Learning **Express.js** as a beginner can be exciting because it's the most popular framework for building **web servers in Node.js**. I'll give you a **step-by-step roadmap**, along with resources, examples, and tips.

Step 1: Prerequisites

Before learning Express.js, you should know:

1. **JavaScript Basics**
   - Variables, functions, loops, objects, arrays.
   - ES6 features: arrow functions, template literals, destructuring.
2. **Node.js Basics**
   - How to create a simple Node.js script.
   - `require` / `import` syntax.
   - `npm` (Node Package Manager) basics: installing packages, `package.json`.
3. **HTTP Concepts**
   - Requests (`GET`, `POST`) and responses.
   - Status codes (200, 404, 500, etc.)

 If you're new to Node.js, start with a simple "Hello World" server:

```
const http = require("http");

const server = http.createServer((req, res) => {
    res.writeHead(200, { "Content-Type": "text/plain" });
    res.end("Hello World!");
});

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

Step 2: Set Up Express.js

1. Create a project folder:

```
mkdir express-demo
cd express-demo
npm init -y
```

2. Install Express:

```
npm install express
```

Step 3: Your First Express Server

Create `app.js`:

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

// Define a route
app.get("/", (req, res) => {
    res.send("Hello, Express!");
});

// Start server
app.listen(3000, () => {
    console.log("Server running on http://localhost:3000");
});
```

Test it:

- Run in terminal: `node app.js`
- Open browser: `http://localhost:3000` → you'll see **Hello, Express!**

Step 4: Learn Routing

Express uses **routes** to handle HTTP requests.

```
// GET route
app.get("/about", (req, res) => {
    res.send("About page");
});

// POST route
app.post("/submit", (req, res) => {
    res.send("Form submitted");
});
```

- `app.get()` → for GET requests
- `app.post()` → for POST requests
- Other methods: `put`, `delete`, `patch`.

Step 5: Handling JSON & Form Data

Install body parser (or use Express's built-in parser):

```
app.use(express.json()); // Parse JSON
app.use(express.urlencoded({ extended: true })); // Parse form data
```

Example:

```
app.post("/data", (req, res) => {
    console.log(req.body);
    res.send("Data received");
});
```

- Send POST request with JSON `{ "name": "Alice" }` → logged in console.

Step 6: Learn Middleware

Middleware functions run **before your route handlers**. They can:

- Log requests
- Authenticate users
- Handle errors

Example:

```
// Simple middleware
app.use((req, res, next) => {
    console.log(`Request URL: ${req.url}`);
    next(); // move to next middleware or route
});
```

Step 7: Use Router for Modular Routes

```
const userRouter = express.Router();

userRouter.get("/", (req, res) => res.send("User list"));
userRouter.get("/:id", (req, res) => res.send(`User ${req.params.id}`));

app.use("/users", userRouter);
```

- Access `http://localhost:3000/users/1` → **User 1**

Step 8: Learn Error Handling

```
app.use((req, res, next) => {
    res.status(404).send("Page not found");
});

app.use((err, req, res, next) => {
    console.error(err.stack);
    res.status(500).send("Something went wrong!");
});
```

Step 9: Practice Projects

1. **Simple To-Do API**
    - CRUD operations (`create`, `read`, `update`, `delete`) using Express.
2. **Blog API**
    - Use routes for posts, comments.
3. **User Authentication**
    - Learn `express-session` or `JWT` for login/logout.

Step 10: Resources for Beginners

- **Official Docs:** https://expressjs.com/
- **FreeCodeCamp Tutorial:** "Express.js Full Course" on YouTube
- **Books:** "Learning Node.js Development" or "Express in Action"
- **Practice Platforms:**
    - Glitch → experiment online
    - Replit → run Express projects in browser