**Introduction to express JS.**

Express is a popular node JS framework. It is very developer friendly. We can require express module. It gives use a function then we need to call this function. After calling this function it gives us a instance object. We can now work with this object like in raw node js we use server.listen(port). Similarly we can use app.listen(port). Then server will be run using express js.

**Row node server:**

const http = require("http");

// here server is event emiter

const server = http.createServer((req, res) => {

  if (req.url === "/") res.writeHead("this is main port");

  if (req.url === "/about") res.write("this is about page");

});

//when a new connection is established means server is hitting

// then this "connection" event will be   fired

server.on("connection", () => {

  console.log("a new connection established");

});

// Note: rather doing connection events. we can pass a callback function to the createServer for each connection request for    each connection request. it will be fired when a new connection is established

// here is ther server register listener

server.listen(4000, () => {

  console.log("server listening on port ", 4000);

});

In raw node project, all the request hit into a callback inside createServer() function. Then check the request method and route. Based on route we perform operation. If a project have 500 api request. It check 500 if else condition then perform task based which condition meet up with route and method. It is huge task for a large application and not scalable, maintainable, extendable

**Express server demo:**

// require express module

const express = require("express");

// call the express function and get a instace of server

const app = express();

// git specific api route wit method and pass a callback function

app.get("/", (req, res) => {

  console.log("this is base api");

});

app.post("/", (req, res) => {

  console.log("this is post method with " / " route");

});

// start the express server

app.listen(4000, () => {

  console.log("Server is running on port 4000");

});

Similarly we can require express module. It gives us a function. When we call this function it gives a server instance object. Through this instance variable we can create api like app.get(‘/users’, (req, res) => { });

Here we use this instance. And define method with route. We pass ‘route’ and callback to the method. When a request is come to the express server first it check the method is called then it check with the route. If matches then it enter into the callback. Now we can do something and response data using res.send().

Express gives use scalable system, modular, maintainable, readable, developer experience etc.

Express have five components:

* Express function
* App object
* Req and res
* And routing

**Introduction to express JS.**

**express() function:**

const express = require("express");

const app = express();

app.use(express.json());

app.get("/", (req, res) => {

  res.send("this is base route");

});

app.post("/", (req, res) => {

  res.send("this is post route ");

});

app.listen(5000, () => {

  console.log("server listening on port: ", 5000);

});

**app.use() and express.json() inside app.use() function**

app.use() function use for which services we want to use my project. Like if we want to use express.json() function inside app.use() function. Our express server will be able to accept ‘application/json’ data from the client. It will not accept the plain text from client request. By using express.json() function inside app.use() method. It enable server to accept the json data from the client body. It parse the json data from the client body. If we don’t use it. Our express will not be able to accept the json data from the client body. Express.json() method return something that used by app.use() function.

So ensure in the client side header that client added the headers as content-type: application/json. So must pass the headers object with content-type as application/json. Otherwise our express server will not be able to accept this data into server.

**app.use(express.raw())**

if we want to received stream buffer raw data we can use express.raw() function. It gives use data as stream of buffer.

If we want to get orginal data we can use toString() method.

**app.use(express.text())**

if in client side header has not any content-type. By default it pass text data to the server. But if we want to use this text data from client we need to use express.plain() function.

app.use(express.urlencoded())

if we pass urlencoded data from the client we should pass content-type in client side as well as we need to use in the server side app.use(express.urlencoded()) middleware.

**express.static()**

if we want to make a folder as public for direct access. Means we want to make a folder which we can access directly by url like localhost:5000/public/masud.jpg.. we can access this directly by making it static folder.

Here public is a folder which we have made as static folder. Now can access the content of this folder directly by url. Anywhere in the project and in client side url

**app.use(express.static(‘dir of the folder’, options(optional)));**

role of option like suppose we have public folder and a view folder inside public folder. Express application search index.html file first by default. But if we pass option to the static function. Express go to this function first. Like localhost:5000/view/ no need to define like locahost:5000/public/view/index.html.

const userRouter = express.Router();

**app.use(userRouter);**

this express.Router() object enable us to create multiple router object with handle multiple router efficiently. We can have multiple router. We can use then router variable like

userRouter.get(), userRouter.post()… adminRouter.get() etc.

const express = require("express");

const app = express();

// app.use(express.json());

// app.use(express.text());

// app.use(express.raw());

// app.use(express.urlencoded());

// enable use to create public static folder

app.use(express.static(`${\_\_dirname}/public/`));

// enable use to create multiple router object and use it

const userRouter = express.Router();

userRouter.get("/", function (req, res) {

  console.log("this is using the router object");

});

app.get("/", (req, res) => {

  res.send("this is base route");

  console.log(req.body);

});

app.post("/", (req, res) => {

  res.send("this is post route ");

});

app.listen(5000, () => {

  console.log("server listening on port: ", 5000);

});