

# Basic **Express** Server Setup



### What is Express.js ?

Express.js is a lightweight and flexible web application framework for Node.js. It helps developers create robust web servers and APIs efficiently. Key benefits of using Express include:

- **Ease of Use:** Simplifies creating routes and handling HTTP requests.
- **Middleware Support:** Offers a way to manage requests with reusable functions.
- **Extensibility:** Compatible with various third-party modules and middleware.

### Why Use Express ?

- **Fast Development:** Streamlines the process of building web servers.
- **Minimal Setup:** Requires only basic configuration to start.
- **Community Support:** Backed by a large community with plenty of resources.



## 1. Install Express

Before starting, ensure **Node.js** is installed on your machine.

### Steps:

1. Run the following command to initialize a new Node.js project. This will automatically create a **package.json** file, which stores metadata about your project and its dependencies:

A terminal window with a dark background. The title bar says "terminal" and has standard window controls (minimize, maximize, close). The command "npm init -y" is entered in the terminal.

```
terminal
```

```
npm init -y
```

2. Install Express as a dependency:

A terminal window with a dark background. The title bar says "terminal" and has standard window controls (minimize, maximize, close). The command "npm install express" is entered in the terminal.

```
terminal
```

```
npm install express
```



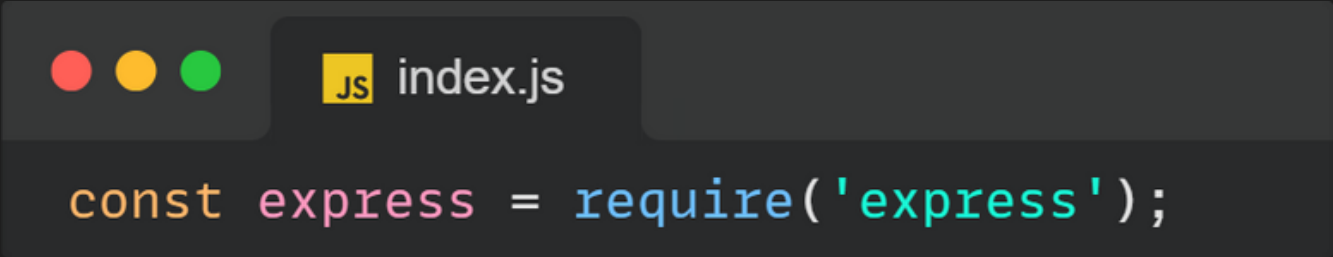
## 2. Create the Server File

### Steps:

1. **Create a New File:** Create a new JavaScript file, typically named `server.js` or `index.js`. This file will contain the code for your server.
2. **Write the Server Code:** Open the file and add the following code to set up your server.

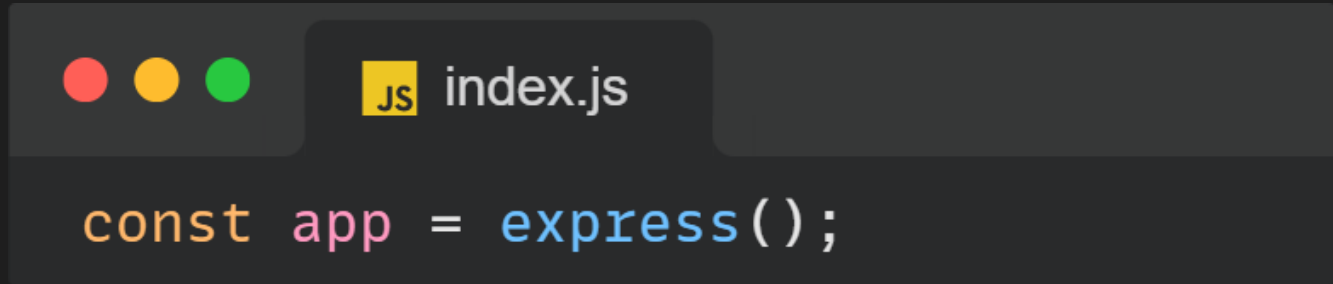
### Code Breakdown:

1. Import Express: Import the Express module to use its features.



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

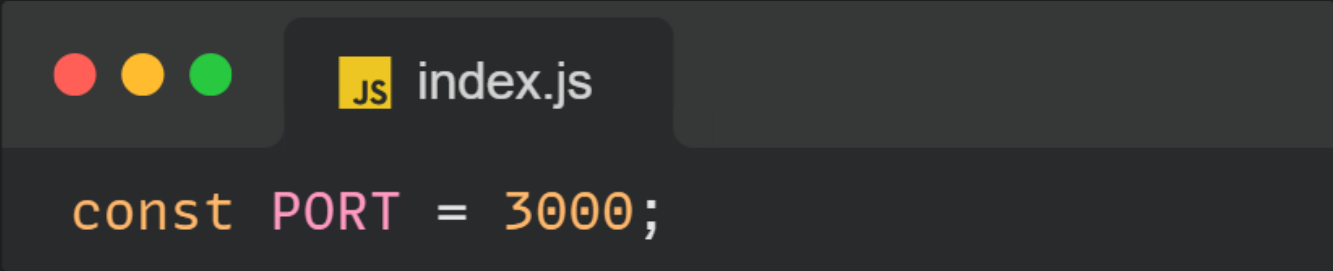
2. initialize the App: Create an Express application instance



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

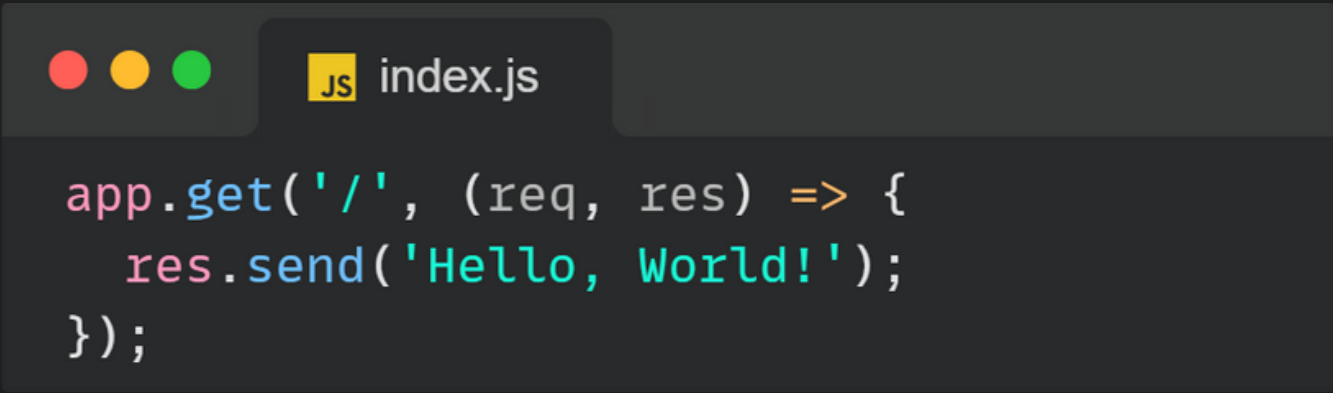


3. Define the Port: Set a port for the server to listen on. The default example uses port 3000.

A code editor window with a title bar containing three colored circles (red, yellow, green) and a tab labeled 'JS index.js'. The code inside is `const PORT = 3000;`.

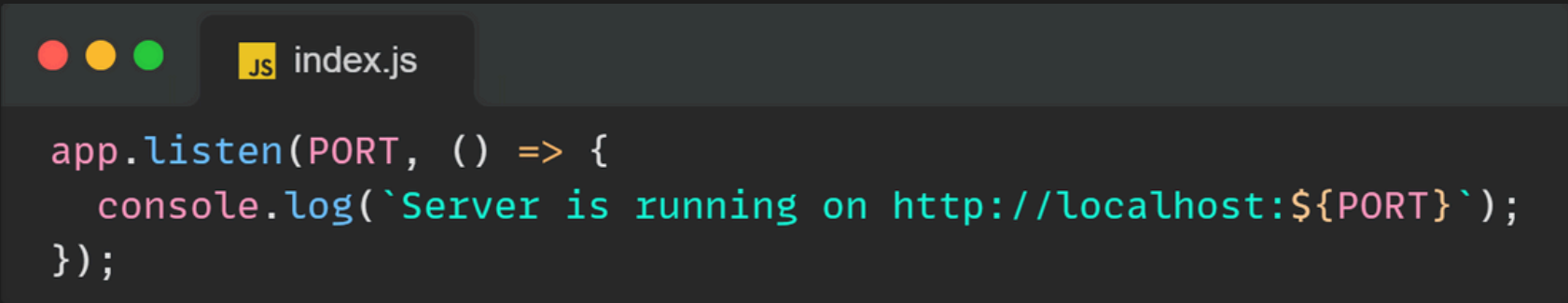
```
const PORT = 3000;
```

4. Define a Route: Create a basic route that responds to HTTP GET requests on the root URL (/).

A code editor window with a title bar containing three colored circles (red, yellow, green) and a tab labeled 'JS index.js'. The code inside is `app.get('/', (req, res) => { res.send('Hello, World!'); });`.

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

5. Start the Server: Listen on the specified port and log a message to indicate the server is running.

A code editor window with a title bar containing three colored circles (red, yellow, green) and a tab labeled 'JS index.js'. The code inside is `app.listen(PORT, () => { console.log(`Server is running on http://localhost:${PORT}`); });`.

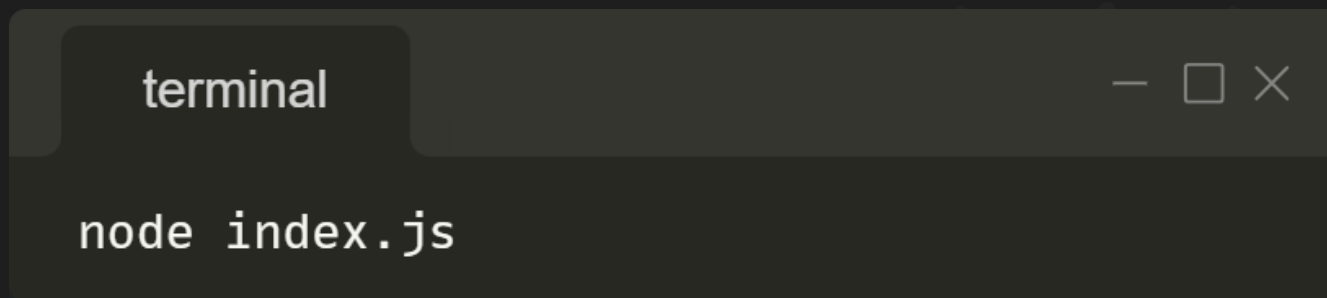
```
app.listen(PORT, () => {  
  console.log(`Server is running on http://localhost:${PORT}`);  
});
```



### 3. Start the Server

To start your server, follow these steps:

1. Open your terminal.
2. Navigate to the directory where your server file (e.g., index.js) is located.
3. Run the following command:

A screenshot of a terminal window. The window has a title bar with the word "terminal" on the left and standard window control buttons (minimize, maximize, close) on the right. The terminal area is dark, and the text "node index.js" is displayed in a light-colored monospace font.

4. Open your browser and visit: <http://localhost:3000>

You should see the message Hello, World! displayed in your browser.





**Hopefully You Found It  
Usefull!**

“Be sure to save this post so you  
can come back to it later”

like

Comment

Share