# \*NPM

# Node Package Manager is a package manager for the js programming language.

# CHALK

# \*chalk is used for colouring

# Const chalk=require(‘chalk’);

# Console.log(chalk.red(add));

# Error outputs – chalk.red

# Normal outputs – chalk.green

# \*GIT Hub – Used for storing codes

# \*Commands for GIT Hub

# git init , git status , git add , git commit –m , git push

# \*git add . 🡪adds all files

# \*git checkout –b “codes”🡪creates new branch

# \*git checkout “codes” 🡪it goes back to previous branch

# \*git push origin “codes” 🡪pushes to particular branch

# DATATYPES

# Var,let and const

# Typeof

# \*Returns string,number,Boolean,undefined.

# \*The typeof operator returns "object" for arrays because in JavaScript arrays are objects.

# If else

# \* ‘===’checks both type and value

# \*’==’checks only value but not type

# Escapechar

# \*The escape sequence \" inserts a double quote in a string.

# BOOLEAN

# \*Except ‘0’ whatever values we pass inside if(),it gives “true”

# What is Callback?

# Callback is an asynchronous equivalent for a function. A callback function is called at the completion of a given task. Node makes heavy use of callbacks. All the APIs of Node are written in such a way that they support callbacks.

# For example, a function to read a file may start reading file and return the control to the execution environment immediately so that the next instruction can be executed. Once file I/O is complete, it will call the callback function while passing the callback function, the content of the file as a parameter. So there is no blocking or wait for File I/O. This makes Node.js highly scalable, as it can process a high number of requests without waiting for any function to return results.

# EXPORT:

# **Syntax:**

# [NodeJS-export-module-2](https://cdn.journaldev.com/wp-content/uploads/2015/04/NodeJS-export-module-2.png)

# What is Node.js used for?

# Node.js is primarily used for non-blocking, event-driven servers, due to its single-threaded nature. It's used for traditional web sites and back-end API services, but was designed with real-time, push-based architectures in mind.

# Companies using nodejs

# Netflix, Linkedin, Trello,Uber,PayPal,Medium,eBay,NASA

# ADVANTAGES OF NODE.JS

# Open Source

# JavaScript as Programming Language

# Scalable

# Better Performance

# Caching Support

# Lightweight and Extensible

# REST API Support

# Unit Testing

# Server Development

# 10.Community Support

# Limitations of Node.js

# There are following limitations of Node.js:

# It doesn’t support multi-threaded programming.

# It doesn’t support very high computational intensive tasks. When it executes long running task, it will queue all the incoming requests to wait for execution, since it follows JavaScript event loop which is single threaded.

# Node good for executing synchronous and CPU intensive tasks.

# 

# EXPRESS GENERATOR

# Use the application generator tool, express-generator , to quickly create an application skeleton. You can run the application generator with the npx command (available in Node.js 8.2.0). ... The app structure created by the generator is just one of many ways to structure Express apps.

# PACKAGE.JSON

# The ****package.json****file defines the application dependencies.

# Bin([www.js](http://www.js))

# The file ****/bin/www**** is the application entry point! The very first thing this does is require() the "real" application entry point (****app.js****, in the project root) that sets up and returns the [express()](http://expressjs.com/en/api.html) application object.

# App.js

# This file creates an express application object (named app, by convention), sets up the application with various settings and middleware, and then exports the app from the module.

# ROUTES

# The route file ****/routes/users.js****  loads the express module and uses it to get an express.Router object. Then it specifies a route on that object and lastly exports the router from the module

# VIEWS

# The views (templates) are stored in the ****/views**** directory (as specified in ****app.js****) and are given the file extension ****.pug****. The method [Response.render()](http://expressjs.com/en/4x/api.html" \l "res.render) is used to render a specified template along with the values of named variables passed in an object, and then send

# the result as a response. In the code below from ****/routes/index.js**** you can see how that route renders a response using the template "index" passing the template variable "title".

# WEB SERVER

# A Web Server is a software application which handles HTTP requests sent by the HTTP client, like web browsers, and returns web pages in response to the clients. Web servers usually deliver html documents along with images, style sheets, and scripts.

# Package-lock.json

# Package-lock.json is automatically generated for any operations where npm modifies either the node\_modules tree, or package.json. It describes the exact tree that was generated, such that subsequent installs are able to generate identical trees, regardless of intermediate dependency updates.

# NPM INSTALL

# npm install downloads a package and it's dependencies.

# npm install can be run with or without arguments.

# When run without arguments, npm install downloads dependencies defined in a package.json file and generates a node\_modules folder with the installed modules.

# When run with arguments, npm install downloads specific modules to the node\_modules folder.

# The package.json file dictates what modules will get installed in the node\_modules folder. It's important that npm install is run in the same location as the package.json file.

# Middleware

# A Middleware is a callback that sits on top of the actual request handlers. It takes the same parameters as a route handler.

# App.set

# app.set(name, data) stores a named property on the app object that can be retrieved later with app.get(name) .

# app.use

# app.use() registers a middleware callback that will be part of the request handler chain for incoming http requests.

# ROOT DIRCTORY

# The root directory, or root folder, is the top-level [directory](https://techterms.com/definition/directory) of a [file system](https://techterms.com/definition/filesystem). The directory structure can be visually represented as an upside-down tree, so the term "[root](https://techterms.com/definition/root)" represents the top level. All other directories within a [volume](https://techterms.com/definition/volume) are "branches" or [subdirectories](https://techterms.com/definition/subdirectory) of the root directory.

# While all file systems have a root directory, it may labeled differently depending on the [operating system](https://techterms.com/definition/operating_system). For example, in [Windows](https://techterms.com/definition/windows), the [default](https://techterms.com/definition/default) root directory is C:\. On [Unix](https://techterms.com/definition/unix) systems and in [OS X](https://techterms.com/definition/os_x), the root directory is typically labeled simply / (a single forward slash). As you move up directories within a file system, you will eventually reach the root directory.

# \_\_dir name

# In Node.js, \_\_dirname is always the directory in which the currently executing script resides (see this). ... Technically, it starts out as your working directory but can be changed using process.chdir() . The exception is when you use . with require()

# HTTP

# The Hypertext Transfer Protocol (HTTP) is an [application protocol](https://en.wikipedia.org/wiki/Application_protocol) for distributed, collaborative, [hypermedia](https://en.wikipedia.org/wiki/Hypermedia) information systems.[[1]](https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol#cite_note-ietf2616-1) HTTP is the foundation of data communication for the [World Wide Web](https://en.wikipedia.org/wiki/World_Wide_Web), where [hypertext](https://en.wikipedia.org/wiki/Hypertext) documents include [hyperlinks](https://en.wikipedia.org/wiki/Hyperlink) to other resources that the user can easily access, for example by a [mouse](https://en.wikipedia.org/wiki/Computer_mouse) click or by tapping the screen in a web browser.

# VERBS OR METHODS OF HTTP

# The primary or most-commonly-used HTTP verbs (or methods, as they are properly called) are POST, GET, PUT, PATCH, and DELETE. These correspond to create, read, update, and delete (or CRUD) operations, respectively.

# HTTPS

# Hypertext Transfer Protocol Secure (HTTPS) is an extension of the [Hypertext Transfer Protocol](https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol) (HTTP). It is used for [secure communication](https://en.wikipedia.org/wiki/Secure_communications) over a [computer network](https://en.wikipedia.org/wiki/Network_operating_system), and is widely used on the Internet.[[1]](https://en.wikipedia.org/wiki/HTTPS#cite_note-1)[[2]](https://en.wikipedia.org/wiki/HTTPS#cite_note-2) In HTTPS, the [communication protocol](https://en.wikipedia.org/wiki/Communication_protocol) is encrypted using [Transport Layer Security](https://en.wikipedia.org/wiki/Transport_Layer_Security) (TLS) or formerly, its predecessor, Secure Sockets Layer (SSL). The protocol is therefore also often referred to as HTTP over TLS,[[3]](https://en.wikipedia.org/wiki/HTTPS" \l "cite_note-3) or HTTP over SSL.

# DIFFERENCE BETWEEN HTTP AND HTTPS

# HTTP is unsecured  HTTPS is secured.

# HTTP sends data over port 80 HTTPS uses port 443.

# HTTP operates at application layer   HTTPS operates at transport

# Layer.

# No SSL certificates are required for HTTP SSL certificate is required and it

# is signed by CA

# 