# Node.js - Day 6: Express.js Introduction + Routes

## ■ Learning Objectives

- Understand what Express.js is and why it’s used.
- Install and set up an Express.js project.
- Create a simple Express server.
- Define routes for handling HTTP methods (GET, POST, etc.).
- Organize routes for better structure.

## ■ Session 1 (45 min) — Introduction to Express.js

**1. What is Express.js?**
Express.js is a minimal and flexible web application framework for Node.js that simplifies building servers and APIs.
It provides routing, middleware, and utilities for handling requests.
Think of Express as "Node.js made easier".

**2. Install & Setup**
mkdir express-intro cd express-intro npm init -y npm install express Create **index.js**:
const express = require('express'); const app = express(); app.listen(3000, () => { console.log('Server is running on http://localhost:3000'); }); Run:
node index.js

**3. Create Your First Route**
const express = require('express'); const app = express(); app.get('/', (req, res) => { res.send('Welcome to Express.js!'); }); app.get('/about', (req, res) => { res.send('This is the about page.'); }); app.listen(3000, () => console.log('Server running...')); Visit:
• http://localhost:3000/
• http://localhost:3000/about

**4. Send Different Types of Responses**
app.get('/json', (req, res) => { res.json({ name: 'Dielli', course: 'Node.js with Express' }); }); app.get('/html', (req, res) => { res.send('Express HTML ExampleHello Students!'); });

**■ Mini Exercise**
Add a route **/contact** returning a message and a route **/info** returning JSON with your name and favorite language.

## ■ Session 2 (45 min) — Express Routes & Parameters

**1. Understanding Routes**
A route defines how the app responds to a request.

app.METHOD(PATH, HANDLER); Example:
app.get('/', (req, res) => res.send('Hello!'));

**2. Route Parameters**
Used to capture dynamic values from URLs.
app.get('/users/:id', (req, res) => { res.send(`User ID is: ${req.params.id}`); }); Example:
GET /users/5 → "User ID is: 5"

**3. Query Parameters**
Sent after a '?' in the URL and accessed via **req.query**.
app.get('/search', (req, res) => { const q = req.query.q; res.send(`You searched for: ${q}`); }); Example:
http://localhost:3000/search?q=express → "You searched for: express"

**4. Handling Different HTTP Methods**
app.post('/add', (req, res) => res.send('New data received!')); app.put('/update', (req, res) => res.send('Data updated!')); app.delete('/delete', (req, res) => res.send('Data deleted!')); Test using Postman or Thunder Client.

**■ Mini Project — Student Info API**
const express = require('express'); const app = express(); const students = [ { id: 1, name: 'Ardit' }, { id: 2, name: 'Nora' } ]; app.get('/', (req, res) => res.send('Welcome to Student Info API')); app.get('/students', (req, res) => res.json(students)); app.get('/students/:id', (req, res) => { const student = students.find(s => s.id == req.params.id); res.json(student || { message: 'Student not found' }); }); app.listen(3000, () => console.log('API running...'));

## ■ Recap Discussion

- What's the difference between Node.js and Express.js?
- How does Express simplify routing?
- What are route and query parameters?
- Why use different HTTP methods?

## ■ Homework

Build a simple Book API:
• /books → List all books
• /books/:id → Show details of a single book
• /addbook → POST new book
• /delete/:id → DELETE a book
Use an array for data:
const books = [ { id: 1, title: 'Clean Code', author: 'Robert Martin' }, { id: 2, title: 'Eloquent JavaScript', author: 'Marijn Haverbeke' } ];