

**Middleware** - middle ware is a function which works between request and response it verify whether made request is eligible or not...

To run middleware on **application level use**  
**app.use(“middleware\_name”)**

If you run middleware on application level it will run for every request.

**next();** - to run next middleware after running the first middleware

- Middleware works orderwise means if we have 3 middleware firstly , 1<sup>st</sup> middleware will run , then 2<sup>nd</sup> middleware , then 3rd middleware after that response is sent
- .

## **TASK**

- Convert get user by id in params

- Create blog data like users data object array
- Create routes to set all blog object data
- Create route to send one blog data using id
- Create route to delete one blog data using id

## **POST REQUEST**

//we use post request because when we send data using get request the data get leaks as it is visible in url clearly to prevent that security threat we use post request method

When to use?

- If want to change state of server you send post request - like adding , removing or updating data
- Improve a security threat – data not visible in url .
- Data push karte time request.body ke andar jayega.
- To read data we will also use request.body

## TASK

Form se data server pr add krna hai , Post se send krna hai data

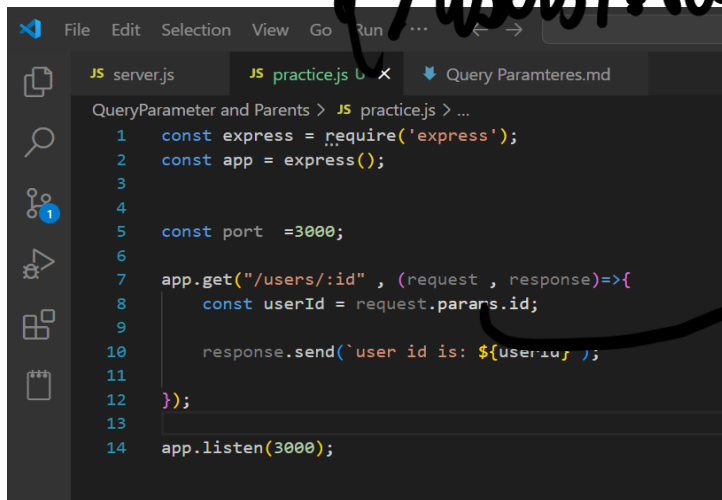
# Params used to handle data and query

Route parameters are parts of url that are variable and can be

directly from URL.

Example → you have route

`/users/:id` route parameter



```
1 const express = require('express');
2 const app = express();
3
4
5 const port = 3000;
6
7 app.get("/users/:id", (request, response) => {
8   const userId = request.params.id;
9
10  response.send(`user id is: ${userId}`);
11 });
12
13
14 app.listen(3000);
```

we using request params

url to use → `localhost:3000/users/12`

output → user id is 12

Query parameter we use key value pairs in ere  
example -localhost:3000/ useradd ? adduser=  
Tushar

In short or can say that params are part of curl  
and query are optional parameters that come  
after? Symbol



This is used for query parameters.

- When the URL is `/search?q=bee&limit=10`, `req.query.q` will be `"bee"` and `req.query.limit` will be `"10"`.

r

- When the URL is `/users/123`, `req.params.id` will be `"123"`.



this is used for params

### Key Differences:

- **Route Parameters ( `params` )**: Are part of the URL path. They're required if the route expects them.
  - Example URL: `/users/:id`
  - Access in Bee.js: `req.params.id`
- **Query Parameters ( `query` )**: Are optional parameters added after the `?` symbol in the URL.
  - Example URL: `/search?q=bee&limit=10`
  - Access in Bee.js: `req.query.q`, `req.query.limit`

**middleware** – execute before the request reaches the route handler or after response is sent

**Built-in middleware** – provided by Express built in


- `Express.json()` Parses incoming JSON requests
- `Express.urlencoded()` parses form data
- `Express.static()` serve static files (, images, etc]

Converting json data into a usable format

This converts form data into usable format of data

#### Example: Using built-in middleware

javascript

 Copy code

```
const express = require("express");
const app = express();

app.use(express.json()); // Parses JSON requests
app.use(express.urlencoded({ extended: true })); //

app.post("/register", (req, res) => {
  console.log(req.body); // Access parsed JSON data
  res.send("User registered!");
});

app.listen(3000, () => console.log("Server running"));
```

**1-2 creating custom middleware** ~ this middleware takes three parameters (req, res, next)

~ req ~ the request object

~ res ~ the response object

~ next () ~ A function that passes control to next middleware

#### Example: A simple logger middleware

4 member

Topic choice is own

6 march

- Blog application
- Only backend
- Fs code  
model/array to  
store data
- Get/post

javascript

Copy code

```
const logger = (req, res, next) => {  
  console.log(`${req.method} request to ${req.url}`)  
  next(); // Moves to the next middleware or route  
};  
  
app.use(logger); // Apply middleware globally  
  
app.get("/", (req, res) => {  
  res.send("Welcome to the Home Page");  
});  
  
app.listen(3000, () => console.log("Server running"))
```

What is node js , working , event loop , async , express , code kya hota hai async , get req post req , query and params , express,json and express.urlencoded , database commands

# MONGO DB

-users.find() ~ return type is array

-users.findOne() ~return type is object

**-filter conditions**

**Db.users.find({name : “Tushar”})** – find ke andr brackets lgake name : “Tushar” se yeh Tushar name wale sab pass karega

**Users.find({age : {\$gt : 20}})** – age greater than 20

Users.findOne({name : “Tushar”}) – return type object ~ return only

**Update - .updateOne();**

- .updateMany();

User.updateOne({ name: name }, { \$set: { email: newEmail } });



**Aggregation** - to do multiple query search we use aggregation pipeline

Homework – learn aggregation from mdn documentation