From Zero to Express: Your First Full-Stack Node.js Setup

Total Duration: 2 Hours

Audience: Beginners with basic programming knowledge (HTML/CSS/JS)
Tools Needed:

- Node.js (latest LTS version)
- Code editor (e.g., VS Code)
- Terminal/Command Line
- Git (optional)

Learning Outcomes

By the end of this session, learners will be able to:

1. Install Node.js and npm correctly.
2. Understand what npm is and how to use it.
3. Initialize a Node.js project.
4. Install and set up Express.js.
5. Render dynamic views using EJS.
6. Connect and interact with a SQL database.
7. Prepare a skeleton for a full-stack web app.

Agenda & Detailed Breakdown

0:00 - 0:10 — Introduction & Goals

- Quick intro to Node.js and npm: what they are and why we use them.
- Overview of the full stack flow: Node.js + Express + EJS + SQL.
- What we'll build by the end of the session.

0:10 - 0:30 — Installing Node.js & npm

Step-by-step:

1. Go to https://nodejs.org/
2. Download **LTS version**

3. Install it (demo for Windows/macOS/Linux if needed)
4. Verify with:

```
node -v

npm -v
```

Concepts Covered:

- What is Node.js (V8 + runtime)?
- npm vs npx
- Global vs local installs

0:30 - 0:45 — Initialize Node Project

Step-by-step:

1. Create a project folder:

```
mkdir myapp && cd myapp
```

2. Initialize npm:

```
npm init -y
```

3. Explain package.json

Concepts Covered:

- Dependency management
- devDependencies vs dependencies

---

3. Install and Set Up Express.js

Express.js is a web application framework for Node.js, designed for building web applications and APIs.

Steps:

1. **Install Express.js:**
    - In the terminal, install Express.js:

```
npm install express
```

2. **Create the Main Server File:**

- In the myapp folder, create a new file named index.js.
- Open index.js in VS Code and add the following code:

```
const express = require('express');
const app = express();
const port = 3000;

app.get('/', (req, res) => {
  res.send('Hello World!');
});

app.listen(port, () => {
  console.log(`Server is running at http://localhost:${port}`);
});
```

3. **Run the Server:**
   - In the terminal, start the server:

```
node index.js
```

   - Open a web browser and navigate to http://localhost:3000. You should see "Hello World!" displayed.

For a detailed tutorial on setting up an Express.js server, refer to onlineresource:

---

4. Integrate EJS for Templating

EJS (Embedded JavaScript) allows you to generate HTML markup with plain JavaScript.

Steps:

1. **Install EJS:**
   - In the terminal, install EJS:

```
npm install ejs
```

2. **Configure Express to Use EJS:**
   - Update index.js to include EJS:

```
const express = require('express');
const app = express();
const port = 3000;

app.set('view engine', 'ejs');

app.get('/', (req, res) => {
  res.render('index', { title: 'Home' });
});

app.listen(port, () => {
  console.log(`Server is running at http://localhost:${port}`);
});
```

3. **Create Views Folder and EJS File:**
   - In the myapp folder, create a new folder named views.
   - Inside the views folder, create a file named index.ejs.
   - Add the following HTML to index.ejs:

```
<!DOCTYPE html>
<html>
<head>
  <title><%= title %></title>
</head>
<body>
  <h1>Welcome to <%= title %> Page</h1>
</body>
</html>
```

4. **Run the Server:**
   - Restart the server:

```
node index.js
```

- Navigate to http://localhost:3000 in your web browser. You should see the rendered HTML with the title "Home".

For more information on using EJS with Express,

---

5. Connect to a MySQL Database

To interact with a MySQL database from your Node.js application:

Steps:

1. **Install MySQL Module:**
   - In the terminal, install the MySQL module:

```
npm install mysql2
```

2. **Set Up Database Connection:**
   - Update index.js to include the MySQL connection:

javascript

CopyEdit

```
const express = require('express');

const mysql = require('mysql2');

const app = express();

const port = 3000;


app.set('view engine', 'ejs');


const db = mysql.createConnection({

 host: 'localhost',

 user: 'your-username',

 password: 'your-password',

 database: 'your-database-name'

});
```

```
db.connect((err) => {
  if (err) {
    console.error('Database connection failed: ' + err.stack);
    return;
  }
  console.log('Connected to database.');
});

app.get('/', (req, res) => {
  res.render('index', { title: 'Home' });
});

app.listen(port, () => {
  console.log(`Server is running at http://localhost:${port}`);
});
```

- Replace 'your-username', 'your-password', and 'your-database-name' with your MySQL credentials.

3. **Create a Sample Table and Data:**
    - Access your MySQL database using a client like **phpMyAdmin** or the MySQL command line.
    - Create a new table:

```
CREATE TABLE users (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(100),
  email VARCHAR(100)
);
```

---

6. Fetch Data from SQL and Render with EJS

We'll now update our Express app to fetch data from the users table and display it in an EJS template.

Steps:

1. Insert Sample Data (if not already done):

Open MySQL terminal or your GUI tool (like phpMyAdmin or MySQL Workbench), and run:

```
INSERT INTO users (name, email)

VALUES

 ('Alice Johnson', 'alice@example.com'),

 ('Bob Smith', 'bob@example.com');
```

---

2. Create a Route to Fetch and Display Users

In your index.js, add a new route:

javascript

CopyEdit

```
app.get('/users', (req, res) => {

 db.query('SELECT * FROM users', (err, results) => {

  if (err) {

   return res.status(500).send('Database query error');

  }

  res.render('users', { users: results });

 });

});
```

---

3. Create users.ejs Template

Inside the views folder, create a new file called users.ejs and add:

```
<!DOCTYPE html>

<html>
```

```
<head>
  <title>User List</title>
</head>
<body>
  <h1>Registered Users</h1>
  <ul>
    <% users.forEach(user => { %>
      <li><strong><%= user.name %></strong> - <%= user.email %></li>
    <% }) %>
  </ul>
</body>
</html>
```

---

4. Test Your Route

- Start your server if it's not already running:

```
node index.js
```

- Visit:
  http://localhost:3000/users

You should see a nicely rendered list of users pulled from your MySQL database.

---

7. Final Project Structure

By now, your project folder (myapp) should look like this:

```
myapp/
 ├── node_modules/
 ├── views/
│   ├── index.ejs
│   └── users.ejs
 ├── package.json
 ├── package-lock.json
└── index.js
```

---

8. Learning Recap for Learners

Let’s summarize what learners achieved in this 2-hour hands-on session:

Key Concepts Covered:

- Installing Node.js and npm using the official installer.
- Using **VS Code's terminal** for Node/npm commands.
- Initializing a Node.js project (npm init).
- Installing and using Express.js.
- Rendering dynamic HTML using EJS.
- Setting up and connecting to a MySQL database.
- Performing basic SQL queries from Node.js.
- Structuring a full-stack starter app.