Beginner's Guide to Creating an Express API with Node.js

Introduction

Node.js is a powerful runtime environment for server-side applications, and Express.js is a minimal and flexible Node.js web application framework that provides robust features for building web and mobile applications. In this beginner's guide, we will walk you through the process of creating a simple Express API and enhancing it by using a module to return data in response to a GET request. By the end of this guide, you will have a basic understanding of how to set up a RESTful API using Express and Node.js.

Prerequisites

Before we get started, make sure you have the following:

- Node.js and npm installed
- Basic knowledge of JavaScript

Step 1: Setting Up Your Project

First, create a new directory for your project and navigate to it in your terminal.

1.1 Create a Project Directory

Begin by creating a new directory for your Express project. You can do this in your terminal:

```
mkdir book-api
cd book-api
```

1.2 Initialize a Node.js Project

Inside your project directory, you'll want to initialize a Node.js project using **npm**. This will create a **package.json** file to manage your project's dependencies:

```
npm init -y
```

1.3 Install Dependencies

Install the Express.js framework as a project dependency:

```
npm install express
```

Step 2: Create a New Module

In your project directory, create a new file called books.js. This module will export an array of books.

```
// books.js
const books = [
  { id: 1, title: 'The Great Gatsby', author: 'F. Scott Fitzgerald' },
  { id: 2, title: 'To Kill a Mockingbird', author: 'Harper Lee' },
  { id: 3, title: '1984', author: 'George Orwell' },
  { id: 4, title: 'Pride and Prejudice', author: 'Jane Austen' },
];

module.exports = books;
```

Step 3: Update Your Express App

In your app.js file, you'll be working on the core of your Express application. First, you'll need to import the necessary dependencies, including Express itself and the books module you created earlier.

```
// app.js
const express = require('express');
const app = express();
const port = 3000;
const books = require('./books'); // Require the books module
```

const express = require('express'); - Here, you import the Express.js framework and create an instance of it called **app**. Express will be the backbone of your API, handling HTTP requests and responses.

const port = 3000; - You define a variable **port** to specify the port on which your Express server will listen for incoming requests. In this case, it's set to **3000**, but you can choose any

available port you prefer.

const books = require('./books'); - This line imports the books module you created in Step 1. This module contains the data that your API will serve.

Next, you'll add middleware to your Express app. Middleware functions are executed before your routes and allow you to perform tasks such as parsing JSON requests.

```
// Middleware to parse JSON requests
app.use(express.json());
```

app.use(express.json()); - Express provides a built-in middleware function **express.json()** that parses incoming JSON requests. This middleware is essential if your API expects JSON data in requests, as it will automatically parse and make the data available for use in your routes.

Now, you'll define a route that responds to GET requests for fetching all books:

```
// Define a route to get all books
app.get('/api/books', (req, res) => {
  res.json(books);
});
```

app.get('/api/books', (req, res) => { - Here, you define a route that listens for GET requests at the URL path /**api/books**. When a GET request is received at this path, the provided callback function is executed.

(req, res) => { - This is the callback function that takes two parameters: **req** (request) and **res** (response). The **req** object represents the incoming HTTP request, and the **res** object is used to send an HTTP response.

Inside the callback function, you send a JSON response using **res.json(books);**. This line sends the **books** array as a JSON response when a GET request is made to **/api/books**.

The completed code for reference below.

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

const books = require('./books'); // Require the books module

// Middleware to parse JSON requests
app.use(express.json());

// Define a route to get all books
app.get('/api/books', (req, res) => {
  res.json(books);
});

// Start the server
app.listen(port, () => {
  console.log(`Server is running on port ${port}`);
});
```

Step 4: Test Your Updated API

Restart your Express server if it's not already running:

```
node app.js
```

1. Open Postman.
2. Create a new request or use the existing one.
3. Set the request type to "GET."
4. Enter the API URL: http://localhost:3000/api/books.
5. Send the request.
6. You should receive a JSON response containing the list of books from the books.js module.

Conclusion:

In this beginner's guide, we've covered the basics of creating a simple Express API with Node.js. We've also demonstrated how to use a module to provide data in response to a GET request. This is just the starting point, and you can expand on this knowledge to build more complex APIs with additional routes, database integration, and advanced features. Express

and Node.js offer a powerful combination for building web applications, and with practice, you can develop robust APIs for your projects. Happy coding!