NAMASTE NODE.JS SEASON 1

Hand Written Notes

-By Shanmuga Priya

www.linkedin.com/in/shanmuga-priya-e-292730284

Namaste Node JS

Day:1

1) what is Node JS? cross platform, open source

Node JS is a JS Runtime Environment built on chrome's V8 Engine. It is used to run JS code outside the web browser not limited to only server it execute JS code everywhere

History of Node JS:

→ It is developed by Ryan Dal in 2009 and it uses SpiderMonkey (JS engine by firefox)

→ It is now managed by Open JS foundation. (within 2 days) V8 engine chrome)

> → wherever there is a JS code, there always be a JS Engine to execute this JS code.

→ Ryan wanted to run webserver so he named it as web.JS. later he renamed it to Node.Js as it can run anywhere.

2) why Node JS was developed?

→ Earlier there is a ~~server~~ Apache HTTP server used for building webservers but it is a blocking one so Ryan wanted to create a server which is Non-blocking I/O one.

→ In 2010, NPM was developed to support Node Js by Issac.

→ In 2011, Node Js given a windows support earlier it is supported only for Macos, Linux by Joyent + Microsoft

→ In 2012, Ryan left the Node Js project & it took over by Issac

→ In 2014, Fedor forked the Node Js open source and named it as ~~Iojs~~ io.Js.

→ In 2015, both Node Js & io.Js are joined together & maintained by Node Js foundation

→ In 2019, Js foundation + Node Js foundation → Open Js foundation & still now Node Js is managed by this Open Js foundation.

3) why Node Js is popular?

It is popular becoz of its event driven architecture & non-blocking I/O

Episode : 2

Node JS – JS on Server

1) what is server?

→ A server is a computer (or) system that provide data, resources to other computers (client) which are always on & available with stable internet connections. whenever we make req domain name map to some IP address that address points to the server.

2) What language V8 engine & NodeJS written on?

→ V8 engine is written on C++ and can be embedded with any C++ application. It is majorly used in chrome & NodeJS.

→ Node JS is also written on C++ so V8 engine is embedded into it.

→ V8 engine follows ECMAScript standards & runs on various systems like windows, MACOS, Linux.

→ V8 engine take the JS & converts it into a machine code that computer can execute.

3) when V8 engine itself execute JS code why we need NodeJS?

though

→ V8 engine is capable of executing JS code NodeJS gives a additional functionality to it like dealing with file system, http & https request, interaction with OS, DB connectivity, npm package Management, streaming and so on.

4) what are all the powers that V8 engine gives?

→ execute JS code

→ Memory Management by memory allocation & garbage collector

→ Optimization by JIT compilation

→ In context of Browser V8 perform DOM Manipulation task, making network req via Browser API.

Why JS engine should follow ECMAScript Standards?

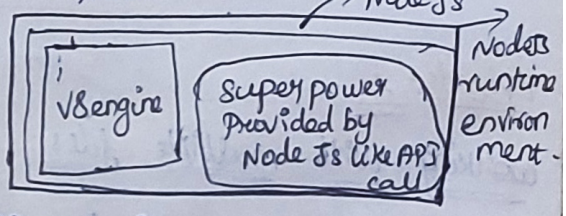
→ The JS code we write should execute in all type of browser & should give same result so the JS engine which execute JS code should adhere with ECMA Script standards.

5) what is Node.js Runtime Environment?

→ Nodejs runtime environment enable us to execute JS code outside the browser. V8 engine + Node JS

→ It includes V8 engine, core Modules, eventloop, Native Binding, npm, libuv

- V8 engine → execute JS code
- core Modules → file handling, networking & more
- eventloop → async operation
- Native Binding → interact with OS
- npm → library & tools
- libuv → Non-blocking I/O operation



6) why V8 engine is written in C++?

→ C++ is used becoz of its efficiency since C++ is a compiled language the code is easily get converted into Machine code for faster execution.

→ C++ gives direct control over memory allocation & deallocation which is imp for creating a garbage collector, a core component of the V8 engine.

Episode: 3

→ whenever we install node npm will also get downloaded automatically.

1) what is node REPL?

R-Read, E-Evaluate, P-Print, L-loop. It is the easiest way of writing program in node.

→ command: node (to start REPL) → It is similar to the Browser's console.

2) what is the command for executing JS file in Node?

→ node fileName

3) what is the name of the global obj in Node?

→ global

→ global obj is one of the super power that Nodejs gives us and it is not a Part of V8 engine.

→ setInterval, setTimeout, clearInterval - Part of global obj not JS

→ Here, this keyword does not point to global obj rather it is empty obj.

Incase of Browser this points to window obj (Node - this - {}

Chrome Browser - this - window)

→ Nodejs give access to V8 engine to use this global obj.

→ To fix the descrepency in words pointing to same global obj in diff run time environment Open JS foundation come up with a new word in 2020 that is globalThis which points to the global obj in all environment no matter it is browser, node & so on.

Episode: 4

working with multiple files:

→ In nodejs each file is called as "modules"

→ There should be only one entry point (a file where the program start) if we have a multiple files we import all those file into the entry point file.

→ To import a file / → require (path)
include a file

→ we cannot access the var, fn, methods defined in one module ~~by~~ into another module by simply requiring it. Bcoz modules protects their var & fn from leaking by default.

→ If we want to access these var & fn we need to export it from that module using module.exports = fn name / var ~~filename~~

→ If we want to export var & fn both we need to wrap it in a obj. eg: module.exports = { x: x, calcSum: calcSum }
exporting multiple things

1)
Why var & fn defined in a module are private unless we export them explictly?

→ It helps to avoid a naming conflict & clean seperation of each module's logic

2) what is module.exports & what are the diff way of writing it?

→ It is an empty obj. module.exports = { x, y, sum }

module.exports.x = x
module.exports.sum = sum

3) what are the types of export / Import in Node JS?

Common JS Modules (cjs)	ES Modules (mjs)
→ Import: require()	→ Import: import {name} from "path"
→ Export: module.exports	→ Export: 2 ways → export {name} → export var
→ Default in Node	→ Enabling required. → by default in React
→ Synchronous / non-strict mode	→ option for async, strict mode is default

4) How to enable ES Modules?

→ In package.json file → {"type": "module"}

5) Why you can't export a module using filename?

→ Nodejs gives more control to developers in what to expose to other modules so it prefers explicit export than exporting whole file.

6) Diff btwn require & import?

Require	import
→ assign the req/ module to a var	→ default import / named import
→ CJS	→ ES6
→ Synchronous loading	→ asynchronous loading of modules.
→ can be used conditionally within functn or block	→ must be at the top level
→ Traditional way including files.	→ Modern way of including files

×

Episode : 5

Diving deep into NodeJS

1) Why fn & var defined in one module cannot be accessed by another module until we explicitly export it?

→ Normally in JS, var & fn defined inside a fn are block scoped (i.e) cannot be accessed outside the fn. Similarly when we create a Module (file) (the var & fn) the code we write are wrapped inside a fn & made it as private behind the scene when we require it. So we can only access them using "module.exports" (or) "export".

2) what happens when we require a file?

→ when we require a module/file all the code written inside that module are wrapped inside a Immediately Invoked function expression (IIFE) and passed into V8 and returns a "module.exports" obj.

3) what is IIFE?

→ Immediately Invoked fn expression is an anonymous fn that is executed right after it is defined and then discard it.
→ This technique is particularly useful for creating private scope.

syntax: (function () { })()

4) why we need to wrap the fn inside Parenthesis?

→ By wrapping the fn inside Parenthesis we turn the fn to fn expression and not fn declaration. It is important becoz fn declaration are hoisted while fn expression are not.

5) why we use IFFE?

→ To immediately invoke the code
→ To keep the var & fn defined inside a module private.

6) How are var & fn are made private in diff modules?

→ It's becoz of IIFE & require() which wraps our code to IIFE.

7) How do you get access to the module.exports & require terms?

→ The module & require term comes from the IIFE Parameter it is given by Nodejs

8) what is module.exports?

→ module.exports is a obj that is returned when a module is "require"d. By default it is an empty obj({}).

→ we can assign any var, fn, obj anything to this obj when we need to expose that.

9) what are the steps involved in requiring a file?

1) Resolving a module:

↳ it checks whether the file path is local, json (or) node module

2) Loading the module:

↳ it loads the content of the module according to the file type.

3) Nodejs wraps the content inside the IIFE

4) Evaluation → Here module.exports is returned

5) Caching: Node caches the required module only once & made it available in any file whenever it requires it than doing all the above steps again.

———— x ————

Episode: 6

LibUV & asynIO

Doubt:

1) what is thread?
→ thread is ntg but a container in which Js code execute in Process.

1) what is synchronous & asynchronous code?

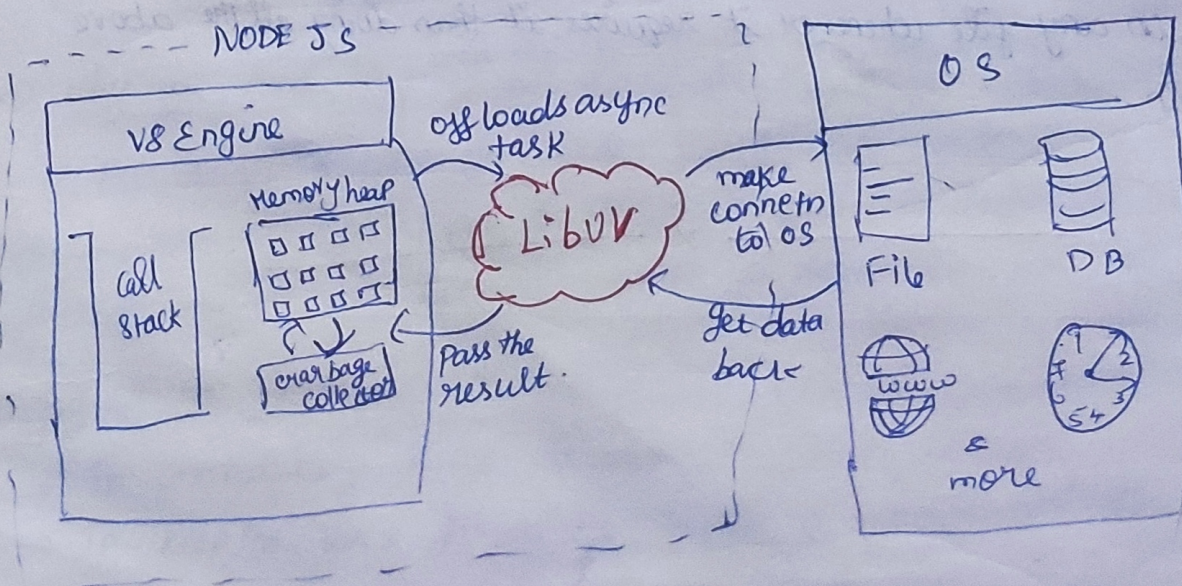
→ Synchronous - Js code execute line by line by default which means one task has to wait for the previous task to get completed only after that the next task will get executed. It is Blocking

→ Asynchronous - one task does not wait for other task to get executed each task perform seperately. Though Js is synchronous & single threaded in default this async operation is possible becoz of various ways like callback, promises, asyn/await, eventloop, setTimeOut, setInterval & so on. It is Non-Blocking

2) what is LibUV?

→ V8 engine does not have power of connecting to DB, Timer, reading file, ~~dea~~ api call & so on. these tasks are performed by the Operating system. V8 engine should needs to connect with this OS for Performing these task. This connection is made by the super hero called "LibUV" Library which nodejs has

3) working diagram of async Program ?



4) Definition of Libuv?

→ It is a multi platform c library that provides support for asynchronous I/O based on event loops and thread pool.

5) why Libuv is written in C?

→ C is low level language. Low-Level-Languages are very suitable for interacting with OS & The major task of Libuv is to interact with OS & make connectn btwn V8 engine & OS. Hence Libuv library is written in C.

Episode: 7

1) How Libuv & V8 engine work together for peforming async task?

→ The V8 engine is capable of executing synchronous code very fast but when it comes to async task it offloads those tasks & callbacks associated with those tasks to libuv. Libuv will interact with OS system and perform the task & send its associated callback fn back to callstack for execution.

2) what happens when you execute a synchronous task which takes lot of time?

The main thread will be blocked until the task finishes.

eg: Fs module "readfilesync" method. Though reading file is OS task it has to be offlooded to Libuv as it is synchronous V8 engine only will execute the task & ~~also cdc offlood cka libuv~~. causing a blocking in the code.

3) How ~~synchronous tasks~~ OS task like reading file from file system is peformed in synchronous way though V8 engine does not have a capability of directly interacting with OS?

→ The synchronous task related to OS are performed by V8 engine though it cannot directly interact with OS behind the scene V8 engine uses C++ bindings of Node.js which

makes a blocking call to the OS & data is read & again it passes back to V8 engine to execute.

4) work flow diagram of synchronous & asynchronous task based on understanding.

Synchronous: (Blocking I/O)

V8 engine → Node.js C++ binding → OS Task → data returned →
(block call)

C++ bindings → V8 engine.

Asynchronous: (Non-Blocking I/O)

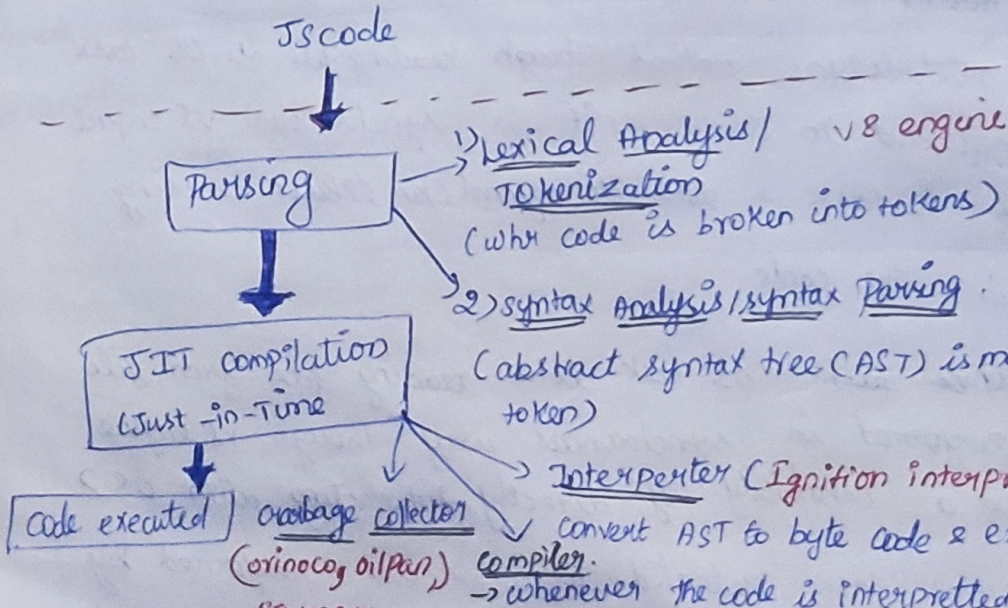
V8 engine → Libuv → OS Task → data returned → libuv →
(Thread pool)

V8 engine (call back execution)

Episode: 8

Deep dive into V8 engine

1) what happens when code is passed to V8 engine?



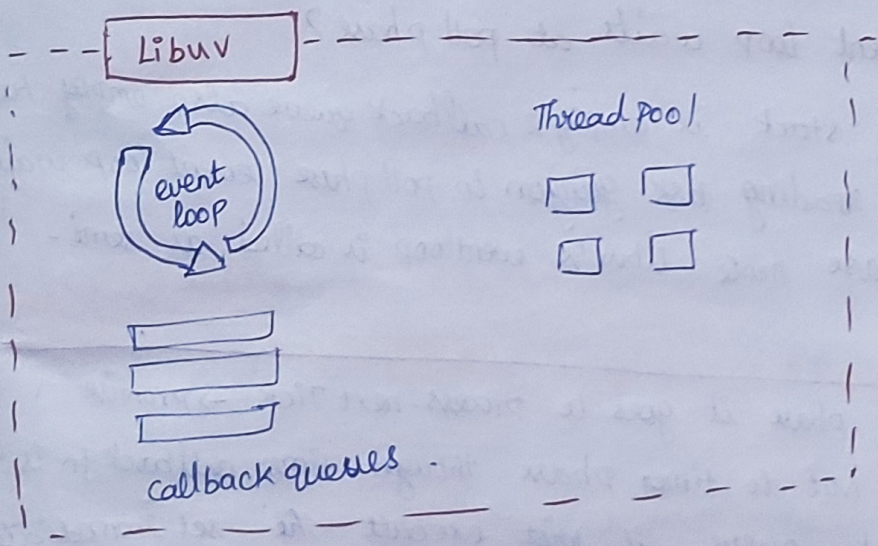
2) why it is called as syntax error?

→ while Parsing when tokens cannot form a Abstract syntax tree it generates syntax error.

Episode: 9

Libuv & Event Loop:

1) Diagram of Libuv



2) what is Eventloop?

Eventloop keeps on checking the callstack & callback queue whenever the callstack is empty & there are some callback fn in the callback queue Eventloop will push this callback fn into the callstack.

3) if there are multiple callback fn waiting in the callback queue how eventloop prioritize it?

→ eventloop give priority to callback fn from promise, process nextTick() → 1st phase → 2nd → 3rd → 4th

4) what are phases in eventloop?

-There are multiphase but these 4 are important

1st Phase: timer (set timeout & set Interval)

2nd phase: Poll (I/O callbacks like API, Fs, crypto, http)

3rd phase: check (set Immediate)

4th phase: close (socket closing, clean up)

→ Before every phase eventloop check if there is any callback fn associated with process.nextTick() & Promises if so it execute this first then move to the phase similarly before executing the next phase event loop again check for process.nextTick() & Promises if so it execute it & then moves to the next phase. This checking will be done before every Phases.

5) why the event loop waits at poll phase?

⇒ when the call stack is empty & callback queue also empty but still some file reading task going on in poll phase event loop waits there at poll phase. Hence libuv's eventloop is called as semi-infinite loop.

→ from the poll phase it goes to process.nextTick → promise → to check phase not to timer phase though timer callback fn is waiting in callback queue it first execute the setImmediate callback fn & then it moves to the next phases.

→ if there is nested nextTick() fn inside another nextTick() fn it will 1st resolve these nested fns until all the nested nextTick() fn are executed. & then only it moves to promise & phases

→ Always nextTick() has more Priority after executing all the nextTick even if it is nested inside a promise it 1st execute nextTick()

6) what is Tick?

→ one whole rotation of a eventloop is called as Tick.

1) what is pending callbacks & idle, prepare phase?

→ These 2 phases comes after the timer phase.

→ pending callbacks : execute I/O callbacks deffered to the next loop iteration

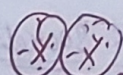
→ idle,prepare : only used internally.

8) ~~when~~ How does Node.js knows when to end the eventloop?

→ eventloop starts as soon as the program starts running and after finishing every phase it check for any pending timers/I/O callbacks are waiting if yes it continue to execute next tick if No it ends the Program.

Episode : 10

Thread Pool in Libuv —

1) Is Nodejs single threaded (or) multi threaded? →  In interview POV

→ It depends on situation.

→ If are only giving just a sync code then Nodejs is single threaded

→ when we perform async task which consume uv_threadpool then it is a multithreaded

2) what is thread pool?

→ thread pool is the group of additional 4 thread that Libuv provides to perform heavy intense task like fs, crypto, dns.lookup, compress & so on.

→ these heavy tasks are performed by each thread in the thread pool when all the 4 threads are engaged the remaining heavy task has to wait until any of the thread finished executing & frees up.

→ we can manually configure thread pool size but upto 1024 threads using process.uv_ThreadPOOL_SIZE By default it is 4 thread

3) How tasks are offloaded to thread pool?

→ eventloop automatically offload the intensive task to the thread pool and it cannot be done manually.

4) what is file descriptor (FDs)?

→ file descriptor is the low level representation (typically an integer) that OS uses to keep track of open files, sockets, pipes, Devices like hardware, USB and so on. whenever Nodejs interact with OS for I/O operatn Fds are created.

5) How file descriptors works in Node js behind the scene?

→ File system:

→ when you perform operatn like 'fs.readFile', a FDS is used to represent the openfile. Since it is blocking I/O it get offloaded to Libuv's thread pool. Here, FDs are used to keep track of file.

→ Network Sockets:

→ when a new HTTP request hits the Server a Tcp socket is created to represent the network connectn btwn client & server.

→ The OS assigns a file descriptor to the socket.

→ the event loop polls these file descriptors, using mechanism like 'epoll' in 'linux', 'kqueue' in 'Macos', 'IOCP' for windows to know when the data in socket is ready for reading or writing.

→ the event loop is notified when the socket is ready for action and it trigger the appropriate callback to continue processing the HTTP req.

6) when to use File descriptor?

→ File descriptor are used whenever Nodejs need to interact with OS for I/O operations, such as opening a file (or) establishing a network connection.

7) when to use threadpool?

→ Thread pool in Node.js is used for operations that require blocking I/O (or) CPU-bound task.

→ Thread poll allows Node.js to perform these tasks in background without blocking the main thread.

8) Tips to be Noted:

→ Don't block the main thread

→ avoid using sync methods, avoid complex regex, complex calculation, recursive loops

---

Episode: 11

Creating a server

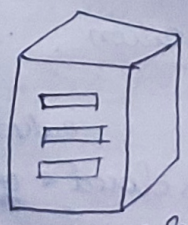
1) what is Server?

→ server can be both hardware & software

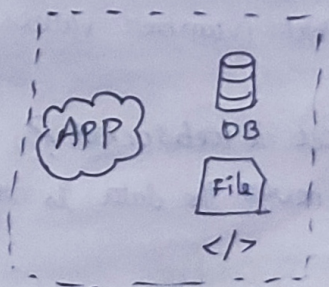
→ Servers are computers which has larger storage, 24/7 internet connectn, NO powercut

eg: AWS servers.

which is otherwise called as "ec2 instance" where we can host our application & we are provided with separate IP address.



Hardware Server
eg: AWS



Software Servers eg: http.createServer()

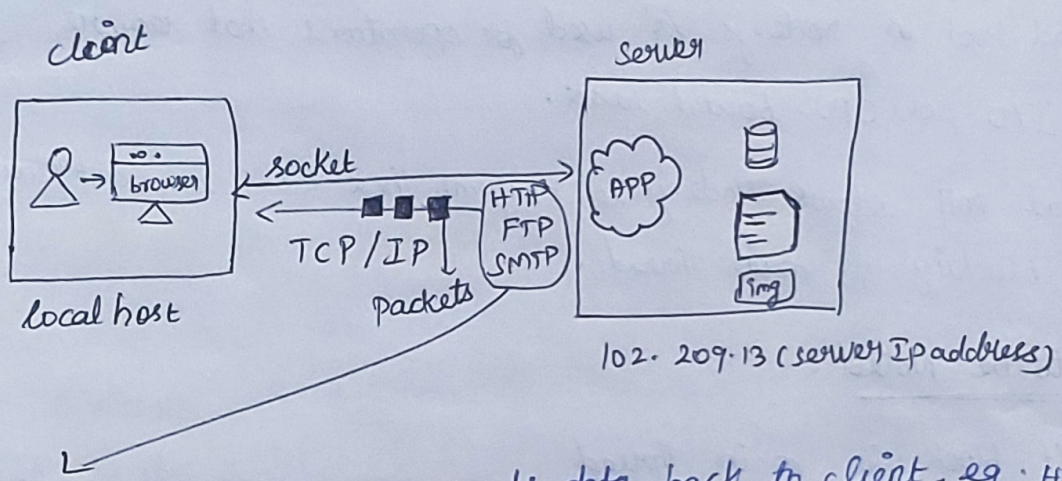
2) client server architecture?

→ whenever a client req something in a browser TCP/IP socket connection is made between client's browser and server.

→ The server returns the data that is requested back to the client

→ once this is done socket connection is closed & for every new req

new socket connection is made.



These are format that server sends data back to client. eg: HTTP- Hyper Text Transfer protocol, FTP- File Transfer Protocol, SMTP- Simple Mail Transfer protocol.

3) what are packets ?

→ whenever the data is send it is break it into smaller chunks These are called packets. These Packets are arranged together again in client side

4) If there are multiple application running on the server then how the req is correctly made to that Particular application though Ip address are same?

→ 'port' through Ip address where the app hosted are same for all the app Port Number varies for each application.

5) what is Socket & websocket?

→ In socket whenever the data is returned socket is closed automatically & opened for further req.

→ But in websocket the connection remains same even after the data is returned we need to manually close it.

6) How to create a server on Nodejs?

→ Nodejs has a built in module "http"

→ It has createServer() in which is used to create the server.

→ req, res → are obj which has many method.

→ res.end() → send data to client

eg:
```
const http = require("http")

const server = http.createServer((req, res) => {
    if (req.url === "/getSecretData") {
        res.end("This is secret!")
    }
    res.end("Hello world")
})

server.listen(3000)
        ↓ Port NO
```

→ In practical real world application we use express to create servers not the http module.

Episode : 12

Databases - SQL & NOSQL

1) what is database?

→ database is an organized collection of data. It is a place where the data is organised in a structured way to poform certain actions easily. → These actions are performed by DBMS.

→ It allows you to efficiently store, manage & retrieve large amount of structured data. These data are usually stored in tabels, row, cole so on makes it easy to query & manipulate.

2) what is DBMS (Database Management System)?

→ DBMS is a software that helps manages databases. It allows user to perform CRUD operation in a database data.

→ eg: MySQL, Oracle

3) what are the types of Databases?

1) Relational DB - MySQL, postgresQL

2) NOSQL DB - MongoDB

3) In memory DB - Redis

4) Distributed SQL DB - Cockroach DB.

5) Time Series DB - influx DB

6) OO DB - db4O → object oriented DB

7) Graph DB - Neo4j (Type of NO SQL DB)

8) Hierarchical DB - IBM IMS

9) Network DB - IDMS

10) cloud DB - Amazon RDS. & so on.
↓
Relational DB.

→ every DB has its own purpose, we can store data in multiple way which made this classification

4) What is RDBMS?

→ RDBMS → came into 70's & 80's

→ eg: MySQL, postgresQL
↓
created by Michael widenium is now managed by Oracle

→ RDB uses a language called SQL (structured Query language)

5) what are the type of NoSQL DB?
→ MongoDB
→ Document DB, key value DB, Graph DB, wide Column DB, multi-model DB.

→ NOSQL (Not-Only SQL) → came into 2000's.

6) what is MongoDB?

→ "10gen" company created MongoDB & renamed its company name to "MongoDB"

→ 2009 - MongoDB became Popular (Nodejs also created at that time)

→ It is based on documents, flexible, very compatible with Nodejs.

→ It is build using c++, Python, Js.

→ collection → Documents → JSON format

*) what is the diff b/w RDBMS & NOSQL ?

RDBMS (MySQL)	NoSQL (Mongo DB)
Table, Rows, column	collection, document, fields
Structured Data	Unstructured Data
Fixed Schema	Flexible Schema
SQL	Mongo (MQL), Neo4J (Cypher)
Tough horizontal scaling	easy to scale horizontally & vertically
Relationships - foreign keys + joins	Nested (Relationships)
Read-heavy apps, transaction workloads	Real time, Big data, distributed computing
eg: Banking apps	eg: Real time analytics, social media

Episode 13

MongoDB and Creating a database

MongoDB Installation:

There are 2 ways of installing MongoDB.

*) Get a package & install it (self managed)

*) MongoDB takes the DB & install it on the server & give access to that cloud platform (Company Manages DB) (Atlas)

2 Types of Edition:

*) Community version

*) free version

Community version:

→ community version is the free version for the developers to use

Enterprise version:

→ It is for company.

→ Both these versions are self managed as well as managed by MongoDB.

MongoDB Compass:

→ It is a GUI for MongoDB & it is free interactive tool to view, query, optimize & analyse MongoDB data.

→ Download & install it. & connect it with "connection string".

→ we can create many DB inside a single cluster.

How to connect our Node app to DB?

Step 1: Install NPM MongoDB package

```
npm i mongodb
```

↓

It will create a node_modules folder, package.json, package-lock.json

Step 2: Requiring the installed module & import the MongoClient class from it

eg:

```
const { MongoClient } = require("mongodb")

const url = "your connection string url"

const client = new MongoClient(url) → creating a new instance of MongoClient class

const dbName = "yourproject"

const db = client.db(dbName) → connecting to DB.

const collection = db.collection("collectionName")
                   ↳ accessing the collection in a db if there is no such collection it will create a new one with this name
```

CRUD operations: → all these methods returns a promise

1) create a Document:

*) insertOne → insert one Document
*) insertMany → insert an array of documents

→ both returns a promise so we need to await the result.

sample code:

insert one:

```
const Doc1 = await collection.insertOne({ name: "Shan" })
```

insert Many:

```
const Docs = [{ name: "Stark" }, { name: "Tom" }]

const Res = await collection.insertMany(docs)
```

2) Read a Document:

*) find → used to find/read a document.

find({}) → retrives all the doc in the collection.

find({ name: "Shan" }) → retrives the docs which matches this filter criteria.

*) findOne → retrive only one document

findone(filter) → retrive the single doc which matches the filter.

3) Update a Document:

*) updateOne → update only one document.

updateOne(doc to be updated, updating value, options)

*) updateMany → update Multiple documents

4) Delete a Document:

*) deleteOne → delete a single doc from a collection which matches the filter.

deleteOne(filter)

*) deleteMany → delete multiple doc from a collection.