Handling GET

How to handle GET requests

Handling GET requests in an Express.js application involves defining routes that respond to GET HTTP requests. Here's a basic example of handling a GET request in an Express.js application:

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

// Define a simple GET route
app.get('/', (req, res) => {
  res.send('Hello, this is a GET request!');
});

// Define a route with a parameter
app.get('/greet/:name', (req, res) => {
  const { name } = req.params;
  res.send(`Hello, ${name}!`);
});

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

In this example:

1. We create an instance of the Express application using `express()`.
2. We define a simple GET route for the root URL (`'/'`) that responds with a message.
3. We define another GET route with a parameter (`'/greet/:name'`) that responds with a personalized greeting based on the parameter.
4. We start the server with `app.listen` on port 3000.

When you run this script (`node filename.js`) and visit `http://localhost:3000` in your browser, you should see the response from the root route. Additionally, visiting `http://localhost:3000/greet/John` should display a personalized greeting for the name "John."

This is a basic example, and in a real-world application, you would likely have more complex routes and logic. Express provides a flexible and powerful routing system, allowing you to handle different HTTP methods, route parameters, query parameters, and more.