Node.js

return to the Linux Foundation

> **Note:** The following is an excerpt from the official About Node.js site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Table 1: Data sheet for Node.js

Title	Node.js
Acronym	
Version	14.3.0
Operating Systems	Linux, macOS, Windows, SmartOS, FreeBSD, OpenBSD, IBM AIX
Downloads	https://nodejs.org/en/download/
Repository	https://github.com/nodejs/node
Supported Languages	JavaScript
License	MIT
Reference	https://nodejs.org/en/docs/

About

> *As an asynchronous event-driven JavaScript runtime, Node.js is designed to build scalable network applications. In the following "hello world" example, many connections can be handled concurrently. Upon each connection, the callback is fired, but if there is no work to be done, Node.js will sleep.*
>
> *Node.js is similar in design to, and influenced by, systems like Ruby's Event Machine and Python's Twisted. Node.js takes the event model a bit further. It presents an event loop as a runtime construct instead of as a library. In other systems, there is always a blocking call to start the event-loop. Typically, behavior is defined through callbacks at the beginning of a script, and at the end a server is started through a blocking call like `EventMachine::run()`. In Node.js, there is no such start-the-event-loop call. Node.js simply enters the event loop after executing the input script. Node.js exits the event loop when there are no more callbacks to perform. This behavior is like browser JavaScript — the event loop is hidden from the user.*
>
> *HTTP is a first-class citizen in Node.js, designed with streaming and low latency in mind. This makes Node.js well suited for the foundation of a web library or framework.*
>
> *Node.js being designed without threads doesn't mean you can't take advantage of multiple cores in your environment. Child processes can be spawned by using our `child_process.fork()`[1) API, and are designed to be easy to communicate with. Built upon that same interface is the `cluster`[2) module, which allows you to share sockets between processes to enable load balancing over your cores.*

1)
https://nodejs.org/api/child_process.html#child_process_child_process_fork_modulepath_args_options

2)
https://nodejs.org/api/cluster.html