ScopeTreasure Courses

NODE.JS & EXPRESS COURSES · LESSON N.2 · CODE SCOPE SERIES

Express Framework: Your First Server, the Easy Way

Plain Node makes you do everything by hand — Express does the heavy lifting: install it, answer requests in five lines, routes, and your first JSON API.

Introduction

In lesson N.1 you built a server with Node's raw `http` module — and every visitor got the same answer, whatever they asked for. Real apps must answer **differently by address**: the home request gets one thing, the students list gets another, a wrong address gets an honest "not found". Doing that by hand means long chains of `if` checks on every request. **Express** is the package that removes that pain: it is a small framework on top of Node that gives you clean **routes** — one small function per address — and helpers that send text, HTML or JSON in one line.

Five lines of Express do what took a page of raw Node. That is why almost every Node backend in the world starts the same way you will in this lesson. (This lesson also covers the roadmap's 6.3 topic — your first API.) *(Roman Urdu: Express server banana asaan kar deta hai — har address ke liye alag jawab, sirf chand lines mein.)*

Key points

- **Express is a package, not a new language:** `npm install express` brings it in; you load it with `require("express")` and call it once — `const app = express()` — to create your application.
- **A route is an address plus an answer:** `app.get("/", handler)` means "when a GET request arrives at `/`, run this function". The function receives `req` (the request) and `res` (the response).
- **`req` carries what the visitor sent:** the address, any values inside it, and any data posted. **`res` is how you answer:** `res.send(...)` for text or HTML, `res.json(...)` for data.
- **`res.json` is the API answer:** APIs do not send pages; they send **JSON** (lesson 3.18). `res.json({ ... })` turns your object into JSON and sends it with the right settings, automatically.
  (Roman Urdu: API ka jawab hamesha JSON hota hai — `res.json` wohi kaam ek line mein karta hai.)
- **Every route needs its own address:** `/`, `/students`, `/about` — Express matches the request's address to the route and runs only that one.
- **`app.listen` opens the door:** the same idea as raw Node — pick a port (3000 is the custom) and print a line so you know the server is awake.
- **A wrong address gets 404:** if no route matches, Express answers "Cannot GET /..." — that honesty is built in; lesson N.9 will make it prettier.

Code examples

Example 1 — The five-line Express server

```
// server.js — run it with: node server.js
const express = require("express");
const app = express();

app.get("/", (req, res) => {
  res.send("Welcome to my Express server!");
});

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

What it does: Three ideas in five lines. Load Express, create the app, and declare one **route**: a GET request to `/` (the home address) is answered with a line of text. `app.listen` then opens port 3000. Compare this with the raw `http` version from N.1 — no manual request checking at all.

What you will see: the terminal prints `Server running on port 3000`. Opening `http://localhost:3000` shows `Welcome to my Express server!` — your first Express answer.

Example 2 — Different addresses, different answers

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

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

app.get("/about", (req, res) => {
  res.send("This app is built by a Code Scope student.");
});

app.get("/contact", (req, res) => {
  res.send("Email: student@example.com");
});

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

What it does: Three routes, three addresses, three answers. Express reads the address of each incoming request and runs **only** the matching route's function — the `if` chain you would have written by hand in raw Node is simply gone. Adding a new page to the app is now just adding one more route.

What you will see: `http://localhost:3000/` shows `Home page`, `/about` shows the about line, and `/contact` shows the email line. A fourth address, like `/xyz`, shows Express's built-in `Cannot GET /xyz` — proof that no route matched.

Example 3 — Your first API: answering with JSON

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

const students = [
  { id: 1, name: "Ali", marks: 70 },
  { id: 2, name: "Sara", marks: 92 },
  { id: 3, name: "Rahim", marks: 64 },
];

app.get("/students", (req, res) => {
  res.json(students);
});

app.get("/students/count", (req, res) => {
  res.json({ total: students.length });
});

app.listen(3000, () => {
  console.log("API running on port 3000");
});
```

What it does: This is the moment a "website" becomes an **API**. Instead of sending display text, the routes send **data** with `res.json`. The data lives in a normal JavaScript array of objects — the same arrays and objects from Series 3. Any app, website or phone can now ask this server for the students list and receive clean JSON it can use.

What you will see: `http://localhost:3000/students` shows the raw JSON of all three students in the browser, and `/students/count` shows `{"total":3}`. The browser displays it plainly because JSON is just text — a real frontend would read it with `fetch`, exactly like lesson 3.22.

Quick recap

- **Express** = the framework that turns raw Node servers into clean routes.
- One route per address: `app.get(address, (req, res) => { ... })`.
- `res.send` answers text or HTML; **`res.json` answers data** — that is an API.

- `app.listen(3000)` opens the door; unmatched addresses get an honest 404.

Common mistakes

1. **Forgetting to restart after every change** — Node loads your file once; editing `server.js` while it runs changes nothing on the port. The fix: Ctrl+C and run `node server.js` again after each edit (later, tools like nodemon automate this).
2. **Two routes, same address, confusion** — if you declare `app.get("/")` twice, only the **first** one ever runs; Express stops at the first match. The fix: one route per address and method; combine the answers inside it.
3. **Answering data with `res.send` and a hand-built string** — building JSON by joining text breaks the moment a name contains a quote. The fix: keep data in real objects and arrays, and always answer with `res.json` — it escapes and formats correctly for you.

Practice

1. Build a server with three routes: `/` sends your name, `/city` sends your city, and `/hobbies` answers with `res.json` and an array of three of your hobbies as objects like `{ "hobby": "cricket" }`. Test all three addresses in the browser.
2. Add a route `/students/top` to the Example 3 API that answers with only the student who has the highest marks (use a loop or `reduce` from lesson 3.11 to find them — do not hard-code the answer).

Practice quiz

1. What does app.get("/", handler) mean?

- It downloads the home page
- When a GET request arrives at /, run this function, which receives req (the request) and res (the response)
- It creates a file called /
- It listens on port /

Answer: When a GET request arrives at /, run this function, which receives req (the request) and res (the response)

2. What is the difference between res.send and res.json?

- There is none
- res.send answers text or HTML, res.json answers data: APIs send JSON, and res.json turns your object into JSON with the right settings automatically
- res.send is for APIs, res.json is for pages
- res.json only works for strings

Answer: res.send answers text or HTML, res.json answers data: APIs send JSON, and res.json turns your object into JSON with the right settings automatically

3. What does Express answer for an address with no matching route?

- The home page
- An honest 404, "Cannot GET /..."
- An empty 200 response
- It crashes

Answer: An honest 404, "Cannot GET /..."

Watch the Code Scope channel: https://www.youtube.com/@CodeScopePK — this lesson's own video joins the channel when it is recorded.

Read online: https://scopetreasure.com/notes/node-express-first-server · More free tools and notes at https://scopetreasure.com