{THE} CODING SKILLS

Middleware in express js

Node JS Tutorial in Hindi #30 What is Middleware in express js

What is Middleware in Express.js?

Middleware in Express.js is a **function** that gets executed **before** the final route handler.

It can be used to check requests, log activities, authenticate users, validate data, and more.

How Middleware Works

```
function middleware(req, res, next) {
  // Do something with req or res
  next(); // Pass control to the next middleware or route
}
```

`req` is the request object

`res` is the response object

`next()` moves to the next middleware or route handler

How the Client, Middleware, and Server Work Together

Client → Sends HTTP request to server (e.g., GET, POST)

Middleware → Intercepts and processes the request

Server → Executes the final route handler and sends a response

Middleware Example (code)

```
import express from "express";

const app = express();

function checkRoute(req, resp, next) {
  console.log("user is acessing " + req.url + " Page");
  next();
}

app.use(checkRoute);

app.get("/", (req, resp) => {
  resp.send("Home Page");
});

app.get("/users", (req, resp) => {
  resp.send("users Page");
});

app.get("/products", (req, resp) => {
  resp.send("Products Page");
});

app.listen(3200);
```

This middleware (`checkRoute`) runs for **every route** and logs the page being accessed before the actual route is handled.

You can also write it like this using an arrow function:

```
app.use((req, resp, next) => {
  console.log("user is acessing " + req.url + " Page");
  next();
});
```

Types of Middleware

Application-level Middleware

Registered using: `app.use()` or `app.get()`

Purpose: Used to apply middleware globally or to specific routes in the main app.

Example Use: Logging, request parsing, etc.

Router-level Middleware

Registered using: `router.use()`

Purpose: Works like application-level middleware, but applies to route instances (e.g., for grouping `/admin` or `/user` routes).

Example Use: Authorization checks in a specific route group.

Built-in Middleware

Comes with Express.js.

Common example: `express.static()` to serve static files (like CSS, images).

Purpose: Provides ready-to-use functionality without extra installation.

Error-handling Middleware

Defined with four arguments: `(err, req, res, next)`

Purpose: Catches and handles errors across the application.

Example Use: Sending custom error messages, logging errors.

Third-party Middleware

Installed via npm, and used with `require()` or `import`.

Purpose: Add advanced functionalities.

Common examples:

`morgan` (logging)

`cors` (Cross-Origin Resource Sharing)

`body-parser` (parsing form data)