MERN-Stack

The 'E' in MERN Stack

What is Express JS ?

Chebbi Hamdi

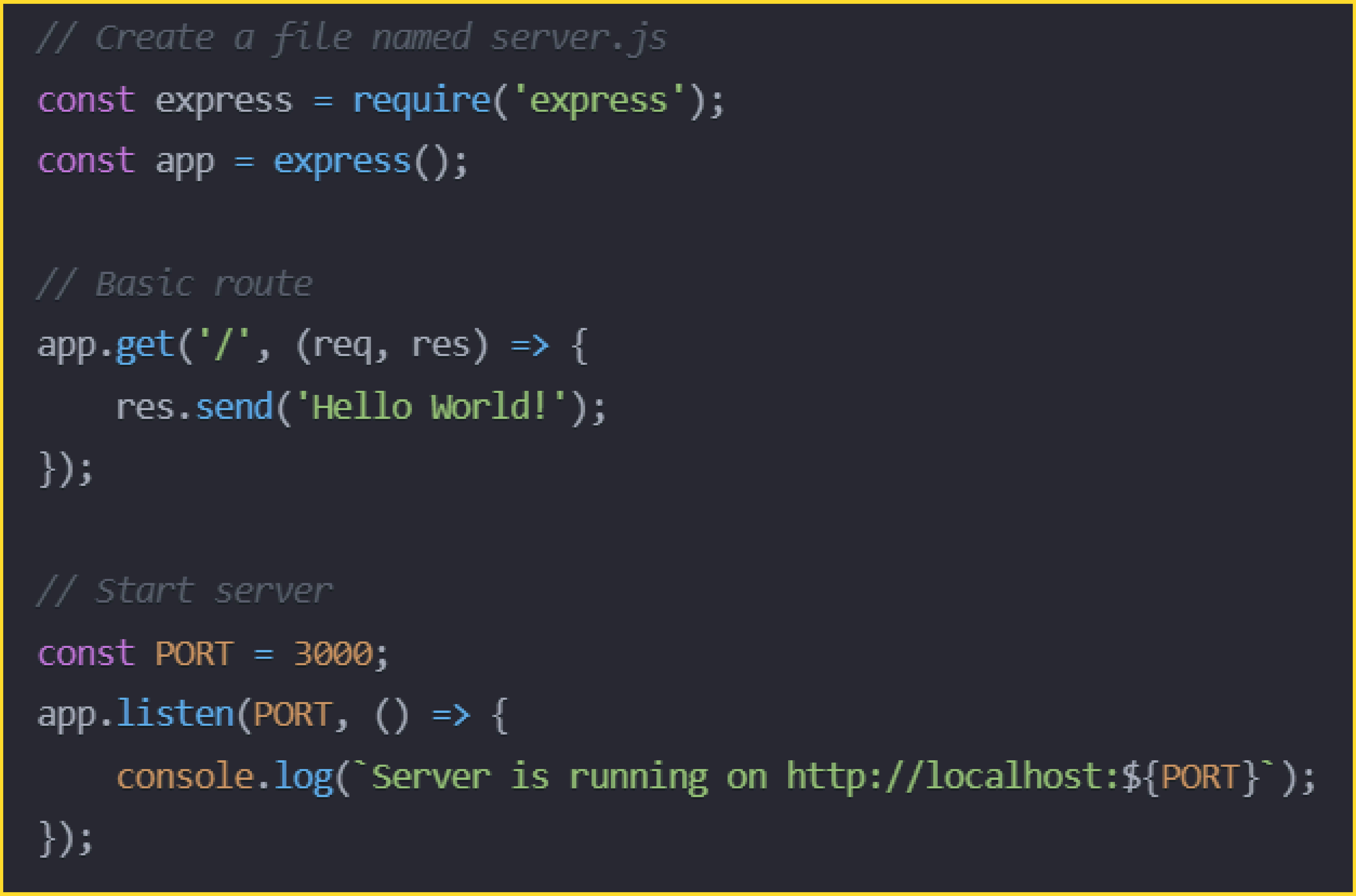
@Chebbi

Express JS is a minimal and flexible Node.js web application framework that provides a robust set of features for building single-page, multi-page, and hybrid web applications. It's the de facto standard server framework for Node.js.

Installation:

```
npm install express
```

1.Create a Simple Express Server



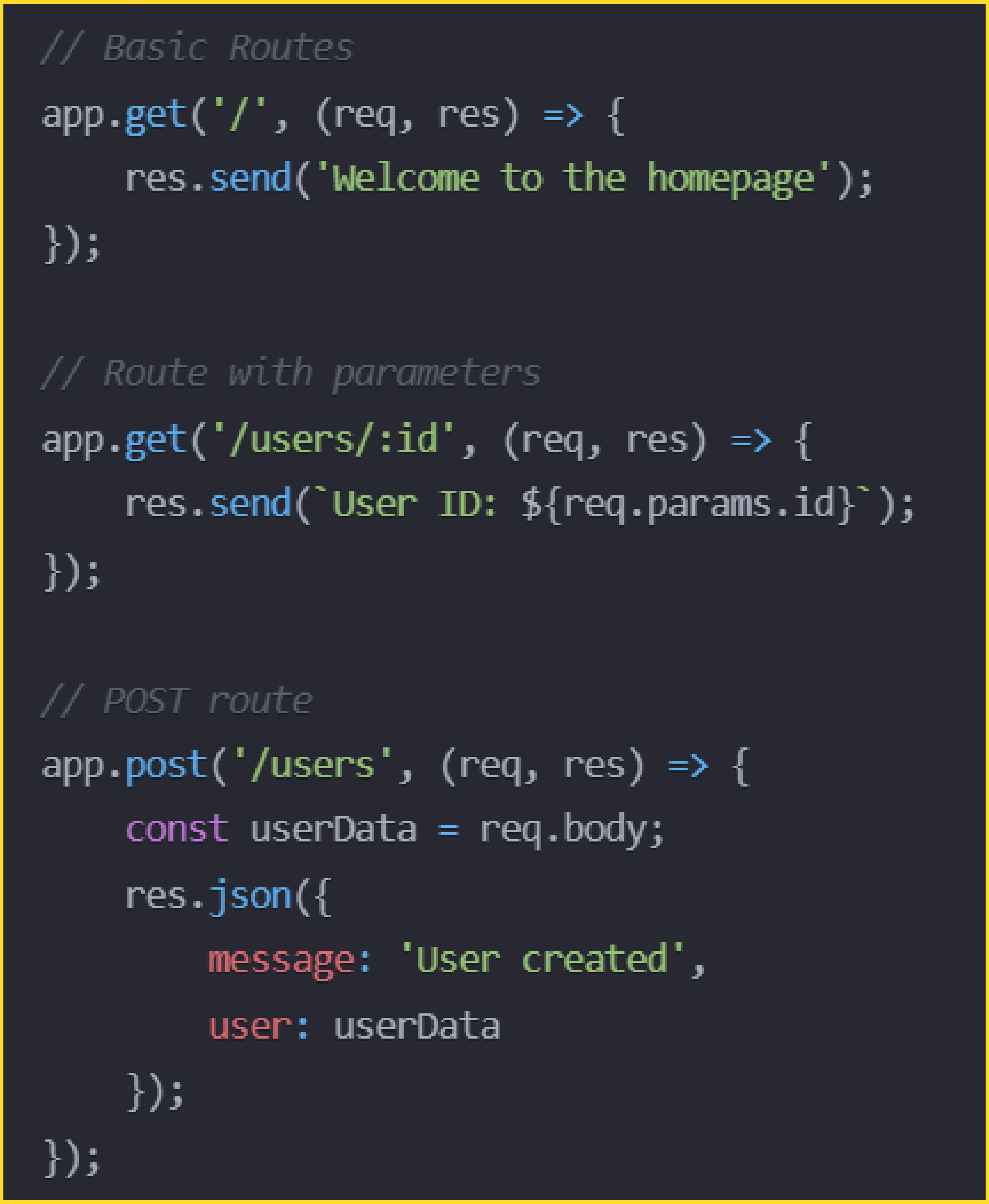
```javascript
// Create a file named server.js
const express = require('express');
const app = express();

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

// Start server
const PORT = 3000;
app.listen(PORT, () => {
    console.log(`Server is running on http://localhost:${PORT}`);
});
```

2.Routing

Express provides a robust routing system for handling HTTP methods (Get, Post, Put, Delete ...)



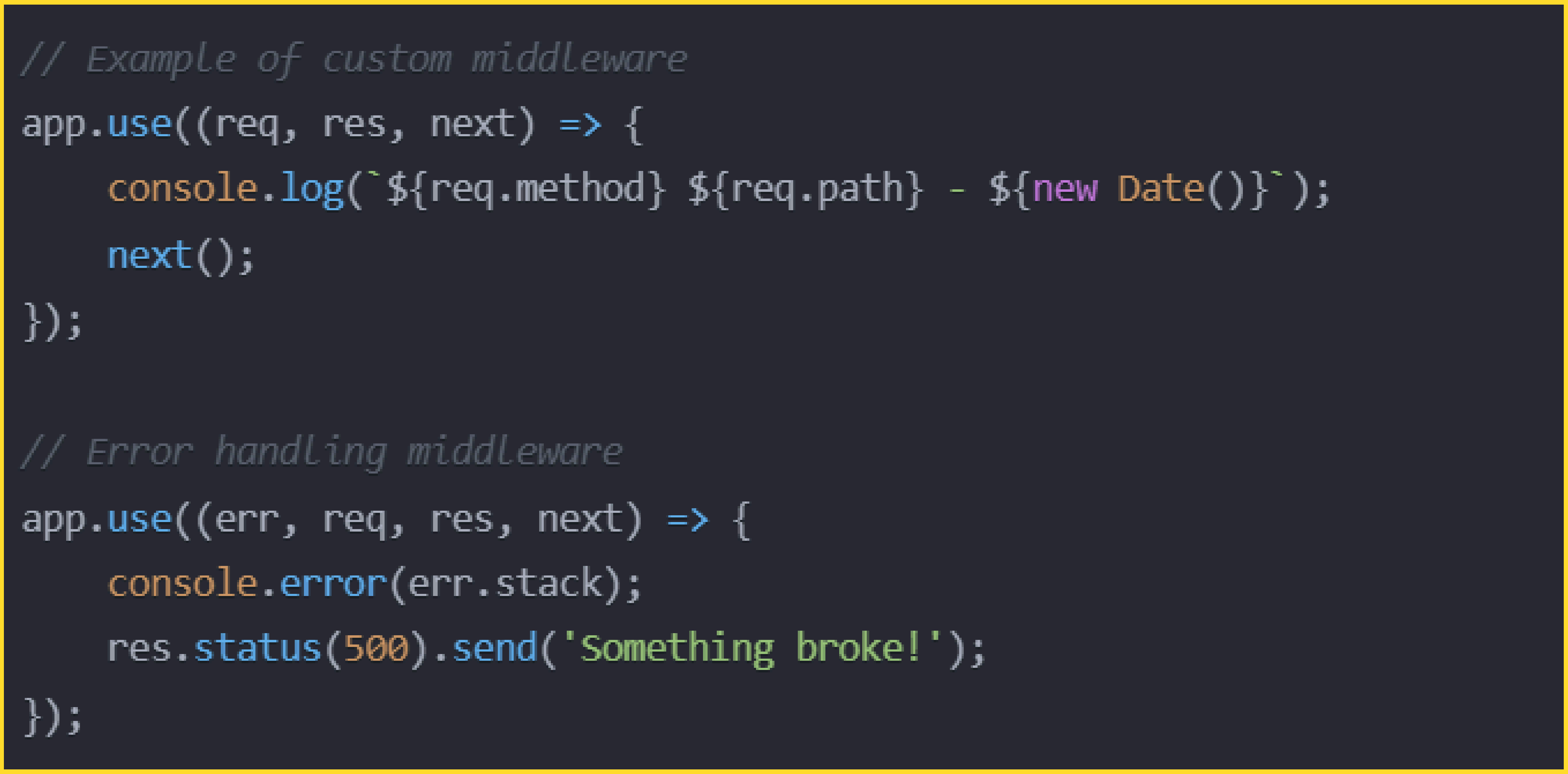
```javascript
// Basic Routes
app.get('/', (req, res) => {
    res.send('Welcome to the homepage');
});

// Route with parameters
app.get('/users/:id', (req, res) => {
    res.send(`User ID: ${req.params.id}`);
});

// POST route
app.post('/users', (req, res) => {
    const userData = req.body;
    res.json({
        message: 'User created',
        user: userData
    });
});
```

3.Middleware

Middleware functions are functions that have access to the request object (req), the response object (res), and the next middleware function in the application's request-response cycle.

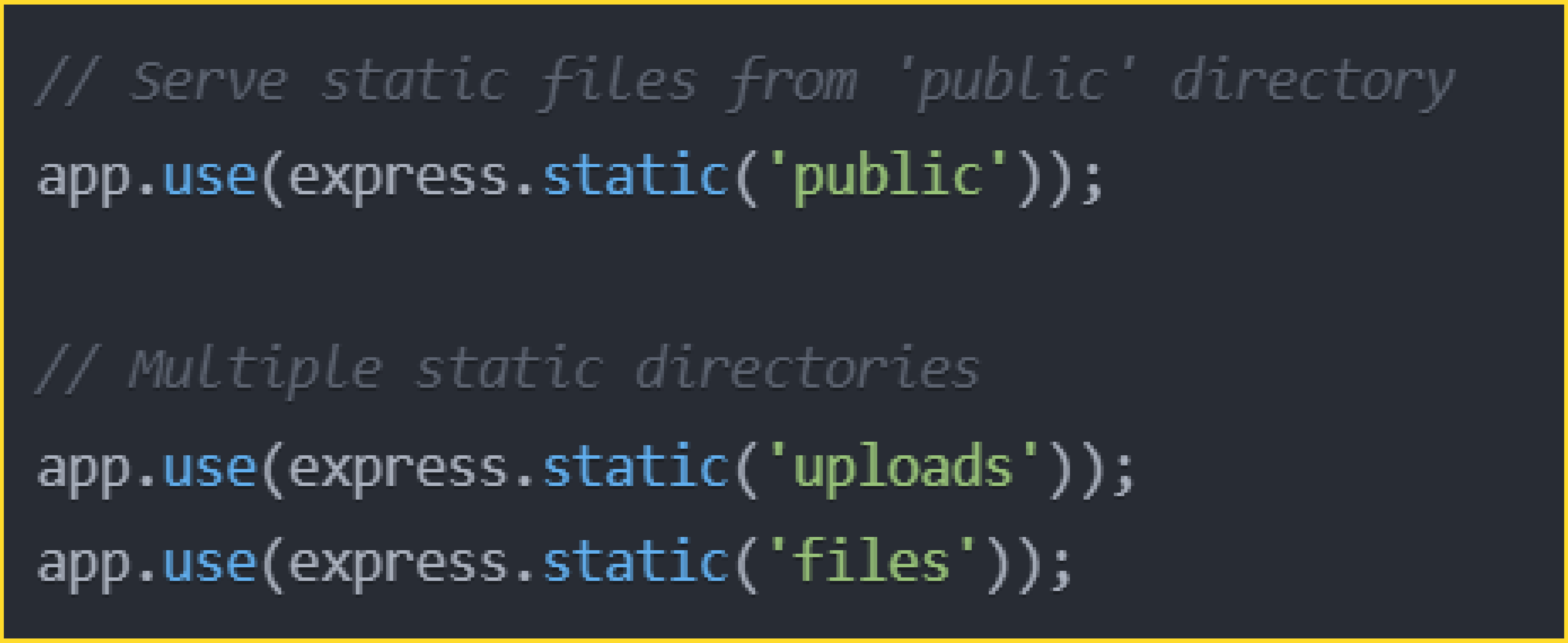


```javascript
// Example of custom middleware
app.use((req, res, next) => {
    console.log(`${req.method} ${req.path} - ${new Date()}`);
    next();
});

// Error handling middleware
app.use((err, req, res, next) => {
    console.error(err.stack);
    res.status(500).send('Something broke!');
});
```

4.Serving Static Files

Express can be Used to Serve static files like images, CSS, and JavaScript

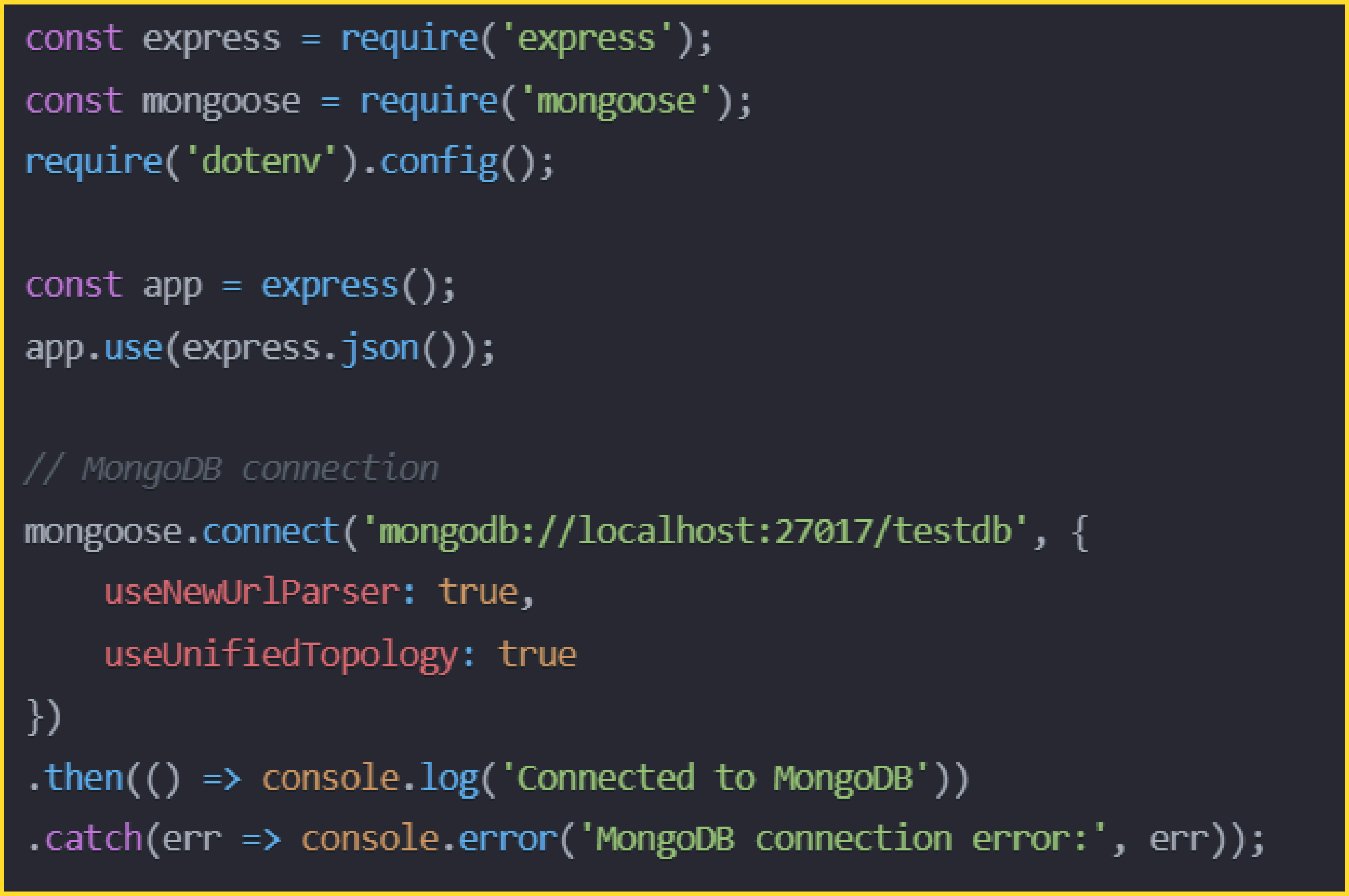


```javascript
// Serve static files from 'public' directory
app.use(express.static('public'));

// Multiple static directories
app.use(express.static('uploads'));
app.use(express.static('files'));
```

5.Connecting Express with MongoDB

To connect to MongoDB, you can use mongoose, a popular ODM (Object Data Modeling) library for MongoDB



```javascript
const express = require('express');
const mongoose = require('mongoose');
require('dotenv').config();

const app = express();
app.use(express.json());

// MongoDB connection
mongoose.connect('mongodb://localhost:27017/testdb', {
    useNewUrlParser: true,
    useUnifiedTopology: true
})
.then(() => console.log('Connected to MongoDB'))
.catch(err => console.error('MongoDB connection error:', err));
```

If you find this helpful, like and share it with your friends

SHARE

Chebbi Hamdi
@Chebbi