Explanation:

1. Install and Import the Express :

```
npm i express
```

    - const express = require('express'); imports the Express.js framework, which is the core of this application.
2. Create an Express application:
    - const app = express(); creates an instance of the Express application. This object will contain all the routing and middleware for your application.
3. Define a route handler:
    - app.get('/', (req, res) => { ... }); defines a route handler for HTTP GET requests to the root URL ("/").
        - req: Represents the incoming HTTP request object.
        - res: Represents the HTTP response object.
        - res.send('Hello from Express.js!'); sends the text "Hello from Express.js!" as the response to the client.
4. Start the server:
    - app.listen(port, () => { ... }); starts the HTTP server and listens for incoming requests on the specified port (3000 in this case).
    - The callback function within listen() is executed once the server is successfully listening.

Code Example :

```
const express = require('express');
const app=express();
app.get('/',(req,res)=>{}
app.listen(port,())=>{}
```

To run this server:

1. Save the code as a .js file (e.g., app.js).

2. Open your terminal or command prompt.
3. Navigate to the directory where you saved the file.
4. Run the script using node app.js.

Now, you can open a web browser and go to http://localhost:3000/. You should see the message "Hello from Express.js!" displayed in the browser.

This is a basic example, but it demonstrates the fundamental concepts of creating an Express.js server:

- Routing: Defining routes (URL patterns) and their corresponding handlers.
- Request and Response Objects: Using the req and res objects to interact with incoming requests and send responses.
- Server Initialization: Starting the server and listening for incoming connections.

Express.js provides a powerful and flexible framework for building web applications in Node.js. You can extend this basic example by adding more routes, using middleware, and integrating with databases, templating engines, and other libraries.
The error message indicates that you're likely trying to run this code in an environment that doesn't support JavaScript's const keyword, such as Python.
To run this Express.js server, you'll need a Node.js environment:

1. Install Node.js and npm:
    - Download and install Node.js from the official website (nodejs.org). This will also install npm (Node Package Manager).
2. Create a file:
    - Save the code above as an .js file (e.g., server.js).
3. Run the server:
    - Open your terminal or command prompt.
    - Navigate to the directory where you saved the server.js file.
    - Run the following command:
      node server.js

Explanation:

- const express = require('express');:
    - Imports the Express.js framework.

- `const app = express();`:
    - Creates an instance of the Express.js application.
- `app.get('/', (req, res) => { ... });`:
    - Defines a route handler for GET requests to the root URL (/).
    - `req` represents the incoming HTTP request.
    - `res` represents the HTTP response that will be sent back to the client.
    - `res.send()` sends the specified string as the response to the client.
- `app.listen(port, () => { ... });`:
    - Starts the server and listens for incoming requests on the specified port (3000 in this case).
    - The callback function within `listen()` is executed once the server starts listening.

This is a very basic example. Express.js offers many more features, such as:

- Routing: Define routes for different HTTP methods (GET, POST, PUT, DELETE) and URLs.
- Middleware: Use middleware functions to perform actions before and after the main route handler (e.g., logging, authentication).
- Static File Serving: Serve static files like HTML, CSS, and JavaScript.
- Templating Engines: Integrate with templating engines like EJS, Pug, and Handlebars to generate dynamic HTML.
- Error Handling: Implement custom error handling middleware.

I hope this explanation is helpful! Let me know if you have any other questions about Express.js.