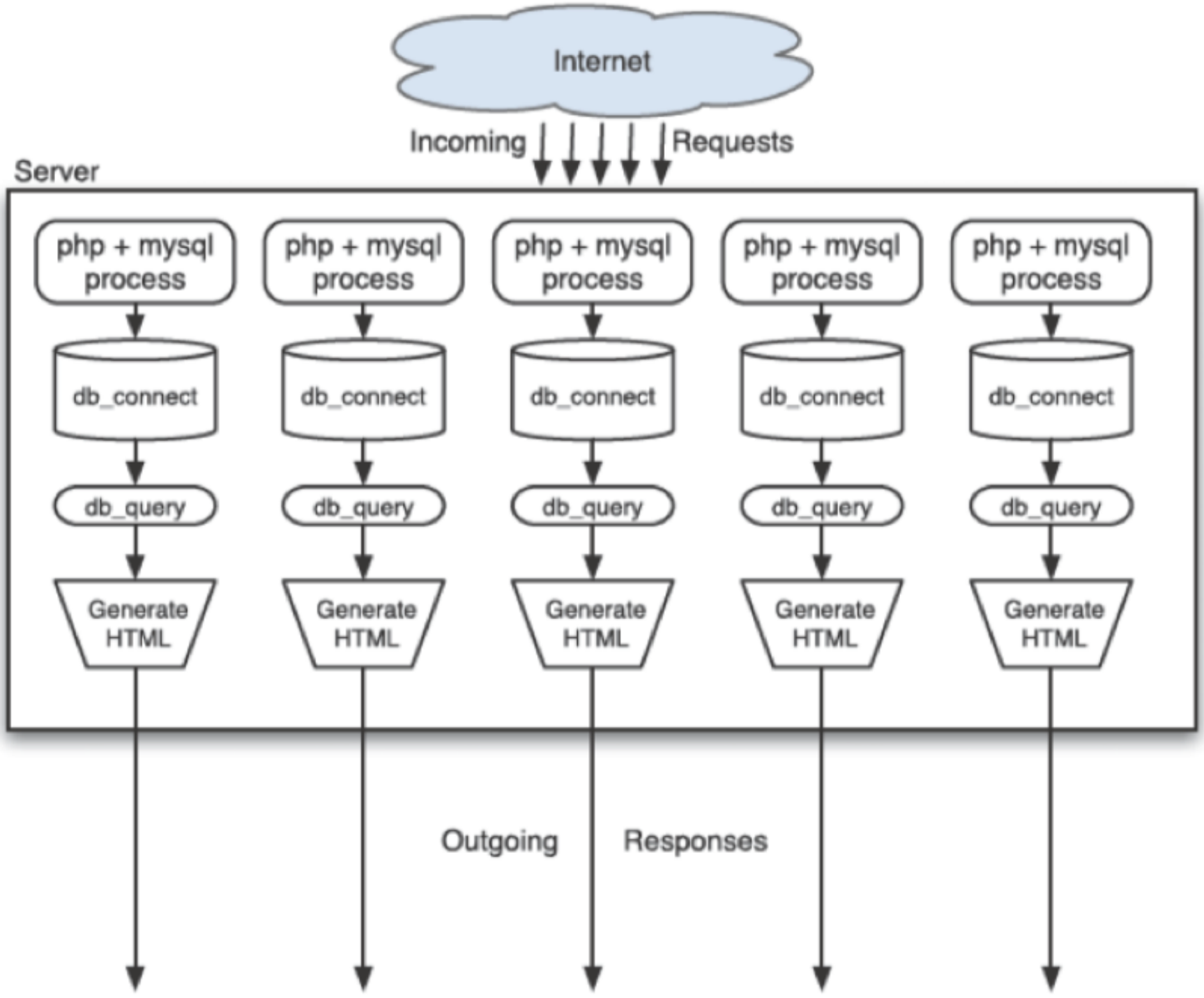
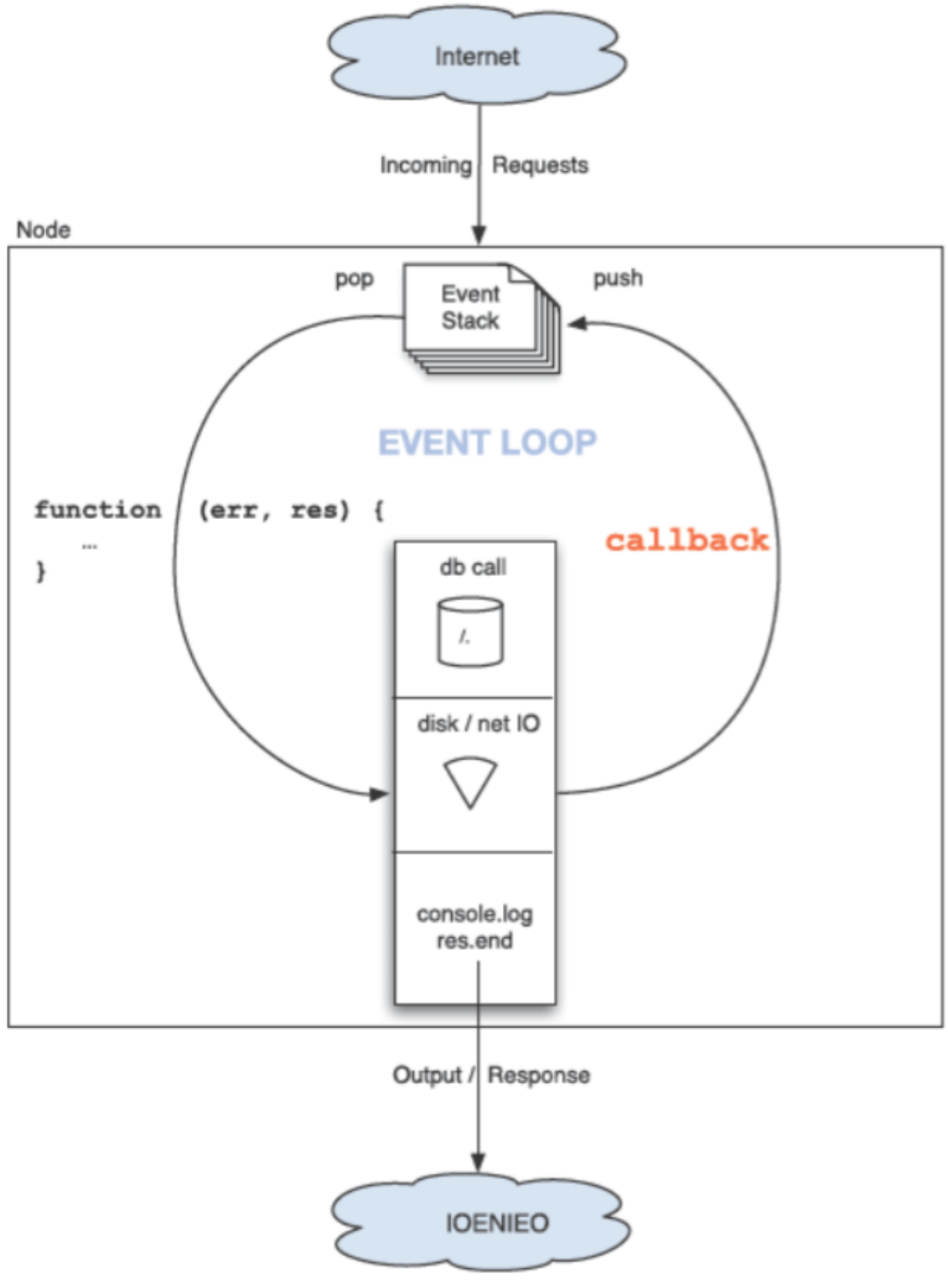
Node Context

Traditional Web Application Structure



Node Structure



What is Node.js

- 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 revolutionise 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

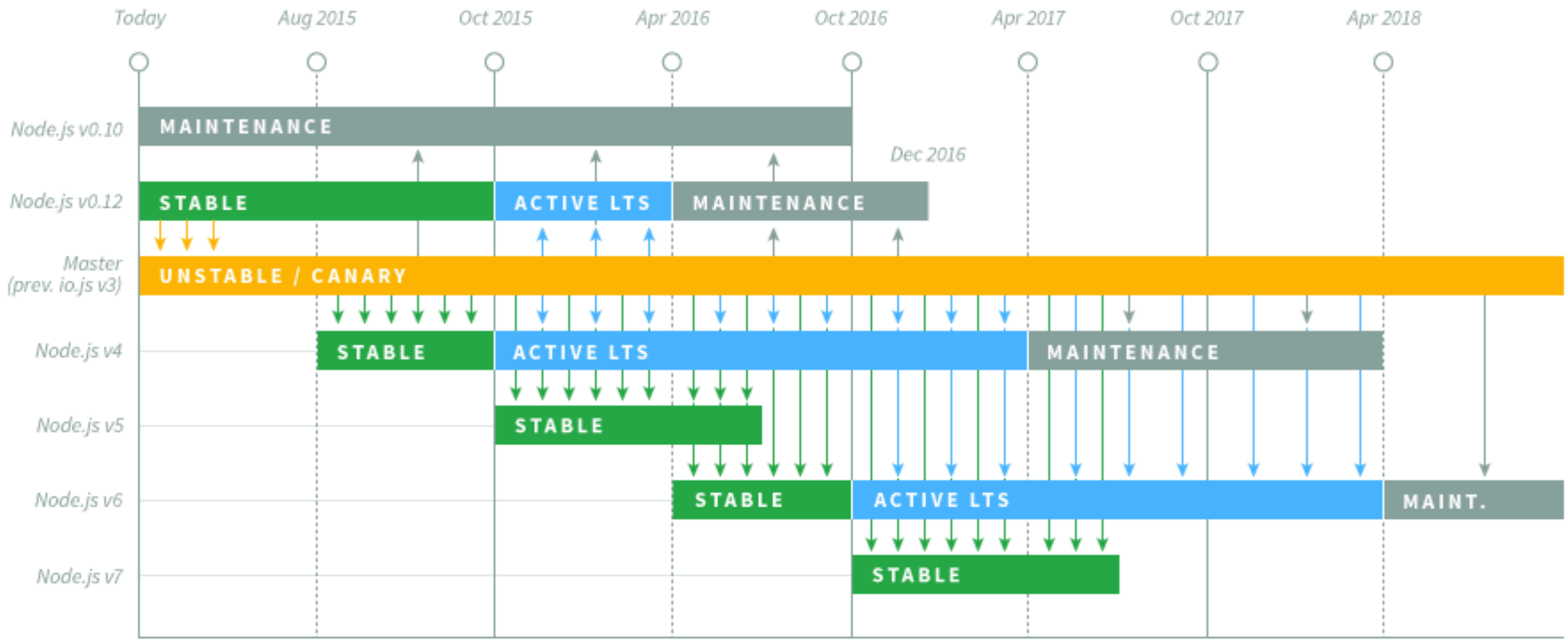
2012

...

2014

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

Node.js Long Term Support Release Schedule



- Since 2014, versions on a predictable trajectory

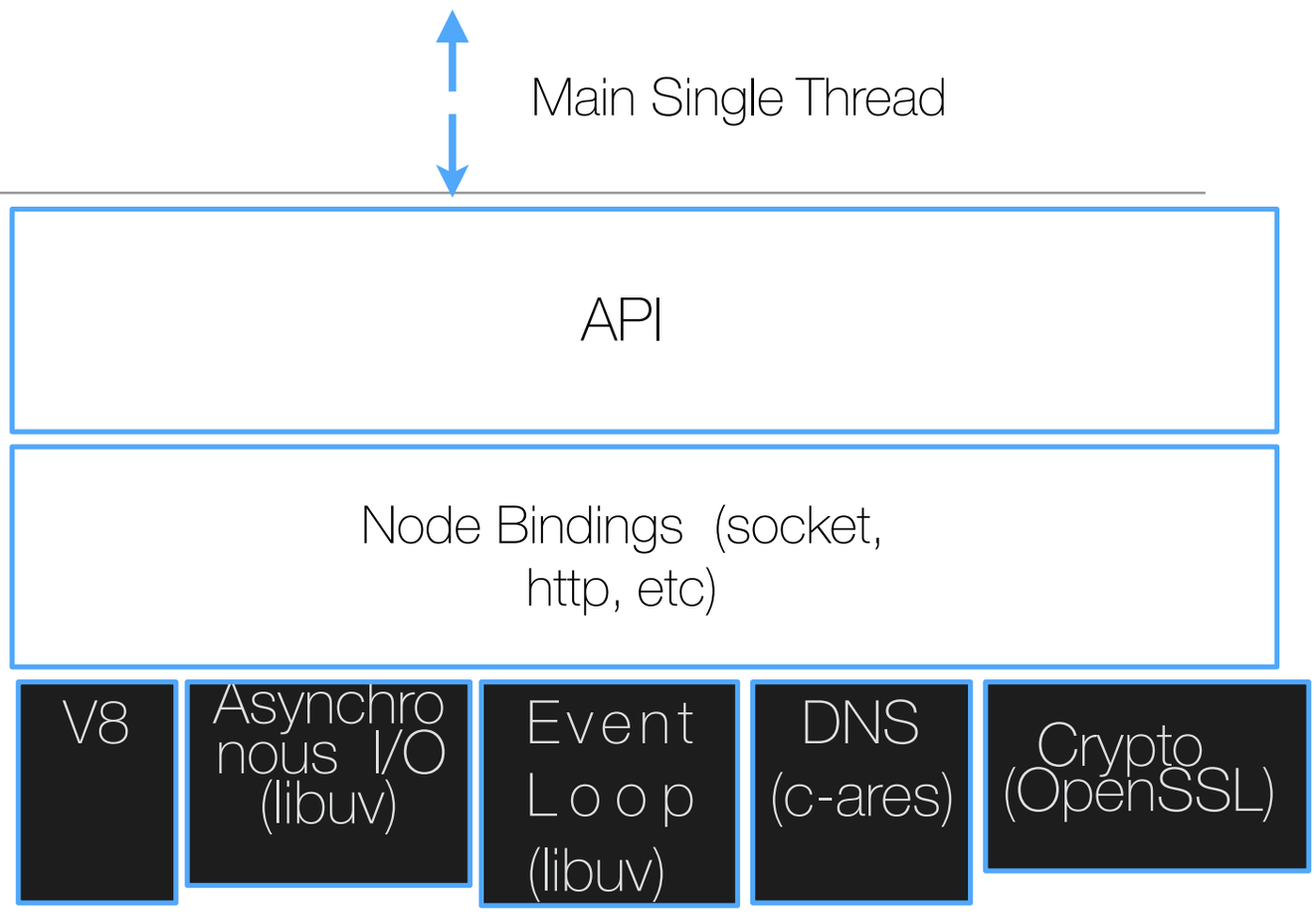
how it works

- 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 efficent, perfect for data-intensive real-time applications that run across distributed devices

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



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

Main Components

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

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

Single Threaded

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

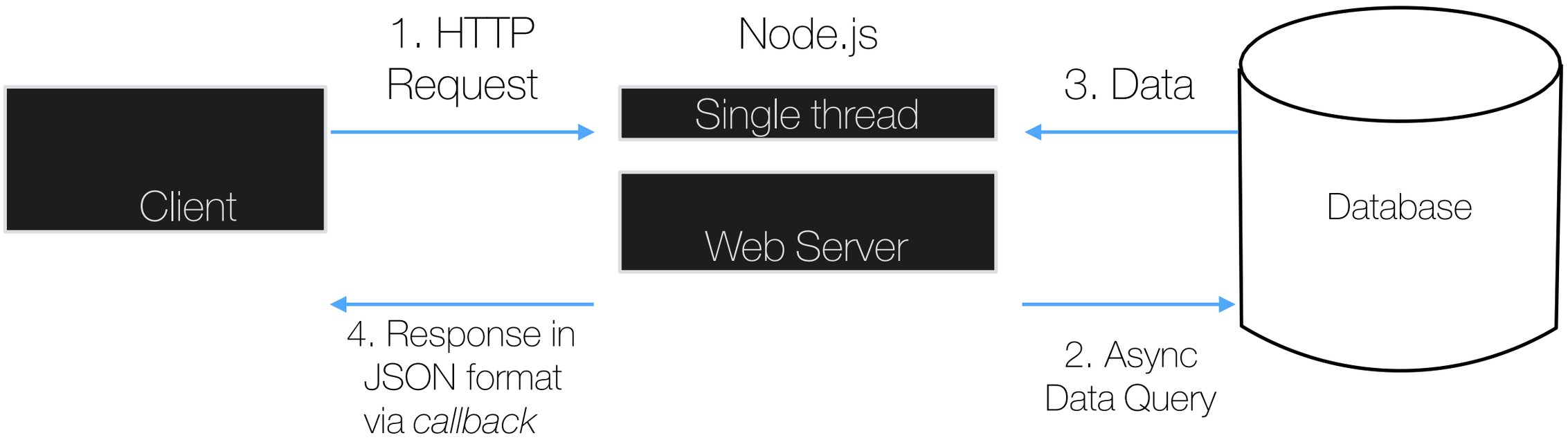
Event Loop

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

Non-blocking

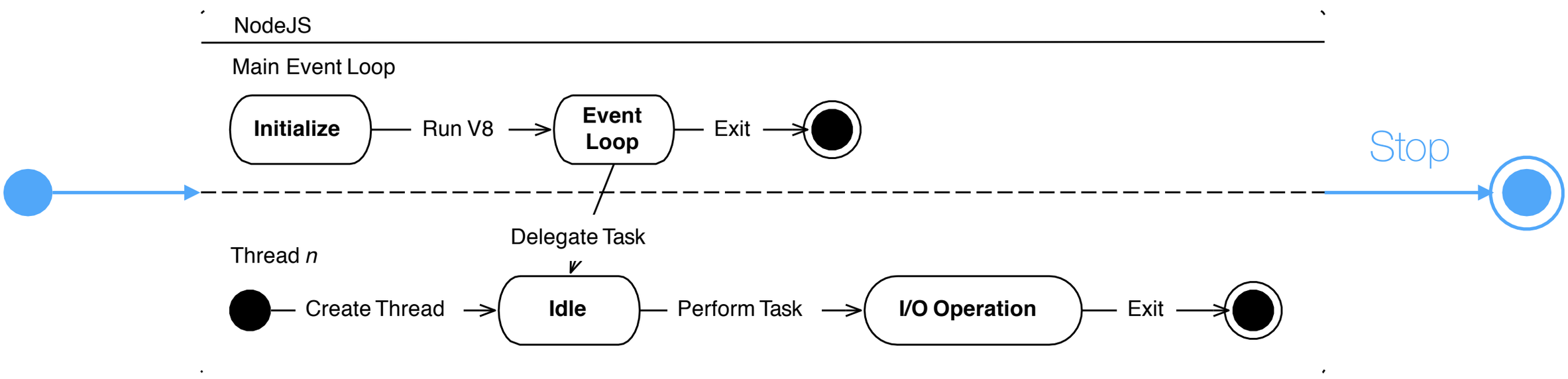
- All requests temporarily saved on heap
- Requests handled sequentially
- Can support nearly 1 million concurrent connections

how it works



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

a generic model



Advantages

- 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

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

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)
- Not suited for CPU-bound tasks
- Not suited for Applications needing to process large amounts of data in parallel, unless using worker processes