What have we learned so far?

> We have learned the basics of Javascript by executing our code inside the NodeJS environment

> We also learnt about Node Package Manager (NPM), which is a tool that lets us download amazing tools and libraries

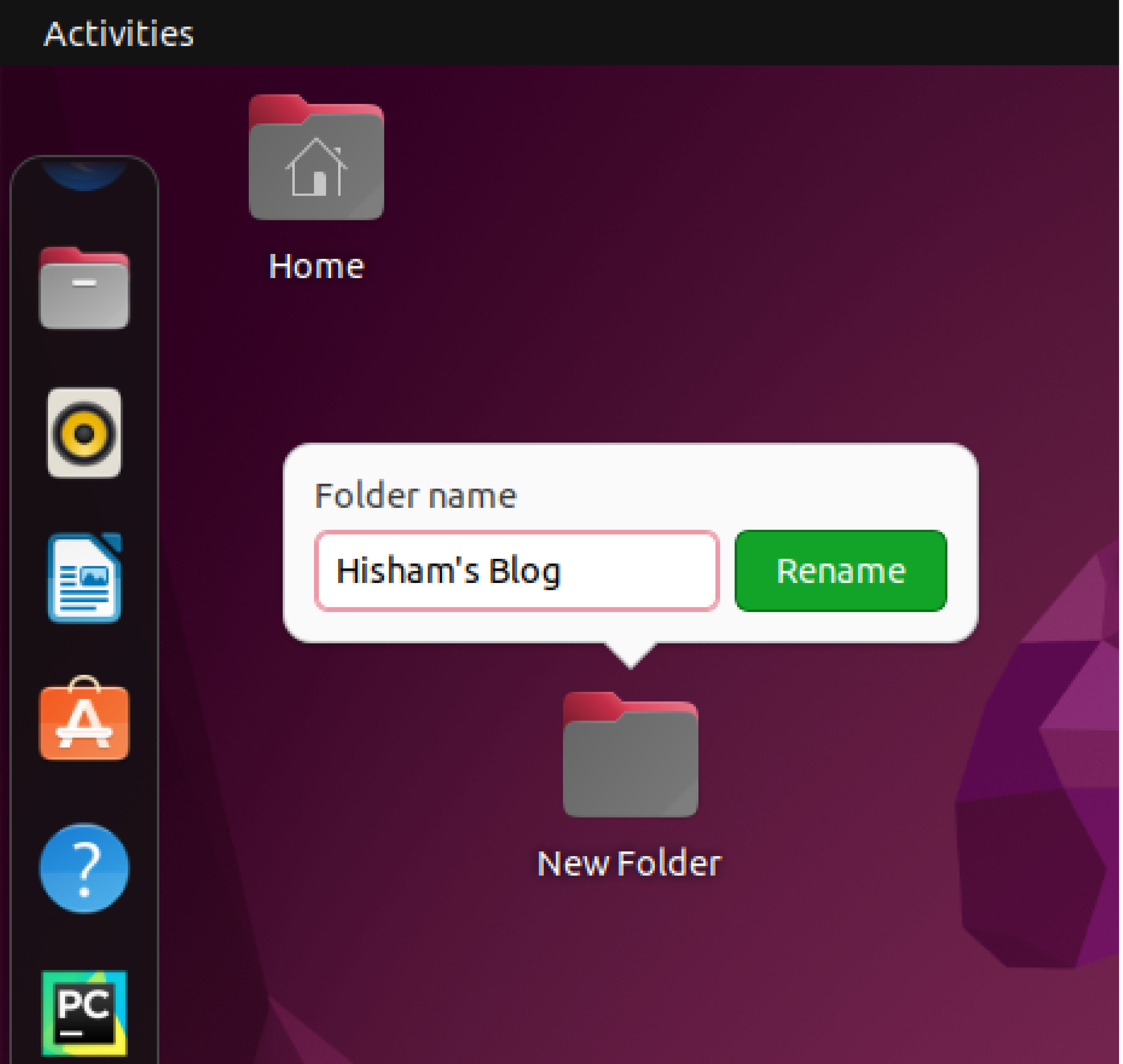
But hey we haven’t really built a website yet?

To start building a website using Javascript inside NodeJS, we need a library!

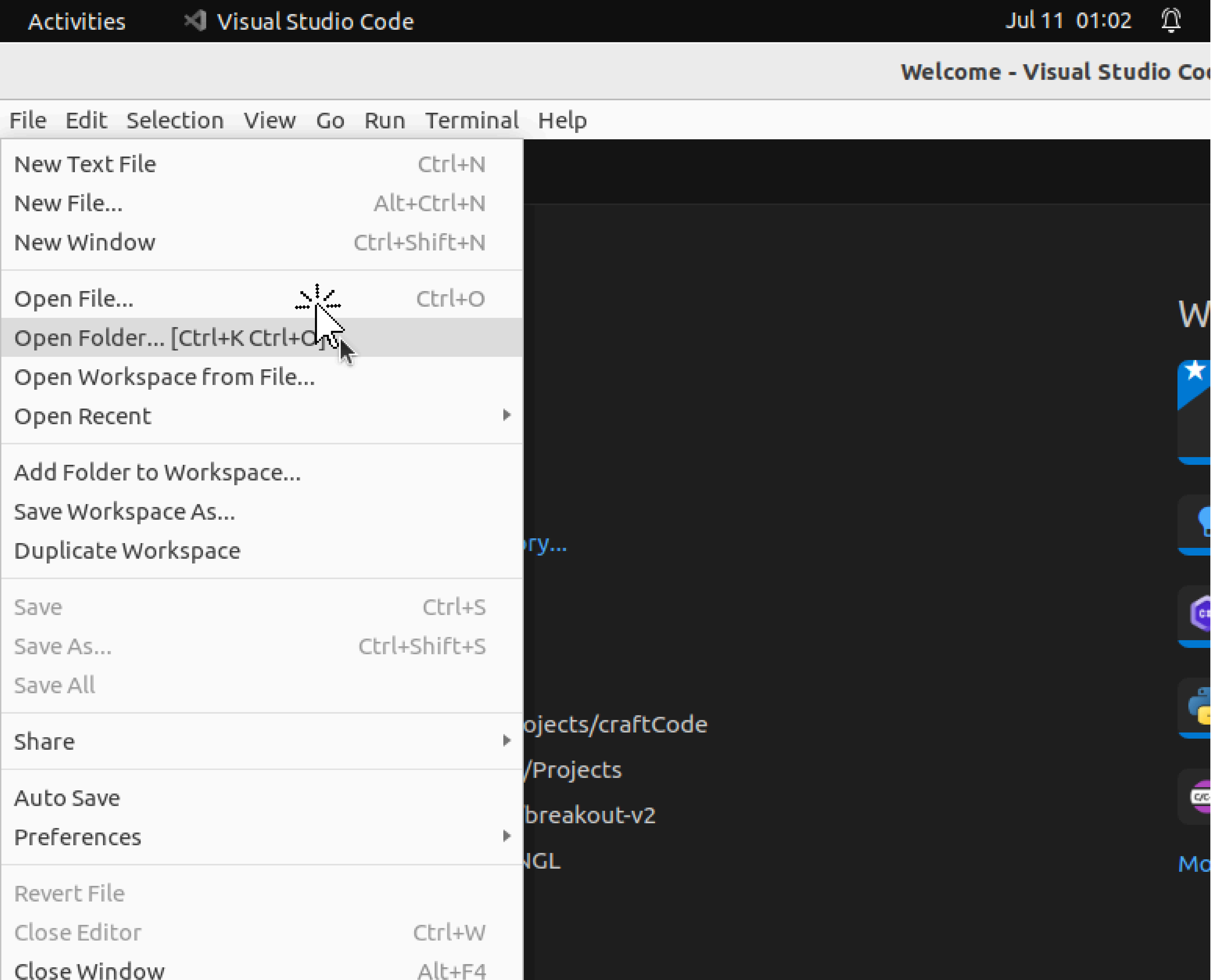
This is where ExpressJS comes into picture!

Express is a library that helps us build website, either full stack or backend for websites

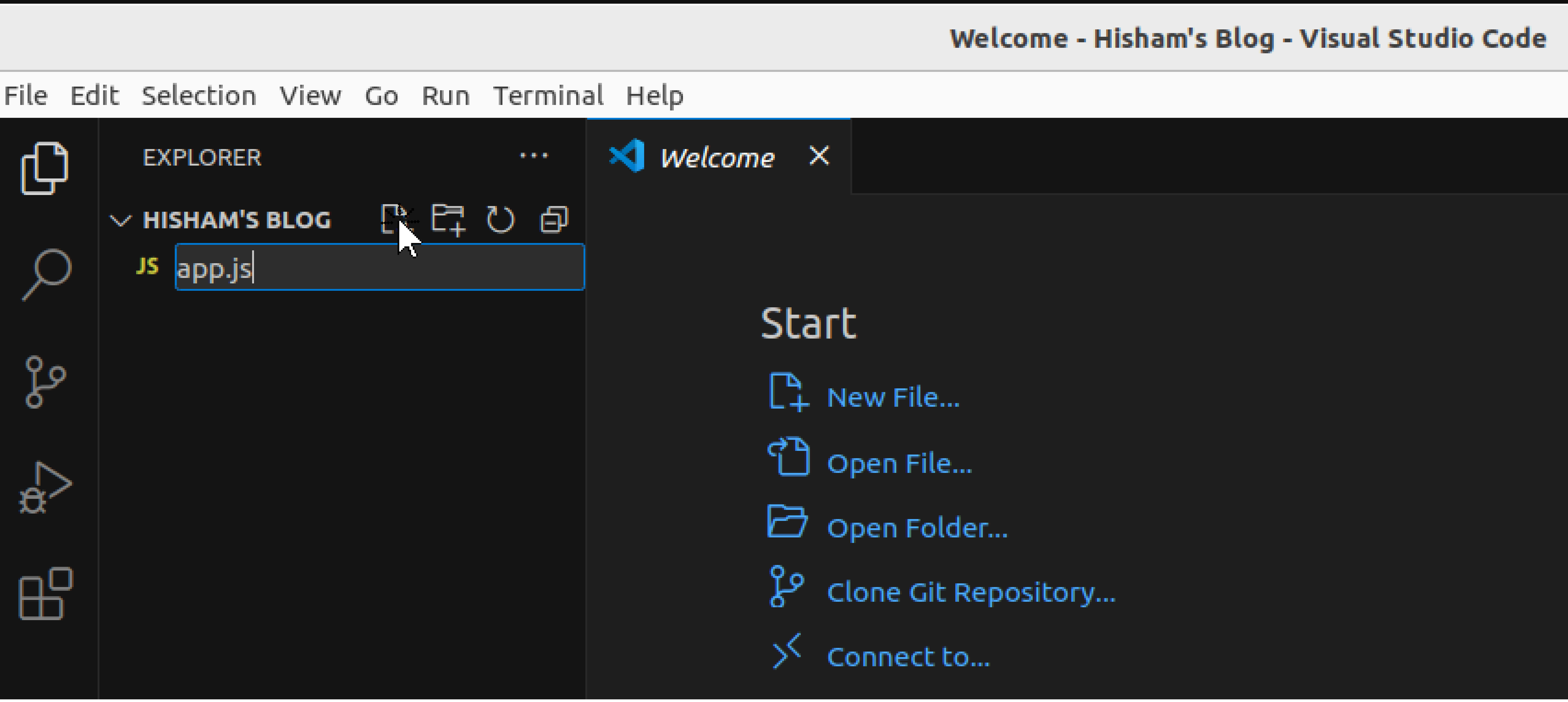
Chapter 1
Installing and Setting up a basic web server with Express



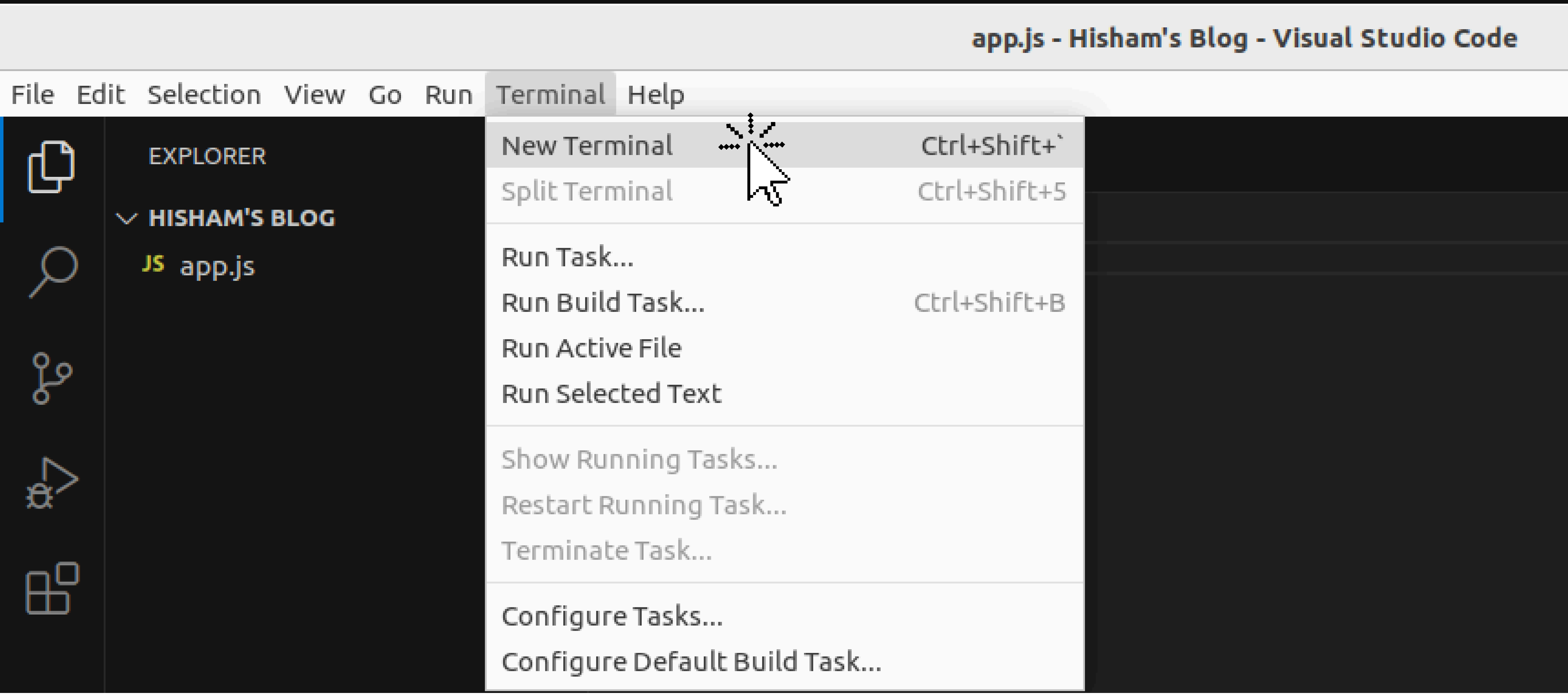
Create your project folder in the the Desktop



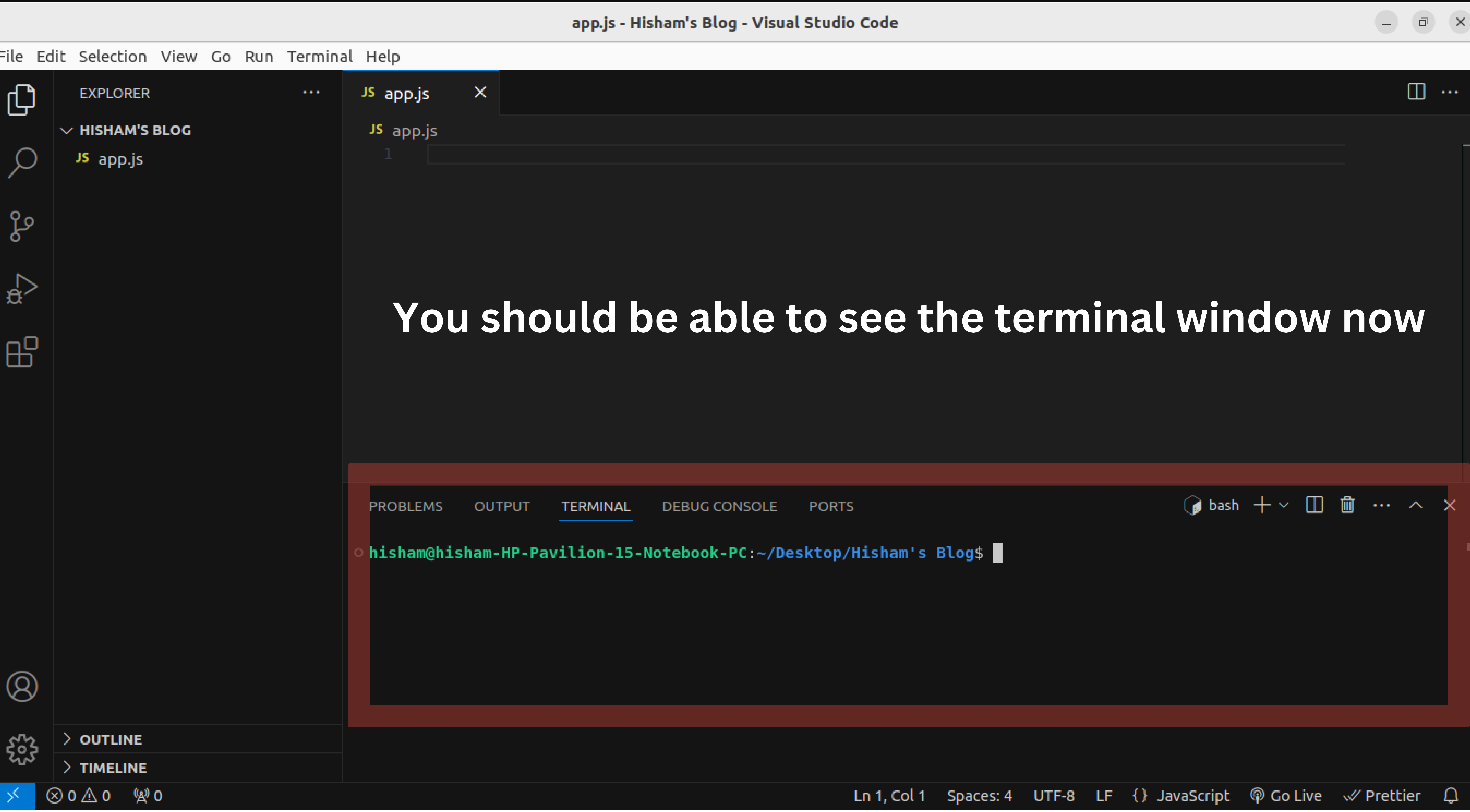
Open the folder with your favorite text-editor or IDE (Here I am using VSCode)



Once opened, create a file app.js



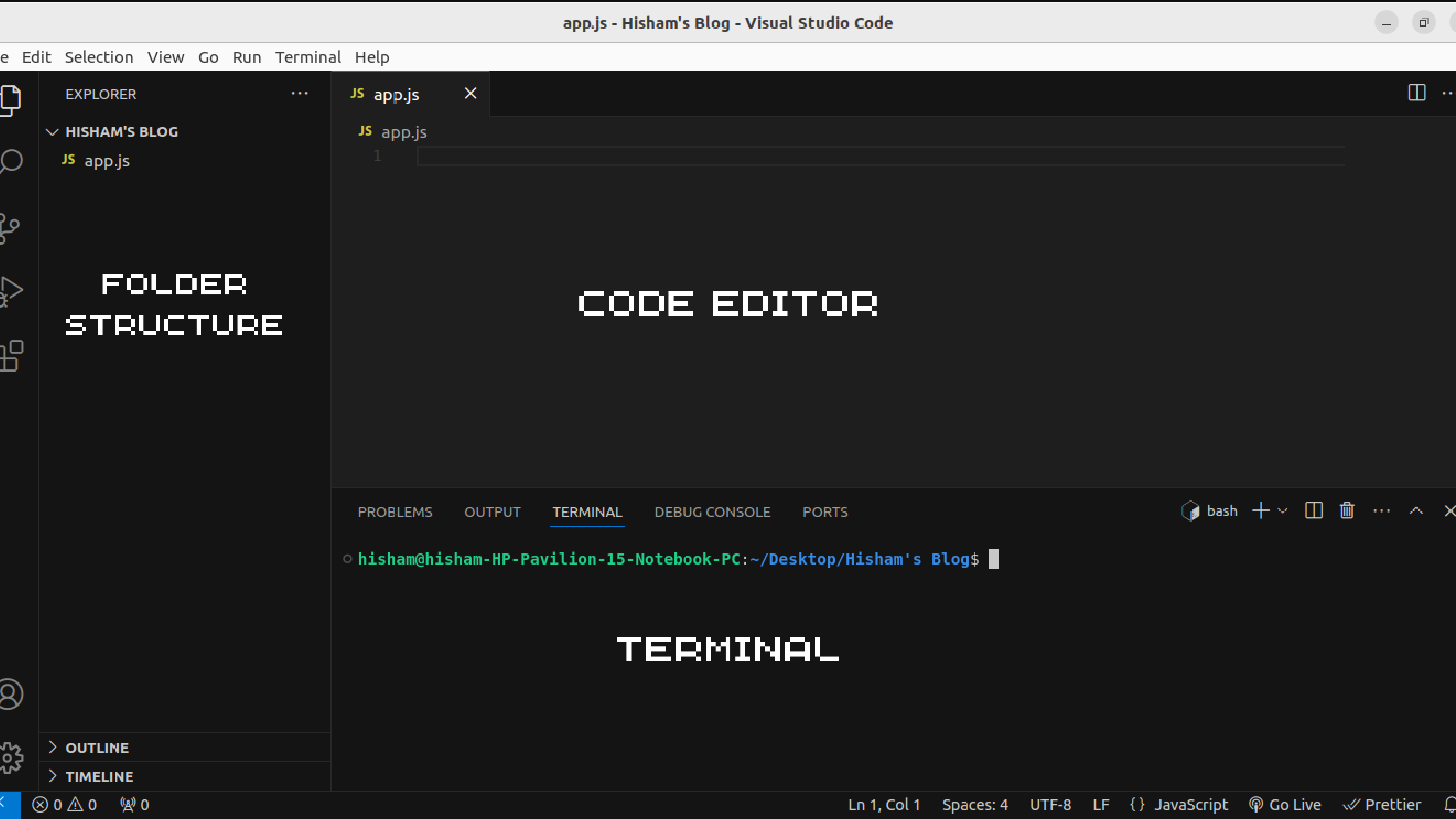
Let us open a new integrated terminal so that we can compile and run our javascript code and install libraries



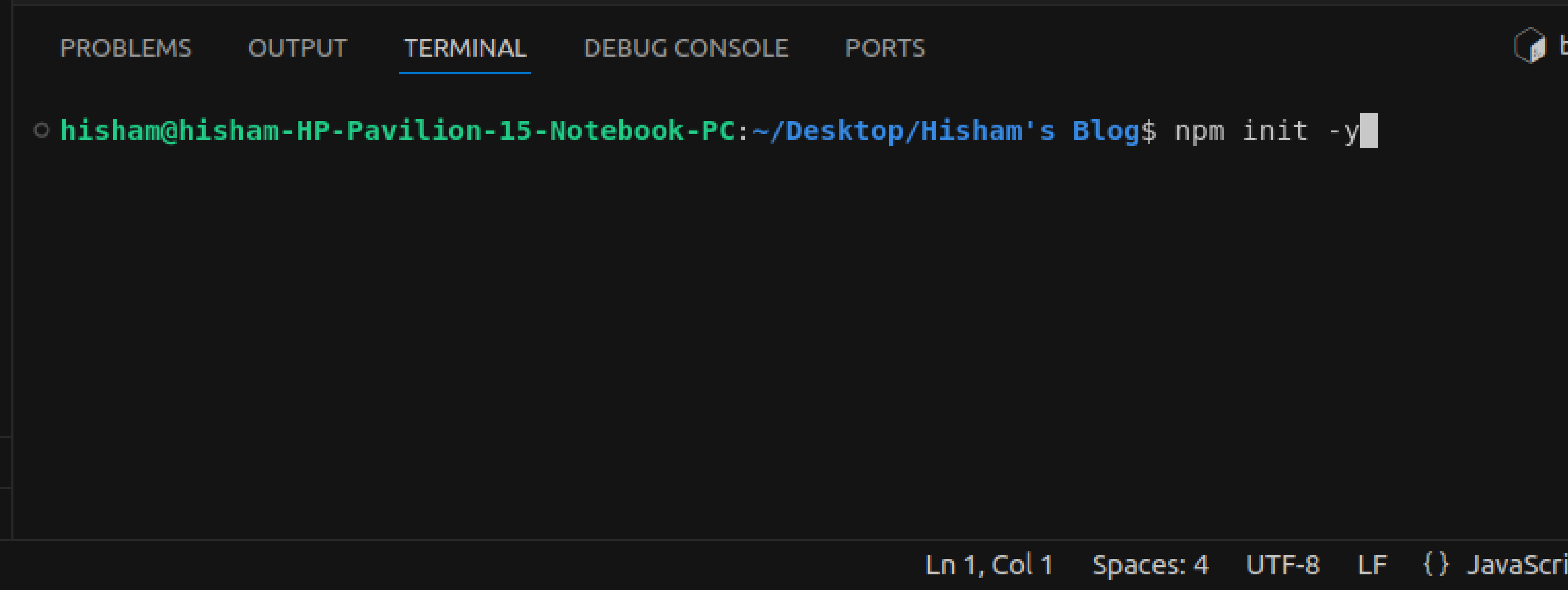
You should be able to see the terminal window now

app.js - Hisham's Blog - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
HISHAM'S BLOG
app.js
OUTLINE
TIMELINE
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE PORTS
bash
Ln 1, Col 1 Spaces: 4 UTF-8 LF JavaScript Go Live Prettier

```
hisham@hisham-HP-Pavilion-15-Notebook-PC:~/Desktop/Hisham's Blog$
```

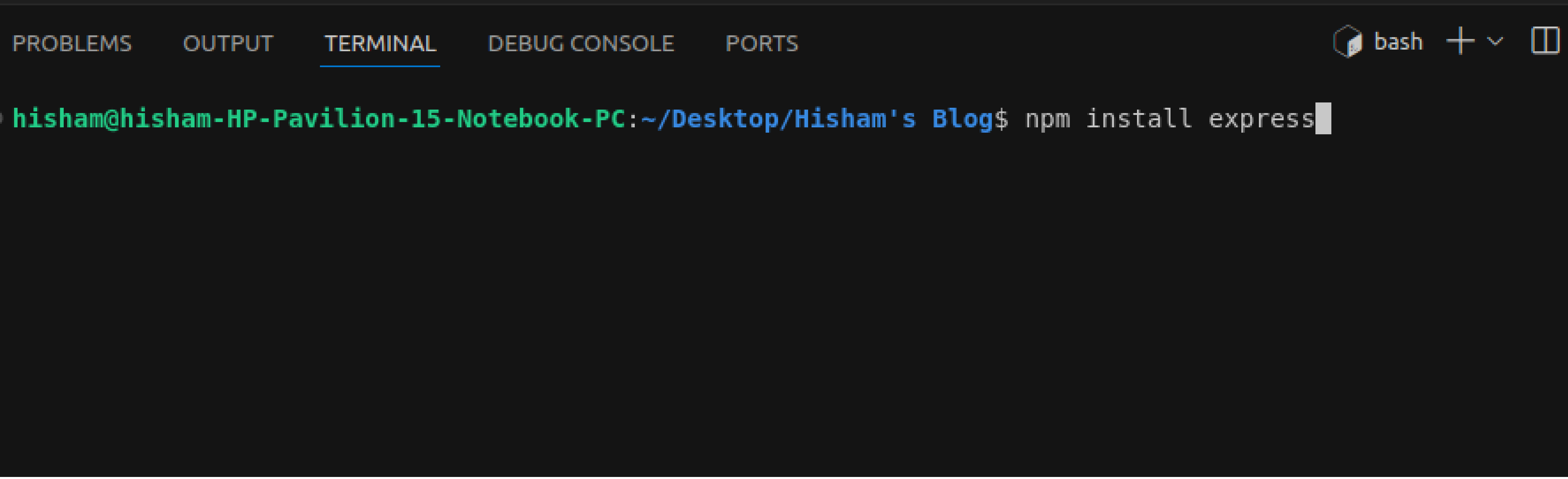

app.js - Hisham's Blog - Visual Studio Code
e Edit Selection View Go Run Terminal Help
EXPLORER
HISHAM'S BLOG
app.js
FOLDER STRUCTURE
CODE EDITOR
OUTLINE
TIMELINE
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE PORTS
bash
TERMINAL
Ln 1, Col 1 Spaces: 4 UTF-8 LF { } JavaScript Go Live Prettier

```
hisham@hisham-HP-Pavilion-15-Notebook-PC:~/Desktop/Hisham's Blog$
```



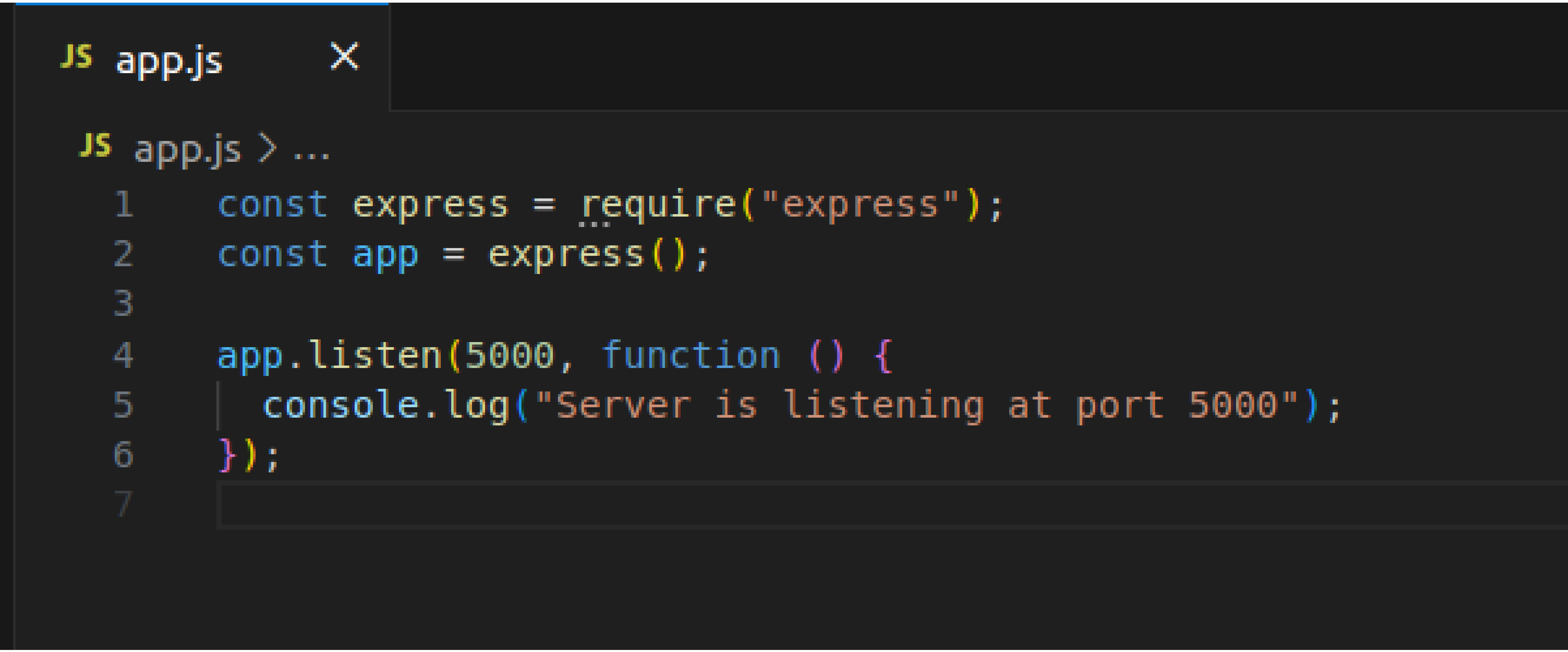
```
hisham@hisham-HP-Pavilion-15-Notebook-PC:~/Desktop/Hisham's Blog$ npm init -y
```

Run the above command in your terminal to setup a default node project



```
hisham@hisham-HP-Pavilion-15-Notebook-PC:~/Desktop/Hisham's Blog$ npm install express
```

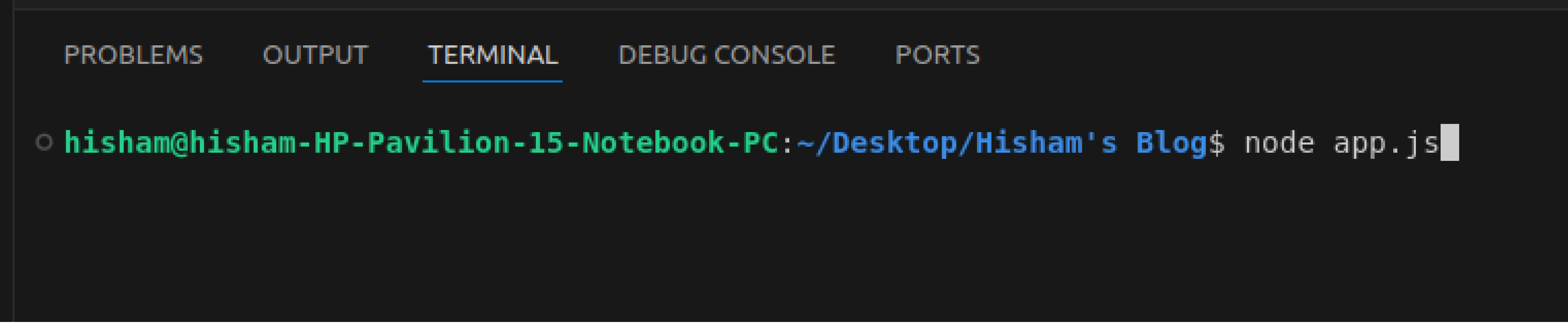
Run the above command to install ExpressJS library


app.js
app.js > ...

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

app.listen(5000, function () {
  console.log("Server is listening at port 5000");
});
```

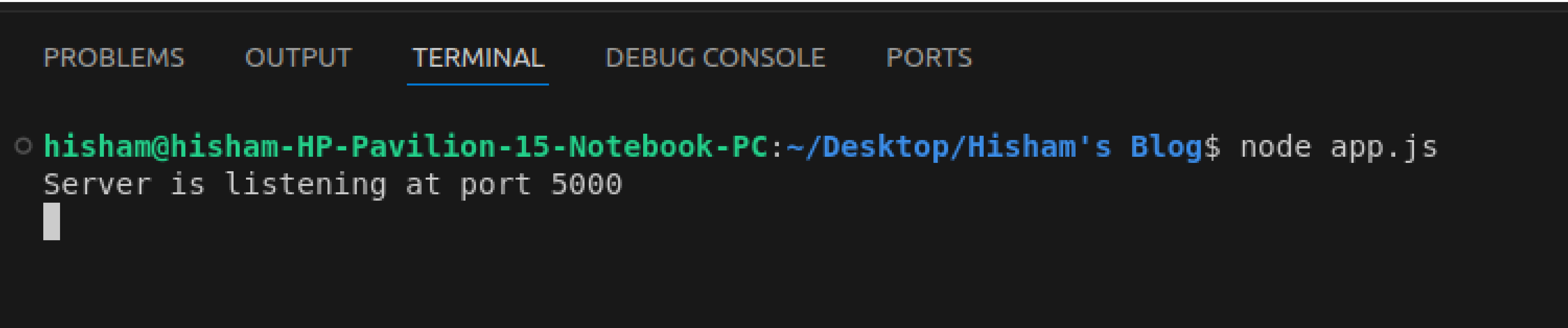
Write the below code to spin up your own web server

Now let us run our project using the command node app.js



```
hisham@hisham-HP-Pavilion-15-Notebook-PC:~/Desktop/Hisham's Blog$ node app.js
```

Output:



```
hisham@hisham-HP-Pavilion-15-Notebook-PC:~/Desktop/Hisham's Blog$ node app.js
Server is listening at port 5000
```

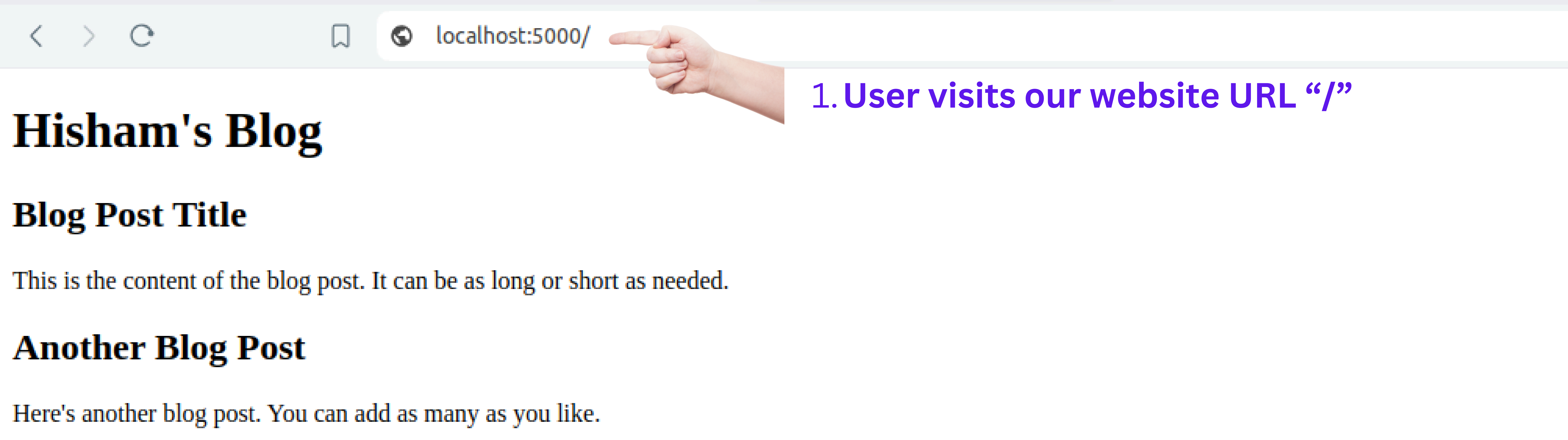
Chapter 2
Setting up our first route and our website’s landing page

In the previous chapter,

- We learnt about an amazing library called ExpressJS that helps us create dynamic backend and full stack web applications
- We installed ExpressJS in our project using NPM and then got a simple web server up and running

In this chapter,

- We will create a route for our website
- Whenever user visits that route, we display our landing page or any HTML page we like



Hisham's Blog

Blog Post Title

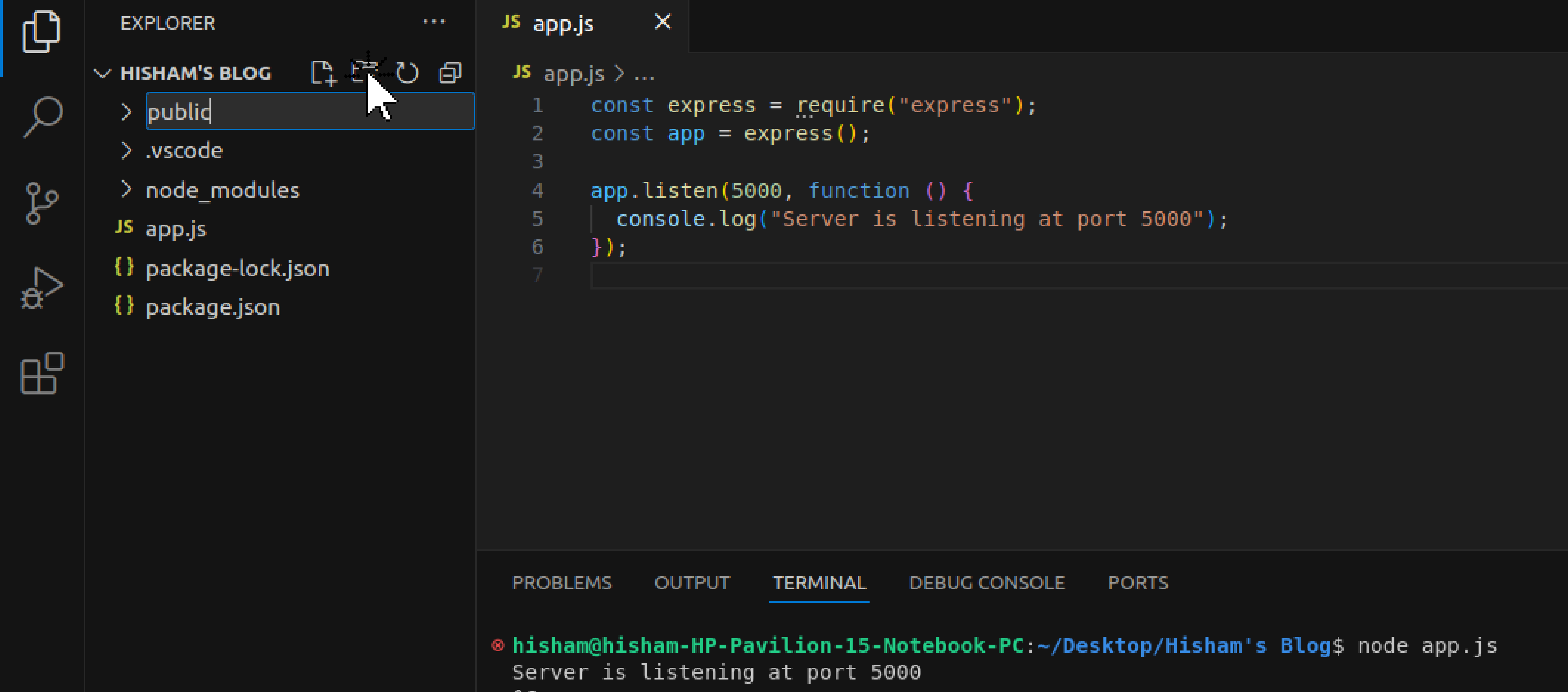
This is the content of the blog post. It can be as long or short as needed.

Another Blog Post

Here's another blog post. You can add as many as you like.

1. **User visits our website URL "/"**

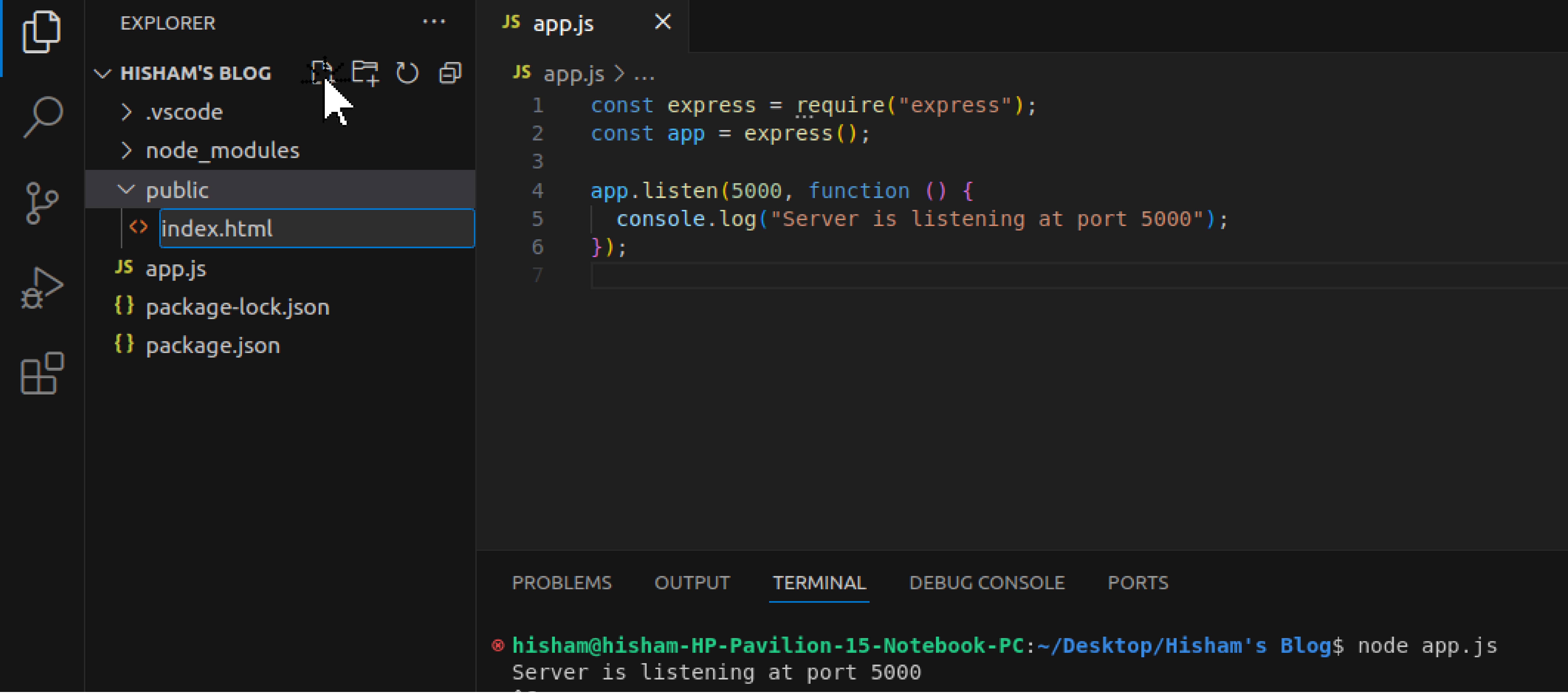
2. The backend app.js processes the request and sends the HTML page in response



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

app.listen(5000, function () {
  console.log("Server is listening at port 5000");
});
```

```
hisham@hisham-HP-Pavilion-15-Notebook-PC:~/Desktop/Hisham's Blog$ node app.js
Server is listening at port 5000
```

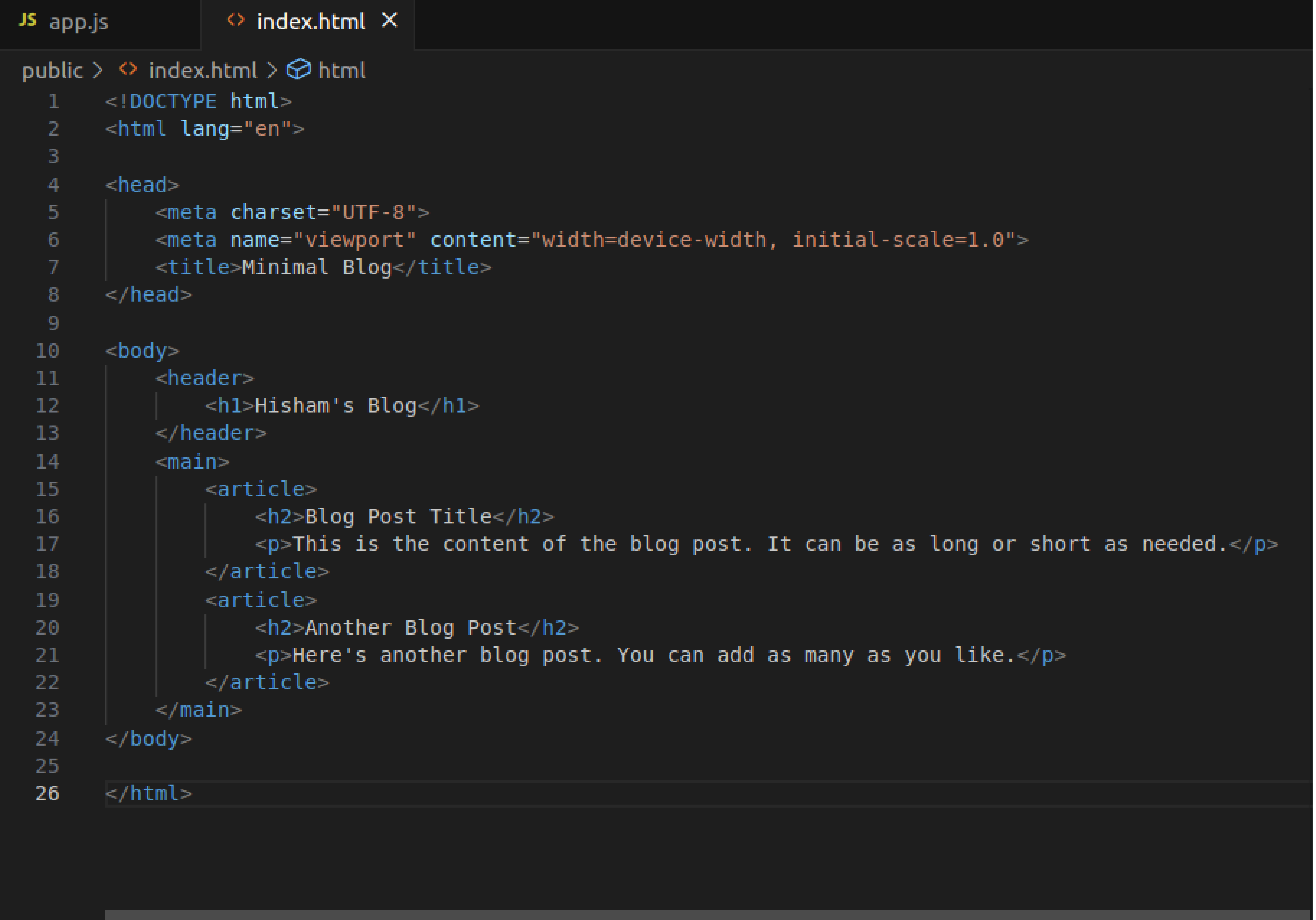
Create a new folder called **public**



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

app.listen(5000, function () {
  console.log("Server is listening at port 5000");
});

```

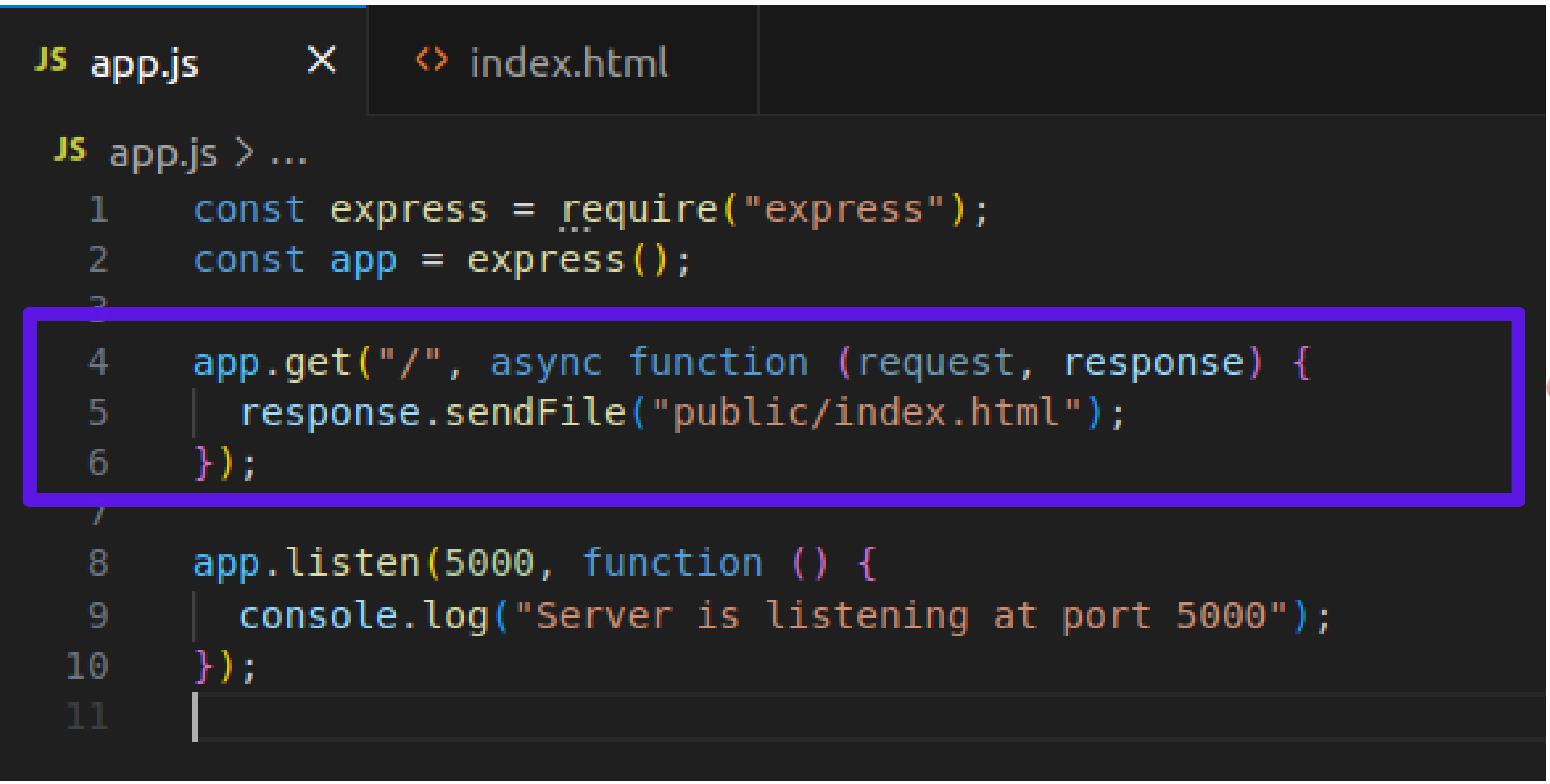
```
hisham@hisham-HP-Pavilion-15-Notebook-PC:~/Desktop/Hisham's Blog$ node app.js
Server is listening at port 5000
```

Create a file index.html inside this public folder



```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Minimal Blog</title>
</head>

<body>
    <header>
        <h1>Hisham's Blog</h1>
    </header>
    <main>
        <article>
            <h2>Blog Post Title</h2>
            <p>This is the content of the blog post. It can be as long or short as needed.</p>
        </article>
        <article>
            <h2>Another Blog Post</h2>
            <p>Here's another blog post. You can add as many as you like.</p>
        </article>
    </main>
</body>

</html>
```

Write your **HTML** code or copy the existing



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

app.get("/", async function (request, response) {
  response.sendFile("public/index.html");
});

app.listen(5000, function () {
  console.log("Server is listening at port 5000");
});

```

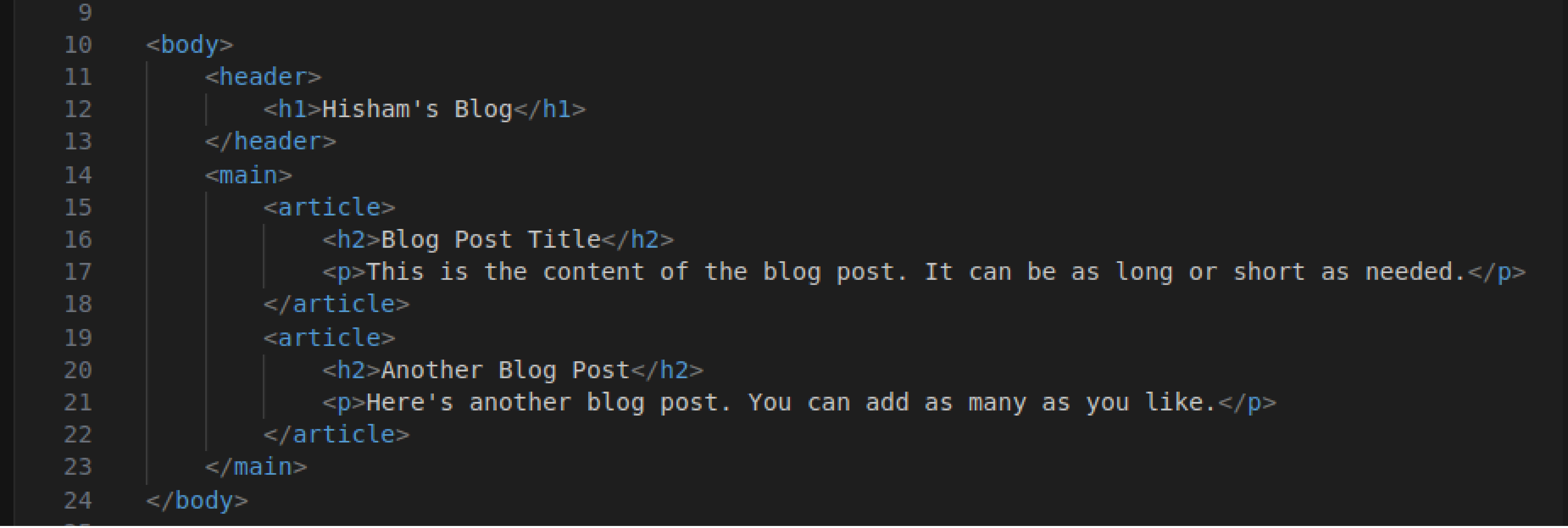
Modify our app.js to say that if any user visits the “/” route of our website, then show him this HTML page we just created

And voila!
Now we have a basic structure for the frontend and backend for our website

But hey, we are just hardcoding the values inside our HTML

Should’nt the frontend be dynamic and changing?

Chapter 3
Sending data from the **backend** to the **frontend**



```html
<body>
    <header>
        <h1>Hisham's Blog</h1>
    </header>
    <main>
        <article>
            <h2>Blog Post Title</h2>
            <p>This is the content of the blog post. It can be as long or short as needed.</p>
        </article>
        <article>
            <h2>Another Blog Post</h2>
            <p>Here's another blog post. You can add as many as you like.</p>
        </article>
    </main>
</body>
```

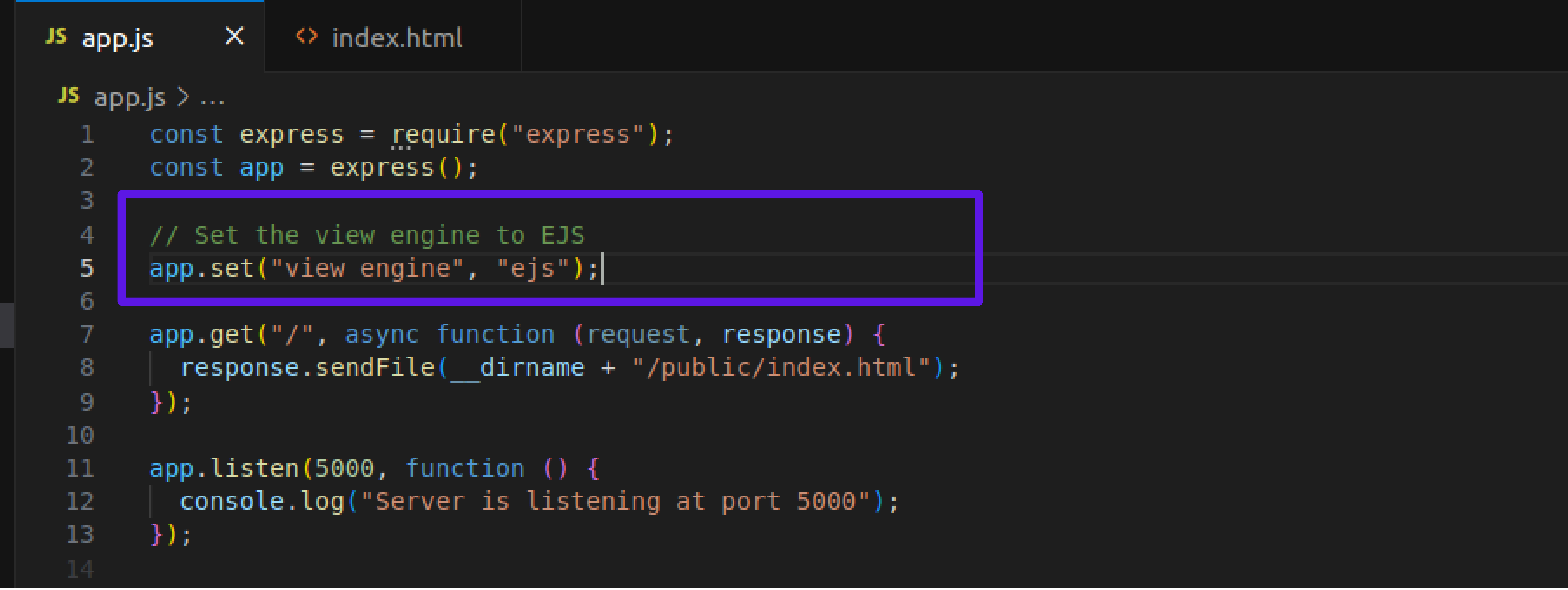
As you can tell, we are hardcoding the values here. But if our frontend cosntantly fetches the data from database then the value should not be hardcoded or fixed, rather it should change!

And that is exactly why we use template engines!

Template Engines allow us to do some logic inside our HTML and it helps transfer data from backend to frontend which is exactly what we need

That is, data is fetched from database to backend and then backend to frontend

Let us transfer data from backend to frontend and we can talk about databases later!

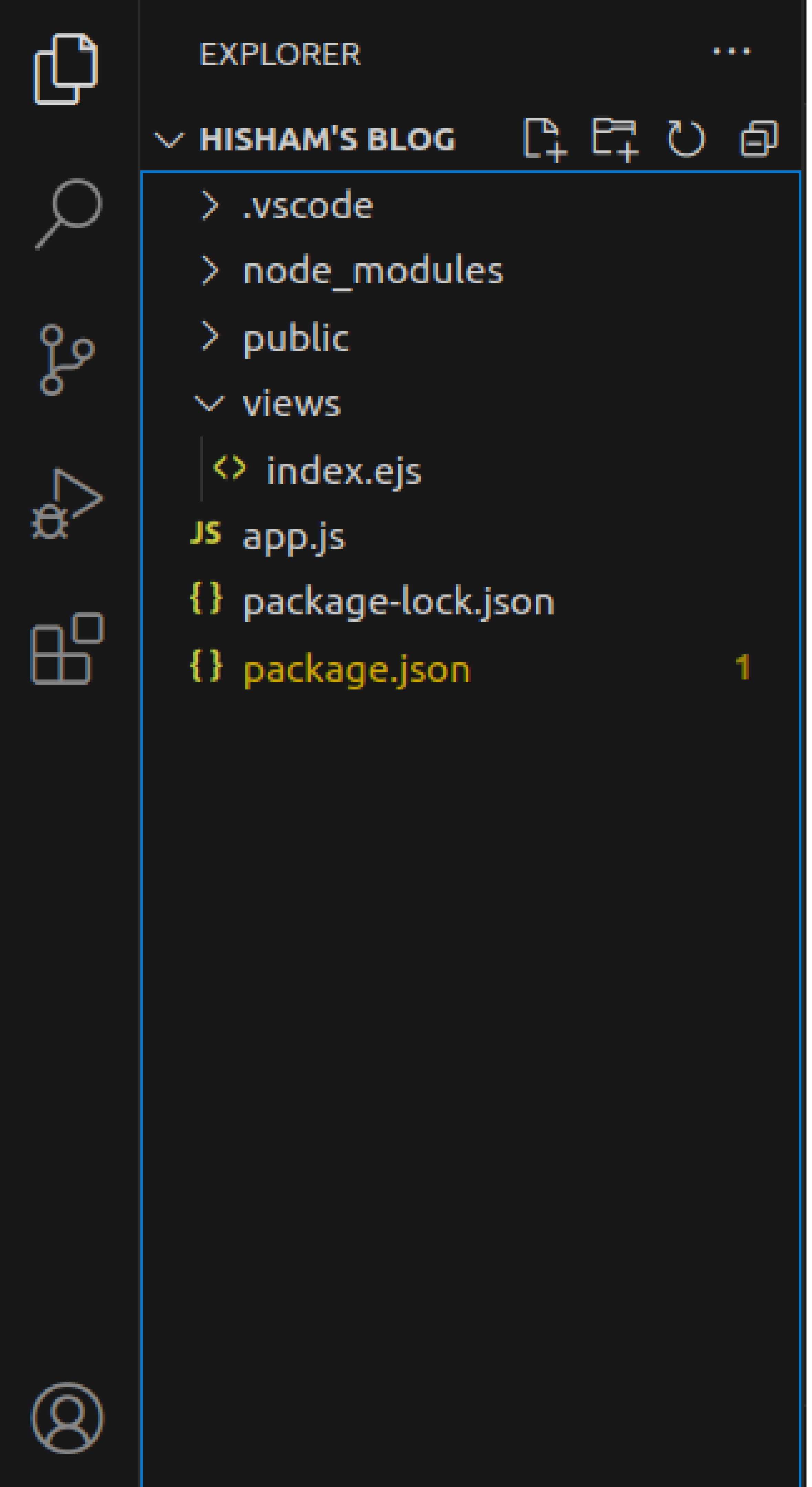


app.js ×   index.html

app.js > ...

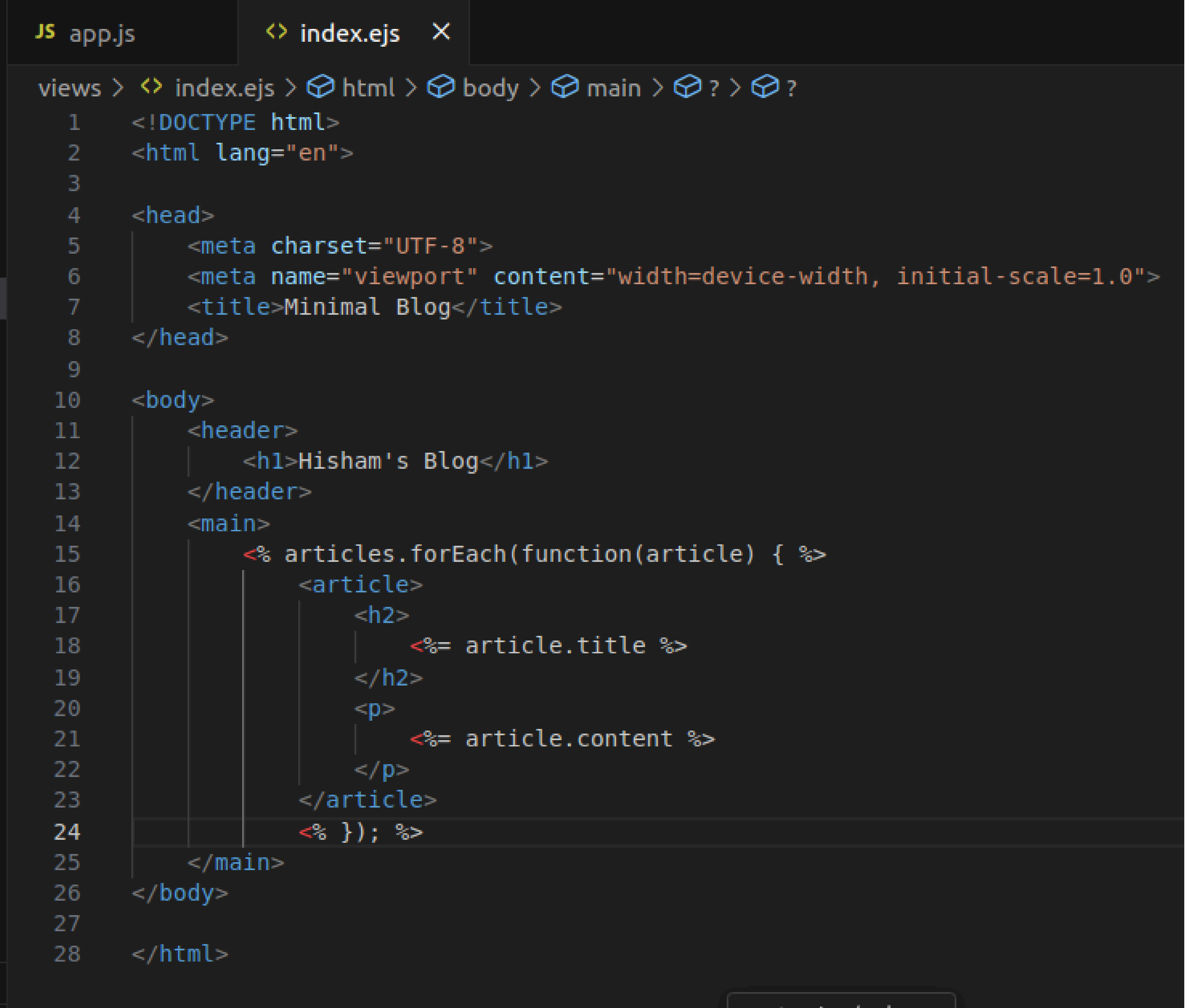
```javascript
const express = require("express");
const app = express();

// Set the view engine to EJS
app.set("view engine", "ejs");

app.get("/", async function (request, response) {
  response.sendFile(__dirname + "/public/index.html");
});

app.listen(5000, function () {
  console.log("Server is listening at port 5000");
});
```

TO tranfer the data, we need template engine and here we use the EJS template engine for same!



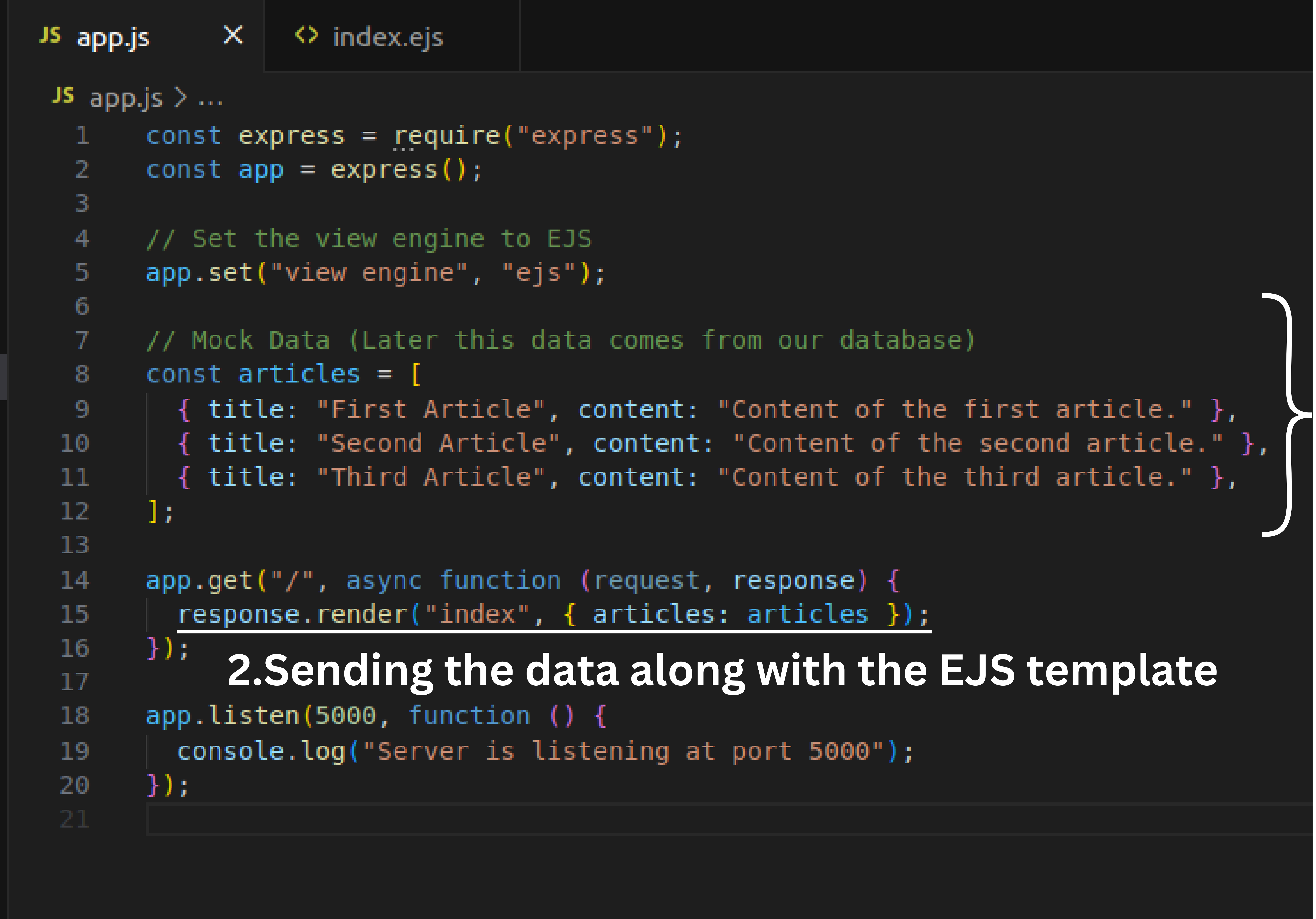
```
EXPLORER
HISHAM'S BLOG
> .vscode
> node_modules
> public
v views
    index.ejs
app.js
package-lock.json
package.json    1
```

Create a new **views** folder and inside it an **index.ejs** file



```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Minimal Blog</title>
</head>

<body>
    <header>
        <h1>Hisham's Blog</h1>
    </header>
    <main>
        <% articles.forEach(function(article) { %>
            <article>
                <h2>
                    <%= article.title %>
                </h2>
                <p>
                    <%= article.content %>
                </p>
            </article>
            <% }); %>
    </main>
</body>

</html>
```

In **index.ejs** we write the exaclty same code as **index.html** but instead we take advantage of the template engine to load the data for us

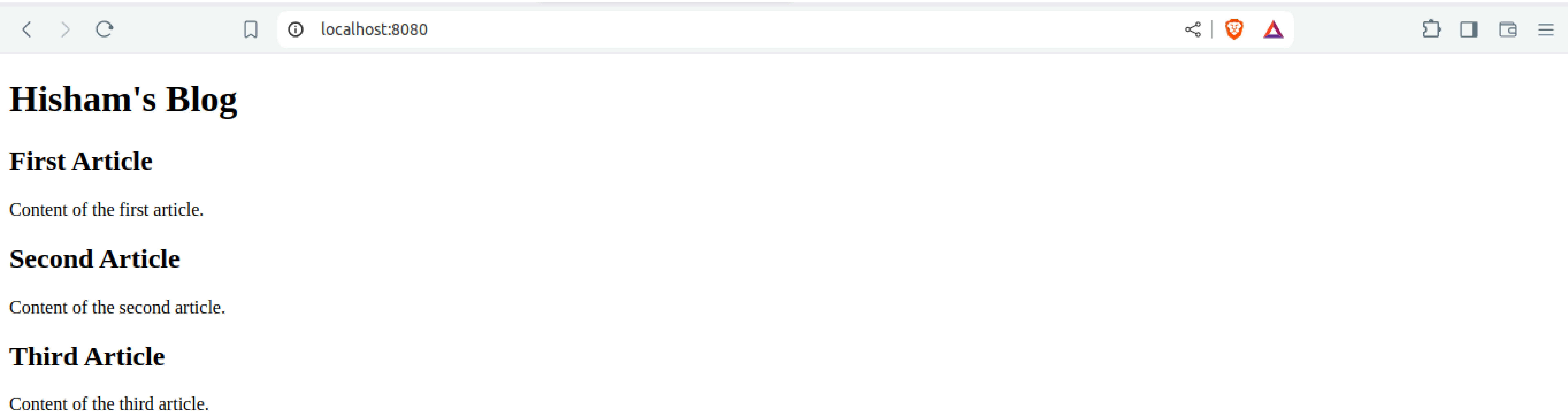


```js
const express = require("express");
const app = express();

// Set the view engine to EJS
app.set("view engine", "ejs");

// Mock Data (Later this data comes from our database)
const articles = [
  { title: "First Article", content: "Content of the first article." },
  { title: "Second Article", content: "Content of the second article." },
  { title: "Third Article", content: "Content of the third article." },
];

app.get("/", async function (request, response) {
  response.render("index", { articles: articles });
});

app.listen(5000, function () {
  console.log("Server is listening at port 5000");
});
```

1.The data we want to send

2.Sending the data along with the EJS template

Finally this is how it looks like!



Hisham's Blog

First Article

Content of the first article.

Second Article

Content of the second article.

Third Article

Content of the third article.