**What is Node.js?**

**Author: Ryan Dahl | Release: May 27, 2009 | Written in: C, C++, JavaScript**

Node.js is an open source, cross platform, JavaScript runtime environment developed on **Chrome’s V8 JavaScript engine(Chrome’s V8 JavaScript engine written in C++)** that compiles the JavaScript directly into the native machine code.

Node.js is a JavaScript runtime environment. It allows you to run JavaScript code on the server side, outside of a web browser. Traditionally, JavaScript was mainly used in web browsers to create dynamic and interactive user interfaces. However, Node.js extends the use of JavaScript to the server, enabling developers to build a wide range of server-side applications.

Key features of Node.js include:

1. Non-Blocking I/O: Node.js is designed around an event-driven, non-blocking architecture. This means that it can handle many concurrent connections efficiently without waiting for each operation to complete before moving on to the next one.

1. Asynchronous Programming: Node.js emphasizes asynchronous programming using callbacks, promises, and async/await. This allows developers to write code that can perform multiple tasks simultaneously without blocking the execution of other code.

3. V8 JavaScript Engine: Node.js is built on the V8 JavaScript engine, which was developed by Google for use in the Chrome web browser. V8 compiles JavaScript into machine code for faster execution.

4. NPM (Node Package Manager): NPM is a package manager that comes bundled with Node.js. It allows developers to easily install and manage third-party libraries and modules, which greatly simplifies building applications.

5. Server-Side Applications: Node.js enables the creation of various types of server-side applications, such as web servers, APIs, real-time applications, command-line tools, and more.

6. Cross-Platform: Node.js is cross-platform and can run on various operating systems, including Windows, macOS, and Linux.

7. Community and Ecosystem: Node.js has a large and active community that contributes to its growth. There is a vast ecosystem of libraries, frameworks, and tools built around Node.js, making it well-suited for various development tasks.

Because of its efficiency, asynchronous nature, and the ability to use JavaScript on both the client and server sides, Node.js has become increasingly popular for building scalable and real-time applications.

(NOTE: Runtime environment is a state of the target machine, which may include software libraries, environment variables, etc., to provide services to the processes running in the system.

Note : js Addons are dynamically-linked shared objects, written in C++, that can be loaded into Node. js using the require() function, and used just as if they were an ordinary Node. js module. They are used primarily to provide an interface between JavaScript running in Node.

Addons provide an interface between JavaScript and C/C++ libraries.)

(NOTE: Node.js is not a JavaScript framework; indeed, several authors have written excellent frameworks specifically for Node.js, including Express.js, Restify.js, and Hapi.js.)

It is generally used for large-scale application development, especially for-

* Video streaming sites
* Single page application
* And other applications including PayPal, LinkedIn, Netflix etc.

Node.js *makes*use of –

* Event-driven
* Non-blocking I/O model

**Node.js Architecture**

Basically, Node.js is a combination of Google’s V8 JavaScript engine, an event loop, and a low-level I/O API. Below diagram denotes a simplified version of Node.js architecture. Following are the 3 main parts

* V8 Engine
* Node.js Bindings (Node API)
* An event Loop

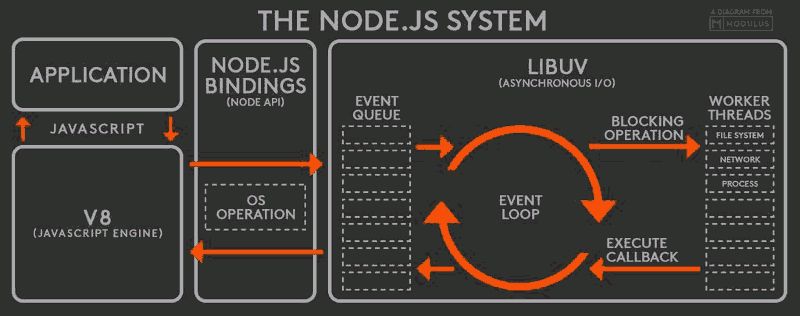
**LIBUV** (Unicorn Velociraptor Library) is a multi-platform C library that provides support for asynchronous I/O based on event loops.

**BINDING** basically are libraries that "bind" two different programming languages so that code written in one language can be used in code written in another library. With the presence of bindings, you don't have to write all the code again just because they are in different languages. Another motivation for bindings is that you can benefit from the advantages of different programming languages. For example, C/C++ are much faster than JavaScript. It might be beneficial to write some code in C/C++ for performance purposes.

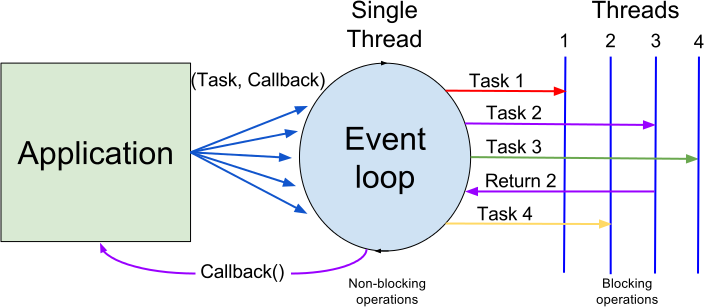
Event Loop is what allows Node.js to perform non-blocking I/O operations — despite the fact that JavaScript is single-threaded.

**EVENT QUEUE** The event loop simply iterate over the event queue which is basically a list of events and callbacks of completed operations.

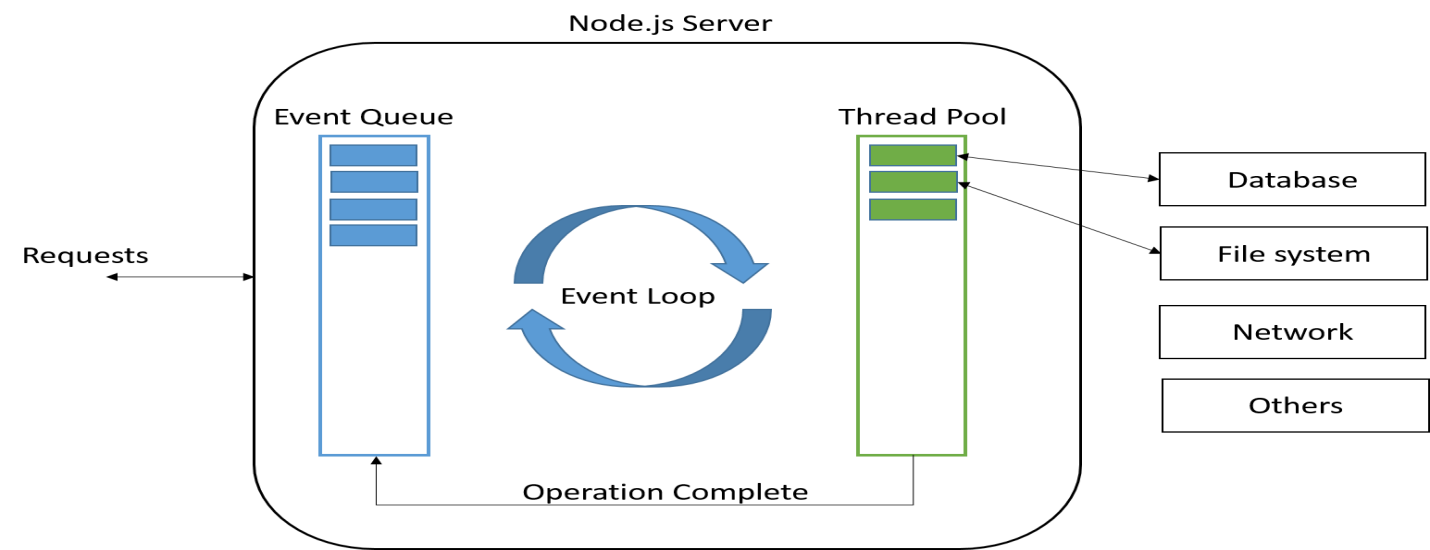
<https://stackoverflow.com/questions/48162575/requests-handling-inside-event-pool-using-nodejs>



Here from application, request passes from v8 engine then node.js binding and goes to event queue.(Node internally works with event queue.)



In the above diagram Threads means Thread Pool (which is a collection of multiple threads). From this multiple threads, one thread deals with File system, one thread deals with Network, one thread deals with database etc.



This makes it a right pick for the data-intensive real-time applications.

(Data-Intensive application means which needs to process large volume of data)

Like any other programming languages, node.js makes use of-

* Packages
* Modules

These are the libraries that contain various functions and are imported from npm (node package manager) into our code and utilized in the programs.

**Features | Advantages | Characteristics of Node.js**

**1) Open Source**

Node.js is an open source . Its community is pretty much active have contributed to add new capabilities to Node.js applications.

**2) Simple and Fast**Since Node.js is built on Google Chrome’s V8 JavaScript Engine, its libraries are capable of fast code execution.

**3) Asynchronous**All the libraries of Node.js are asynchronous which means the Node.js based servers never wait for an API to send back the response and move on to the next API.

**4) High Scalability**  
Because of the event mechanism, Node.js is highly scalable and aids the server in a non-blocking response.

**5) Single-Threaded**With the help of event looping, Node.js is able to follow the single-threaded model. This lets a single program to handle multiple requests.

**6) No Buffering**One of the major functionalities of Node.js applications is that it never buffers any data.

### (Note: While Node.js allows you work with data streams (ie. chunked data), that does not necessarily mean that "Node.js never buffers data" - that is up to the developer to handle.

Typical streaming is done by creating a callback which receives data over a period of time, does something with that data, or handles it off to another stream.

This means that the consumer of the data can get it more readily without having to wait for the entire operation to complete.

Bufferring is actually key to working with streamable data. In the sense that the actual data will reside in a Buffer until your Callback pulls the data out of the stream to work with.

This is why Node.JS has Buffer classes and libraries such as stream-buffers, to facilitate stream data access.)

**7) Cross-Platform**  
Node.js can be easily built and deployed on various platforms like Windows, MAC, and Linux.



**About Node.js Foundation**

Node.js is used by tens of thousands of organizations in more than 200 countries and collect(amasses) more than 3 million active users per month.

It is the runtime of choice for high-performance,enterprise applications, robots, API engines, and mobile websites.

The Foundation is made up of a diverse group of companies.

Platinum members includes IBM, Intel, Microsoft, PayPal and Red Hat.

Gold members include GoDaddy.

Silver members include DigitalOcean.

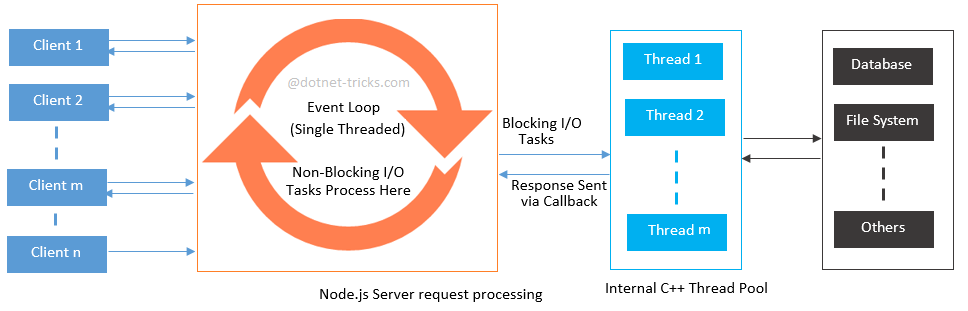
Node.js Foundation is a Collaborative Project at The Linux Foundation.

.

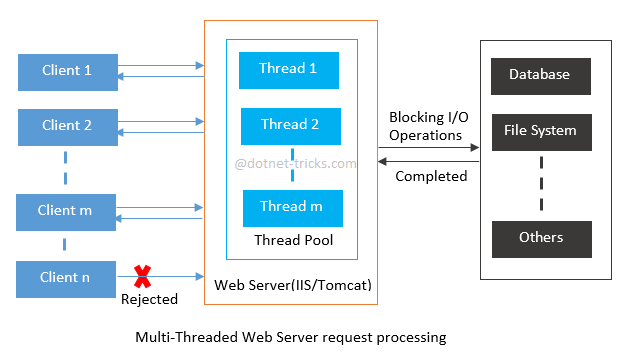
**Node.js v/s Others Server Side Framework**

Today, Node.js is the most popular and widely used in server side for small, large and any sized web app and web application development.

Node.js is different from existing server-side frameworks because it is based on asynchronous events via JavaScript callback functionality and uses the JavaScript as a programming language. Moreover, everything inside Node.js runs in single thread.



While existing server-side framework like ASP.NET, JSP and Php etc. are based on multiple threads web server (IIS/Tomcat). In multiple threads system, there is a limit of maximum number of threads, beyond which the throughput decreases.



There are following issues with Multi-threaded systems:

1. Under heavy load a multi-threaded web server consumes a large amount of memory.
2. Most of the time threads wait till some I/O operations finish.
3. Context-switching(context switching is the mechanism that enables a CPU to switch between different processes or threads in order to achieve concurrent execution in multitasking or multi-threaded environments) and scheduling increases drastically with large number of threads.