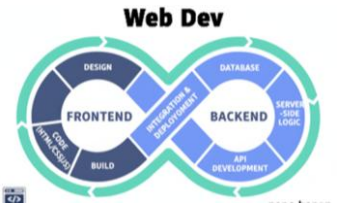
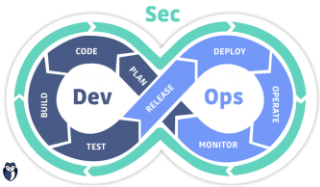




Day 44

"Web Development + Security"

Introduction to Express Js:

What is Express.js?

Express.js is a backend web application framework for Node.js. It helps you build web servers and APIs easily — handling routes, requests, and responses efficiently.

Why Express.js?

Feature	Description
Minimal & Fast	Lightweight, unopinionated — you add only what you need.
Routing	Easy way to define endpoints (like /home, /login, etc.).
Middleware	Functions that run before your route logic — used for logging, authentication, etc.
Error Handling	Built-in support for handling errors gracefully.
Integration	Works smoothly with databases (MongoDB, MySQL, etc.).

Why Express.js Was Created

Express.js was built on top of Node.js to simplify the process of:

- Building servers
- Managing routes
- Handling requests and responses
- Adding middleware (like authentication, logging, etc.)

What Express Adds to Node.js

Feature	Node.js	Express.js
Server creation	Manual with http module	One line express()
Routing	Manual if/else checks	Simple .get(), .post()
Middleware	You build from scratch	Built-in system
JSON handling	Manual parsing	express.json()
Error handling	Manual try-catch	Centralized system
Scalability	Complex for large apps	Organized with Routers & Middleware

Installing express.js:

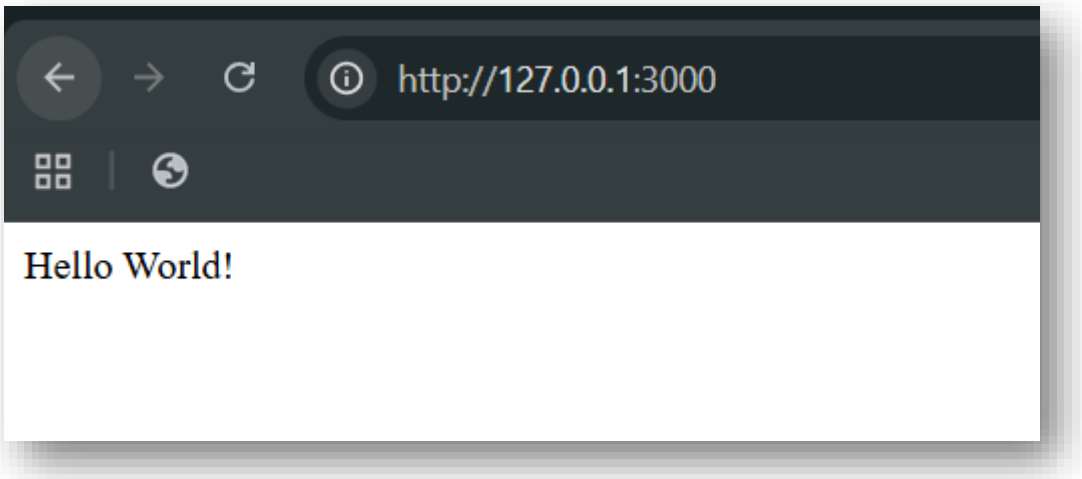
```
PS E:\FullStackDevelopment\Day41-50\Day44> npm i express

added 68 packages, and audited 69 packages in 3s

16 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS E:\FullStackDevelopment\Day41-50\Day44>
```

Now, creating a minimal application in express.js: just copied and pasted the code from the official documentation.

Main.js:

```
JS main.js > ...
1   const express = require('express')
2   const app = express()
3   const port = 3000
4
5   app.get('/', (req, res) => {
6     res.send('Hello World!')
7   })
8
9   app.listen(port, () => {
10    console.log(`Example app listening on port ${port}`)
11  })
12
```

Output:



How It Works

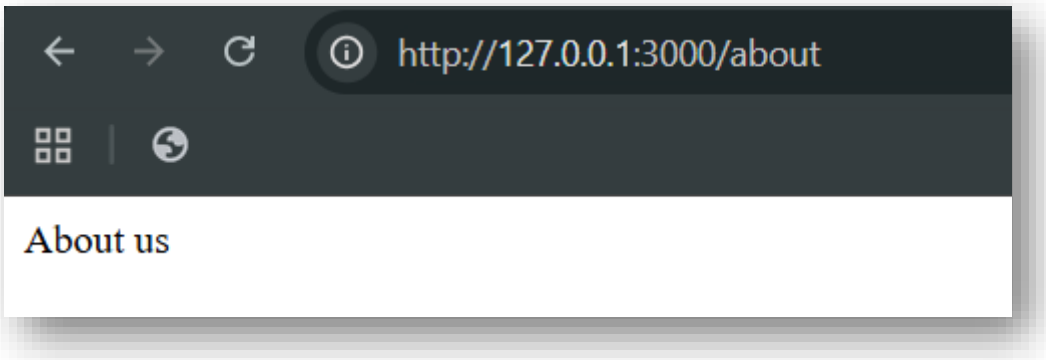
1. require("express") → Imports the Express library.
2. express() → Creates an app object (your server).
3. app.get("/", callback) → Sets up a route for GET requests at /.
4. res.send() → Sends back a response to the browser.
5. app.listen() → Starts your server on a specific port.

Now, we can add other locations too:

Code:

```
JS main.js > ...
1   const express = require('express')
2   const app = express()
3   const port = 3000
4
5   app.get('/', (req, res) => {
6     res.send('Hello World!')
7   })
8   app.get('/about', (req, res) => {
9     res.send('About us')
10  })
11
12  app.listen(port, () => {
13    console.log(`Example app listening on port ${port}`)
14  })
```

Output:



So, do we need to make so many endpoints one by one? No. We can use parameters and queries to do so.

Example:

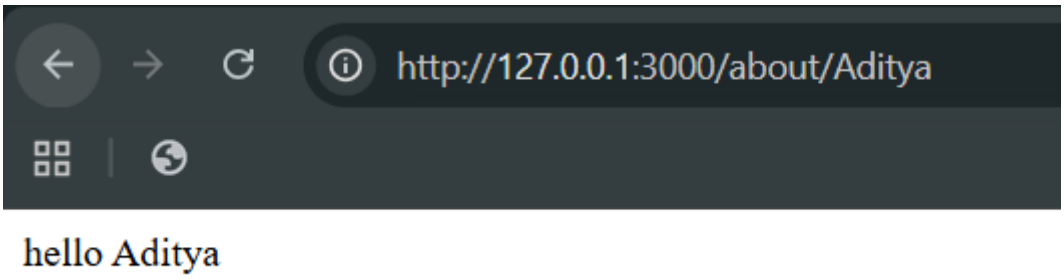
```
JS main.js > ...
16  const express = require('express')
17  const app = express()
18  const port = 3000
19
20  app.get('/', (req, res) => {
21    res.send('Hello World!')
22  })
23  app.get('/about', (req, res) => {
24    res.send('About us')
25  })
26  app.get('/about/:slug', (req, res) => {
27    res.send(`hello ${req.params.slug}`)
28  })
29
30  app.listen(port, () => {
31    console.log(`Example app listening on port ${port}`)
32  })
33
```

Now, to get the object: we need to log it

Main.js:

```
JS main.js > ...
16  const express = require('express')
17  const app = express()
18  const port = 3000
19
20  app.get('/', (req, res) => {
21    res.send('Hello World!')
22  })
23  app.get('/about', (req, res) => {
24    res.send('About us')
25  })
26  app.get('/about/:slug', (req, res) => {
27      console.log(req) //added
28    res.send(`hello ${req.params.slug}`)
29  })
30
31  app.listen(port, () => {
32    console.log(`Example app listening on port ${port}`)
33  })
```

Output:



Terminal: clearly request is sent as an object to the terminal

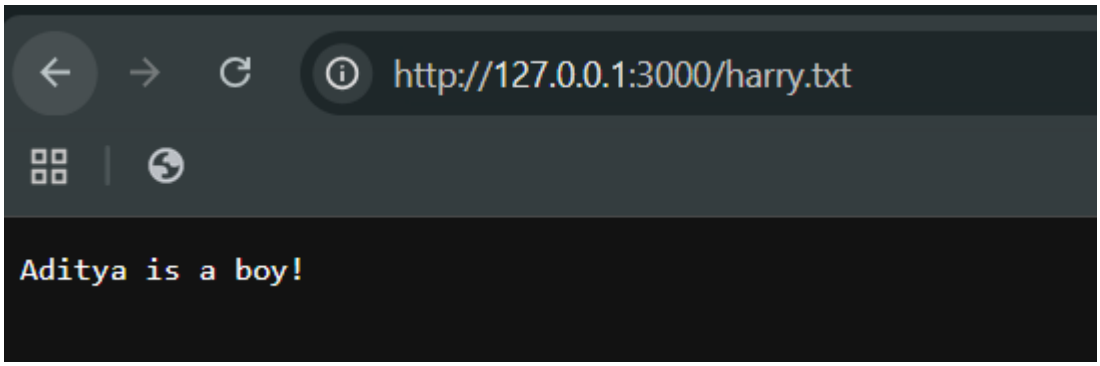
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS E:\FullStackDevelopment\Day41-50\Day44> node "e:\FullStackDevelopment\Day41-50\Day44\main.js"
Example app listening on port 3000
<ref *2> IncomingMessage {
  _events: {
    close: undefined,
    error: undefined,
    data: undefined,
    end: undefined,
    readable: undefined
  },
  _readableState: ReadableState {
    highWaterMark: 16384,
    buffer: [],
    bufferIndex: 0,
    length: 0,
```

Now, suppose we want to give the file harry.txt to the public, then we will first keep it in the 'public' folder and then do as shown below: it will be served as a static file.

Code:

```
16  const express = require('express')
17  const app = express()
18  const port = 3000
19  app.use(express.static('public'))
20
21  app.get('/', (req, res) => {
22    res.send('Hello World!')
23  })
24  app.get('/about', (req, res) => {
25    res.send('About us')
26  })
27  app.get('/about/:slug', (req, res) => {
28      console.log(req) //added
29    res.send(`hello ${req.params.slug}`)
30  })
31
32  app.listen(port, () => {
33    console.log(`Example app listening on port ${port}`)
34  })
35
```

Output:



--The End--