Benjamin San Souci & Maude Lemaire

INTRODUCTION

What is Node.js anyway?

- a complete software platform for scalable server-side and networking applications
- open-source under the MIT license
- comes bundled with a JavaScript interpreter
- runs on Linux, Windows, Mac OS & most other major operating systems

TIMELINE

2009

- Created by Ryan Dahl
- Version 1 in 2009 to revolutionize web applications
- Inspired by Ruby Mongrel web server

2010

- Joyent sponsors Node.js development

2011

- First released version of Node.js available to the public
- Initial version only available for Linux.
- Microsoft partners with Joyent to provide Windows support

2012

- Complete rewrite of central libraries

...

2014

- Latest release v0.10.26
- Still several improvements away from a stable v0.12 and a finalized v1.0

HUGE SUCCESS

The New York Times

WHY?

- Up until recently, the web was a stateless environment.
- Interactive features were encapsulated within Flash or Java Applets
- Node establishes real-time, two-way connections!

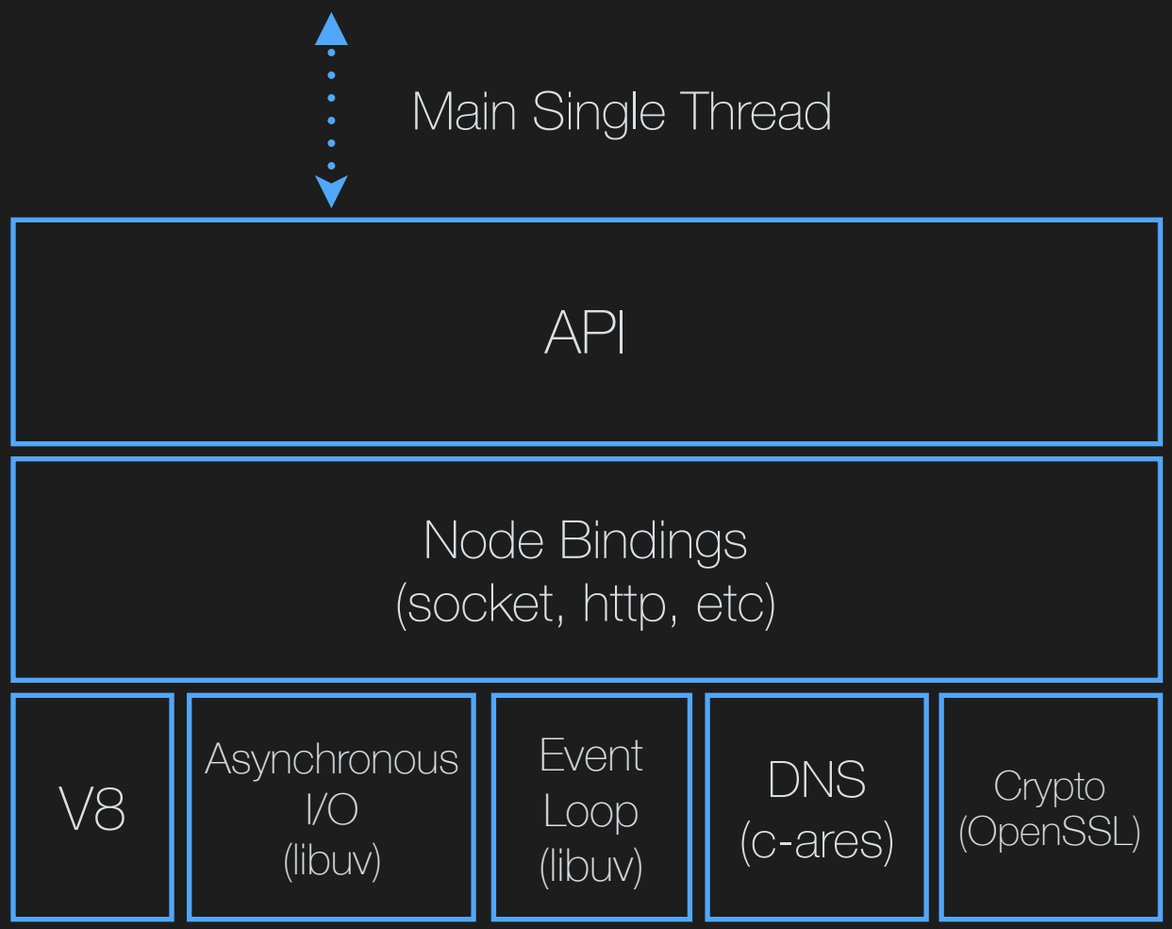
HOW IT WORKS

- Built on Chrome's V8 JavaScript runtime for easily building fast, scalable network applications
- Uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices

OVERALL STRUCTURE

- Two major components:
  - Main core, written in C and C++
  - Modules, such as Libuv library and V8 runtime engine, also written in C++

OVERALL STRUCTURE



- All requests handled by the Main Single Thread
- API in JavaScript
- Node bindings allow for server operations
- Relies on Google's V8 runtime engine
- Libuv responsible for both asynchronous I/O & event loop

V8 RUNTIME ENGINE

- *Just in Time* compiler, written in C++
- Consists of compiler, optimizer, and garbage collector

LIBUV

- Responsible for Node's asynchronous I/O operations
- Contains fixed-size thread pool

MAJOR INFLUENCES

- Heavily influenced by architecture of Unix operating system
- Relies on a small core and layers of libraries and other modules to facilitate I/O operations

MAJOR INFLUENCES

- Built-in package manager contributes to the modularity of Node

MAJOR FEATURES

1. Single threaded
    - Most other similar web platforms are multi-threaded
    - With each new request, heap allocation generated
    - Each request handled sequentially

MAJOR FEATURES

2. Event Loop

- Typically implemented using library sand a block call, but Node is non-blocking throughout!
- Implemented using language construct
- Automatically terminated
- Tightly coupled to V8 engine

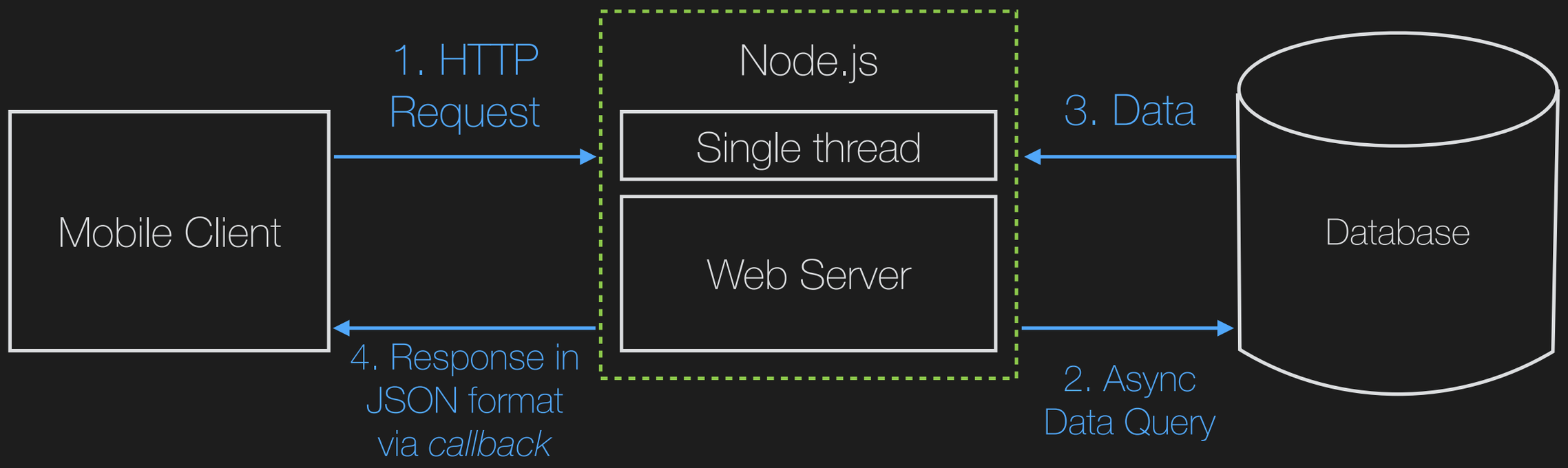
MAJOR FEATURES

3. Non-blocking I/O

- All requests temporarily saved on heap
- Requests handled sequentially
- Can support nearly 1 million concurrent connections

HOW IT WORKS

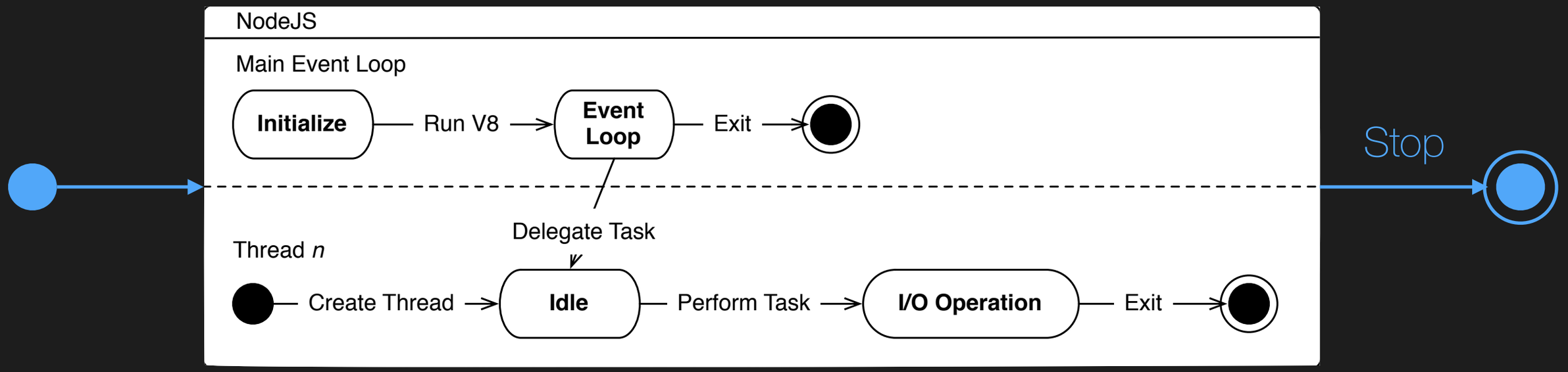
A simple example: accessing data from a database



* Here Node.js, acknowledges the request right away before writing any data to the database.

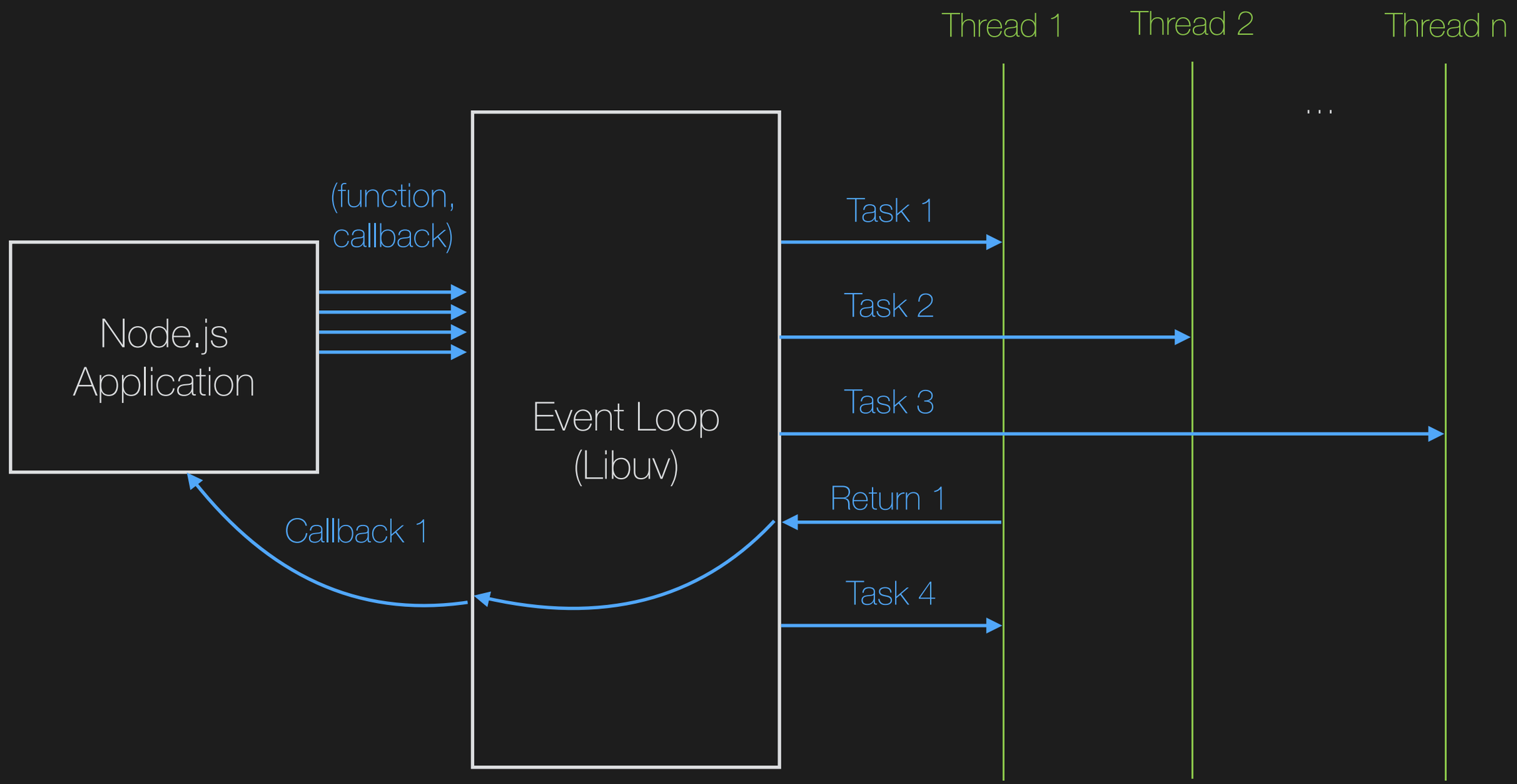
HOW IT WORKS

A generic model of Node.js



HOW IT WORKS

A closer look at the Event Loop



MAJOR ARCHITECTURAL STYLES

- Distributed
- Hierarchical
- Data Flow
- Implicit Asynchronous

CRITICAL ANALYSIS

Benefits:

- Because of its single-threaded, non-blocking scheme, Node can support nearly 1 million concurrent connections
- Asynchronous, event-based scheme allows for scalability, lower memory usage & CPU overhead
- Can be used to implement an entire JavaScript-based web application
- Requests are acknowledged quickly due to asynchronous nature
- Native JSON handling
- Easy RESTful services
- Speedy native bindings in C
- Due to its real-time nature, it's possible to process files while they are being uploaded

CRITICAL ANALYSIS

Best suited for:

- REST + JSON APIs
- Backend for single-page web apps with same language for client and server
- Quick prototyping
- Rapidly evolving applications: media sites, marketing, etc.
- Chat applications
- Ideal for computing and orchestration tasks divided using worker processes

CRITICAL ANALYSIS

Limitations:

- Node & V8 runtime engine are tightly coupled
- Because it is single-threaded, it has a single point of failure for all requests (low fault-tolerance)
- Developers beware of exception handling
- Currently lacking standards regarding code quality
- Without a complete v1.0, backwards-compatibility is bogging down code base

CRITICAL ANALYSIS

Not suited for:

- CPU-bound tasks
- Applications needing to process large amounts of data in parallel, unless using worker processes

THE FUTURE OF NODE

- Larger clients are seeking to integrate Node.js into their mobile platforms.
- Increasing enterprise influence vs. popularity among autonomous developers
- v1.0 release expected by the end of 2014