# \*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#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#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

# Physical Layer (Layer 1) :

# The lowest layer of the OSI reference model is the physical layer. It is responsible for the actual physical connection between the devices. The physical layer contains information in the form of**bits.** It is responsible for the actual physical connection between the devices. When receiving data, this layer will get the signal received and convert it into 0s and 1s and send them to the Data Link layer, which will put the frame back together.

# https://media.geeksforgeeks.org/wp-content/uploads/computer-network-osi-model-layers-bits.png The functions of the physical layer are :

# **Bit synchronization**

# **Bit rate control**

# **Physical topologies**

# **Transmission mode**

# \* Hub, Repeater, Modem, Cables are Physical Layer devices. \*\* Network Layer, Data Link Layer and Physical Layer are also known as **Lower Layers** or **Hardware Layers**.

# 2. Data Link Layer (DLL) (Layer 2) :

# The data link layer is responsible for the node to node delivery of the message. The main function of this layer is to make sure data transfer is error free from one node to another, over the physical layer. When a packet arrives in a network, it is the responsibility of DLL to transmit it to the Host using its MAC address. Data Link Layer is divided into two sub layers :

# Logical Link Control (LLC)

# Media Access Control (MAC)

# The packet received from Network layer is further divided into frames depending on the frame size of NIC(Network Interface Card). DLL also encapsulates Sender and Receiver’s MAC address in the header.

# The Receiver’s MAC address is obtained by placing an ARP(Address Resolution Protocol) request onto the wire asking “Who has that IP address?” and the destination host will reply with its MAC address. https://media.geeksforgeeks.org/wp-content/uploads/computer-network-osi-model-layers-framing.png The functions of the data Link layer are :

# **Framing**

# **Physical addressing**

# **Error control**

# **Flow Control**

# **Access control**

# \* Packet in Data Link layer is referred as ***Frame***.\*\* Data Link layer is handled by the NIC (Network Interface Card) and device drivers of host machines.\*\*\* Switch & Bridge are Data Link Layer devices.

# 3. Network Layer (Layer 3) :

# Network layer works for the transmission of data from one host to the other located in different networks. It also takes care of packet routing i.e. selection of the shortest path to transmit the packet, from the number of routes available. The sender & receiver’s IP address are placed in the header by network layer. The functions of the Network layer are :

# **Routing**

# **Logical Addressing**

# Segment in Network layer is referred as **Packet**. https://media.geeksforgeeks.org/wp-content/uploads/computer-network-osi-model-layers-packet.png \*\* Network layer is implemented by networking devices such as routers.

# 4. Transport Layer (Layer 4) :Transport layer provides services to application layer and takes services from network layer. The data in the transport layer is referred to as Segments. It is responsible for the End to End delivery of the complete message. Transport layer also provides the acknowledgment of the successful data transmission and re-transmits the data if an error is found.

# **• At sender’s side:** Transport layer receives the formatted data from the upper layers, performs **Segmentation** and also implements **Flow & Error control** to ensure proper data transmission. It also adds Source and Destination port number in its header and forwards the segmented data to the Network Layer. Note: The sender need to know the port number associated with the receiver’sapplication. Generally, this destination port number is configured, either by default or manually. For example, when a web application makes a request to a web server, it typically uses port number 80, because this is the default port assigned to web applications. Many applications have default port assigned.

# **• At receiver’s side:** Transport Layer reads the port number from its header and forwards the Data which it has received to the respective application. It also performs sequencing and reassembling of the segmented data.

# The functions of the transport layer are :

# **Segmentation and Reassembly**

# **Service Point Addressing**

# The services provided by transport layer :

# **Connection Oriented Service:**

# It is a three-phase process which include – Connection Establishment – Data Transfer – Termination / disconnection In this type of transmission, the receiving device sends an acknowledgment, back to the source after a packet or group of packet is received. This type of transmission is reliable and secure.

# **Connection less service:** It is a one phase process and includes Data Transfer. In this type of transmission, the receiver does not acknowledge receipt of a packet. This approach allows for much faster communication between devices. Connection oriented Service is more reliable than connection less Service.

# \* Data in the Transport Layer is called as ***Segments***.\*\* Transport layer is operated by the Operating System. It is a part of the OS and communicates with the Application Layer by making system calls.Transport Layer is called as ***Heart of OSI*** model.

# 5. Session Layer (Layer 5) :

# This layer is responsible for establishment of connection, maintenance of sessions, authentication and also ensures security. The functions of the session layer are :

# **Session establishment, maintenance and termination**

# **Synchronization**

# **Dialog Controller**

# \*\*All the below 3 layers(including Session Layer) are integrated as a single layer in TCP/IP model as “Application Layer”.\*\*Implementation of these 3 layers is done by the network application itself. These are also known as ***Upper Layers*** or ***Software Layers***.

# 

# SCENARIO: Let’s consider a scenario where a user wants to send a message through some Messenger application running in his browser. The “Messenger” here acts as the application layer which provides the user with an interface to create the data. This message or so-called Data is compressed, encrypted (if any secure data) and converted into bits (0’s and 1’s) so that it can be transmitted. https://media.geeksforgeeks.org/wp-content/uploads/computer-network-osi-model-layers-session.png

# 6. Presentation Layer (Layer 6) :

# Presentation layer is also called the **Translation layer**.The data from the application layer is extracted here and manipulated as per the required format to transmit over the network. The functions of the presentation layer are :

# **Translation**

# **Encryption/ Decryption**

# **Compression**

# 7. Application Layer (Layer 7) :

# At the very top of the OSI Reference Model stack of layers, we find Application layer which is implemented by the network applications. These applications produce the data, which has to be transferred over the network. This layer also serves as a window for the application services to access the network and for displaying the received information to the user. Ex: Application – Browsers, Skype Messenger etc. \*\*Application Layer is also called as Desktop Layer. *https://media.geeksforgeeks.org/wp-content/uploads/computer-network-osi-model-layers-application.png*

# The functions of the Application layer are :

# Network Virtual Terminal

# FTAM-File transfer access and management

# Mail Services

# Directory Services

# MICROSERVICES

# Microservices are a [software development](https://en.wikipedia.org/wiki/Software_development) technique —a variant of the [service-oriented architecture](https://en.wikipedia.org/wiki/Service-oriented_architecture) (SOA) structural style— that arranges an [application](https://en.wikipedia.org/wiki/Application_(computing)) as a collection of [loosely coupled](https://en.wikipedia.org/wiki/Coupling_(computer_programming)) services.[[1]](https://en.wikipedia.org/wiki/Microservices#cite_note-1) In a microservices architecture, services are [fine-grained](https://en.wikipedia.org/wiki/Service_granularity_principle) and the [protocols](https://en.wikipedia.org/wiki/Protocol_(computing)) are lightweight.

# LAYERS IN MICROSERVICE

# Layer 1: The Hardware Layer.

# Layer 2: The Communication Layer.

# Layer 3: The Application Platform.

# Layer 4: The Microservice Layer.

# HTTP Request Structure from Client

# A simple request message from a client computer consists of the following components:

# A request line to get a required resource, for example a request GET /content/page1.html is requesting a resource called /content/page1.html from the server.

# Headers (Example – Accept-Language: EN).

# An empty line.

# A message body which is optional.

# All the lines should end with a carriage return and line feed. The empty line should only contains carriage return and line feed without any spaces.

# HTTP Response Structure from Web Server

# A simple response from the server contains the following components:

# HTTP Status Code (For example HTTP/1.1 301 Moved Permanently, means the requested resource was permanently moved and redirecting to some other resource).

# Headers (Example – Content-Type: html)

# An empty line.

# A message body which is optional.

# All the lines in the server response should end with a carriage return and line feed. Similar to request, the empty line in a response also should only have carriage return and line feed without any spaces.

# Node.js | package.json

# The ****package.json**** file is the heart of Node.js system. It is the manifest file of any Node.js project and contains the metadata of the project. The package.json file is the essential part to understand, learn and work with the Node.js. It is the first step to learn about developmnent in Node.js.

# The metadata information in package.json file can be categorized into below categories: 1. Identifying metadata properties: It basically consist of the properties to identify the module/project such as the name of the project, current version of the module, license, author of the project, description about the project etc. 2. Functional metadata properties: As the name suggests, it consists of the functional values/properties of the project/module such as the entry/starting point of the module, dependencies in project, scripts being used, repository links of Node project etc.

# Creating a package.json file: A ****package.json**** file can be created in two ways: 1. Using npm init : Running this command, system expects user to fill the vital information required as discussed above. It provides users with default values which are editable by the user. Syntax:

# npm init

# 2. Writing directly to file : One can directly write into file with all the required information and can include it in the Node project.

# Example: A demo package.json file with the required information.

{

"name": "GeeksForGeeks",

"version": "1.0.0",

"description": "GeeksForGeeks",

"main": "index.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1",

"start": "node start.js",

},

"engines": {

"node": ">=7.6.0",

"npm": ">=4.1.2"

},

"author": "GeeksForGeeks",

"license": "ISC",

"dependencies": {

"body-parser": "^1.17.1",

"express": "^4.15.2",

"express-validator": "^3.1.2",

"mongoose": "^4.8.7",

"nodemon": "^1.14.12",

},

"devDependencies": {},

"repository": {

"type": "git",

"url": "https://github.com/gfg/gfg.git" //sample git repo url

},

"bugs": {

"url": "https://github.com/gfg/gfg/issues"

},

"homepage": "https://github.com/gfg/gfg#readme"

}

# Explanation:

# name: The name of the application/project. version: The version of application. The version should follow semantic versioning rules. description: The description about the application, purpose of the application, technology used like React, MongoDB, etc. main: This is the entry/starting point of the app. It specifies the main file of the application that triggers when the application starts. Application can be started using npm start. scripts: The scripts which needs to be included in the application to run properly. engines: The versions of the node and npm used. These versions are specified in case the application is deployed on cloud like heroku or google-cloud. keywords: It specifies the array of strings that characterizes the application. author: It consist of the information about the author like name, email and other author related information. license: The license to which the application confirms are mentioned in this key-value pair. dependencies: The third party package or modules installed using npm are specified in this segment. devDependencies: The dependencies that are used only in the development part of the application are specified in this segment. These dependencies do not get rolled out when the application is in production stage. repository: It contain the information about the type and url of the repository where the code of the application lives is mentioned here in this segment. bugs: The url and email where the bugs in the application should be reported are mentioned in this segment.

# Note: Here, “body-parser”, “express”, “express-validator”, “mongoose” and “nodemon” are the modules/packages installed using npm (Node Package Manager).