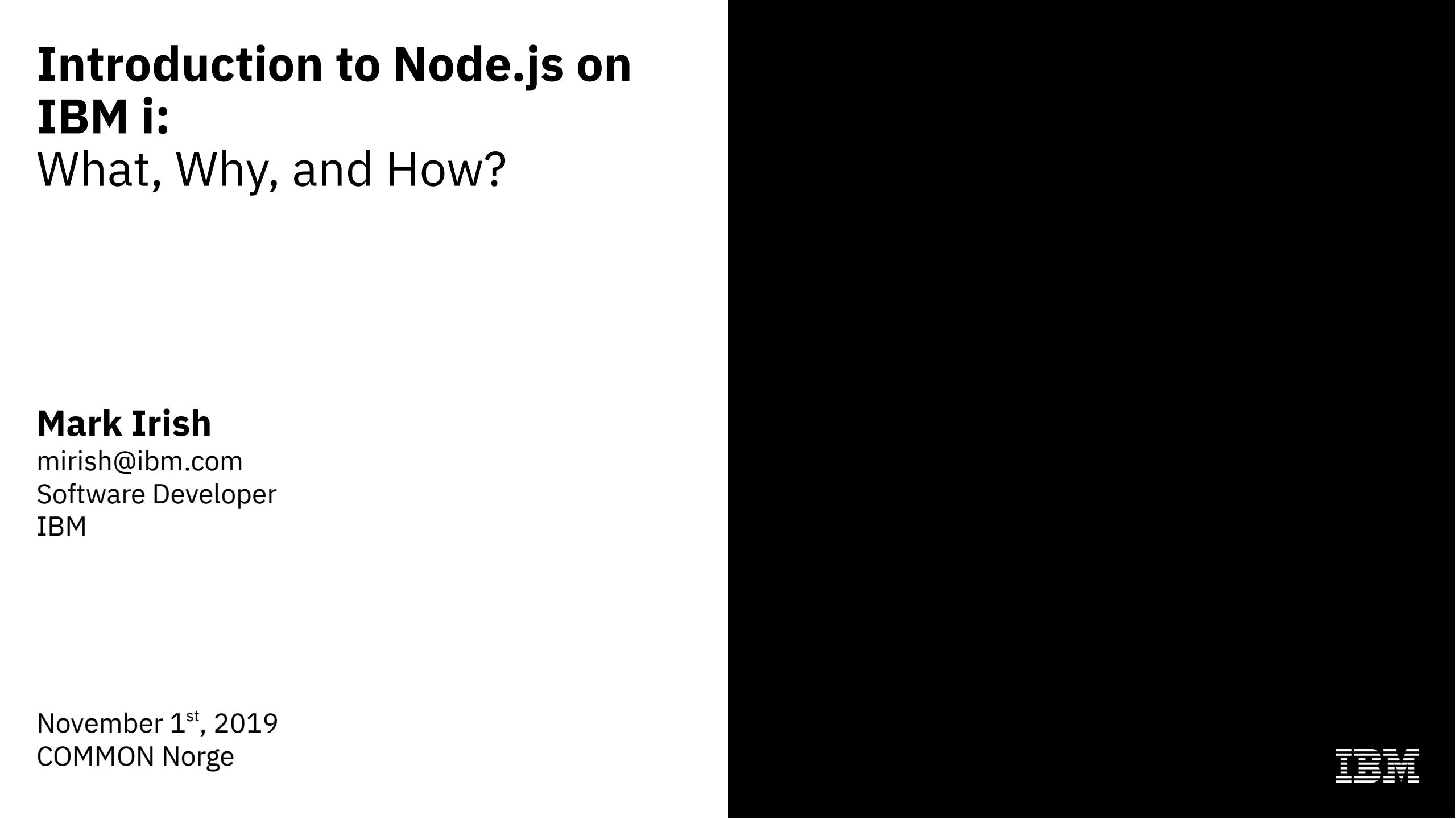
Introduction to Node.js on IBM i:
What, Why, and How?

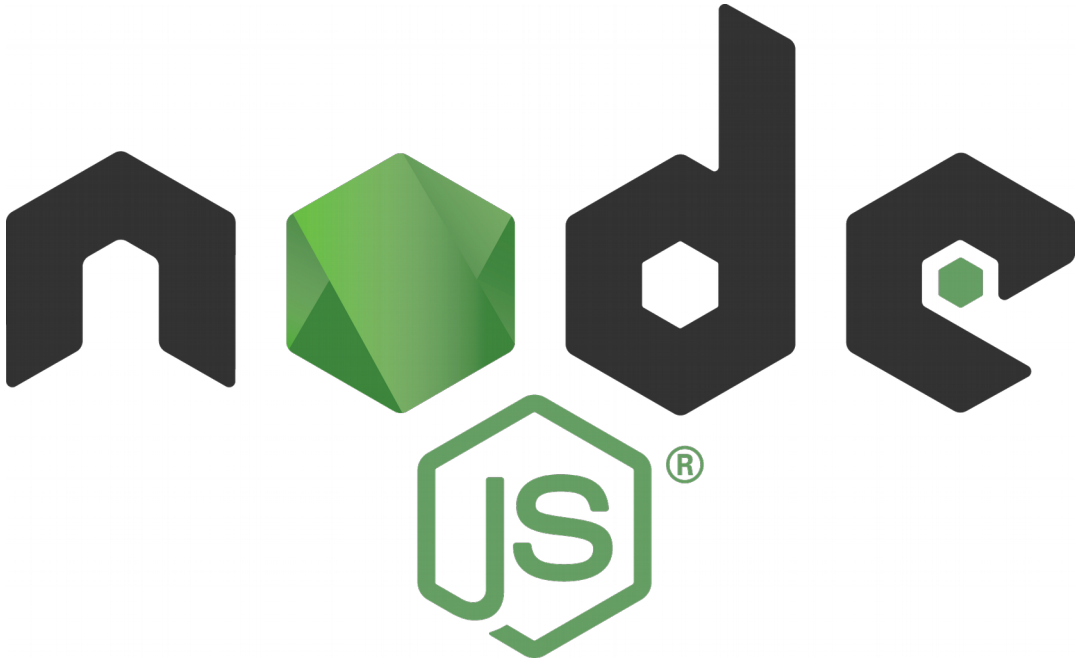
Mark Irish
mirish@ibm.com
Software Developer
IBM

November 1st, 2019
COMMON Norge



Outline

- Introduction
- What is Node.js?
  - JavaScript Runtime
  - Node.js APIs
  - Event Loop
  - npm: The Node Package Manager
- Why should I use Node.js
  - Too many reasons to list!
- How do I use Node.js?
  - Installing Node.js on IBM i
  - Writing your first program
- Conclusion



What? Why? How?

Node.js: A JavaScript Runtime Environment

- Node.js is an environment that runs JavaScript code outside of a browser
- Node.js uses Google's V8 Engine, written in C++, to quickly interpret and execute code
- Released in 2009, Node.js is a **stable, mature environment** for developing applications

A JavaScript* Refresher

- JavaScript is a high-level, interpreted, weakly-typed language
- JavaScript was written in the mid 90s for scripting in web-browsers
- In Node.js, JavaScript is used for server-side scripting (think PHP)

* This has nothing to do with Java!

JavaScript Front-End *AND* Back-End?

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

- Jeff Atwood: Author, Entrepreneur, Co-founder of StackOverflow

JavaScript Front-End *AND* Back-End?

Browser (front-end):

- JavaScript to access DOM elements such as `window` and `document`
- Allows manipulation of data in a web browser, like changing HTML or element styles

Node.js (back-end):

- JavaScript to access Node.js APIs such as `fs`, `child_process`, and `https`
- Interacts with the Node.js runtime, which can access your system (files, threads, the internet, your network, etc.)

Node.js APIs

Node.js exposes a number of APIs that allow you to interact with your machine.

APIs exist for:

- Access the file system
- Throwing and handling errors
- Getting information from the operating system
- Sending/receiving HTTP and HTTPS requests
- Spawning child processes
- And more!

API Example

File System (`fs`)

- “The fs module provides an API for interacting with the file system in a manner closely modeled around standard POSIX functions”

Example:

```
fs.readFile(path[, options], callback)

fs.mkdir(path[, options], callback)
```

If you download a package like PDFKit, you call its functions, which in turn call the `fs` API.

More Than Just JavaScript!

N-API (Node API) allows you to write C programs for Node.js

- Maintained as an official part of Node.js
- Engine agnostic, so it can run on V8 or any future engine
- ABI (Application Binary Interface) stable, so it doesn’t need to be recompiled on newer version of Node.js
- node-addon-api is a C++ wrapper for N-API

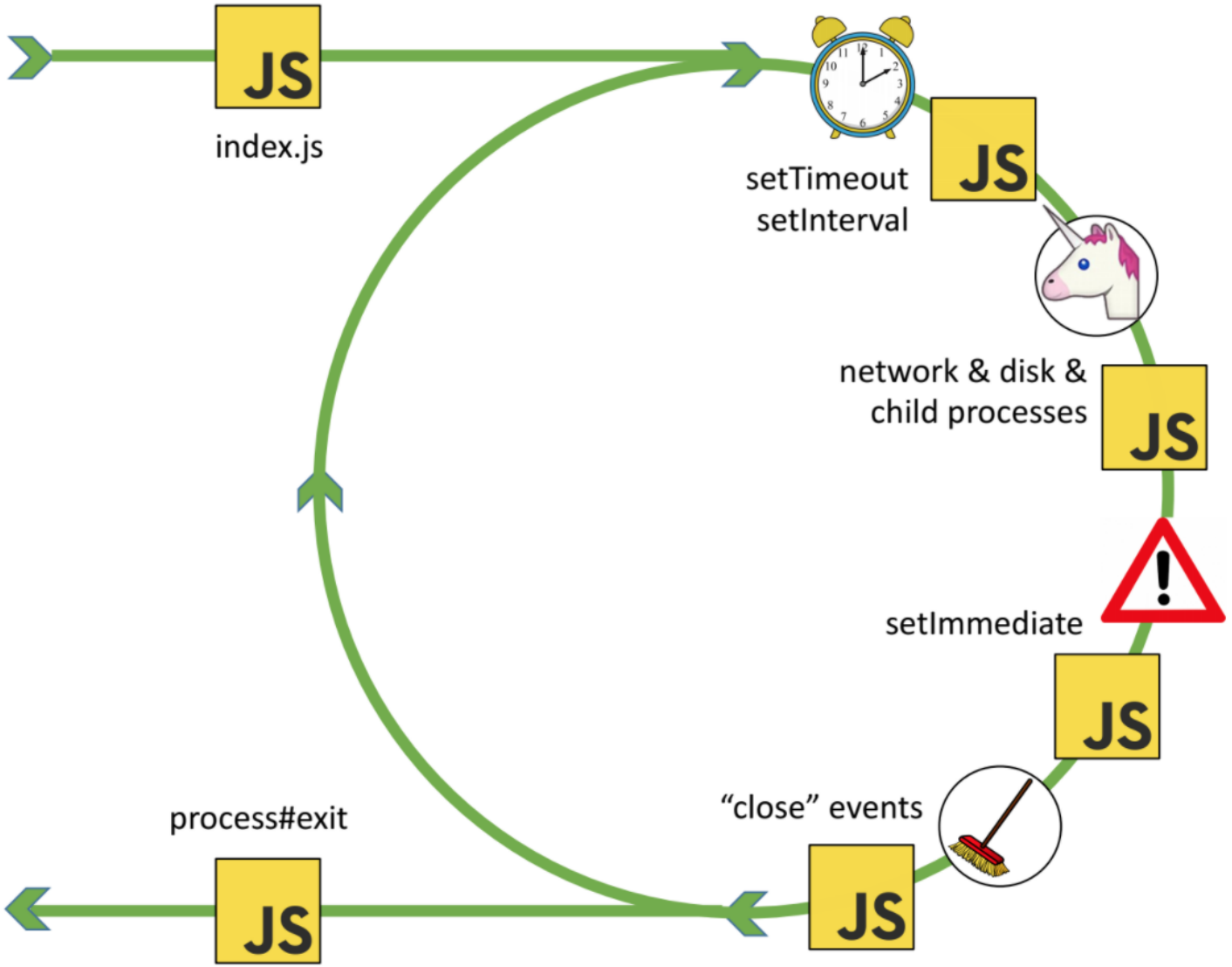
Node.js Event Loop

Very important information! You can shoot yourself in the foot if you don’t understand how Node.js executes code!

Node.js is single-threaded!

- Both user space (your code) and the internal code run on the same thread
- To avoid lockup, you need to use asynchronous code (callback functions and/or promises) to break up your code
- A simplified view of the event loop might look like...

Node.js Event Loop



Prevent Blocking the Event Loop

Using callback functions…

```
fs.readFile('/etc/passwd', (err, data) => {
  if (err) throw err;
  console.log(data);
});

/*
The (err, data) => { … } is the callback function. When readFile is done running
internally (where it sends I/O work to a libuv thread), it will call the
function, and the code between the { … } will be executed. In the meantime, it
won’t be blocking the event loop.
*/
```

Prevent Blocking the Event Loop

(callback hell)

```
function iIds(startAt, showSessionRoot, iNewNmVal, endActionsVal, iStringVal, seqProp, htmlEncodeRegEx) {
    if (SbUtil.dateDisplayType === 'relative') {
        iRange();
    } else {
        iSelActionType();
    }
    iStringVal = notifyWindowTab;
    startAt = addSessionConfigs.sbRange();
    showSessionRoot = addSessionConfigs.elHiddenVal();
    var headerDataPrevious = function(tabArray, iNm) {
        iPredicateVal.SBDB.deferCurrentSessionNotifyVal(function(evalOutMatchedTabUrlsVal) {
            if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
                iPredicateVal.SBDB.normalizeTabList(function(appMsg) {
                    if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
                        iPredicateVal.SBDB.detailTxt(function(evalOrientationVal) {
                            if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
                                iPredicateVal.SBDB.neutralizeWindowFocus(function(iTokenAddedCallback) {
                                    if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
                                        iPredicateVal.SBDB.evalSessionConfig2(function(sessionNm) {
                                            if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
                                                iPredicateVal.SBDB.iWindow2TabIdx(function(iURLsStringVal) {
                                                    if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
                                                        iPredicateVal.SBDB.idx7Val(undefined, iStringVal, function(getWindowIndex) {
                                                            if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
                                                                addTabList(getWindowIndex.rows, iStringVal, showSessionRoot && showSessionRoot.length > 0 ? show
                                                                    if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
                                                                        evalSAllowLogging(tabArray, iStringVal, showSessionRoot && showSessionRoot.length > 0 ?
                                                                            if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
                                                                                BrowserAPI.getAllWindowsAndTabs(function(iSession1Val) {
                                                                                    if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
                                                                                        SbUtil.currentSessionSrc(iSession1Val, undefined, function(initCurrentSe
                                                                                            if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
                                                                                                addSessionConfigs.render(matchText(iSession1Val, iStringVal, eva
                                                                                                    id: -13,
                                                                                                    unfilteredWindowCount: initCurrentSessionCache,
                                                                                                    filteredWindowCount: iCtrl,
                                                                                                    unfilteredTabCount: parseTabConfig,
                                                                                                    filteredTabCount: evalRegisterValue5Val
                                                                                                }] : [], cacheSessionWindow, evalRateActionQualifier, undefined,
                                                                                                    if (seqProp) {
                                                                                                        seqProp();
                                                                                                    }
                                                                                                });
                                                                                            }
                                                                                        });
                                                                                    }
                                                                                });
                                                                            }
                                                                        });
                                                                    }
                                                                });
                                                            }
                                                        }, showSessionRoot && showSessionRoot.length > 0 ? showSessionRoot : startAt ? [startAt] : []);
                                                    }
                                                });
                                            }
                                        });
                                    }
                                });
                            }
                        });
                    }
                });
            }
        });
```

Prevent Blocking the Event Loop

...or Promises.

```
const contents = await fsPromises.readFile('/etc/passwd');
console.log(contents);

/*
FsPromises.readFile returns a Promise. By using the ‘await’ keyword, Node.js
knows to do other things on the event loop until the Promise resolves, then puts
the execution of more code on a the queue.
*/
```

Prevent Blocking the Event Loop

Don’t do long-running functions without breaking them up!

```
for (let i = 0; i < Integer.MAX_SAFE_INTEGER; i++) {
    // this will stop all other execution
    console.log(`i is ${i}`);
}

/* The above code will block the entire Node.js process until the loop finishes
running! Break up loops like above into smaller chunks, otherwise it will block
everything.
*/
```

Node.js Event Loop

In reality, most Node.js I/O is built to be non-blocking by default

If your program starts to have poor performance, look for places where there are long-running functions

npm (Node.js Package Manager)

npm is a public repository with over **one-million open-source Node.js packages**

You can download frameworks, libraries, even single functions that will **greatly reduce development time**

npm is the **most powerful aspect of Node.js**

npm Orgs and npm Enterprise

Private repositories to store your business's proprietary packages

Modularize your Node.js application, publish modules privately, and import them only as needed

All the power of npm to download packages, but with privacy and security for your business

`package.json`

Holds the metadata for your application

- application name
- license
- dependencies to download from npm

Copy your `package.json` and source code to a new location, then reinstall dependencies from the internet

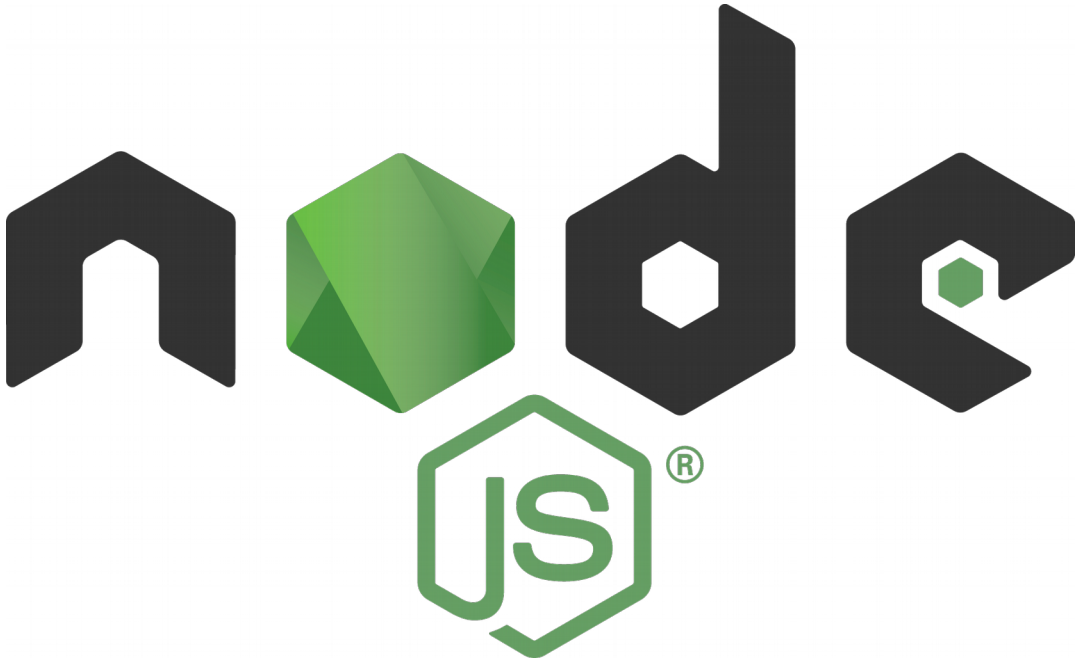
I'll teach you more in the **How?** section

Node.js sounds huge!

Some fun numbers:

- 20 MB: The size of the IBM i RPM
- 18 MB: The entire size of Node.js
- 60 ms: The time it takes to start Node.js

Node.js is small and fast, meaning low overhead for your applications



What? **Why?** How?

JavaScript Makes Programming Easy

JavaScript is an incredibly easy language to learn

- Weakly-typed
- Interpreted (no precompilation needed, all JIT)
- Fewer “gotchas” compared to other languages

Less time spent reading documentation, more time spent developing!

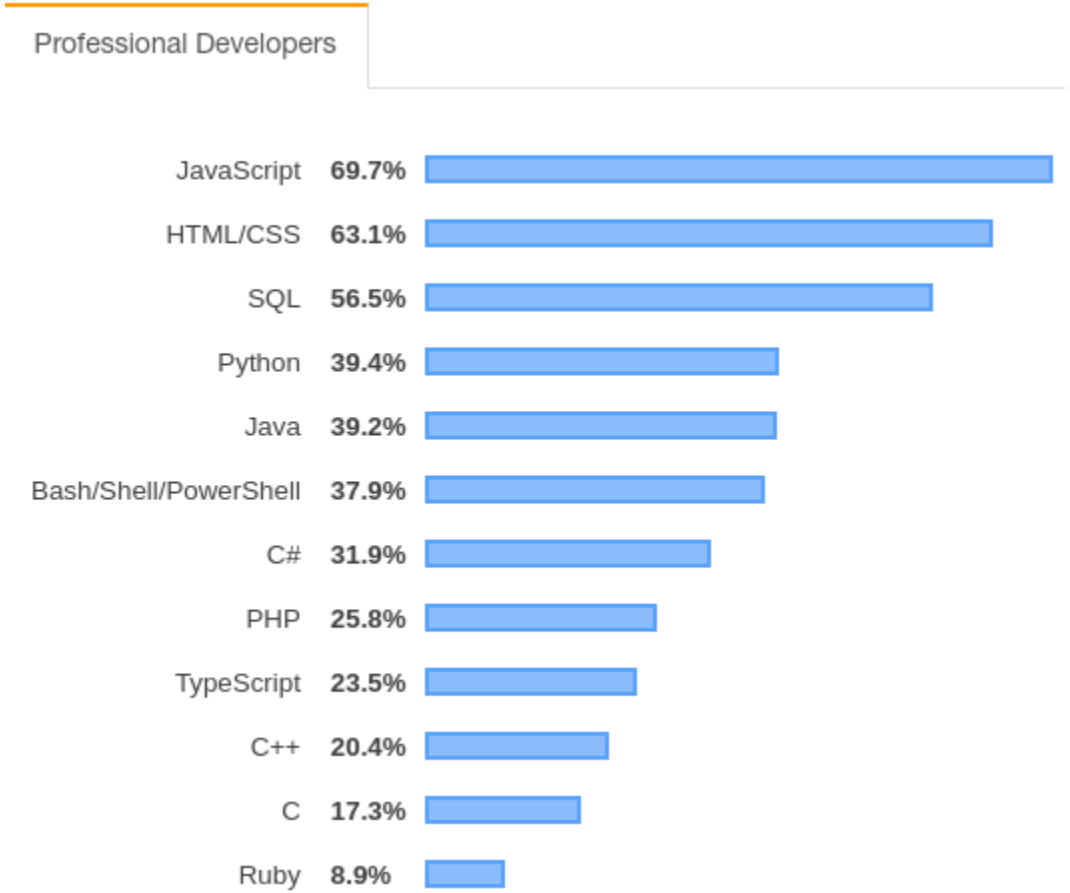
Easy(ier) to find JavaScript Talent

Developers have been using JavaScript for almost 25 years

Developers have been doing Node.js for almost 10 years

StackOverflow 2019 Survey

Almost 70% use JavaScript



Node.js is Versatile

Node.js is a JavaScript runtime that has no opinions on what it can be used for

- Serving web applications
- Create REST APIs
- Scraping websites
- Creating or editing files
- Watson API
- Running cron jobs
- Communicating with Db2
- And more!

npm Supercharges Development

Packages exist for nearly every task

Do you want to create...

...a web application?

- Express, Koa, hapi, Strapi, Sails, Restify

...a web scraper?

- Request + Zombie + Crawler

...an IoT dashboard?

- Node-red, tuyapi

- (These are just examples, no endorsement intended)

More Tools Than You Can Use

npm is the largest software repository in the world

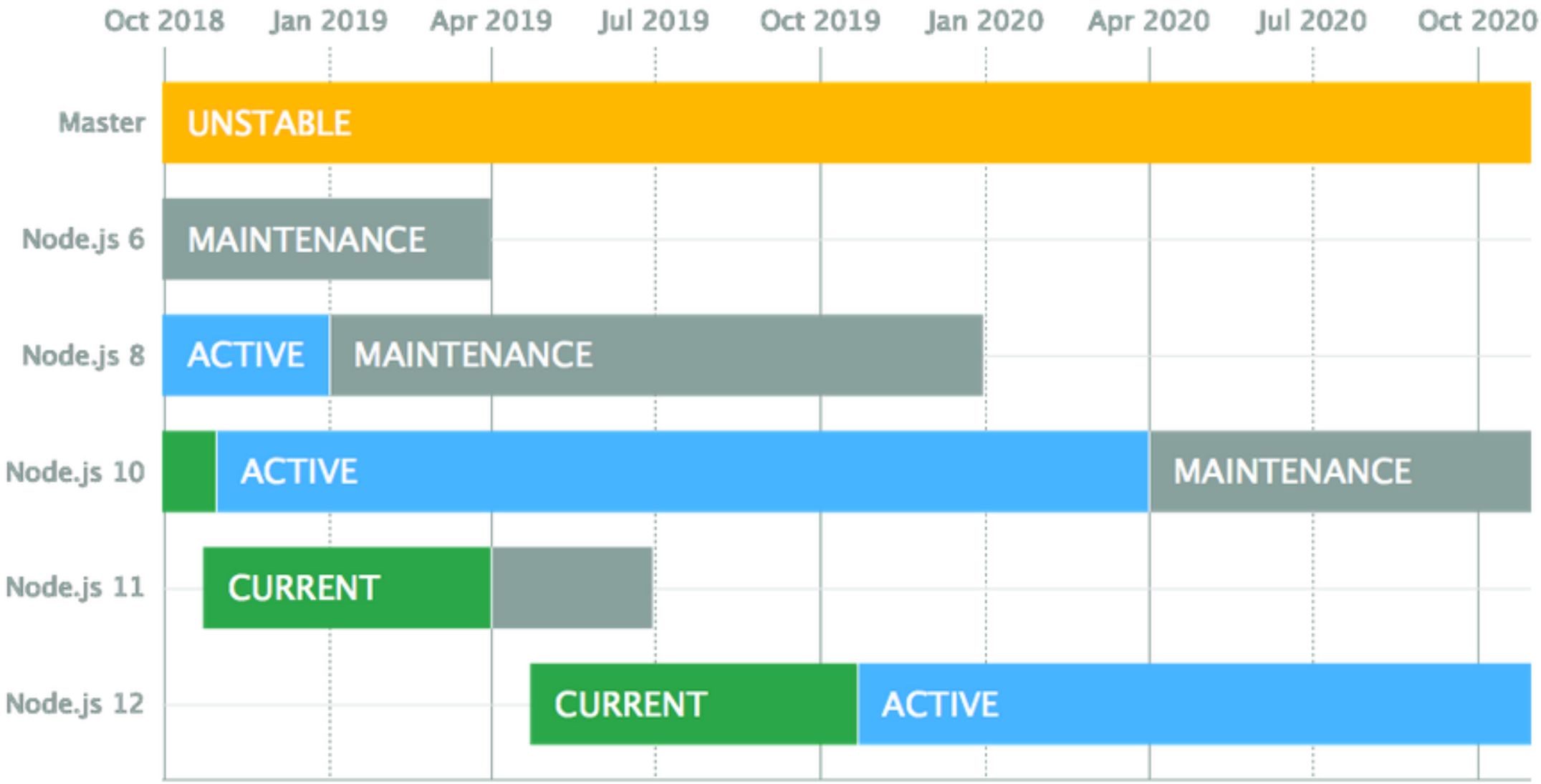
- Over one-million open-source packages available for you to use
- Many packages have been used and tested for years
  - Low-risk for vulnerabilities
  - Hundreds of tutorials for developers to follow

Packages Reduce Development Time

When developers don’t have to reinvent the wheel, they can…

- Focus on writing application logic for your business’s needs
- Get support from others who use the same package
- Be more certain that components are secure

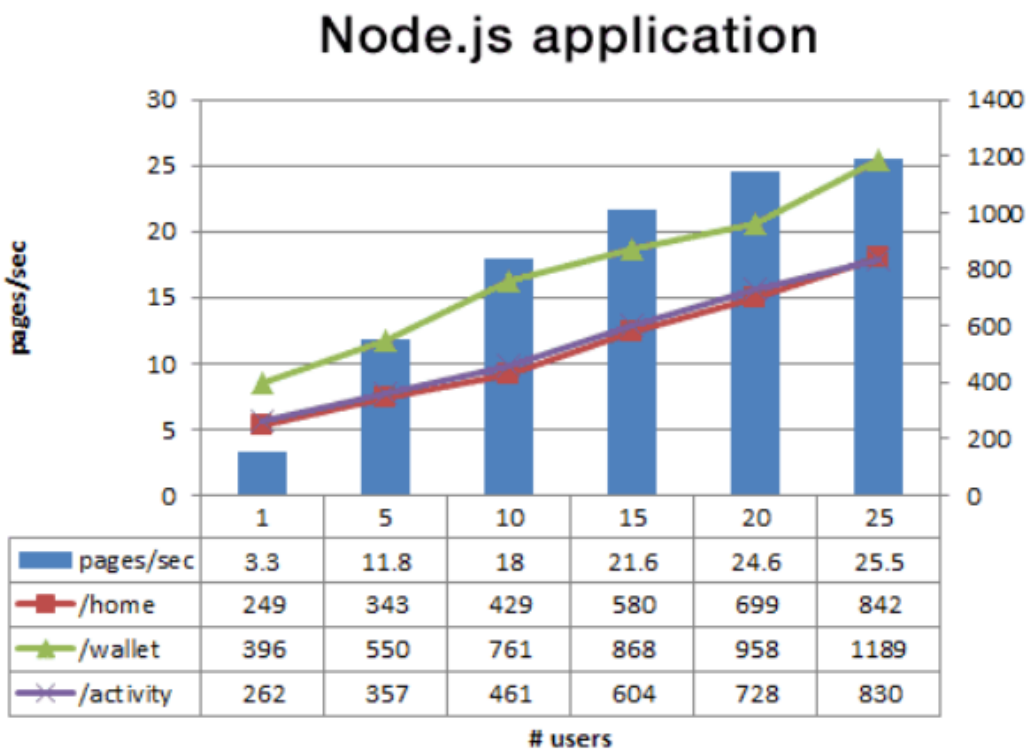
Stable Release Schedule



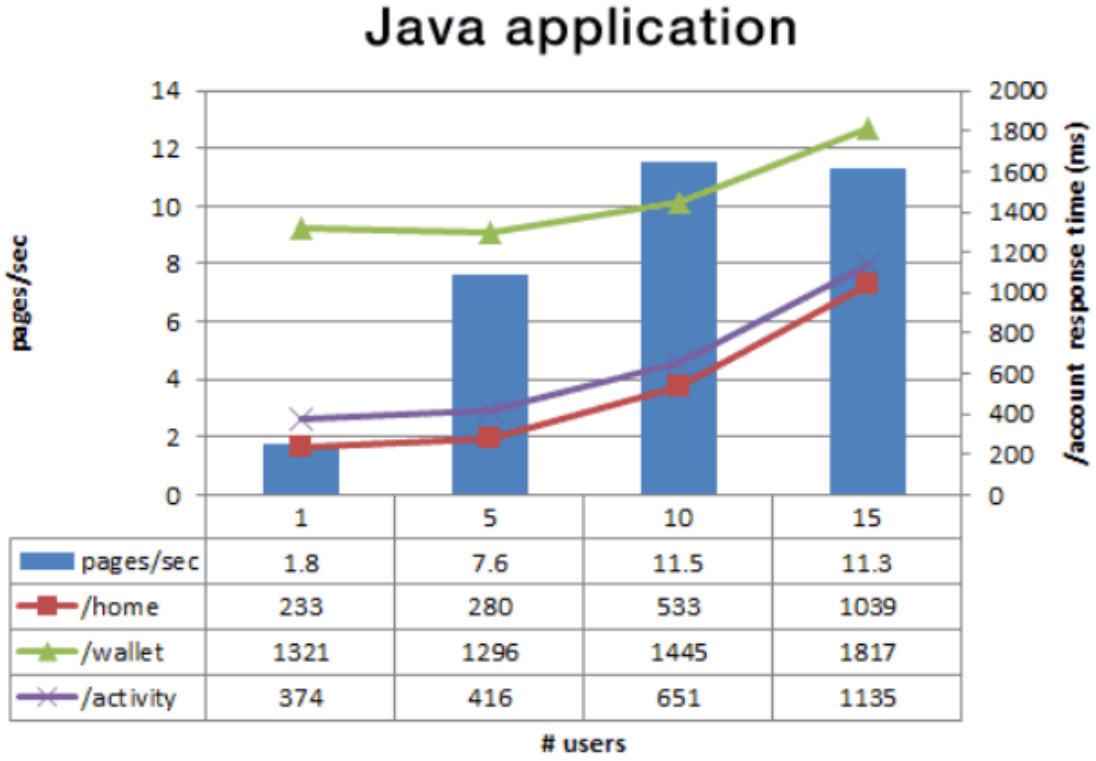
Performance Comparisons with Java

Node.js can handle large volumes of data throughput

Comparison between scaling in Node.js and Java:



	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



	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

Performance Comparisons with Java

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

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

Supported by Large Organizations

"When an organization becomes a member of the OpenJS Foundation, they are making a tangible commitment to the ongoing success and sustainability of many critical projects in the JavaScript ecosystem." https://openjsf.org/members/

Google IBM Joyent Microsoft

GoDaddy intel accenture

...Including IBM

IBM's strategy

- Keep Node.js viable for enterprise applications
- Ensure the level of security, performance, and support reaches or exceeds that of other traditional enterprise technologies

IBM Node.js Community Leadership

Participation in Technical Steering Committee

Michael
Dawson

IBM Node.js Community Leadership

9 Core Collaborators

Michael Dawson

Ben Noordhuis

Gireesh Punathil

Bethany Griggs

Yi-Hong Wang

Sam Roberts

Steven Loomis

Richard Lau

Ryan Graham

Node.js and Package Support Through IBM

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

http://ibmsystemsmag.com/blogs/open-your-i/december-2018/a-game-changer-for-open-source-support/

Proven Results

Node.js has a proven track record of delivering more value in less time

Node.js is a go-to technology for many Fortune 500 companies and governmental agencies

Don’t just take my word for it...

PayPal Success Story

Parallel development of Account page in both Java and Node.js

Node.js version was built:

- Twice as fast
- Had 33% fewer lines of code
- 40% fewer files
- With a smaller team

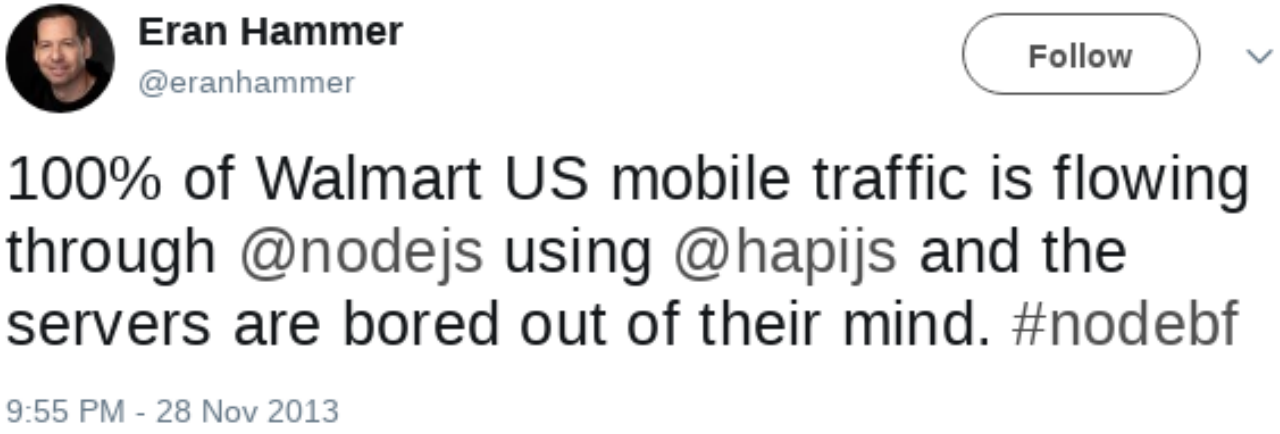
Node.js could handle double the requests-per-second, with 35% reduction in average response time

PayPal transitioned all development to Node.js

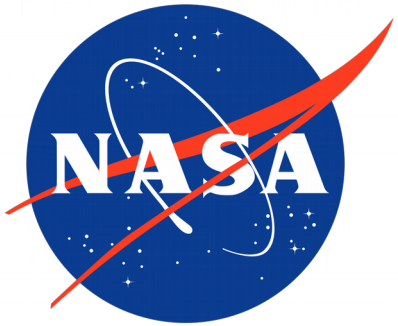
Walmart Success Story

Walmart saw the potential for Node.js so developed Hapi.js, an open-source web framework

In 2013, Hapi.js was used for the walmart.com mobile website during Black Friday...



Many Success Stories



IBM i and Node.js

We don’t want you to rewrite all of your C, C++, RPG, COBOL, or CL code in JavaScript

Instead, you should use Node.js enhance and extend your applications

Many ways to connect to your IBM i:

- itoolkit

  All in one solution for calling PGMs, CL commands, interact with LIBs, call Db2, and more!

- idb-ṗconnector

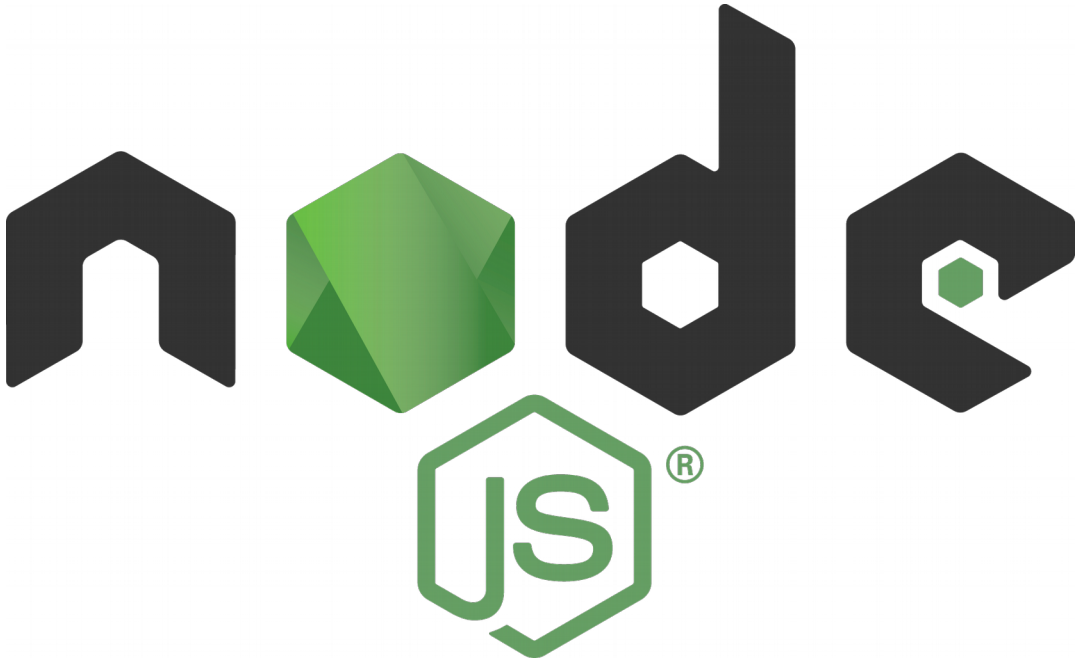
  Connecting to Db2 from on your IBM i system using CLI

- odbc

  Connecting to Db2 from IBM i, Windows, or Linux using ODBC

Summary

- JavaScript makes development fast and easy by its interpreted, weakly-typed nature
- Massive pool of talent makes it easy to find Node.js developers
- Great performance and scalability compared to other technologies like Java, Ruby on Rails, or PHP
- Mature technology that has been available for nearly a decade means no surprises
- Over one million packages on npm is evidence of a robust ecosystem, wherein you can find the right tools for any job (or make your own)
- Support from large corporations who have a vested interest keeping Node.js relevant and modern
- Proven value through many case-studies at large corporations
- Tools to integrate Node.js directly with your existing IBM i business applications
- Developer satisfaction with Node.js is much higher than other technologies. Programming with Node.js is (more) fun!



What? Why? **How?**

Installing the Open-Source Environment

A little outside the scope of this presentation, but…

Node.js is delivered by the IBM i OSS team through the yum package manager (no more PTFs!)

- To get the open-source environment, visit http://ibm.biz/ibmi-rpms

Installing Node.js

On your Windows or Mac machine, you can download the installation package at https://nodejs.org/en/download

On your Linux (or IBM i) machine, you should use your package manager

IBM i:

```
yum install nodejs12
```

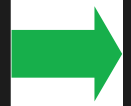
This will install node and npm

- `node` will be used to run our appliaction
- `npm` is used to download packages and keep track of metadata

`npm init` and `package.json`

Create a directory for your application and use `npm init` to create a `package.json` file, which holds application metadata

```
$ mkdir norge-app
$ cd norge-app
$ npm init
Press ^C at any time to quit.
package name: (norge-app)
version: (1.0.0)
description: Our example app
entry point: (index.js)
test command:
git repository:
keywords:
author: Mark Irish
license: (ISC)
…
Is this OK? (yes)
```



```
{
  "name": "norge-app",
  "version": "1.0.0",
  "description": "Our example app",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit
1"
  },
  "author": "Mark Irish",
  "license": "ISC"
}
```

`norge-app/package.json`

`package.json`

`package.json` is confusing at first, but its actually really simple

Your application can be recreated with just `package.json` and the source code

When you download dependencies through npm, **you also include your dependencies' dependencies** (and so on).

- These are all placed in a directory called `node-modules`
- This directory can be **magnitudes larger than your source code**

You don't need to transfer or track `node_modules`, as npm will read `package.json` and install all the required files from npm when you run

```
$ npm install
```

Let’s See It In Action

```
$ cat package.json
{
  "name": "bookstore",
  "version": "1.0.0",
  "description": "Restful API with authentication
using passport.js",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit
1"
  },
  "author": "",
  "license": "MIT",
  "dependencies": {
    "body-parser": "^1.18.3",
    "connect-flash": "^0.1.1",
    "ejs": "^2.6.1",
    "express": "^4.16.3",
    "express-session": "^1.15.6",
    "idb-pconnector": "^0.1.1",
    "itoolkit": "^0.1.3",
    "passport": "^0.4.0",
    "passport-local": "^1.0.0"
  }
}
```

Express.js example (express_books)

Without dependencies downloaded, size of express_books directory is 340 kB. That is source, package.json, and other smaller files.

package.json lists 9 dependencies

Downloading Dependencies

```
$ npm install
added 132 packages from 96 contributors and audited
269 packages in 27.359s
found 0 vulnerabilities
$ du -k
...
17948   ./node_modules
18324   .
```

269 packages? But we only had 9 as dependencies!

Remember, your dependencies have dependencies

express_books directory now has 18324 kB, 53x larger than before

All dependencies are in node_modules directory, where Node.js will look for them

A Simple Application

You have all the tools you need to create a Node.js application

Let’s make a Hello World Express.js web application

(Example is not IBM i specific so that you can run it directly on your machines today)

Set Up Application Directory

1. Create a directory called norge-app

```
$ mkdir norge-app
```

2. Navigate to that directory and run

```
$ npm init -y
```

3. Install Express.js (a popular web framework for Node.js)

```
$ npm install express
```

Using Downloaded Packages

In JavaScript files, you use the require module

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

require will resolve the package name you pass in, and give you that package's main exported object

You can also 'require' an absolute path if you want to import objects that aren't in your `node_modules` folder

Creating a JavaScript File

Create a file called `app.js` with the following code:

```
const express = require('express');
const app = express();
const port = 3000;

app.get('/', (req, res) => res.send('Hello World!'));

app.listen(port, () => console.log(`App running`));
```

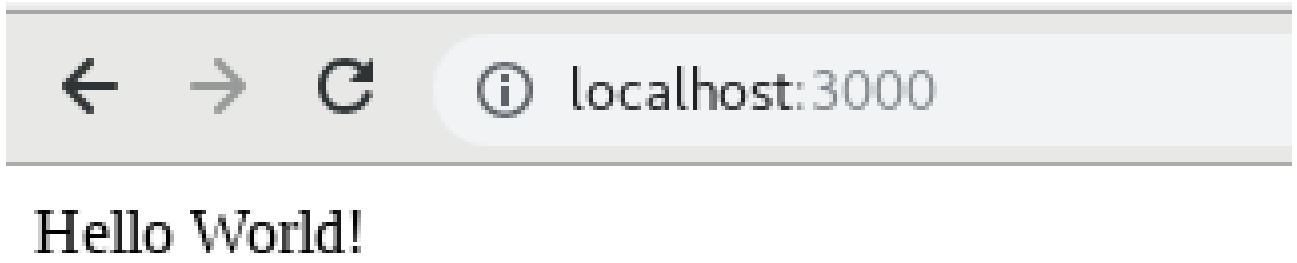
Run Your Program

From the norge-app directory, run

$ node app.js

Your terminal should display ‘App running’. Your application will run until you kill it (CTRL + c)

Go to your browser and type ‘localhost:3000’ in the address bar



Your First Application

Underwhelmed?

- Web server with 5 lines of code (plus the use of Express.js)
- Took about 20 seconds to make
- Runs faster than PHP or Java serving the same page

You used your knowledge!

- `$ npm init`
- `$ npm install express`
- `require('express')`
- `node app.js`

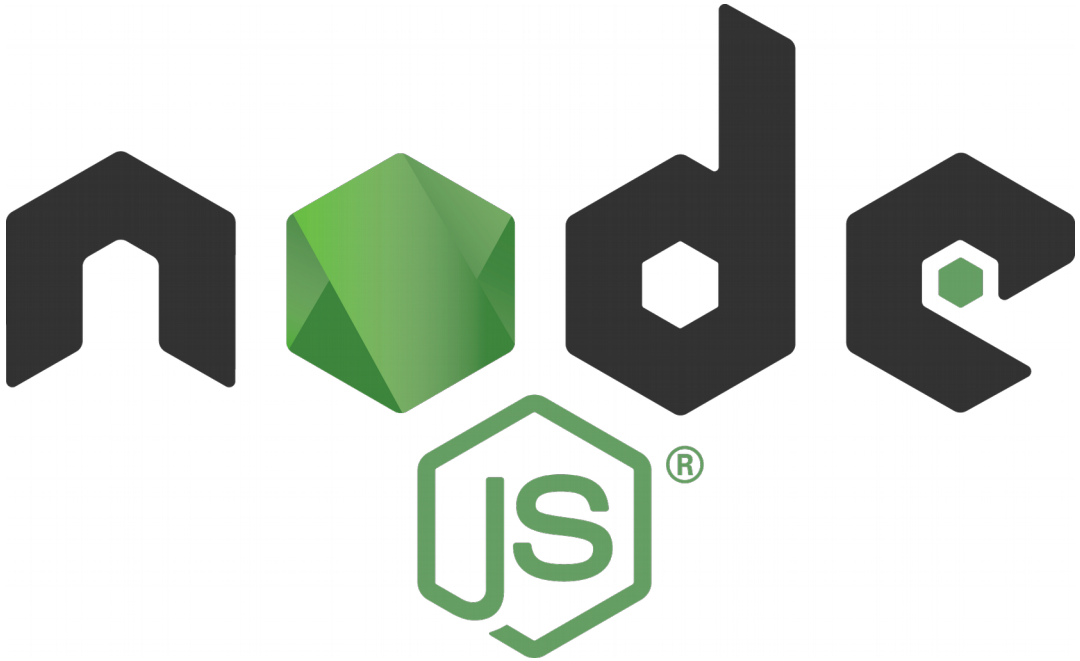
Best Way to Learn is to Fail

Node.js is simple, but like everything it can seem confusing at first

If you want to learn to use Node.js...

...you should start to use Node.js

You will make mistakes, but you will quickly see why Node.js is so popular



The End