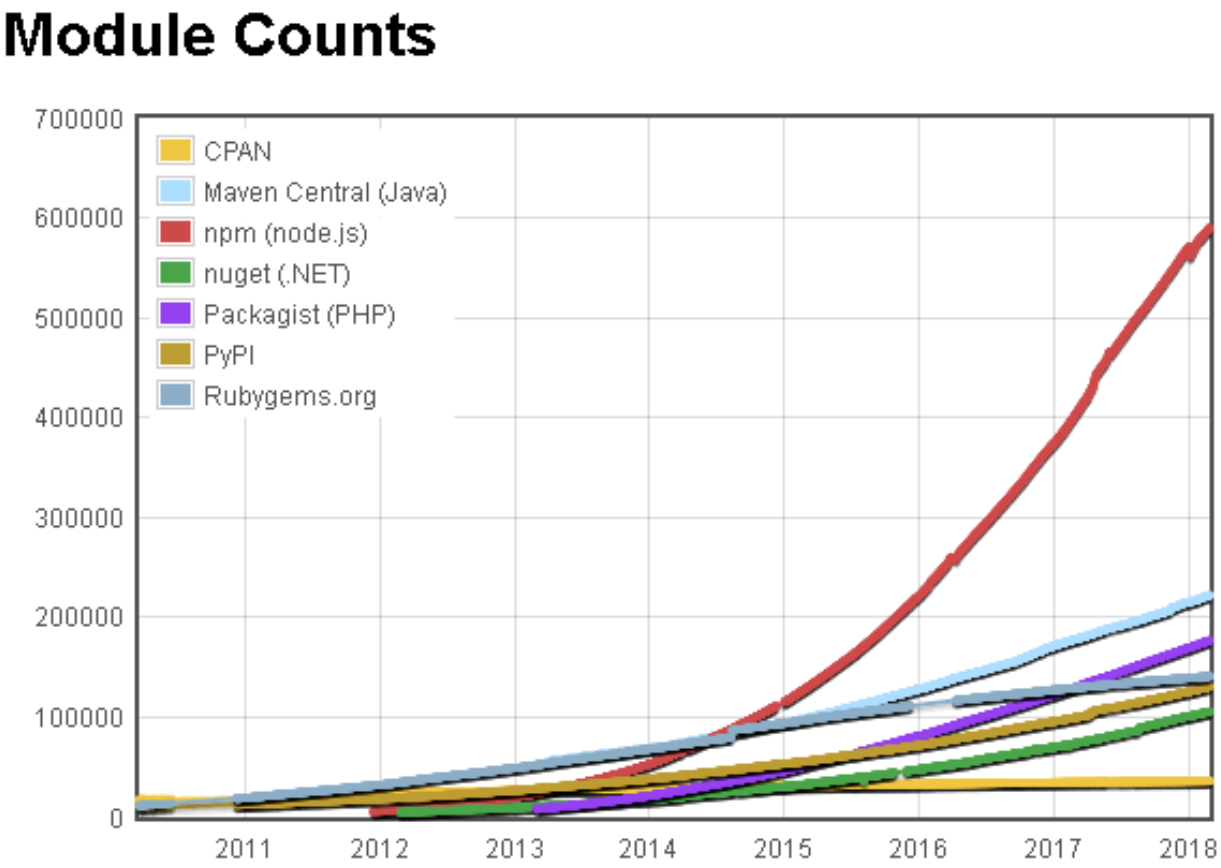
WW1 - Why Node.js – What is it?

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

WW2 - Why Node.js? - Ecosystem

- There is a module for that
  - 600k+ modules
  - #1 on module counts
  - 3x growth rate versus other runtimes
- #1 on Github (#projects)

http://www.modulecounts.com/



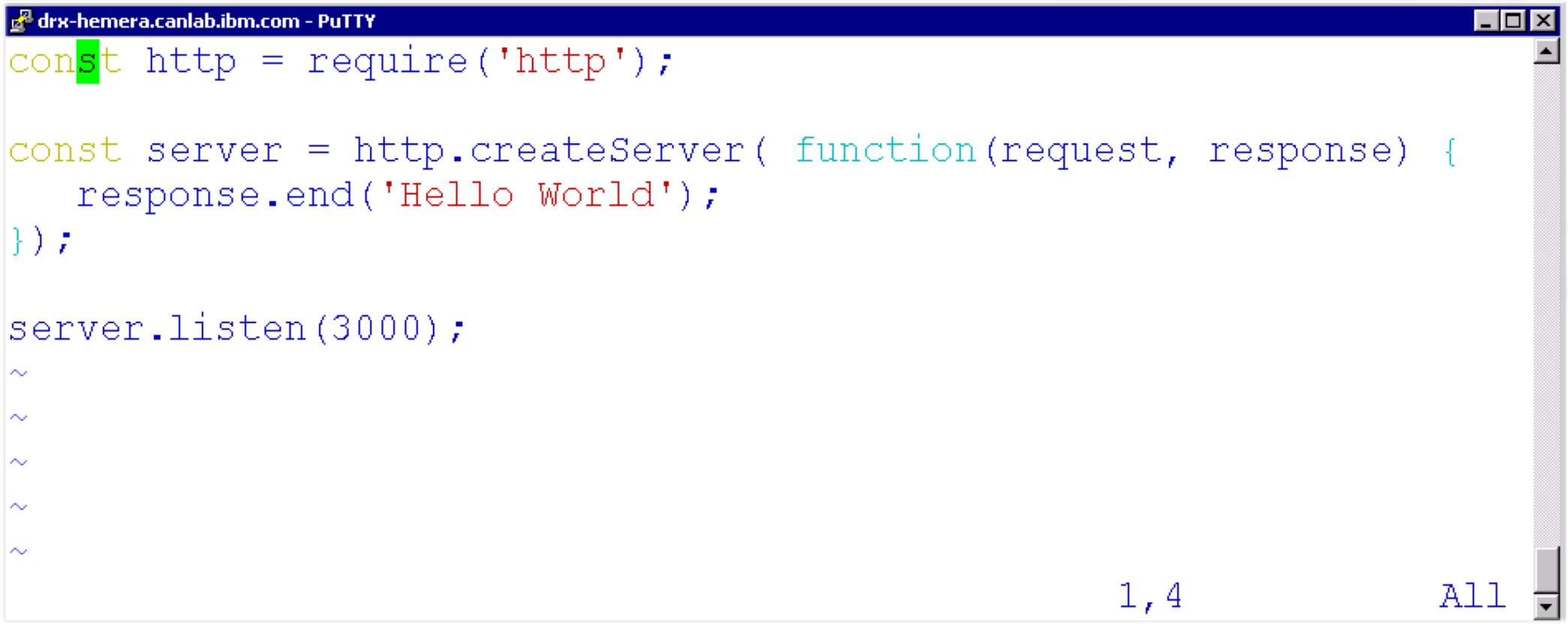
WW2 - Why Node.js? - Productivity

- **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

WW2 - Why Node.js? – Productivity

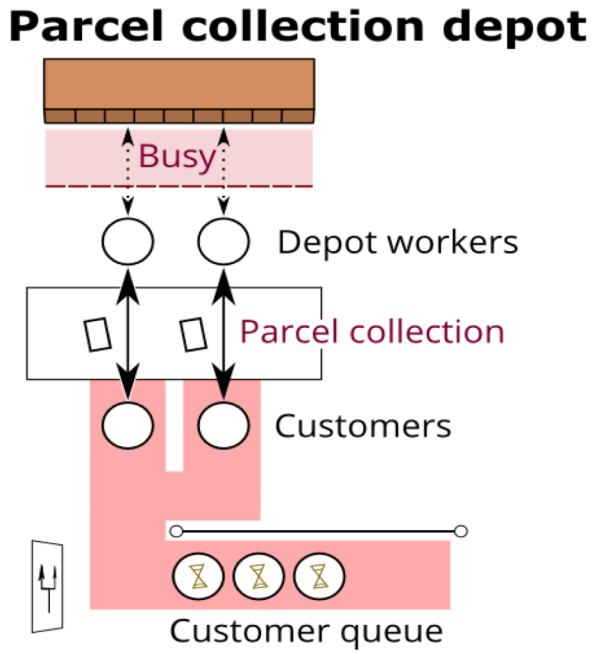
- Reuse of “isomorphic” code components
- Availability of JavaScript talent
- Developer satisfaction

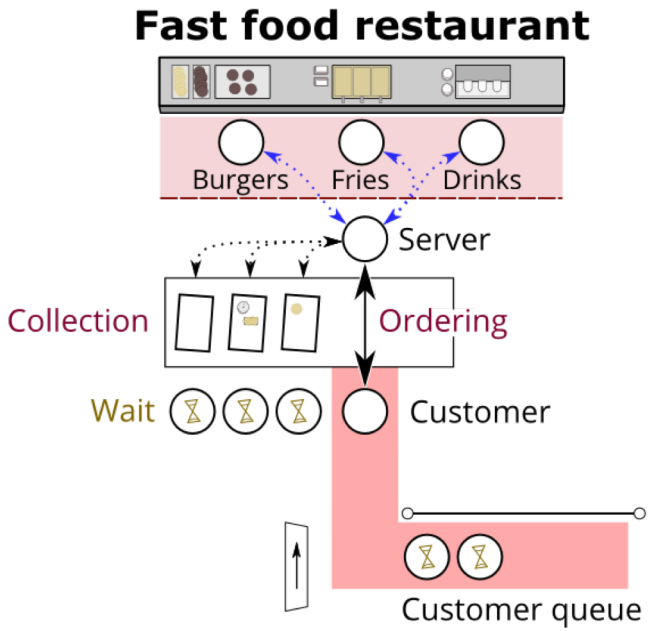
WW2 - Why Node.js? = Productivity



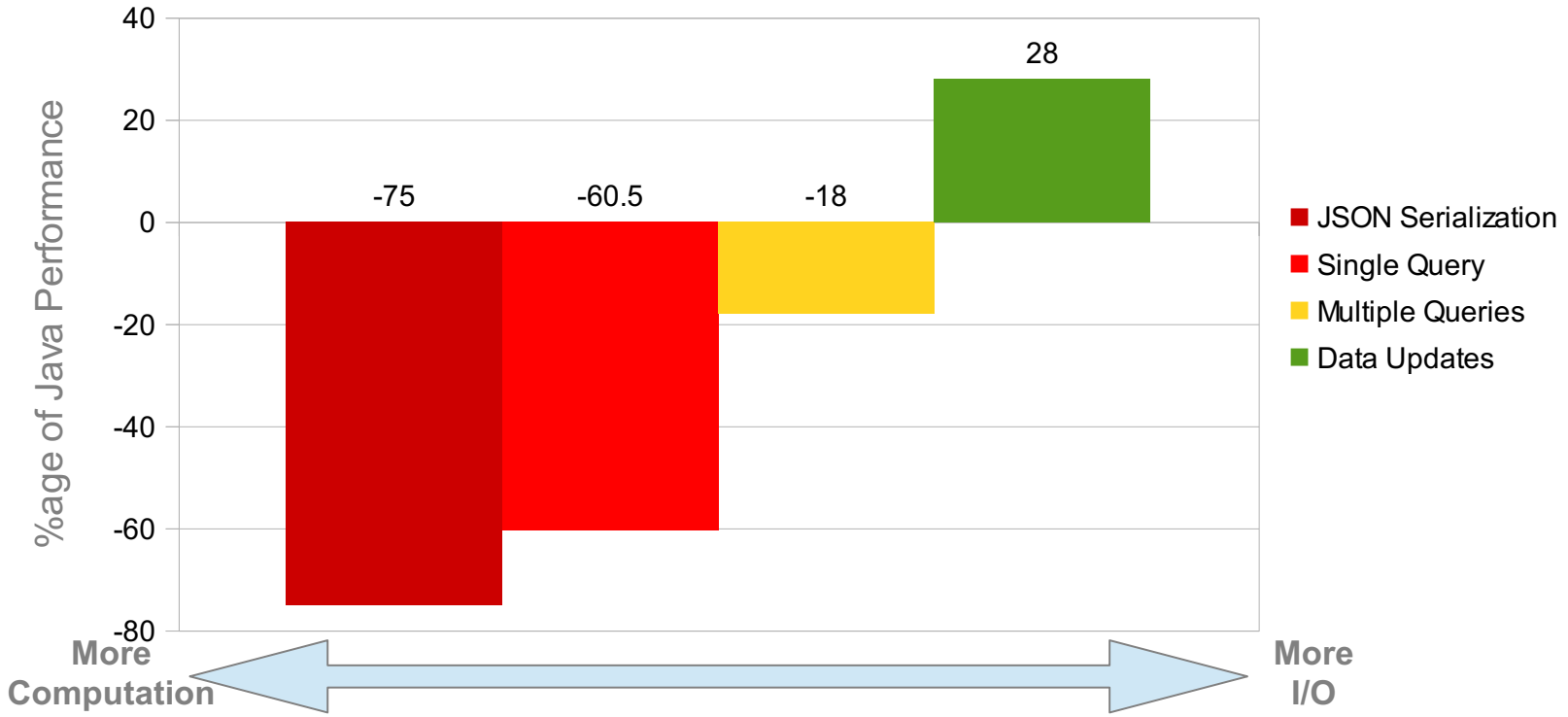
WW2 - Why Node.js? - Performance

Event based: perfect fit for asynchronous non-blocking I/0





WW2 - Why Node.js? - Performance



WW2 - 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%

WW2 - Why Node.js? – Compact/Fast

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

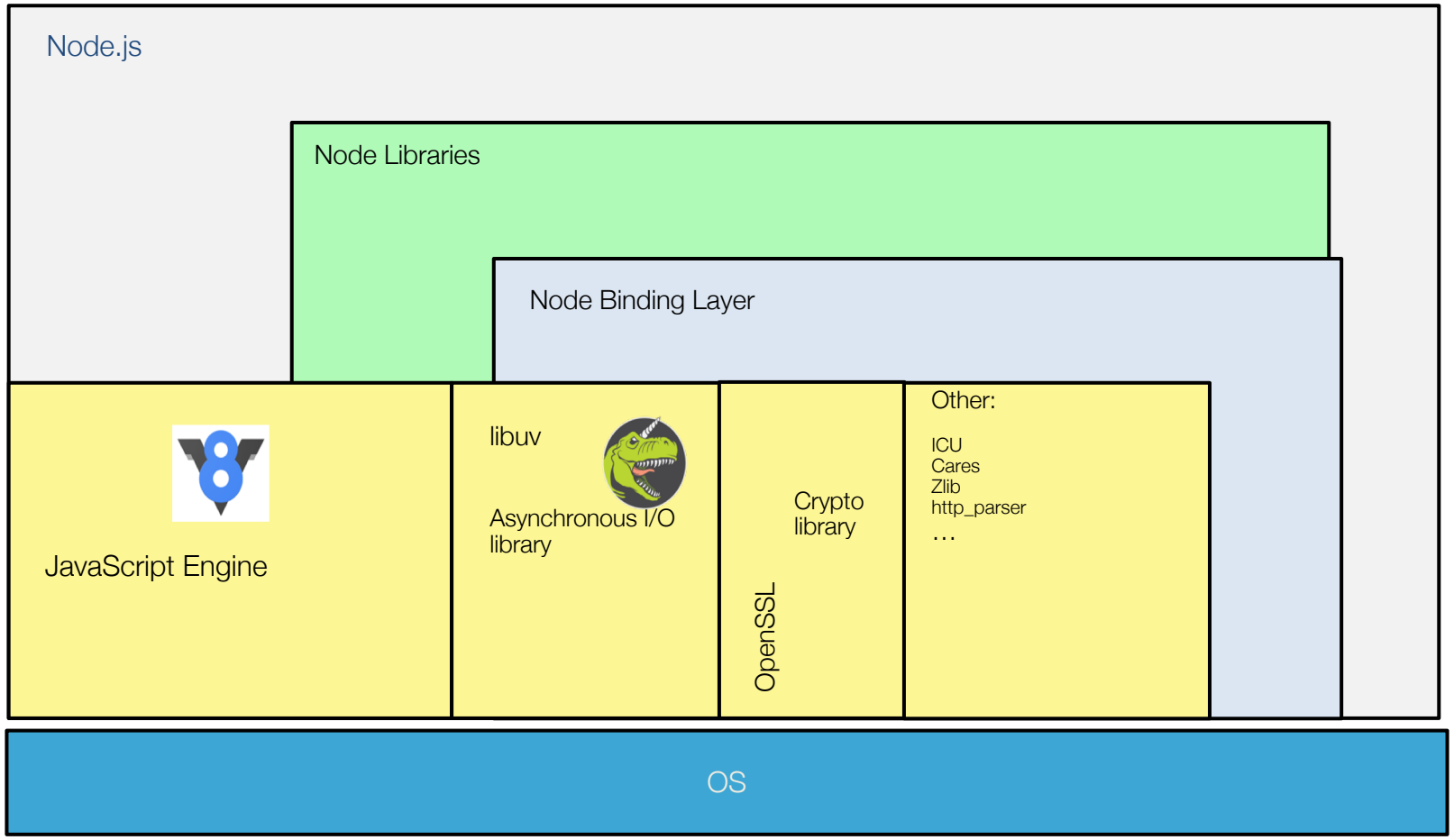
https://nodejs.org/en/download/

- Fast startup
  - 60 ms

https://benchmarking.nodejs.org/

- Small footprint
  - 18 MB

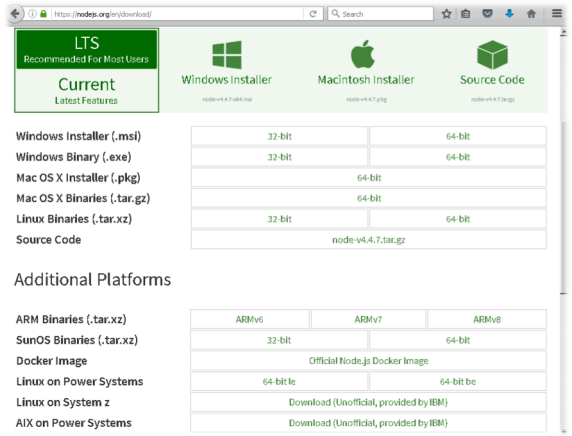
WW3 - Key Components



C/C++ Compiler

WW4 – Platform Support

- Community Binaries
  - Linux on Z
  - Linux on P
  - AIX
  - Linux on x and ARM
  - Windows, Mac, SmartOS



- IBM Binaries
  - System I
  - z/OS

WW5 - Use Cases

- 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

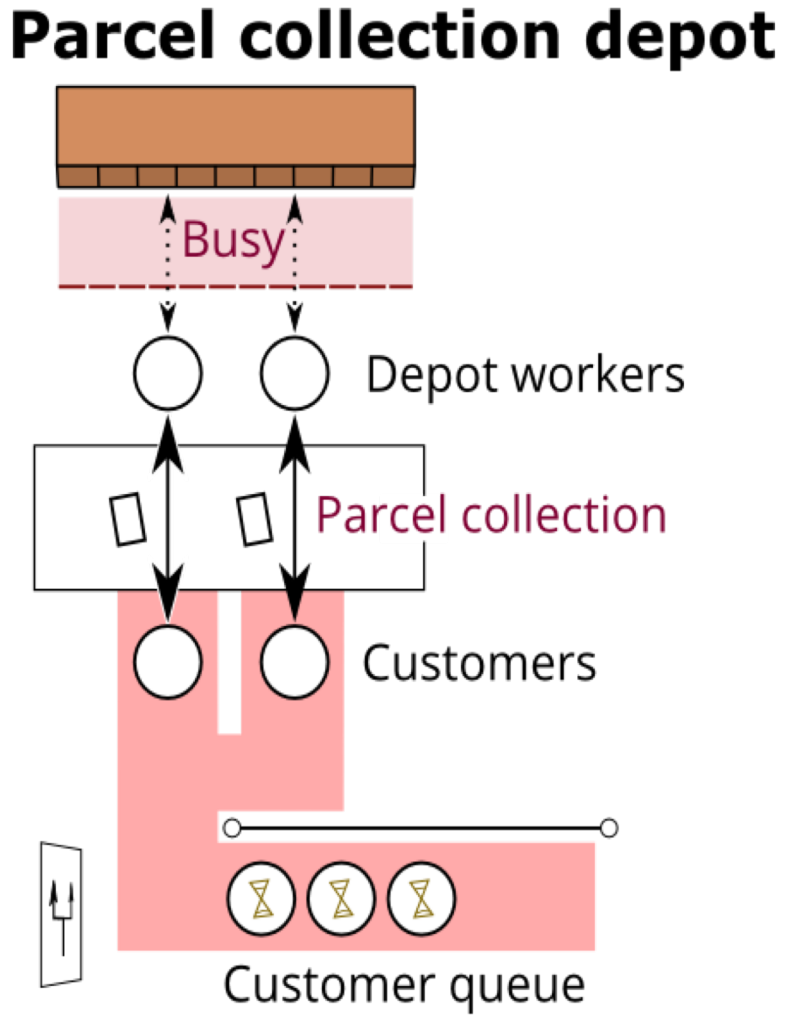
https://github.com/nodejs/benchmarking/blob/master/docs/use_cases.md

WW6 - Node.js versus Java

- Strengths and weaknesses
- Choosing the right language
- Often used together

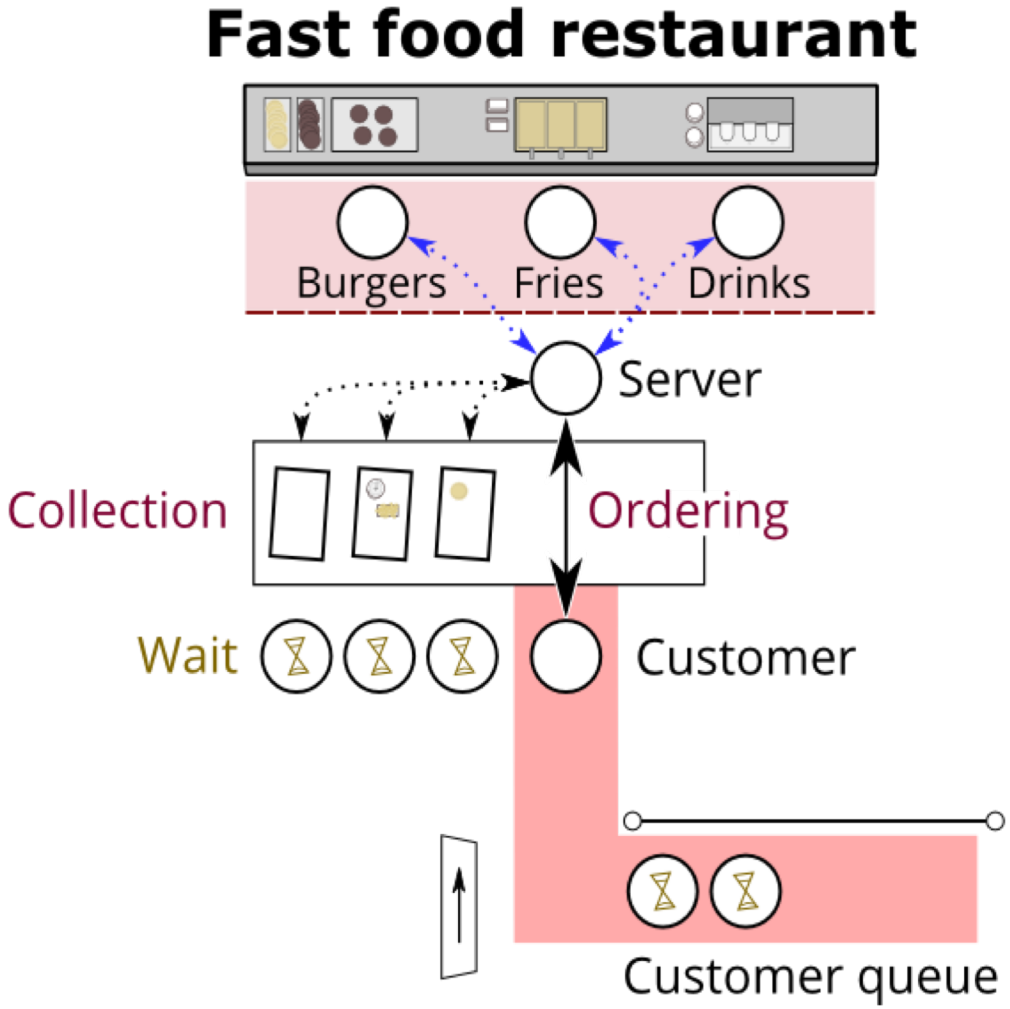
WW6 - Node.js versus Java – Scaling with Java

- 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

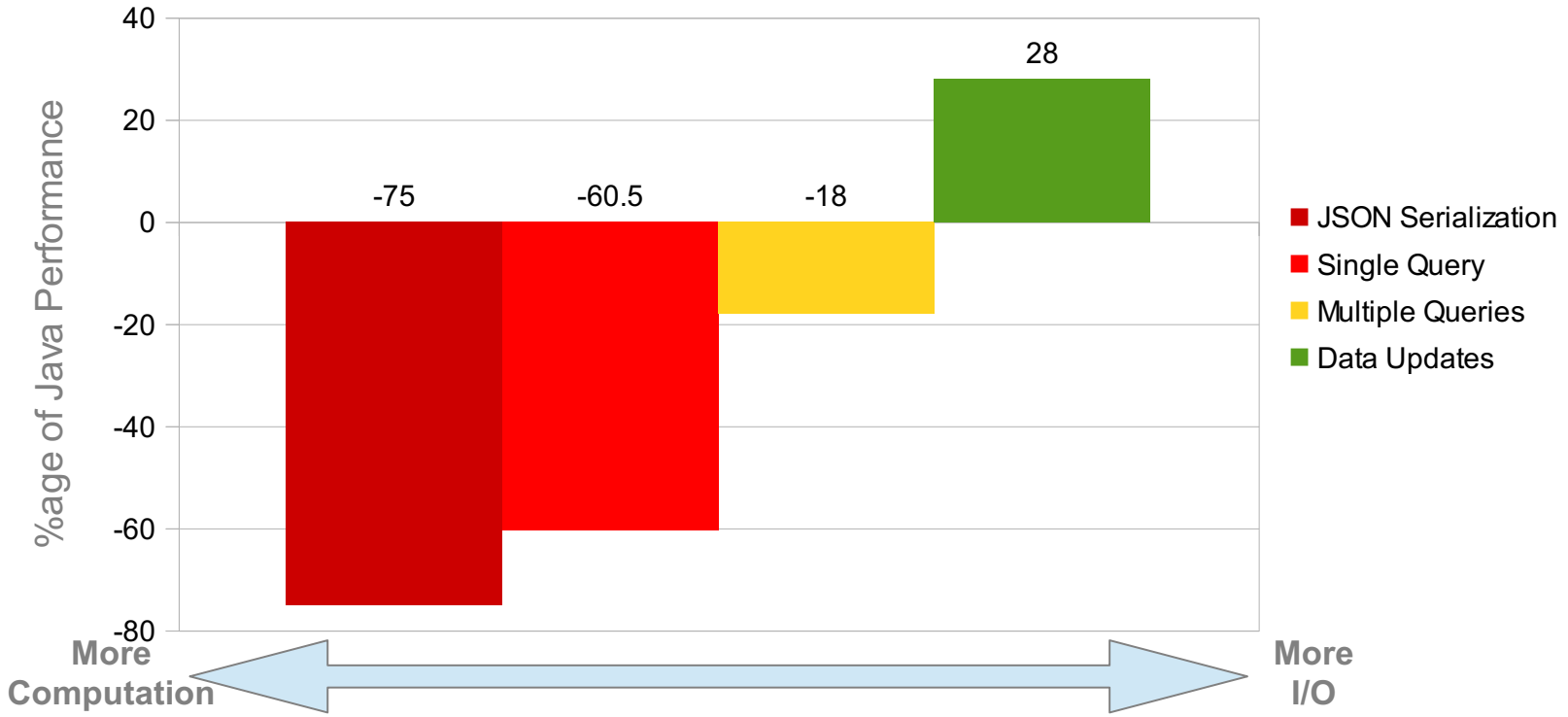


WW6 - Node.js versus Java – Scaling with Node.js

- 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



WW6 - Node.js versus Node.js – Tradeoffs



WW6 - Node.js versus Java – Choosing the Right Language

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

WW6 - Node.js versus Java – Choosing the Right Language

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

WW6 - Node.js versus Java – Choosing the Right Language

+

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

Common Questions – Project Organization

- PO1 - What does the leadership for the project look like and how is the direction set
- PO2 - What is the Node.js foundation and how does it interact with the technical work
- PO3 – What is the history of Node.js
- PO4 - What is semver and how does the Node.js project use it
- PO5 - What are LTS releases
- PO6 - What version of Node.js should I use

Common Questions – Project Organization

- PO7 - How does the project operate day to day
- PO8 - What does the community do in order to ensure good quality
- PO9 - How do I get started in contributing to the Node.js project
- PO10 - What are Node.js working groups, and how do I get involved

PO1 - Leadership

- **Board**
- **TSC**
- **Community Committee**
- **WGs**
- **Teams**

PO2 - 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

PO3 - Node.js Community - History

- 2009 – written by Ryan Dhal
- 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 - …

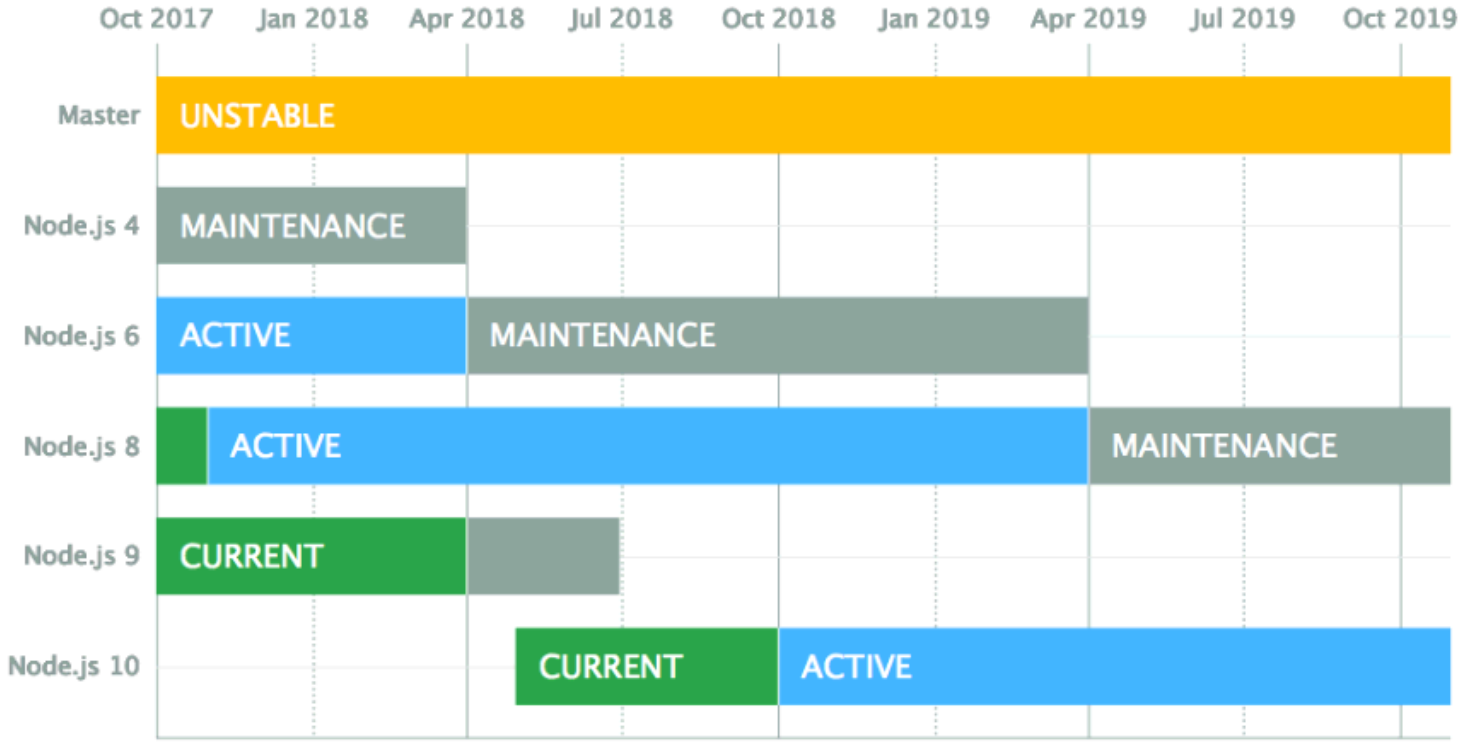
PO4 - Semver

X.Y.Z:

- *X* – Major: backwards incompatible changes
- *Y* – Minor: additive, new features
- *Z* – Patch: no API changes or new features

PO5 - Node.js Long Term Support (LTS)

- Current Release
  - every 6 months
  - Semver major
- LTS release every October
  - Even semver majors
  - 30 months of support



https://github.com/nodejs/lts

PO6 – Versions

- Most stable – LTS
  - Latest gives you longest runway
  - Plan to upgrade at least 6 months in advance
  - Changes already validated in Current
- Current – Live closer to the edge
  - Most up to date fully tested release
  - More rapid pace of chance, less settling time
- Nightly
  - Experiment with new features in master

PO7 - 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
  - Teams https://github.com/orgs/nodejs/teams
- Community Committee
  - Teams (ex user-feedback)
  - Working Groups (to come)

PO8 – Quality with Speed?

- Different release types
- Change flow processes
- Enhancement Proposal process
- Automation and Testing
  - Functional Tests
  - Module Testing
  - Stress Testing (Future)
  - Platform/OS coverage (Future)
  - Development Workflows (Future)
- Performance Benchmarks
- Tools

PO9 – I want to contribute, where to start ?

- Node Todo: http://nodetodo.org/
- http://coverage.nodej.org
- Issues
  - Follow/comment on issues
  - “Good first contribution tag”
  - Tests/doc, lots to do here
- Working Groups
  - find one that interests you!
- Strategic Initiatives

https://github.com/nodejs/TSC/blob/master/Strategic-Initiatives.md
https://github.com/nodejs/community-committee/blob/master/STRATEGIC-INITIATIVES.md

Common Questions – Production Concerns

- PC1 - What are some of the common use cases
- PC2 - How does a company typically start using Node.js
- PC3 - How do I monitor applications
- PC4 - What kinds of tools do I need for a production app
- PC5 - What about web frameworks
- PC6 - How/where do I run my Node.js applications

PC1 – Common Use Cases

- 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

https://github.com/nodejs/benchmarking/blob/master/docs/use_cases.md

PC2 - How does a company start using Node.js

- Starts using it internally for non-critical
- Expands to more critical but still internal uses
- After success and experience, uses it externally

PC3 - How do I monitor applications?

- Aggregate logs: Splunk, Loggly, Syslog, …
- Graph your metrics: ELK, statsd/graphite, appmetrics
- Consider higher level tools: Newrelic, Appdynamics, IBM BAM/APM,…

PC4 – Tools for production app ?

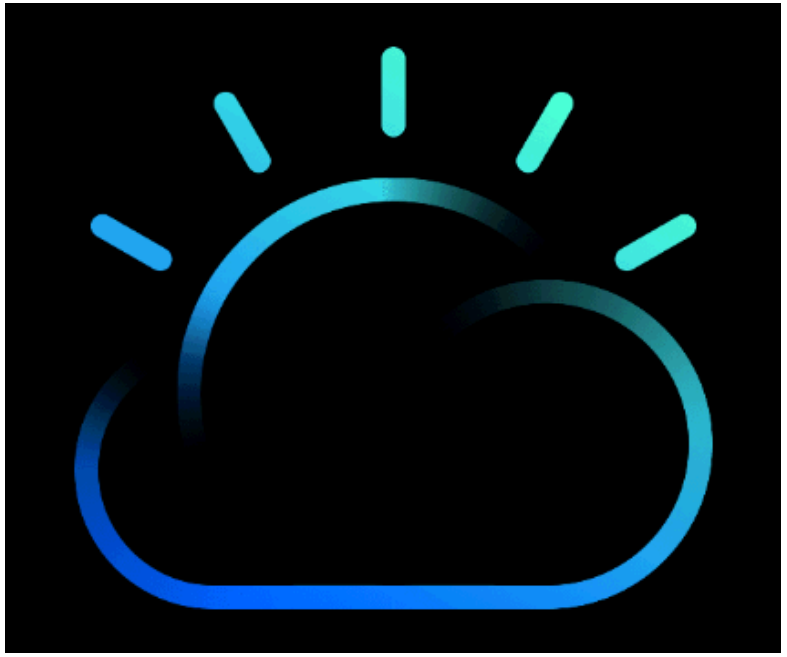
- heapdump (appmetrics has it pre-compiled)
  - dumps can be analyzed with Chrome Dev Tools
  - https://strongloop.com/strongblog/how-to-heap-snapshots/
- node-report – human readable first failure information
- core dump on uncaught exception
  - core files can be analyzed with llnode

PC5 – What about web frameworks

- Pick one!
- express: bare bones, build it yourself, good way to tinker
- hapi, restify, koa, sails, loopback: when you want more

PC6 – Where to run my applications

IBM Cloud

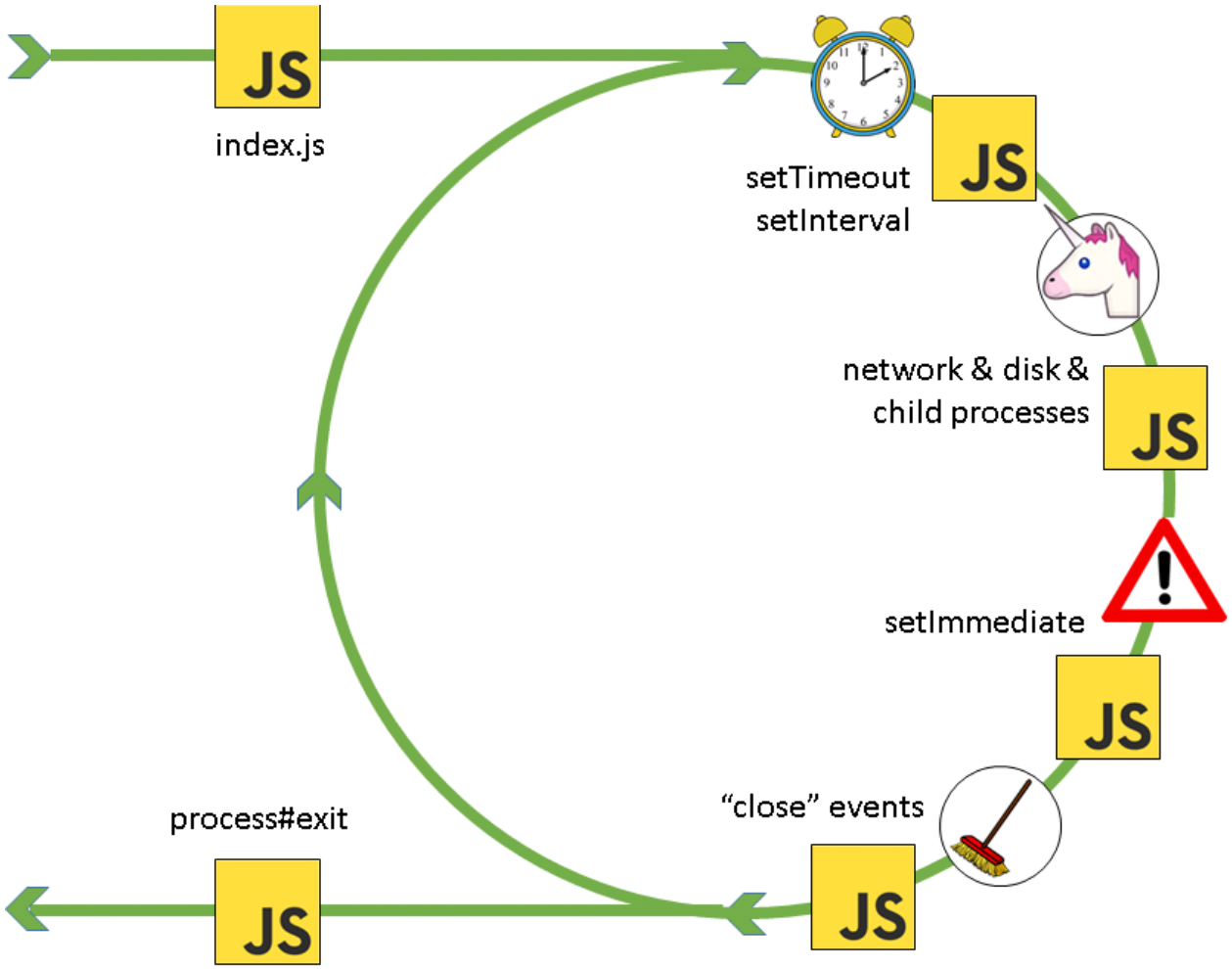


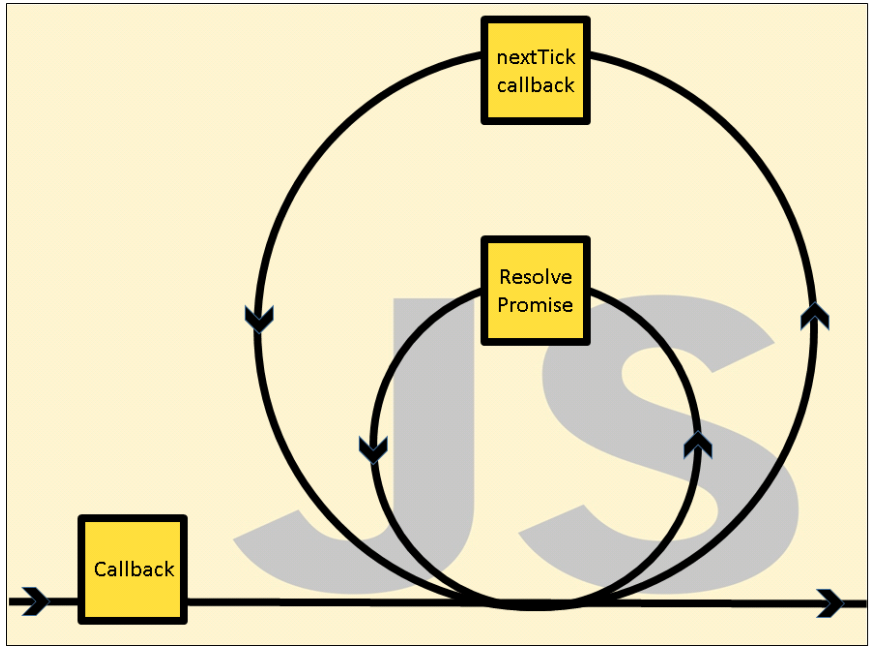
- And other clouds of course …
- Node is always one of the top tier languages
- And works great on premise if that's still your thing
  - Your choice of hardware due to broad platform support

Common Questions – Technical

- T1 - Whats this event loop thing
- T2 - How should I use semver and manage project dependencies
- T3 - What is the Node.js programming model
- T4 - How do you integrate with Native code
- T5 - Why do I have to recompile my native modules for major versions
- T6 - Tools to deal with asynchrony
- T7 - Common tools (beside npm, git)

T1 – Event Loop





T2 – Managing dependencies

- Use “loose” dependency specifications
- Freeze packages at deploy time, https://strongloop.com/strongblog/node-js-deploy-production-best-practice
- Keep up to date!

T3– 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$
```

T3– 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) {});
```

T4– Native Code

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

```
#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);
```

Note should consider N-API when writing your modules
https://nodejs.org/dist/latest-v10.x/docs/api/n-api.html#n_api_n_api

T4– Native Code

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

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

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

T5 – Why do I have to recompile for each release

- Direct use of V8
  - Fast pace of change
- Nan, helps but recompile still needed …
- N-API
  - https://nodejs.org/dist/latest-v10.x/docs/api/n-api.html#n_api_n_api

T6 - Tools to deal with asynchrony

- Promises: pros/cons
- callback-based (use async): pros/cons
- Read blogs! Lots of traps for beginners (especially with promises).

T7 – Common tools (other than npm, git)

- Lodash
- Eslint
- Package scripts
- Chrome Dev Tools

Common Questions – Security

- SEC1 – What tools should I be using
- SEC2 – What Node.js version should I use
- SEC3 – What should I be watching for updates
- SEC4 – What is the bug bounty program
- SEC5 – What about vulnerabilities in modules

SEC1 – What tools should I be using

- snyk
- nsp
- Not strictly security, but
  - eslint
  - coverity

SEC2 – What Node.js version should I use

- **8.x!** It's the best LTS so far

SEC3 – What should I be watching for updates

- Watch https://nodejs.org/en/blog/ to keep up to date.
- Subscribe to nodejs-sec google group

SEC4 – What is the bug bounty program?

- Managed through the HackerOne platform
- https://hackerone.com/nodejs

Impact	Amount
High *Demonstrate that remote exploitation of this bug can be easily, actively, and reliably achieved.*	$1,500+
Medium *Demonstrate that remote exploitation of this bug is very likely (e.g. good control a register).*	$1,000
Minimum *Demonstrate the presence of a security bug with probable remote exploitation potential.*	$500

Only vulnerabilities in Node.js core will be considered for eligibility. **Submissions related to *nodejs.org* and other project websites are explicitly NOT eligible.**

SEC5 – What about vulnerabilities in modules?

- https://github.com/nodejs/security-wg takes reports
- **https://github.com/nodejs/security-wg/blob/master/processes/third_party_vuln_process.md**
- **Uses HackerOne**