**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**

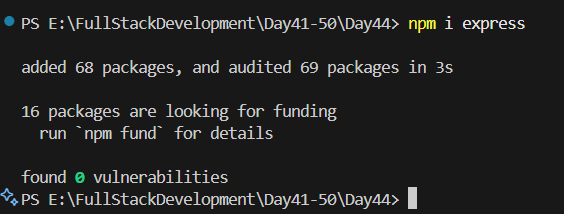
Express.js was built on top of Node.js to simplify the process of:

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

**What Express Adds to Node.js**

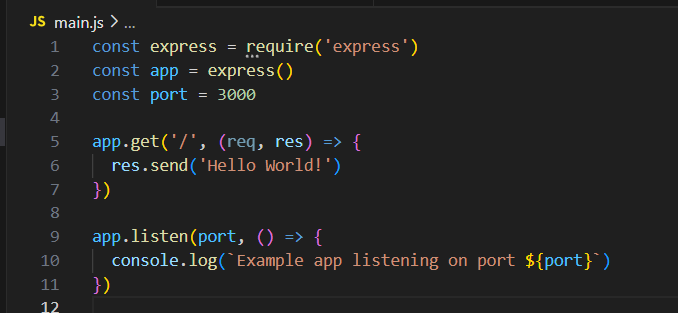
| **Feature** | **Node.js** | **Express.js** |
| --- | --- | --- |
| Server creation | Manual with http module | One line express() |
| Routing | Manual if/else checks | Simple .get(), .post() |
| Middleware | You build from scratch | Built-in system |
| JSON handling | Manual parsing | express.json()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

Installing express.js:

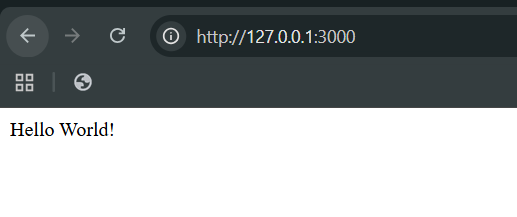


Now, creating a minimal application in express.js: just copied and pasted the code from the official documentation.

Main.js:



Output:

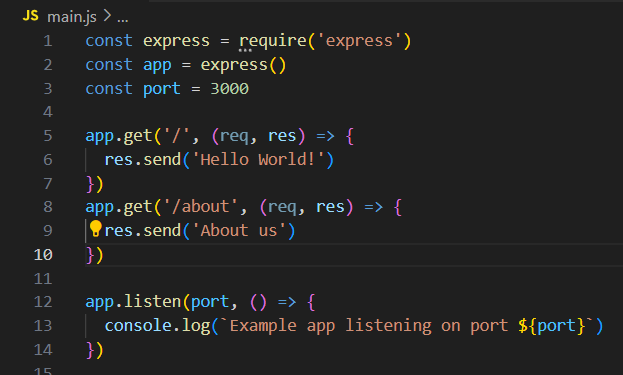


**How It Works**

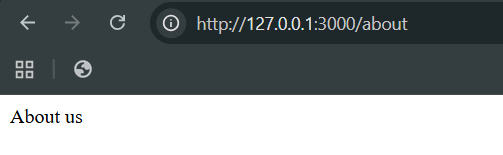
1. require("express") → Imports the Express library.
2. express() → Creates an app object (your server).
3. app.get("/", callback) → Sets up a route for GET requests at /.
4. res.send() → Sends back a response to the browser.
5. app.listen() → Starts your server on a specific port.

Now, we can add other locations too:

Code:



Output:

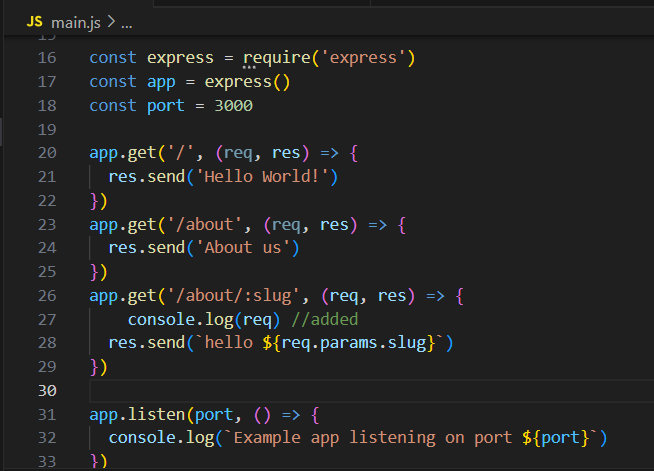


So, do we need to make so many endpoints one by one? No. We can use parameters and queries to do so.

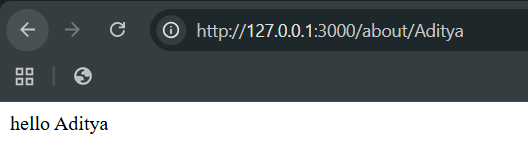
Example: 

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

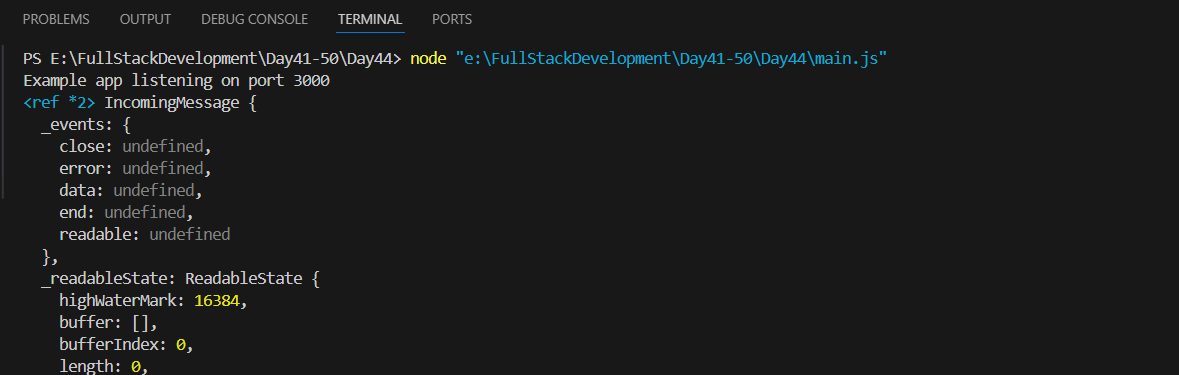
Main.js:



Output:



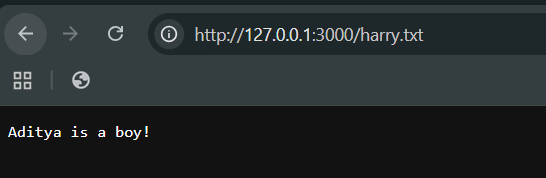
Terminal: clearly request is sent as an object to the terminal



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:  


Output:



--The End--