GUI-

* Graphical User Interface that helps user to graphically interact with the app.
* Communication between user and app.
* Eg:-Icons on the screen that helps user to interact with app. (Facebbok.com)

API-

* Application Programming Interface is an interface(software) that connect/communicates between two apps.
* Eg:-Travel Website, login with google,paypal in ecommerce website.
* Eg:-Weather App shows map data of Google maps through Google Map API Key. [Google map API helps to interact between Weather APP and Google Map.]
* Eg:- Waiter establish communication between customer and chef.

REST(An style to perform API by transfer data in many formats)-

* Representational State Transfer is an architectural style for API that uses HTTP requests to perform CRUD.
* We will be having same URL for CRUD.
* We are transferring the state of a resource when requested.6 constraints.
* State of resource at any particular time is resource representation.
* Only using mongoose then we will have different URLs for CRUD.

require()=

* -it reads a JavaScript file
* -executes the file
* -return the exports object
* var example = require('./sample.js')

     example = {

                  message: "hi",

                  say: [Function]

                }

Module.exports ~ exports-

* Instructions that tell Node.js which bits of code (functions, objects, strings, etc.) to export from a given file so that other files are allowed to access the exported code.

express.Router()  =

* When var app = express() is called, an app object is returned. Think of this as the main app.
* When var router = express.Router() is called, a slightly different mini app is returned.
* The idea behind the mini app is that each route in your app can become quite complicated, and you'd benefit from moving all that code into a separate file.
* Each file's router becomes a mini app, which has a very similar structure to the main app.

express.json() =

* It parses incoming requests with JSON payloads(content-type-to make request headers) and is based on **body-parser**.
* It parses arguments from an incoming request and uses them as inputs to invoke the corresponding controller method

Middleware

It is those methods that are called BETWEEN processing the Request and sending the Response in your application method. It is called before making call to the database(work).

You NEED  express.json() and express.urlencoded() for POST and PUT requests, because in both these requests you are sending data (in the form of some data object) to the server and you are asking the server to accept or store that data (object), which is enclosed in the body (i.e. req.body) of that (POST or PUT) Request

1. Express provides you with middleware to deal with the (incoming) data (object) in the body of the request.

express.json() is a body parser for post request except html post form [converts request body to JSON ]

 express.urlencoded({extended: false}) is a body parser for html post form.[ converts request body to JSON, converting form-data to JSON ]

*app.use()* middleware is basically used to define the handler of the particular request made by the client.

app.use(express.json())

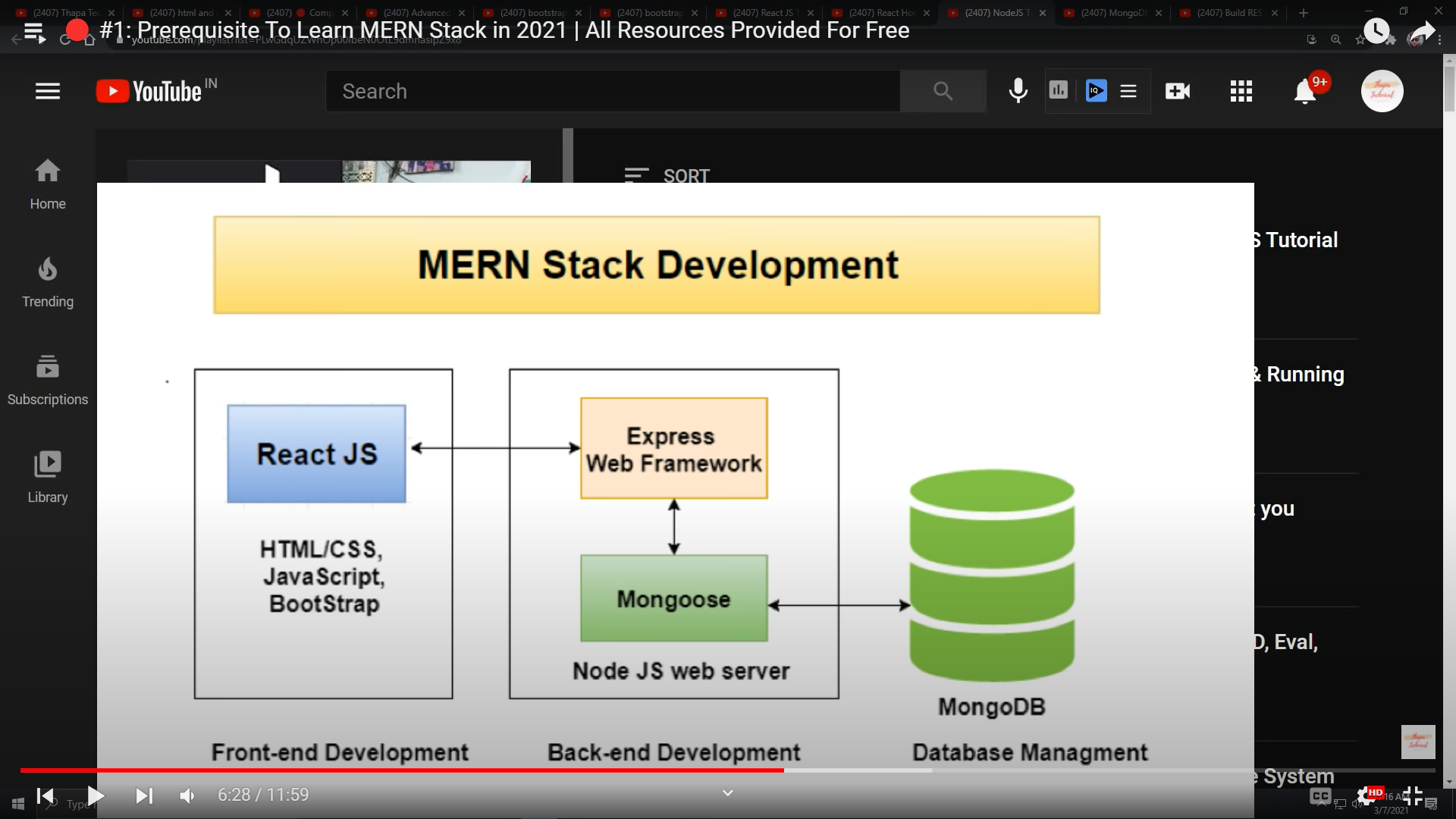
app.use(express.urlencoded({ extended: false }))

process = object in Node.js is a global object that can be accessed inside any module without requiring it. it provides various information sets about the runtime of a program.

How it will recognize that which specific middleware or which function to execute?

Next()=Callback argument to the middleware function. To call a middleware function(say a function) and return back.

Simply a way to call a function(say x) and then returns back from that function(x function) to the line after next().



Middleware? Why middleware?Adv.?

www

MVC

Framework (Frame+Work)

EG:- a ‘Photo frame’ in which a photo must be of shape and size that of frame to work.

EG:- Brick Frame,House Frame(Everything will built inside/according to that frame)

Web Framework means you have to code inside a skeleton that is predefined.

Arrangements in which software provides greater functionality that can be extended by additional user written code.

Advantages:-

* Allow standard way of creating applications.
* Have functions resuable.
* Simply the process of creating web app-basic is already been created.(Not scratch).
* Hiding startup things(focus on main part)
* A way of processing requests(MVC Model)(Just logic not to understand flow).
* Have updated features.
* Attach great third party.

# app.listen() function binds and listen to the connections(requests) of client on specified host and port.

# Its Prepare the server (start the server)(Executes the server)(make it live.) on that port number and host.

# Now, when client comes to that port and host, it finds that app running. That is, its listening to incoming client requests.

# Server-serve data/service to everywhere