NODE.JS KNOWLEDGE

C O N T E N T S

INTRODUCTION TO NODE.JS

Node.js is a server-side platform built on Google Chrome's JavaScript Engine (V8 Engine). Node.js was developed by Ryan Dahl in 2009.

Node.js is a platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications. Node.js 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.

Node.js is an open source, cross-platform runtime environment for developing server-side and networking applications. Node.js applications can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux.

FEATURES OF NODE.JS

- Node.js is an open source server environment
- Being built on Google Chrome's V8 JavaScript Engine, Node.js library is very fast in code execution.
- Node.js is free
- Node.js runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- Node.js uses JavaScript on the server
- Node.js can generate dynamic page content
- Node.js can create, open, read, write, delete, and close files on the server
- Node.js can collect form data
- Node.js can add, delete, modify data in your database
- All APIs of Node.js library are asynchronous, that is, non-blocking. It essentially means a Node.js based server never waits for an API to return data. The server moves to the next API after calling it and a notification mechanism of Events of Node.js helps the server to get a response from the previous API call.
- Node.js uses a single threaded program and the same program can provide service to a much larger number of requests than traditional servers like Apache HTTP Server.
- Node.js applications never buffer any data. These applications simply output the data in chunks.
- It's worth pointing out that Node.js is also great for situations in which you'll be reusing a lot of code across the client/server gap.
- It runs Javascript, so you can use the same language on server and client, and even share some code between them (e.g. for form validation, or to render views at either end.)

- It has become the defacto standard environment in which to run Javascript-related tools and other web-related tools, including task runners, minifiers, beautifiers, linters, preprocessors, bundlers and analytics processors.
- Easy learning curve. Knowing JavaScript gives a developer a good start with Node.js. Of course, you need to know the backend development principles, however, the knowledge of the programming language will simplify things a lot.
- Large community. Node.js, being an open-source project, encourages support and contribution aimed at the improvement and adoption of the platform.
- Robustness - Using Node.js allows organizing full-stack JavaScript development ensuring the speed and performance of the application.
- Scalability - This is a true jewel of the Node.js development environment, as it allows building applications that can easily grow with your business. Node.js works great in systems using the microservices architecture or containerization where the scalability and flexibility can be achieved quickly and easily.
- Great ecosystem - Browse npm (Node.js package manager) for 650,000 free code packages that you can reuse with Node.js.
- The single-threaded event-driven system is fast even when handling lots of requests at once, and also simple, compared to traditional multi-threaded Java or ROR frameworks.
- The ever-growing pool of packages accessible through NPM, including client and server-side libraries/modules, as well as command-line tools for web development. Most of these are conveniently hosted on github, where sometimes you can report an issue and find it fixed within hours! It's nice to have everything under one roof, with standardized issue reporting and easy forking.
- Node.js is released under the MIT license

WHEN TO USE NODE.JS

- I/O bound Applications
- Data Streaming Applications
- Data Intensive Real-time Applications (DIRT)
- JSON APIs based Applications
- Single Page Applications - All client-side scripts are loaded into a single HTML page that works as the main entry point of the application, while all partial views are loaded into this central template on demand.
- Microservices Architecture - Microservices architecture is a way of developing an application as a group of independent, small, and modular services each of which runs a unique single process and plays a specific role in the business logic.
- Internet of Things - IoT (Internet of Things) is a network of devices such as sensors, beacons, actuators, and any other items embedded with electronics that enables them to send and exchange data. Normally, IoT systems pass data from devices to servers and from servers to applications that process it and display it to users.
- Node.js is especially suited for applications where you'd like to maintain a persistent connection from the browser back to the server.
- Application that sends updates to the user in real time, there Node.js is useful.
- When you use something like Node.js, the server has no need of maintaining separate threads for each open connection. This means you can create a browser-based chat application in Node.js that takes almost no system resources to serve a great many clients. Any time you want to do this sort of long-polling, Node.js is a great option.
- For Online games, collaboration tools, chat rooms, or anything where what one user (or robot? or sensor?) does with the application needs to be seen by other users immediately, without a page refresh.

- Node.js seems quite suitable for prototyping, agile development and rapid product iteration.
- Sockets only servers like chat apps, irc apps, etc.
- Social networks which put emphasis on realtime resources like geolocation, video stream, audio stream, etc.
- Handling small chunks of data really fast like an analytics webapp. As exposing a REST only api.
- Applications that are highly event driven & are heavily I/O bound
- Applications handling a large number of connections to other systems
- Real-time applications (Node.js was designed from the ground up for real time and to be easy to use.)
- Applications that juggle scads of information streaming to and from other sources
- High traffic, Scalable applications
- Mobile apps that have to talk to platform API & database, without having to do a lot of data analytics
- Build out networked applications
- Applications that need to talk to the back end very often

WHEN NOT TO USE NODE.JS

- It runs Javascript, which has no compile-time type checking.
- Added to that, many of the packages in NPM are a little raw, and still under rapid development. Npmjs.org has no mechanism to rate packages, which has lead to a proliferation of packages doing more or less the same thing, out of which a large percentage are no longer maintained.
- Dealing with files can be a bit of a pain. Things that are trivial in other languages, like reading a line from a text file, are weird enough to do with Node.js.
- Your server request is dependent on heavy CPU consuming algorithm/Job.
- Node.JS itself does not utilize all core of underlying system and it is single threaded by default, you have to write logic by your own to utilize multi core processor and make it multi threaded.
- Node will support most of databases but best is mongodb which won't support complex joins and others.
- Compilation Errors...developer should handle each and every exceptions other wise if any error accord application will stop working where again we need to go and start it manually or using any automation tool.
- A complex calculation requiring a lot of processing resources may block the flow and cause delays.
- Applications with heavy computing server-side. Since Node.js uses only one CPU core, heavy computations on the server will block all other requests. In this case, the event-driven non-blocking I/O model which is the strongest side of Node.js will become useless, and the application performance will suffer.

- CRUD applications - In this case, using Node.js does not automatically mean poor performance. However, if you are building a simple CRUD app with data coming directly from the server and no API is needed, Node.js may be excessive, as its powerful features will be simply wasted.
- Server-side web applications with relational databases. The reason for Node.js poor performance in this case is that its relational database tools are not as advanced as those created for other platforms.

WHO USES NODE.JS?

PayPal, Netflix, Walmart, LinkedIn, Groupon, Uber, GoDaddy, Dow Jones

NODE.JS TUTORIALS

Reference Website -

Visit ReadyMadeCode.com For Node.js Tutorials And Code