Node.js Authentication: Sessions / Cookies & JWT

2 Academic Hours — Beginner Friendly

Learning Objectives

• Understand authentication concepts
• Learn sessions and cookies login
• Learn JWT authentication
• Protect routes in Node.js
• Compare Sessions vs JWT

SESSION 1 — Sessions & Cookies

Authentication answers the question: Who are you?

Cookies store small data in the browser.
Sessions store user data on the server and send a session ID to the browser.

Login Flow:
Client → Login → Server creates session → Session ID saved in cookie

Install:
npm install express express-session

Session Setup: app.use(session({ secret: "secret-key", resave: false, saveUninitialized: true }));

Login Route: app.post("/login", (req, res) => { if (req.body.username === "admin") { req.session.user = "admin"; res.json({ message: "Login successful" }); } });

Protected Route: app.get("/dashboard", (req, res) => { if (req.session.user) { res.json({ message: "Welcome" }); } else { res.status(401).json({ message: "Not logged in" }); } });

SESSION 2 — JWT Authentication

JWT (JSON Web Token) is a signed token stored on the client.
The server does not store session data.

JWT Flow:
Login → Token → Client sends token → Server verifies token

Install:
npm install jsonwebtoken

JWT Login: const token = jwt.sign({ username }, "jwt-secret", { expiresIn: "1h" });

JWT Middleware: function auth(req, res, next) { const token = req.headers.authorization?.split(" ")[1]; if (!token) return res.sendStatus(401); jwt.verify(token, "jwt-secret", (err, user) => { if (err) return res.sendStatus(403); req.user = user; next(); }); }

Protected Route: app.get("/profile", auth, (req, res) => { res.json({ message: "Welcome " + req.user.username }); });

Sessions vs JWT

Sessions store data on server — easier but harder to scale.
JWT stores token on client — scalable and best for APIs.

Homework

Create a login system using JWT with:
• /login endpoint
• /profile protected route
• Token expiration (1 hour)