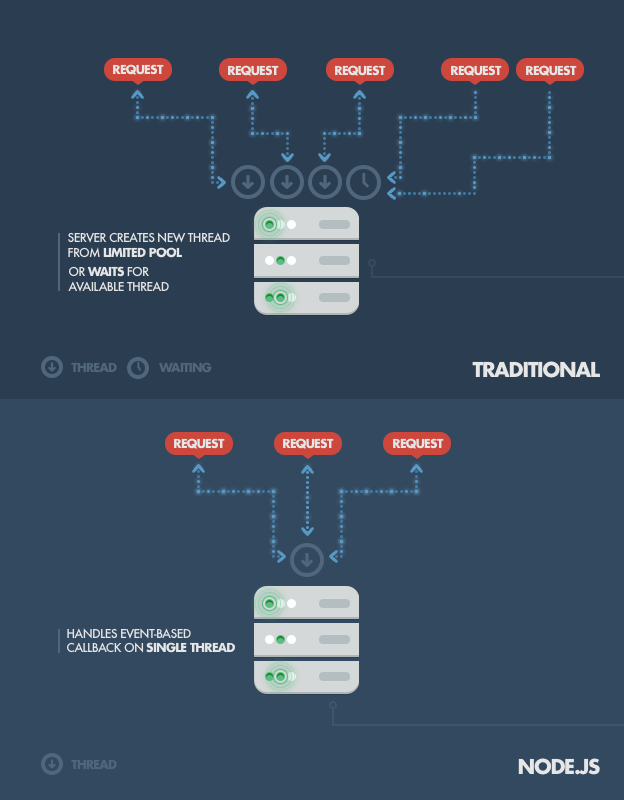
## Node.js and how it works



Node.js is server-side JavaScript built on Google’s V8. It’s event driven (event loop) and was introduced to build highly concurrent, real time network programs.

No function should directly perform I/O from a network or disk. Callbacks are used so not to block the rest of the programs. This is achieved via “promises” which emit either success or error.

Node.js can be used as simple HTTP server or as a HTTP streaming server (web sockets). <http://s3.amazonaws.com/four.livejournal/20091117/jsconf.pdf>

Node Package Manager (NPM) has over 64k packages written by active community as well as companies. Node.js is very light weight and easily customisable as it has many packages.

## Who is using node.js and how

Node.js is gaining popularity with big enterprise companies such as Paypal, [WalMart](http://www.nearform.com/nodecrunch/node-js-becoming-go-technology-enterprise#.UymGoKh_tHX) , Linkedin and more.

Wallmart, PayPal and linkedIn

Because Node.js was designed for non CPU intensive tasks it is a prime candidate for REST APIs as it doesn’t require extensive hardware support.

Assuming each thread in traditional web servers e.g. Apache “has an accompanying [2 MB of memory](http://nodejs.org/about/) with it, running on a system with 8 GB of RAM puts us at a theoretical maximum of 4000 concurrent connections, plus the cost of [context-switching between threads](http://www.slideshare.net/marcusf/nonblocking-io-event-loops-and-nodejs). That’s the scenario you typically deal with in traditional web-serving techniques. By avoiding all that, Node.js achieves scalability levels of over 1M concurrent connections ([as a proof-of-concept](http://blog.caustik.com/2012/08/19/node-js-w1m-concurrent-connections/)).” Source: <http://www.toptal.com/nodejs/why-the-hell-would-i-use-node-js>

This is one of the reasons why these enterprises chose Node.js to save costs and handle millions of concurrent connections with far less servers and engineers as the this example from PayPal shows “PayPal had five engineers redoing the PayPal Wallet application using Java when he [Bill Scott] decided to assign a two-person engineering team to work on the same task using Node.js. Within two months, that team had caught up to the five-person team” source: <http://www.informationweek.com/cloud/software-as-a-service/paypal-finds-nodejs-secret-to-successful-makeover/d/d-id/1127734>

## Node.js and Databases

Node.js developers prefer NoSQL databases especially MongoDB perhaps it’s because, the way it stores data JSONB pretty much JSON format. In additions to this

Relationship database are not very popular with Node.js developers because the tools are premature and hard to work with. Data mapper or active record data access are also premature.

Summary

Node.js is no replacement for existing technologies by no means however, there are areas where it shines than most.

Pros

* Web scraping – fully rendered html responses
* Requires less server resources i.e. number of servers, RAM and CPU
* Easy to learn and scale applications
* Fast to develop because there are lots of resource as its popular with the community
* Very good for message queues with either RabbitMQ/ZeroMQ with redis
* Real-time applications

Cons

* CPU intensive operation block thread
* Not suitable for relational databases
* Not mature enough as its yet to be battle tested
* Doesn’t like blocking I/O

Node.js is gaining popularity in big enterprise companies such as [eBay](http://www.ebaytechblog.com/2011/11/30/announcing-ql-io/#.UymGFah_tHW), [WalMart](http://www.nearform.com/nodecrunch/node-js-becoming-go-technology-enterprise#.UymGoKh_tHX) , Linkedin, Dailymail, and more.

Each use for different purpose for example eBay used it to build REST API whereas Linkedin have built a hybrid mobile application.

Node.js is very light weight and easily customisable by excluded any modules that you don’t need as [eBay](http://www.ebaytechblog.com/2013/05/17/how-we-built-ebays-first-node-js-application/#.Uyl0nah_tHU) have done.

The denominator behind their main reasoning was that they wanted reail-time and highly scalable apps without the need to boost RAM or CPUs.

Pro

* Fast reliable apps both mobile and web
* Can be used hybrid with native mobile application
* Very good for message queues with either RabbitMQ/ZeroMQ
* Node.js + nginx – magic happens
* Monitoring server performance i.e. dashboards
* NPM

Cons

* Not good with relational databases
* Not good with CPU intensive operations
* End-to-end is limited to application such as online chats