Introduction

- Node.js is an open source, cross-platform built on Chrome's JavaScript runtime for fast and scalable server-side and networking applications. Being an interface to the V8 JavaScript runtime, it enables event-driven programming to the web servers through super-fast JavaScript interpreter that runs in the Chrome browser.
- Google's V8 Engine is an Open Source JavaScript Engine for the Chrome Web browser. V8 Engine is completely written in C++ Language.
- Ryan Dahl is a big fan of "Google's Gmail". He was impressed by Gmail performance and started created new Platform.
- Ryan Dahl has developed a Server-Side Platform "Node JS" in 2009 on top of Google's V8 Engine. That means Node JS applications run on V8 Engine.
- Unlike some UI Frameworks for example Angular JS, use to write Client-side JavaScript, Node JS Platform is used to write Server-side JavaScript.
- Node JS applications can be run on most of the OS Platforms like Windows, Linux, and Mac etc.
- Official Website for Node JS Platform: https://nodejs.org/en/
- Node JS is a Platform. It is not a Framework or Web Framework. It is not a Language.
- It is a Platform which contains thousands of modules to develop wide variety of applications. We can install base Node JS Platform and update all application required modules very easily.
- Node JS "Module" is also known as Package. Now onwards, when we refer Module means Node JS Package.
- Node JS is not for Multi-Threaded applications. It follows Single Thread with Event Loop architecture.

Advantages of Node JS

- **One Language and One Data Format:** We need to use one and only one language to write whole applications i.e. Java Script. We need to code all layers from UI to Controller to Business Layer to Persistence Layer (From Front-End to Back-End) in Java Script only. All layers from Front-End to Back-End support same Data Format i.e. JSON (Java Script Object Notation). UI Layer Angular JS Supports JSON Format, Server-Side Scripts written in Express JS supports JSON Format and Back-end MongoDB NO SQL supports same JSON Format.
- **Open Source:** Node JS is Open Source Module. It provides many modules for free to use in our applications.
- **Highly Scalable:** There are two ways to scale any application:
    - Horizontal Scaling
    - Vertical Scaling

Vertical Scalability means adding more resources to a single node. Horizontal Scalability means adding more nodes to a system. Node JS uses Horizontal Scaling instead of Vertical Scaling to our applications. It's consistent with today's Cloud Technology trend.

- **Better Performance and Low Latency:** I/O operations often blocks our application because it can take a long time to read a big file, or make a large query against a database, or wait for any stream of data to fully transmit. As Node JS follows Non-Blocking IO architecture and also uses only JavaScript to develop entire application, we can achieve better performance and low latency. If we develop our UI and Data Intensive applications by using MEAN Stack, we can achieve very high performance. **MEAN** Stands for **MongoDB, Express JS, Angular JS and Node JS**.
- **Caching Modules:** Node JS Modules are cached once they are loaded into application for first time. Multiple calls to load a module may not cause the module code to be executed multiple times.
- **Less Problems with Concurrency:** Node JS does not follow Multi-Thread architecture. It follows Single-Thread with Event Loop Architecture. It serves any number of requests without any issues. Node JS does NOT need to handle pooling of requests to a set of threads. That's why it has less overhead to handle multiple requests concurrently.
- **Easy to Extend and Lightweight:** We can install only required modules and use them to develop applications. Whenever we need to add new feature or module, we can easily extend it. By following this approach, we can also achieve very light-weight nature into our applications.
- **Faster Development and Easy to Maintain:** As Node JS Platform had already provided support for many modules, we can use them and develop wide variety of applications within no time. We can develop and maintain Node JS applications very easily.
- **REST API:** Node JS Platform supports developing RESTful Web Services API very easily.

Limitations of Node JS

- It does NOT support Multi-threading programming.
- It does support for Computational Intensive Tasks. Node JS struggles in handling of very high computational intensive tasks, because whenever it does something long running task, it will queue all remaining incoming requests, because it follows Single-Thread Architecture with Event Loop.
- Don't use Node JS for Blocking/Synchronous and CPU-intensive tasks.
- Unstable API – Node JS is still in Beta stage and most of its modules are in unstable state. It's not ready for Production or Live Systems.

Download Node.js

❖ The official Node.js website has installation instructions for Node.js: https://nodejs.org

First Program:

Once you have downloaded and installed Node.js on your computer, let's try to display "Hello World" in a web browser.

Create a Node.js file named "myfirst.js", and add the following code:

```
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.end('Hello World!');
}).listen(8080);
```

Save the file on your computer: C:\Users*Your Name*\myfirst.js

The code tells the computer to write "Hello World!" if anyone (e.g. a web browser) tries to access your computer on port 8080.

Node.js has a built-in module called HTTP, which allows Node.js to transfer data over the Hyper Text Transfer Protocol (HTTP). To include the HTTP module, use the require() method.

The HTTP module can create an HTTP server that listens to server ports and gives a response back to the client. Use the createServer() method to create an HTTP server

If the response from the HTTP server is supposed to be displayed as HTML, you should include an HTTP header with the correct content type.

The first argument of the res.writeHead() method is the status code, 200 means that all is OK, the second argument is an object containing the response headers.

The function passed into the http.createServer() has a req argument that represents the request from the client, as an object (http.IncomingMessage object).

Command Line Interface

Node.js files must be initiated in the "Command Line Interface" program of your computer.

How to open the command line interface on your computer depends on the operating system. For Windows users, press the start button and look for "Command Prompt", or simply write "cmd" in the search field.

Navigate to the folder that contains the file "myfirst.js", the command line interface window should look something like this:

Initiate the Node.js File

The file you have just created must be initiated by Node.js before any action can take place.

Start your command line interface, write node myfirst.js and hit enter:

C:\Users*Your Name*>node myfirst.js

Now, your computer works as a server!

If anyone tries to access your computer on port 8080, they will get a "Hello World!" message in return!

Start your internet browser, and type in the address: http://localhost:8080