**1. What is Middleware?**

In **Node.js (especially Express.js)**,  
a **middleware** is simply **a function that runs between** the **request coming from the client** and the **final response sent by the server**.

Think of it like **security guards, baggage checks, and ticket checks** you go through before entering a concert hall — each step is a middleware.

### **Basic Syntax**

js

CopyEdit

app.use((req, res, next) => {

console.log("Middleware 1");

next(); // Pass control to the next middleware in line

});

### **Key points about Middleware**

✅ They **have access** to:

* req (request object)
* res (response object)
* next() (function to move to the next middleware)

✅ They can:

* Modify req or res
* End the request-response cycle
* Call next() to move forward

## ****Detailed Example****

js

CopyEdit

const express = require('express');

const app = express();

app.use((req, res, next) => {

console.log("Middleware 1");

next();

});

app.use((req, res, next) => {

console.log("Middleware 2");

next();

});

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

console.log("Route Handler");

res.send("Hello World");

});

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

### **What happens when you visit** http://localhost:3000/

1. **Browser sends a request** → Node receives it.
2. Express starts **looping through middleware stack**:
   * Middleware 1 → Runs → Calls next()
   * Middleware 2 → Runs → Calls next()
   * Route / handler → Runs → Sends response
3. After res.send() is called, Express **stops** and sends data back to the browser.

## ****2. Will the order of middleware functions matter?****

**Yes, 100%.**

In Express, middleware is executed **in the order you define it** in your code.  
If you change the order, your app may behave differently (or even break).

### **Example — Order Matters**

js

CopyEdit

app.use((req, res, next) => {

console.log("First middleware");

next();

});

app.use((req, res, next) => {

console.log("Second middleware");

next();

});

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

res.send("Hello World");

});

**Request flow for / route:**

sql

CopyEdit

First middleware

Second middleware

Hello World

### **If you change the order:**

js

CopyEdit

app.use((req, res, next) => {

console.log("Second middleware");

next();

});

app.use((req, res, next) => {

console.log("First middleware");

next();

});

Now output will be:

sql

CopyEdit

Second middleware

First middleware

Hello World

## ****3. When order is important****

* **Logging before authentication** → Logs user attempt but before checking credentials
* **Authentication before route handler** → Ensures only valid users reach the handler
* **Error-handling middleware** → Must be placed **at the end**