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

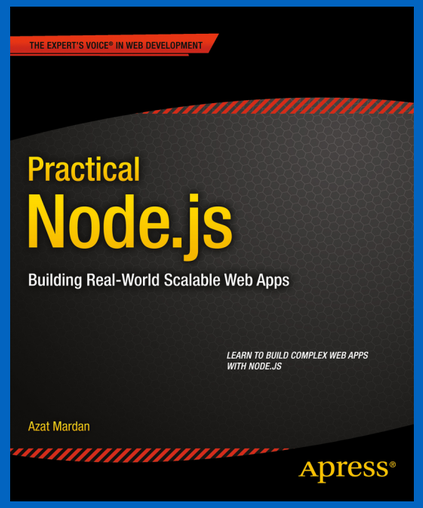
You may also want/need a local data store!

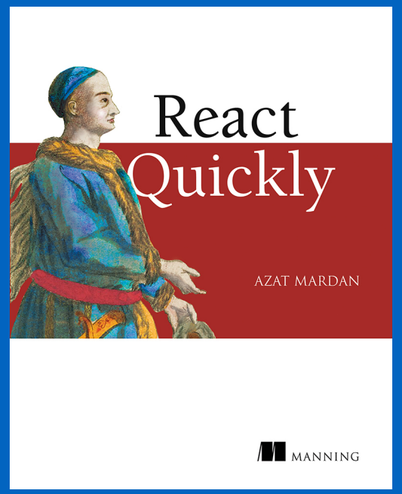
- MongoDB
- MySQL
- Postgresql

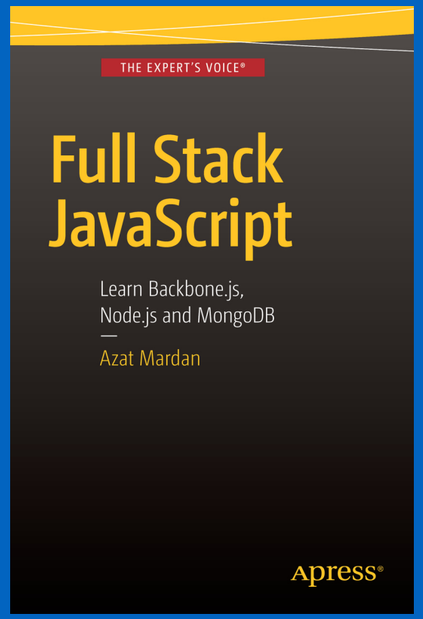
Introductions

Instructor: Azat Mardan

- Work: Capital One, Storify, FDIC, NIH, DocuSign
- Books: ReactQuickly, Full Stack JavaScript, Practical Node.js, Pro Express.js, Mongoose Course







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?

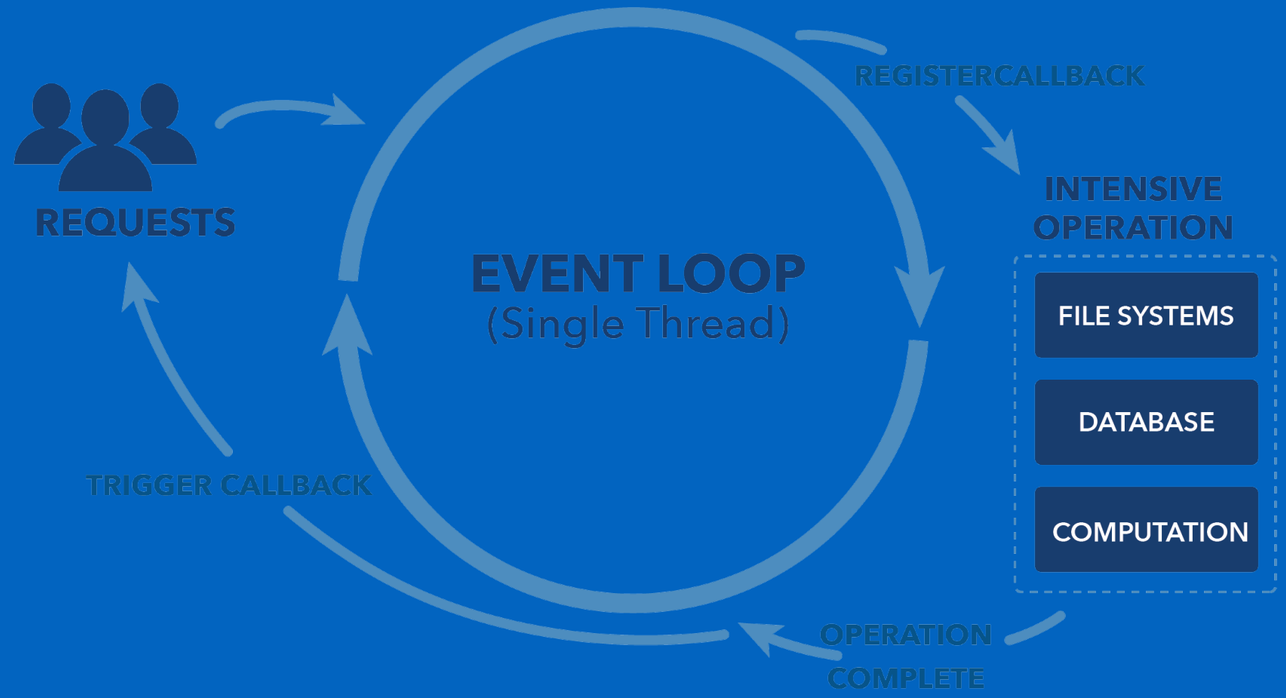
- Non-blocking I/O: performant
- Fast: browser arms race (V8)
- **One language across the stack**
- Expressive: don't waste time on setup
- Solid standard (ECMA)

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.

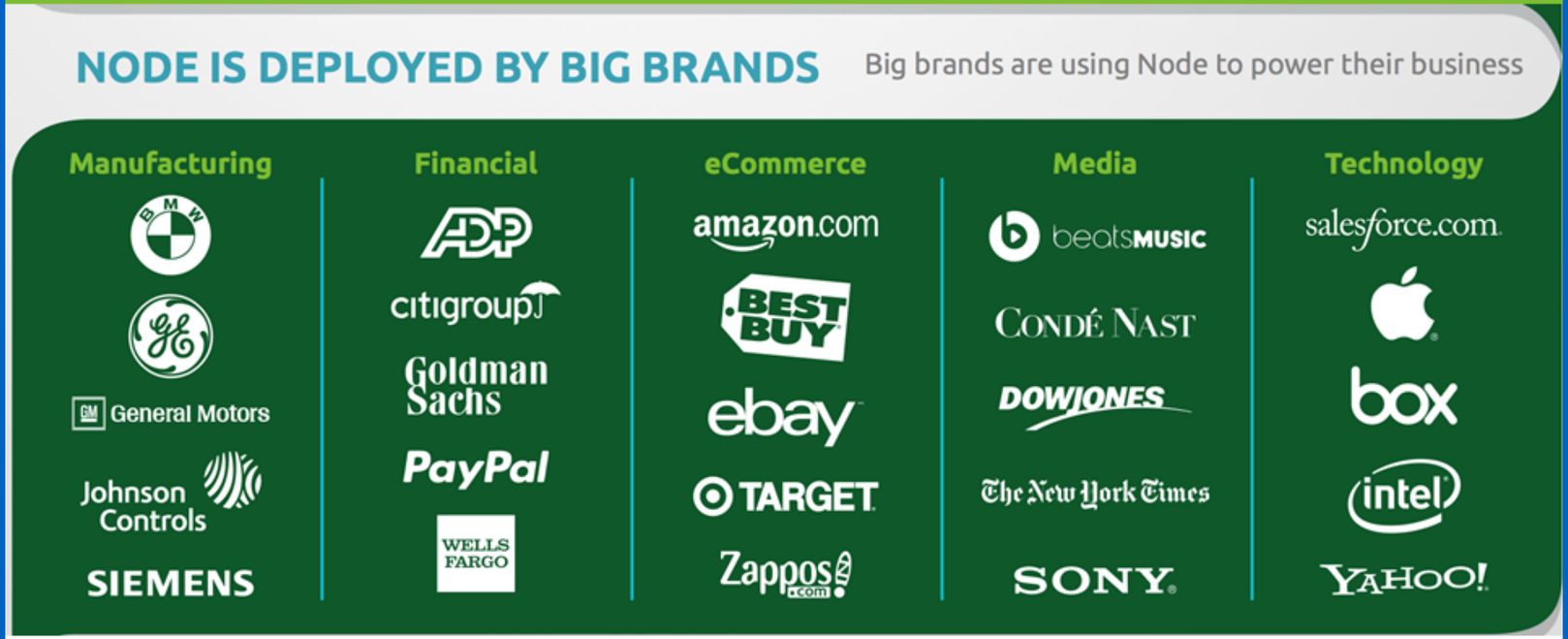
Downsides of JavaScript (Not only Node)

- 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 !

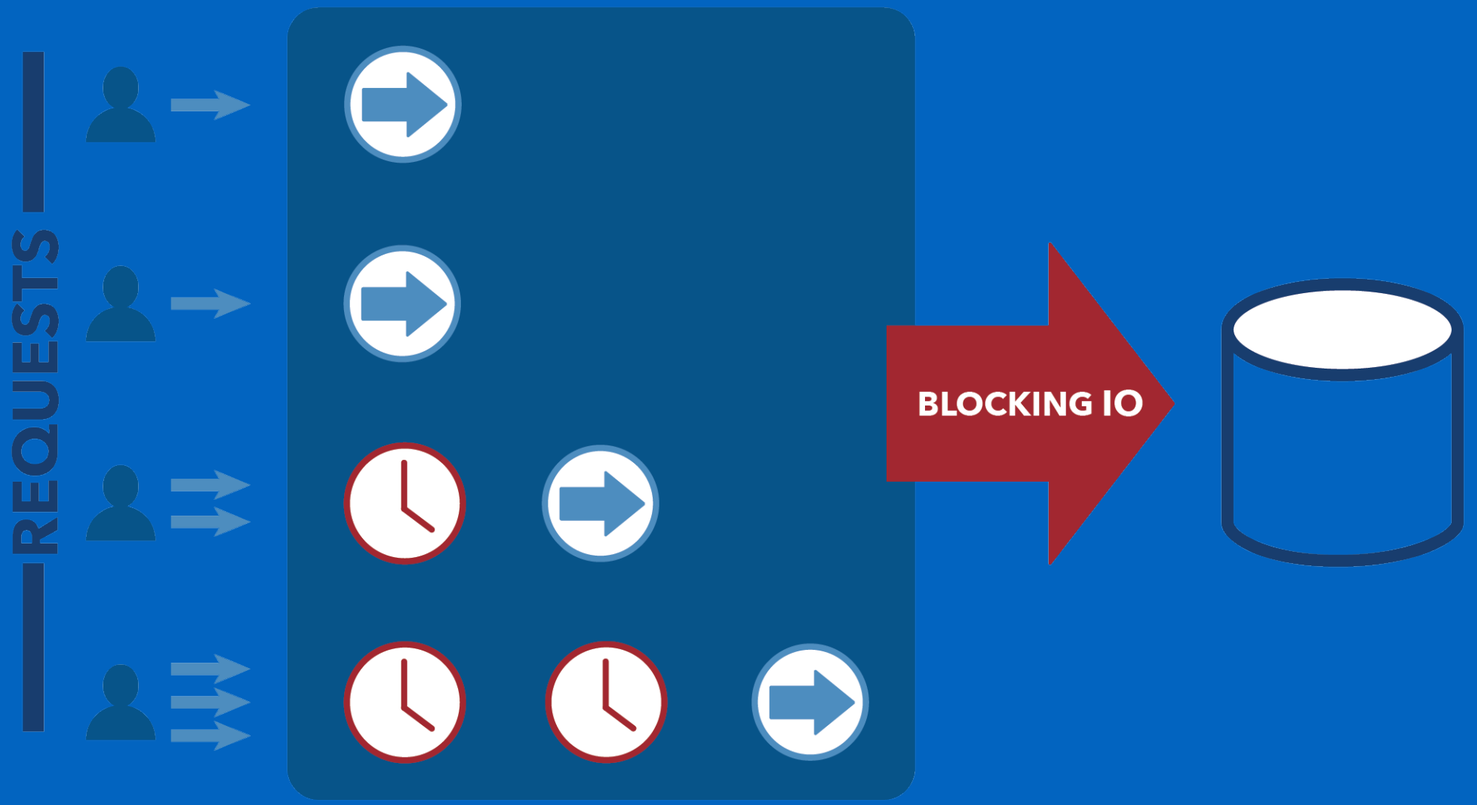


Node is Single-Threaded

Node.js is single-threaded by design to make asynchronous processing simpler. Multi-threading can be very complex: racing condition, deadlocks, priority inversions...

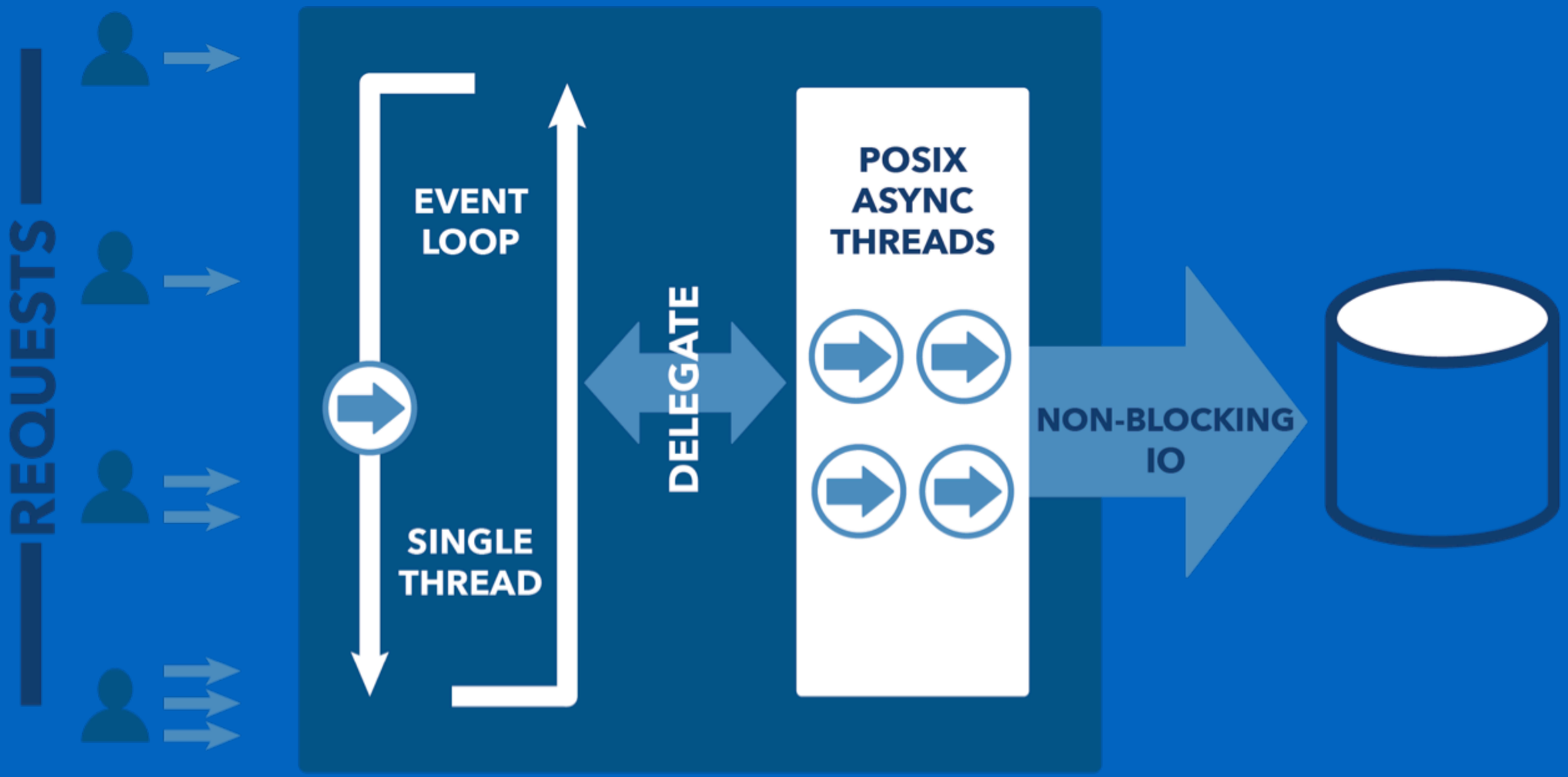
It turned out for web-based application, single-threaded asynchronous event-loop based non-blocking I/O is very performant!

MULTI THREADED SERVER



THREAD PROCESSING

THREAD WAITING



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

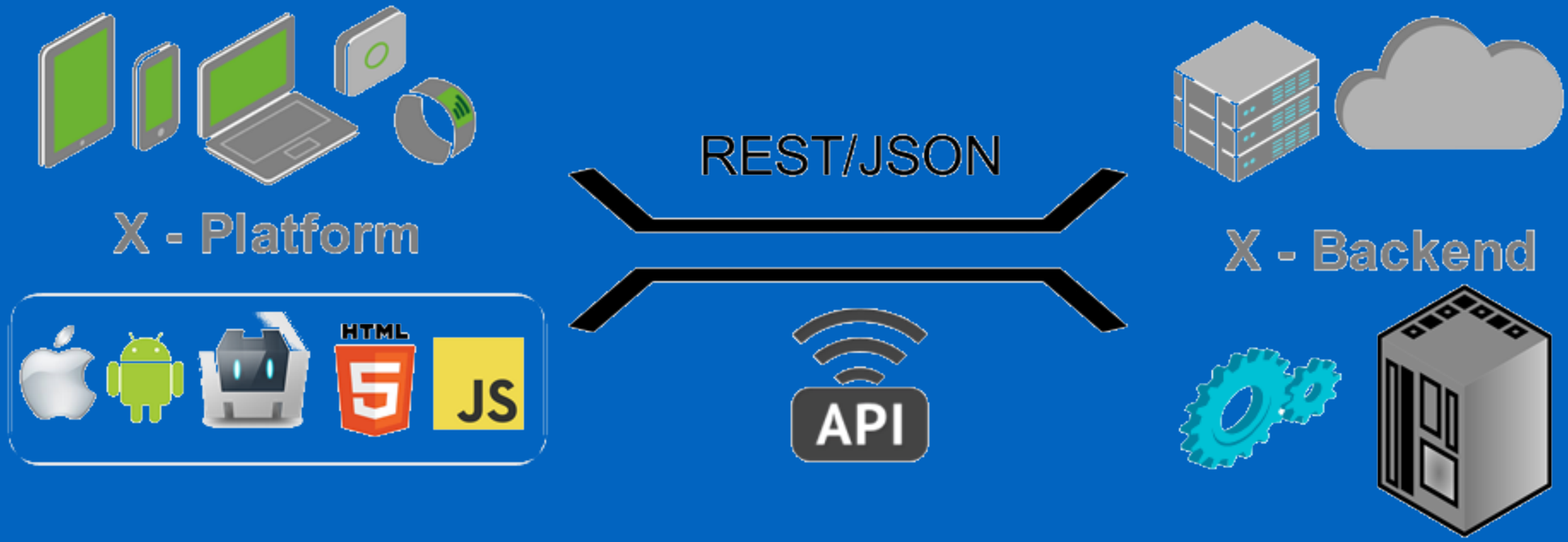
- Backbone
- Angular (e.g., M.E.A.N)
- Ember
- React
- MV*

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++

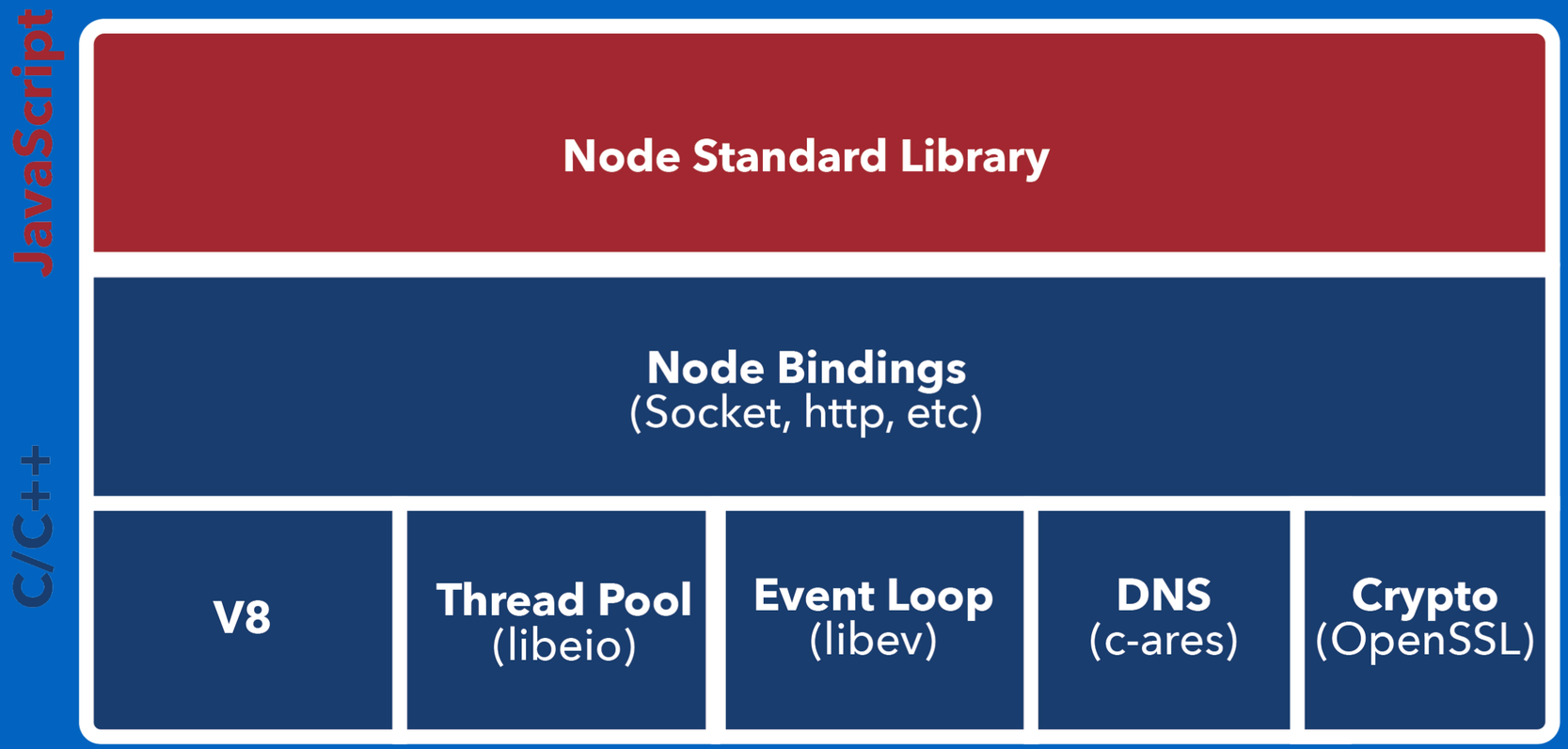
Browser JS != Node

- Modules
- Scopes
- window vs. global and process
- `fs` and other modules

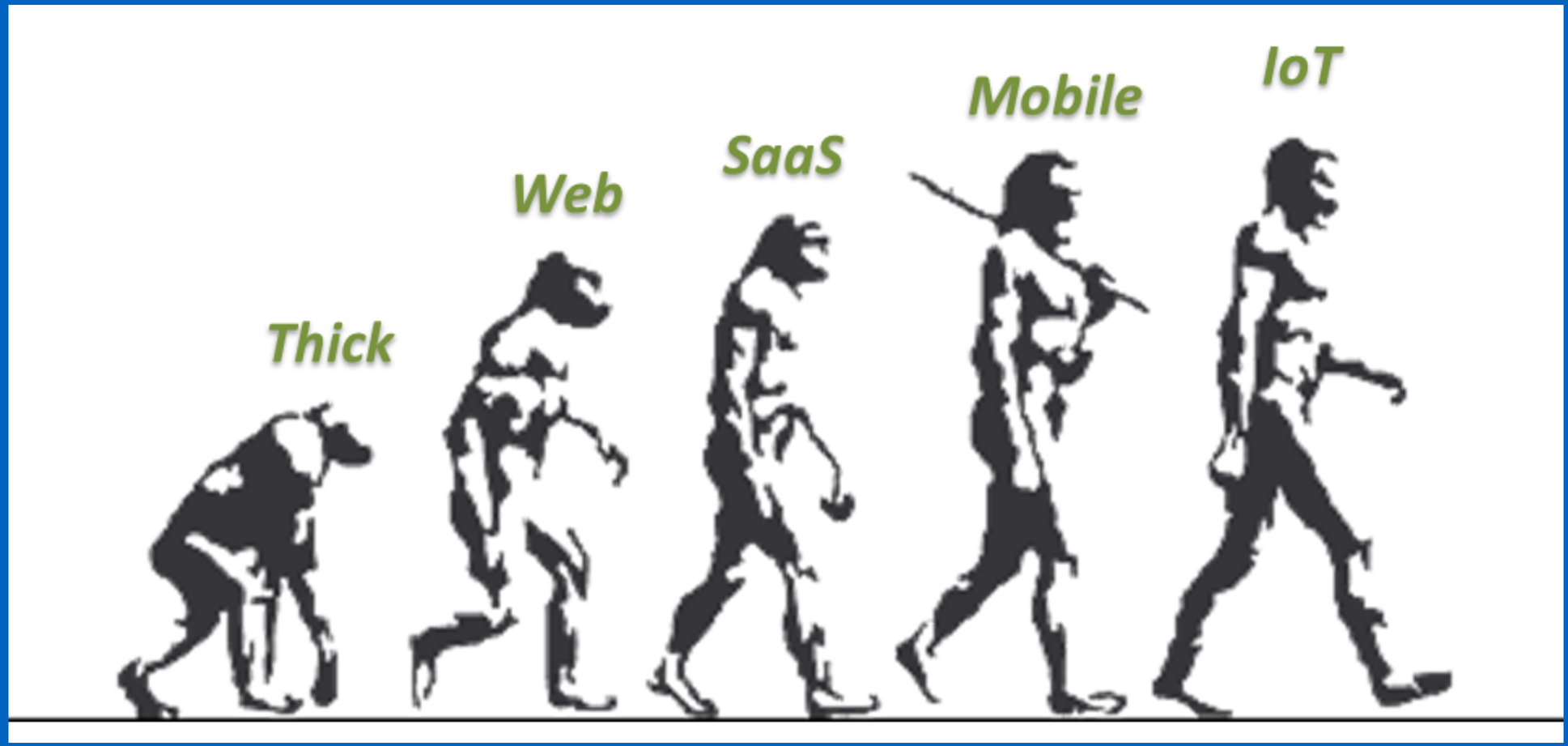
Node Core: V8, libev, and libeio

- 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. 😁