NODE PROGRAM

INTRODUCTION

NODE.JS VERSION: 5.1
LAST UPDATED: JAN 2016

BEFORE WE START...

YOU'LL NEED:

- NODE.JS AND NPM
- CODE EDITOR
- COMMAND LINE
- INTERNET CONNECTION
- SLIDES & SAMPLE CODE

SLIDES AND EVERYTHING ELSE

HTTPS://GITHUB.COM/AZAT-CO/NODE-REACT

```
git clone git@github.com:azat-co/node-react.git
```

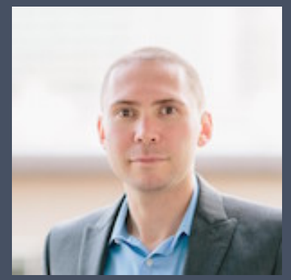
INSTALLING NODE.JS

- HTTP://NODEJS.ORG
- `$ brew install node`
- NODE IN 30S

YOU MAY ALSO WANT/NEED A LOCAL DATA STORE!

- MONGODB
- MYSQL
- POSTGRESQL

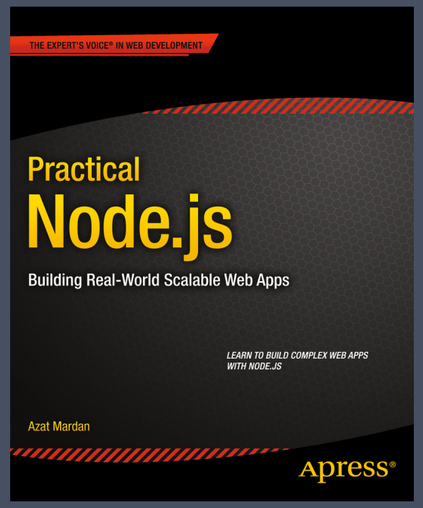
INTRODUCTIONS

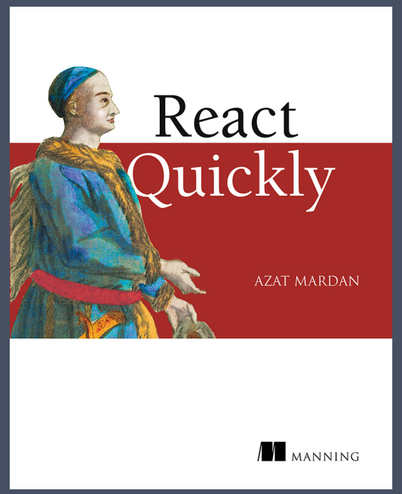


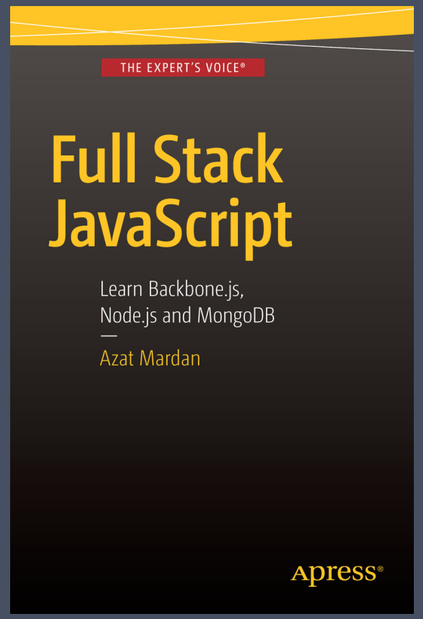
INSTRUCTOR: AZAT MARDAN

- WORK: CAPITAL ONE. STORIFY. FDIC. NIH. DOCUSIGN
- BOOKS: REACTQUICKLY. FULL STACK JAVASCRIPT. PRACTICAL NODE.JS. PRO EXPRESS.JS. MONGOOSE COURSE


THE EXPERT'S VOICE® IN WEB DEVELOPMENT
Pro
Express.js
MASTER EXPRESS.JS FOR YOUR DEVELOPMENT
Azat Mardan
Apress®


THE EXPERT'S VOICE® IN WEB DEVELOPMENT
Practical
Node.js
Building Real-World Scalable Web Apps
LEARN TO BUILD COMPLEX WEB APPS WITH NODE.JS
Azat Mardan
Apress®


React
Quickly
AZAT MARDAN
MANNING


THE EXPERT'S VOICE®
Full Stack
JavaScript
Learn Backbone.js,
Node.js and MongoDB
Azat Mardan
Apress®

INTRODUCE YOURSELF

1. WHAT IS YOUR TECH BACKGROUND/LANGUAGE?
2. WHAT IS YOUR PROJECT?
3. HOW DO YOU PLAN TO USE NODE.JS?

OUTCOME

- BUILD SERVER-SIDE WEB APPLICATIONS WITH THE NODE.JS PLATFORM UTILIZING THE JAVASCRIPT LANGUAGE
- USE NODE.JS FRAMEWORK EXPRESS.JS
- USE NOSQL DATABASE MONGODB
- GET FAMILIAR WITH METEOR
- GRASP REACT AND ISOMORPHIC JAVASCRIPT

HIPCHAT ROOM

HTTPS://WWW.HIPCHAT.COM/GI1LG5C2Q

INTRODUCTION

WHY SERVER-SIDE JAVASCRIPT?

NODE WAS ORIGINALLY BORN OUT OF THIS PROBLEM – HOW CAN YOU HANDLE TWO THINGS AT THE SAME TIME

– RYAN DAHL, THE CREATOR OF NODE.JS

WHY SERVER-SIDE JAVASCRIPT?

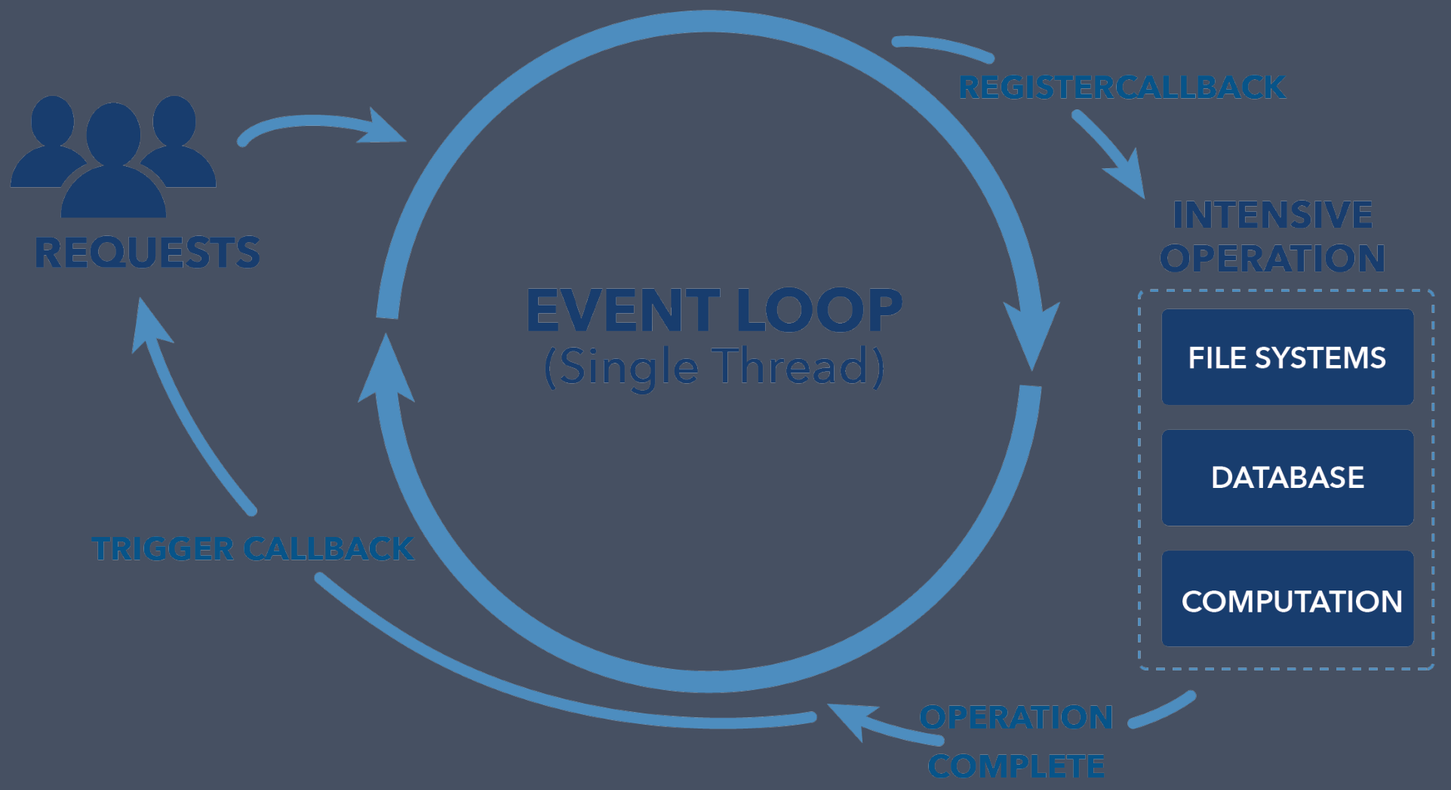
- NON-BLOCKING I/O: PERFORMANT
- FAST: BROWSER ARMS RACE (V8)
- ONE LANGUAGE ACROSS THE STACK
- EXPRESSIVE: DON'T WASTE TIME ON SETUP
- SOLID STANDARD (ECMA)

ADVANTAGES OF NODE.JS

- NON-BLOCKING I/O
- SUPER FAST (V8)
- VIBRANT ECOSYSTEM (NPM)
- ABILITY TO RE-USE CODE ON BROWSER AND SERVER
- ABILITY TO USE FRONT-END DEVS FOR BACK-END AND VICE VERSA

NON-BLOCKING I/O

IT'S KIND OF A BIG DEAL



DISADVANTAGES OF NODE.JS

- DEVS HAVE TO THINK IN ASYNC AND FUNCTIONAL+PROTOTYPAL
  - FRAMEWORKS AND TOOLS ARE NOT AS MATURE AS IN RUBY, JAVA, PYTHON (YET)
    - JAVASCRIPT "QUIRKS" (MOSTLY FIXED IN ES6!)

NODE GOTCHA

DON'T USE NODE.JS FOR CPU-INTENSIVE TASKS. HAND THEM OVER TO OTHER WORKERS.

DOWNSIDES OF JAVASCRIPT (NOT ONLY NODE)

- CALLBACK HELL
- PROTOTYPAL INHERITANCE

JAVASCRIPT IS OPTIONAL IN NODE.JS

IT'S POSSIBLE TO USE OTHER LANGUAGES FOR NODE.JS THAT COMPILE INTO JAVASCRIPT. E.G., COFFEESCRIPT. TYPESCRIPT. AND CLOSURESCRIPT.

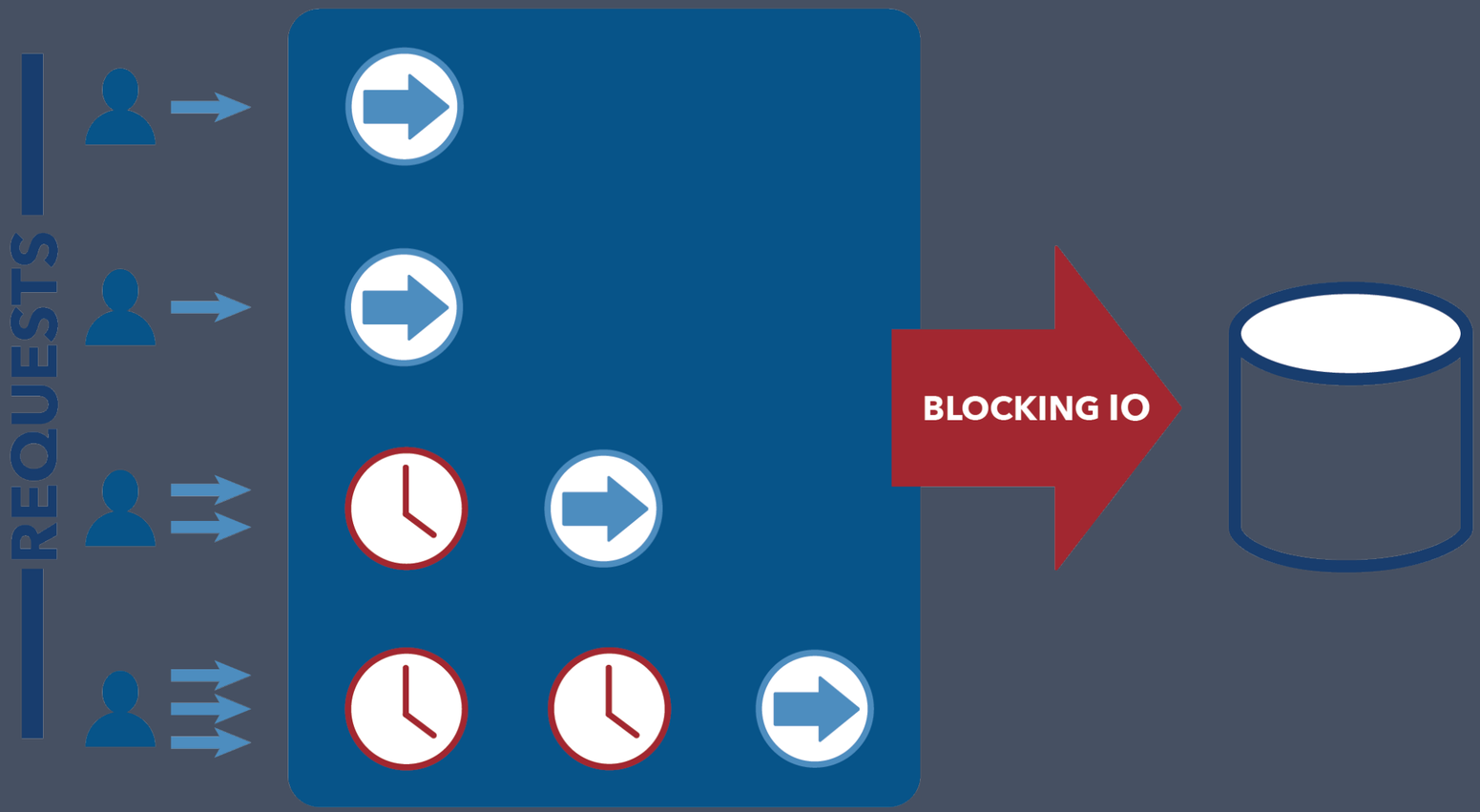
Nodies are not just Silicon Valley hipsters !

NODE IS SINGLE-THREADED

NODE.JS IS SINGLE-THREADED BY DESIGN TO MAKE ASYNCHRONOUS PROCESSING SIMPLER. MULTI-THREADING CAN BE VERY COMPLEX: RACING CONDITION, DEADLOCKS, PRIORITY INVERSIONS...

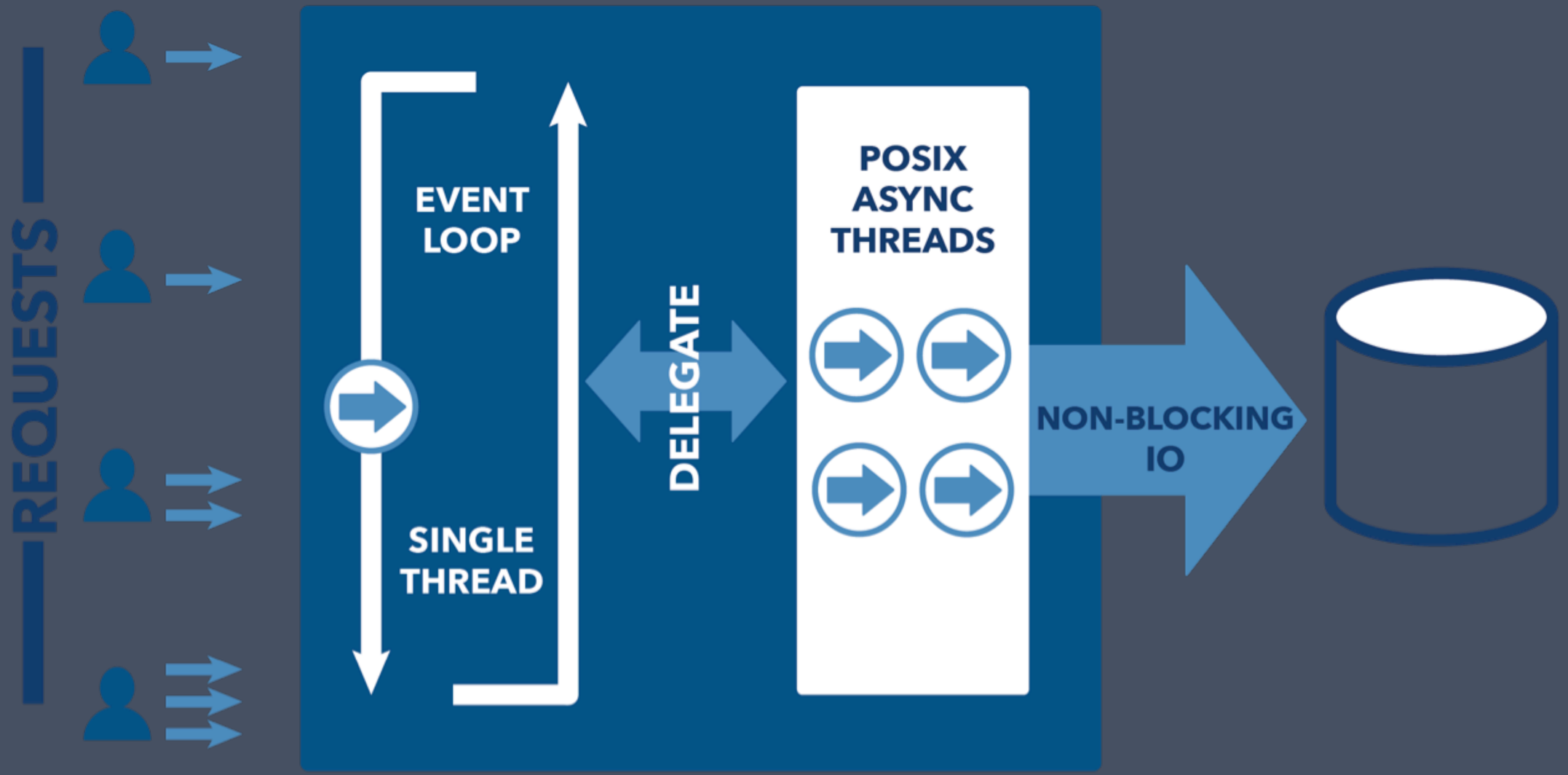
IT TURNED OUT FOR WEB-BASED APPLICATION, SINGLE-THREADED ASYNCHRONOUS EVENT-LOOP BASED NON-BLOCKING I/O IS VERY PERFORMANT!

MULTI THREADED SERVER



THREAD PROCESSING

THREAD WAITING


REQUESTS
EVENT LOOP
SINGLE THREAD
DELEGATE
POSIX ASYNC THREADS
NON-BLOCKING IO

SCALING NODE VERTICALLY

TO SCALE NODE VERTICALLY, YOU CAN TAKE ADVANTAGE OF MULTIPLE CPUS CORES OR COMPUTE UNITS (MULTI-THREADING) WITH CLUSTERING (E.G., STRONGLOOP'S PM).

THE IDEA IS TO HAVE MULTIPLE PROCESSES FROM THE SAME CODE BASE TO LISTEN ON THE SAME PORT FOR REQUESTS.

INTEGRATION

- NOSQL
- SQL
- OAUTH 1.0/2.0
- REST
- SOAP

DATABASES

- MYSQL
- POSTGRESQL
- ORACLE
- MS SQL
- MONGODB
- CASSANDRA

NODE + CLIENT MVC ARCHITECTURE

SINGLE-PAGE APPLICATIONS A.K.A. BYOC: REST API IN NODE + SPA

- BACKBONE
- ANGULAR (E.G., M.E.A.N)
- EMBER
- REACT
- MV*

SERVER-SIDE RENDERING

- JADE
- HANDLEBARS
- EJS
- HOGAN

MANY MORE: HTTP://GARANN.GITHUB.IO/TEMPLATE-CHOOSER

NODE FOR SOA / REST

SO WHAT IS ECMASCRIPT?

ES AS A LANGUAGE SPECIFICATION

- BROWSER IMPLEMENTATIONS (LIKE CHROME'S V8)
  - NODE BUILDS ON V8 WITH C++

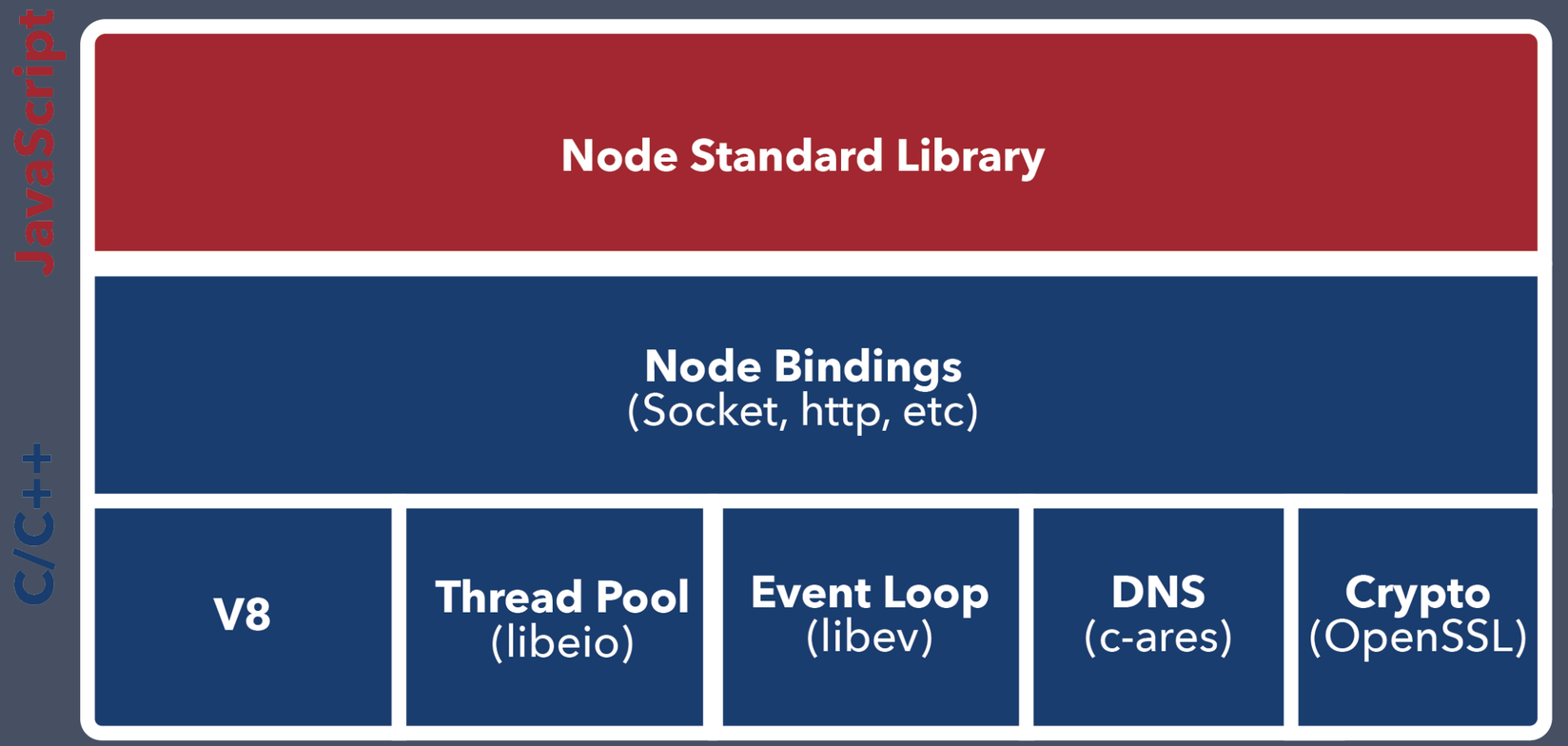
BROWSER JS != NODE

- MODULES
- SCOPES
- WINDOW VS. GLOBAL AND PROCESS
- `fs` AND OTHER MODULES

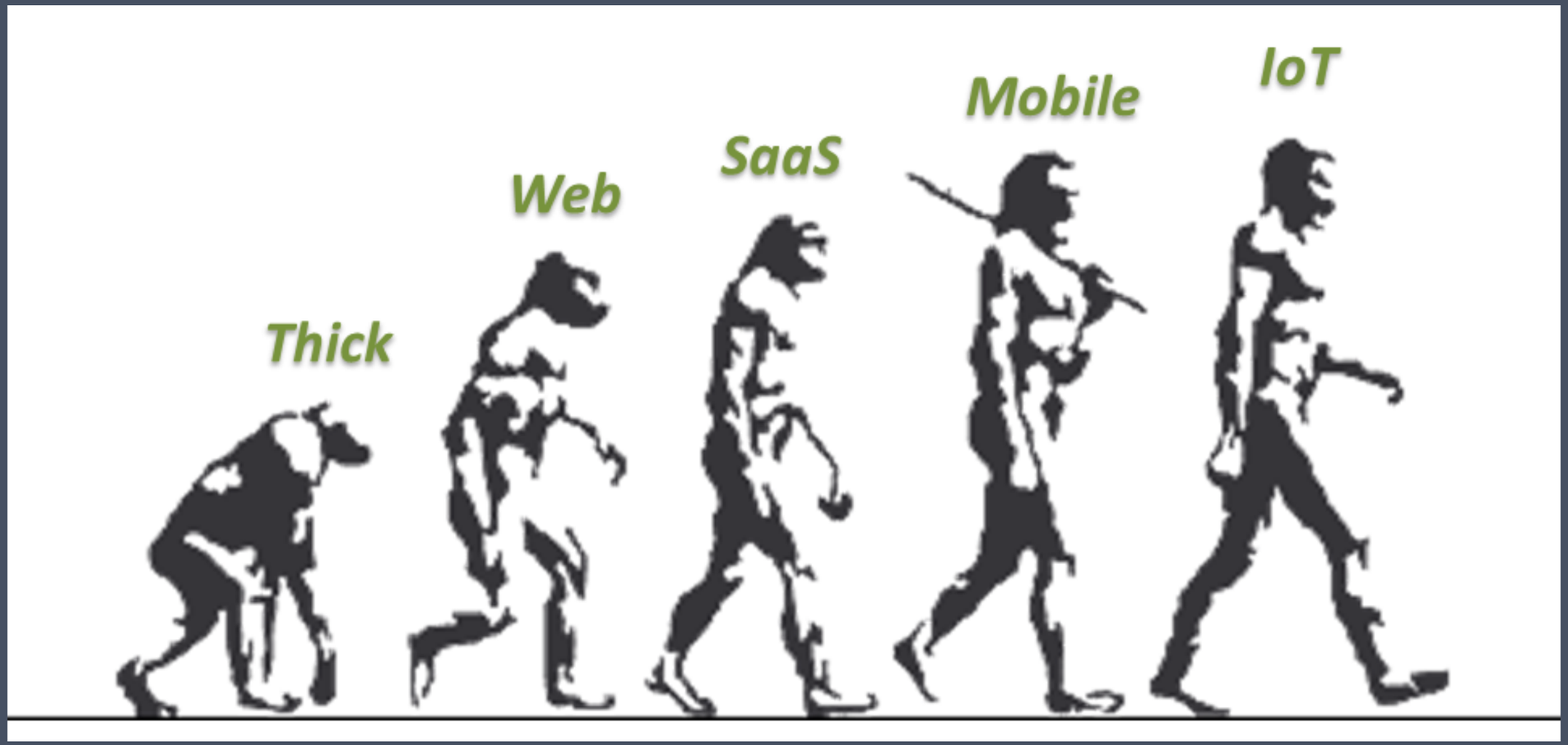
NODE CORE: V8, LIBEV, AND LIBEIO

- LIBEV: THE EVENT LOOP
- LIBEIO: ASYNC I/O
- LIBUV: ABSTRACTION ON LIBEIO, LIBEV, C-ARES (FOR DNS) & IOCP (FOR WINDOWS)

NODE CORE ARCHITECTURE



PATTERNS EVOLVE TO SERVE MARKET NEEDS



FRAMEWORK CATEGORIES

- KISS SERVERS: SMALL CORE, SMALL MODULES
- CONVENTION: FOLLOW THE LEADER, STEEP LEARNING CURVE
- CONFIGURATION: OPEN PATH, MANUAL EFFORT FOR ADVANCED
- ORM & ISOMORPHIC: MODEL-DRIVEN, SHARED CODE, STEEP LEARNING

FRAMEWORK EXAMPLES

- KISS SERVERS: NODE CORE
- CONVENTION: EXPRESS. RESTIFY. TOTAL.JS
- CONFIGURATION: HAPI. KRAKEN
- ORM & ISOMORPHIC: LOOPBACK. SAILS. METEOR*

NODE PROGRAM

EFFECTIVE LEARNING

50% WORKSHOPS +

50% LECTURES +

50% Q&A/OFFICE HOURS

(YES. WE DELIVER 150%!)

WORKSHOPS = CODING + COLLABORATION + PAIR PROGRAMMING + SOLO PROGRAMMING + DISCUSSIONS + READING + SOLVING PROBLEMS (✋ IF STUCK)

AGENDA

NODE.JS DAY:

- 9-11:00: LECTURES: INTRO, SETUP AND NODE.JS BASICS
- 11:00-12:00: WORKSHOP
- 12:00-1:00: LUNCH

AGENDA

NODE.JS DAY:

- 1:00-2:00 LECTURES: MONGODB, EXPRESS
- 2:00-3:00: WORKSHOP
- 3:00-3:15 BREAK
- 3:15-4 LECTURES: METEOR
- 4-5 WORKSHOP

AGENDA

REACT DAY:

- 9-11: LECTURES
- 11-12: WORKSHOP
- 12-1: LUNCH

AGENDA

REACT DAY:

- 1-2: LECTURES
- 2-3: WORKSHOP
- 3-5: OFFICE HOURS AND INDIVIDUAL TRACKS

INDIVIDUAL TRACKS

1. DEPLOYMENT
2. SINGLE-PAGE APPLICATION
3. REST API
4. YOUR OWN PROJECT/IDEA

QUESTIONS AND EXERCISES

WRITE THEM DOWN AND ASK AT THE END OF THE LESSON: YOU'LL HAVE 5 OPEN FRAMES TO ASK QUESTIONS. USE THEM FULLY!

NO WORKSHOP FOR THIS LESSON. 😁