what is node js?

it is a runtime environment that allows you to run JavaScript code outside of browser , which means that nodejs allows you to run outside of browser normally js is used inside web browsers to make web pages interactive. But with node js you can run js directly on a server or computer.

Working of node js?

It is non-blocking (**Doesn't wait**)

which means tasks don't wait for previous tasks to complete , instead they start and move on , while pending tasks run in background

it is event driven (**responds to events**)

it works based on events, when something happens(like request from a user or a file being read) node js listens to event and responds when it is ready.

1. **Receives Requests** → Node.js receives multiple requests from clients.
2. **Checks if Immediate Execution is Possible** → If the task is simple (like a calculation), it executes immediately.
3. **Delegates I/O Tasks** → If the request involves a time-consuming task (like file reading, database query, or API call), Node.js **sends it to background threads** to process.
4. **Continues Executing Other Tasks** → While waiting for the background task to complete, Node.js **continues handling other requests**.
5. **Executes Callback & Sends Response** → When the background task is done, the **callback function** runs, and Node.js sends the response to the client.

what is async?

unctions run **without blocking** the main thread, allowing the program to continue executing other tasks while waiting for a response (e.g., from a database or API). When a function is declared with async, it **automatically returns a Promise**. Inside an async function, you can use await to pause execution **until a Promise is resolved**.

📌 What is Express.js?

Express.js is a **fast, lightweight, and flexible** web framework for **Node.js**. It helps developers build **web applications and APIs** easily by handling things like **routing, middleware, and HTTP requests**.

◇ GET Request (Retrieve Data)

- Used to **fetch data** from the server.
- Data is sent in the **URL (query parameters)**.
- **No body** in the request.
- Example: Searching on Google (https://google.com/search?q=Node.js).
- **Not secure** for sensitive data (as data is visible in the URL).
- 🔹 **Example of GET in Express.js**

```
javascript
CopyEdit
app.get('/user', (req, res) => {
    res.send("Fetching user data...");
});
```

- 📌 **URL:** `http://localhost:3000/user`
  📌 **Response:** `"Fetching user data..."`

◇ POST Request (Send Data)

- Used to **send data** to the server (e.g., form submissions, API data).
- Data is sent in the **request body** (not in the URL).
- **More secure** than GET, as data is hidden from the URL.
- Used for **creating/updating** data in databases.

◇ Example of POST in Express.js

javascript

CopyEdit

```
app.post('/user', (req, res) => {
  res.send("User data received!");
});
```

📌 Request Body (Sent from client):

json

CopyEdit

```
{
  "name": "Tushar",
  "email": "tushar@example.com"
}
```

📌 **Response:** "User data received!"

Query se request ese bhejte hai - http://localhost:3000/user?name=Tushar&age=21

```
app.get('/user', (req, res) => {
  const name = req.query.name;
  const age = req.query.age;
  res.send(` User Name: ${name}, Age: ${age}` );
});
```

Params se ese bhejete hai - http://localhost:3000/user/Tushar/21

```
app.get('/user/:name/:age', (req, res) => {
  const name = req.params.name;
  const age = req.params.age;
  res.send(` User Name: ${name}, Age: ${age}` );
});
```

Feature	Query Parameters (req.query)	Route Parameters (req.params)
Usage	Optional filters or extra data	Required part of the URL
Format	?key=value&key=value	/value1/value2
Access in Express	req.query	req.params
Example URL	/user?name=Tushar&age=21	/user/Tushar/21

What Is express.json()?

It is built-in middleware function that parses incoming **JSON data** from request body and make it available in req.body , in short if you need json data in your requests you need to use express.json()

What is express.urlencoded({extended : true});

Built-in middleware function and parses incoming form data sent via a **form** and makes it available in req.body

It is used when handling **HTML form submissions** where data is sent in the **URL-encoded format**.

```
<form action="/submit" method="POST">
  <input type="text" name="name" value="Tushar">
  <input type="number" name="age" value="21">
  <button type="submit">Submit</button>
</form>
```

Db 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();

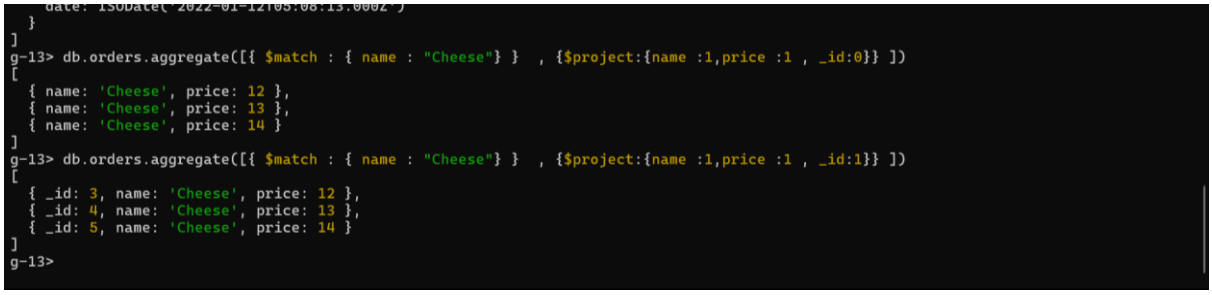
- .updateMany();

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

Db.users.deleteOne({})

Aggregation – it has 3 stages

- $match – filter those which matches database
- $group
- $soof
- $project – return only required key:value pairs



Syntax :

```
Db.users.find(
[
{
//stage 1
$match
}
,
{//stage 2
$group
}



]
)
```