Node JS

- Not a programming language
- It provides a runtime environment for javascript to run ouside browser
 - For running the JS code, we need Javascript Engine. Browser has its own JS Engine
    eg: Chrome: V8 Engine
        Mozilla firefox: spider monkey
 - So inorder to execute the JS code without browser, node js runtime environment is using
 - Node JS built on top of V8 Engine
 - It provides some javascript libraries along with it.
 - It is open source, we can freely use it. Also we can make modification to it, if needed

  Features
 =========
 1) It is built on top of V8 Engine
 2) Extremely faster
 3) It has capacity to resolve multiple requests
 4) It is light weight
 5) In some systems, it accepts only one request, resolve, back the response, then only
    accepts the new request. But Node js can accept multiple request at a time
 6) Node js is Single threaded, that means it can handle one request at a time,
    then HOW it handles multiple request?
    Event-loop: event loop is an infinite loop that runs in the background of Nodejs.
    Event loop makes node js to handle multiple request at a time

7) Node js single thread, that means, even if it accepts multiple requests it only process
one request at a time. But if we look at Java, .net. it mas many threads to accept the request.
Even if it can process one request at a time, it is very fast is processing the request

   - Js code consists of both Synchronous fns and asynchronous fns.
    Synchronous Fn: execute line by line
                     eg: console.log
                        mathematical expressions
    Asynchronouse: It will take some time to execute, not in line by line
                   eg: API calls, file operation, call back fn, timer fn

    Timer fns in js
    ----
    setTimeout
    setInterval
    setImmediate
    clearInterval

clearTimeout

There are two queues, 1) Mocrotask queue 2) call back queue
Microtask queue holds async fns like API calls, timer fns
Callback queue holds async fns like call back fns

First Js engine scans the entire code, when it finds a synchronous fn, it is added to callstack,
if it finds a asynchronous fns it is added to either microtask queue or callback queue based on its type
Event loop is an infinite loop that runs in the background, every time it checks the callstack and queues.
When every synchronous fn in the callstack executed, event loop take API fns in microtask queue and
given to call stack, after executing the API fns, it take timer fn in microtask queue given to call stack.
when microtask queue becomes empty it takes fns from callback queue and given to callstack

file extension: .js
Modules in Node js: Javascript library inside node js environment is called Modules
    2 types: a) Custome modules - user defined
             b) Built-in
                fileSystem
Inorder to use this module, we have to import it. In node js we use require("") instead of import
For exporting a fumnction or module we use module.exports

Inbuilt modules-
---
1) File system (fs)
    This module help us to work with file
    Common Uses
        - Read a file : fs.readFile()
        - Create a file : fs.open()
        - Update a file : fs.writeFile()
        - Delete a file : fs.unlink()
        - Rename a file: fs.rename()
import statement
 var fs = require("fs")

Error first callback fns: first argument is err and second argument is our actual response
eg:
 fs.readFile('./test.txt','utf-8', (err, data) => {

 }

2) HTTP module:

```
         It allows Node js to transfer data over HTTP
         - By using http module, we can create an http server, that listens to a
server port
         and gives response back to client
         createServer() method is used to create a server

    3) Event Modules
        Used to create events
        We use a class called eventEmitter

 How to create a Node js server
 ==============================
 1) create a folder with application name
 2) To initalize that folder as a node js application, go inside of that folder
    and type: npm init -y

Client-server Architecture: Client is sending the request, server resolves the
request and send back
the response back to client
API - Application programming interface
CRUD: basic data base operations
        Create
        Read
        Update
        Delete
CORS: Cross Origin resource sharing
    - in a client server architecture, client is running on one domain and port ,
server is
    running on another domain and port (some times domain can be same). Here Node
js server
    by default does not support request form outside domain or port. to solve this
issue we make
    use of CORS module

3) Create an index.js file
4) Edit package.json file and a new script item
        script:{
            "start":"node index.js"
        }
5) Install below node js packages
    a) express
        Express package is used to create server
        npm install express
    b) cors
        To enable cross origin resource sharing
        npm install cors
    c) dotenv
        by using this module, we can access global variables that is decalred in
.env file
        by using process.env.variable_name
```

```
        npm install dotenv


MVC Architecture
=================
Model View Controller

  Postman
  ============
  postman is a tool used for testing the APIS. Usually while we are creating APIs
in backend,
  we have to check whether it is working fine before giving that APIs to front end
developers.
  For checking whether this APIs are working and returning expected response, we
are using POSTMAN
  In postman we can create a collection and inside that collection we can create
request

 MVC Architecture
====================
Model-View-Controller

View: front end (Angular, react, vue application)
From front-end we are sending the request to backend (eg: registration request,
login, getuserdetails)
Backend - consists of 3 parts
1) Routing: part of back end, where paths are configured
2) Controller: It contains the logic to resolve the request
3) Model: model defines, structure in which data can be stored

Database: which stores the data (eg: mysql, mongodb)

Implementing Router
====================
1) create a folder Router
2) create a file router.js inside it
  a) import express
  b) create an object for the class Router in Express
  c) define path and which controller fn to execute the specifc request
      syntax
      ------------
      router. httpmethod('path', ()=>{
        how to resolve the request
      })
  d) export router

  Creating Controller
  ----------------------
  inside the controller, we are difiing the logic to resolve the request and
setting the response
```

```
  that need to be sent back to front-end
  1) create a folder named controller
  2) create a file inside it (eg: userController.js)
  3) define function and exports it
      syntax:

      exports.function_name= (req,res)=>{
          write logic to resolve the request
          set reaponse that need to be sent back to front-end
      }


  Databases
  ===========

Mainly DBs are 2 types
1) Relational Database
      - table like structure
      - only structured data can be inserted
      - eg: Mysql, SqlServer
2) Non-Relational Database
      - document type structure
      - datas are stores as JSON objects (key -value pairs)
      - it can deal with non structured data
      - eg: MongoDB

      - individual data is called document (same as row in relational data base)
      - group of related documents is called collection( same as tables in
relational database)
      - Group of collection is called Database

MongoDB compass is a Ui tool inorder to access locally installed mongoDB. It
provides a user interface
to perform all opeartions on database
eg: Creating a DB, creating a collection, add document, etc

MongoDB Atlas is a web service provided by mongodb to create and manage mongodb in
cloud

Basic Mongo DB Commands
----------------------
show databases      - list all databases inside that mongodb
use database_name   - to enter into particular DB (eg: use projectfairb2)
show collections    -  to list all collections inside that db (eg: show
collections)
db.collection_name.find() - to list all documents inside that collection
                            eg: db.users.find()
db.collection_name.findOne({key:"value"})   -  to list only one data based on
specific criteria
                            eg: db.users.findOne({usernmae:"sean"})
```

```
                              eg:
db.users.findOne({usernmae:"sean",password:"sean123"})

db.collection_name.find().limit(number)    - list specific number of documents
                                 eg: db.users.find().limit(2)

db.collection_name.find().sort({keyname:1/-1}) - sort the values of specific
collection  wrt specific key provided
    if 1, it sort data on ascending order
    if -1 it sort data on descending order
    eg: db.users.find().sort({username:1})
    eg: db.users.find().sort({username:-1})


db.collection_name.insertOne({......})  - to insert one document
    eg: db.users.insertOne({username:"emily", email:"emily@gmail.com"})
db.collection_name.insertMany([{},{}])
    eg: db.users.insertMany([{username:"emily", email:"emily@gmail.com"},{......}])

db.collection_name.countDocuments()   - to get total number of documenst in that
collection

- to find documents based on less than or gretaer than
      db.uses.find({age:{$gt:17}})
      db.users.find({age:{$lt:20}})
  less than or equal to or greater than or equal to
      db.uses.find({age:{$gte:17}})
      db.users.find({age:{$lte:20}})
  in between
      db.users.find({$and:[{age:{$gt:17}},{age:{$lte:20}}]})

to list documents only with specific key name:
    db.users.find({uname:{$exists:true}})

to list documents only without specific key name:
    db.users.find({uname:{$exists:false}})

updateone(), updateMany()


mongoose
==========
this is a nodejs package used to connect mongoDB with server
npm i mongoose


JWT
===========
json web token
- it defines a compact and self contained way for securily transamitting
```

information
  between parties as json object
- when a user logged in, the backend create a token and send back to front-end as response,
  in front-end we are storing the token in session storage and for every request from front-end,
  we have to pass this token in header.
  when the request reaches the backend, backend varifies the token, whether the request is coming
  from correct user
  module used: jsonwebtoken
  npm i jsonwebtoken
  method to create token:
  const jwt = require("jsonwebtoken")
  const token = jwt.sign(argument1, argument2)
    first argumnet: the information that need to be secretly transmitted
    second argument: a secret key
    optional: time validity

to verify the token
        jwt.verify(token,secretkey)

Middlewares in Node JS

As the the name suggest, it act in middle between request and response
- it has the ability to control requet-response cycle

 consider an example, when a user send a request from front-end, in the backend we need to check,
  whether the user is logged in user or not. The request from the client first reaches router,then
  before calling the controller method, we must have to check the credibilty of user, in this case
  we can make use of middlewares

  Two types of middlewares
1) Application middlewares
    Every route in the application will go through this middleware
    So we have to import this in index.j
2) Router specific middleware
    the request coming to particular route only passes through the middleware

Middleware requires 3 arguments, req, res and next
    next() method is used to exit from the middleware
    and pass the control to controller function

multer
----
multer is a node js package, used to deal with form data
By using multer we can store the files that received in the request

it is used as a middleware for handling multipart/form-data
it consist of body object and file object
npm i multer

React- notes

For data sharing between react components, there are 3 methods
1) State lifting
2) redux
3) context-api