Express.js Complete Notes

📘 Table of Contents

1. Introduction to Express.js
2. Why Use Express.js
3. Installing Express.js
4. Creating Your First Express Server
5. Express Application Structure
6. Routing in Express
7. Request & Response Objects
8. Middleware in Express
9. Built-in Middleware
10. Custom Middleware
11. Error Handling Middleware
12. Serving Static Files
13. Handling Forms & JSON Data
14. REST API Design with Express
15. MVC Pattern in Express
16. Connecting Express with MongoDB
17. Authentication Basics in Express
18. Environment Variables in Express
19. Security Best Practices
20. Production Best Practices

✂️ Introduction to Express.js

- Express.js is a **minimal and flexible Node.js web application framework**.
- Built on top of Node's HTTP module.
- Makes backend development **faster and easier**.

✒️ Why Use Express.js

- Simple routing system
- Middleware support
- Easy REST API development
- Widely used in industry (MERN stack)

🛏️Installing Express.js

```
npm init -y
npm install express
```

Optional (for development):

```
npm install --save-dev nodemon
```

🌂Creating Your First Express Server

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

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

app.listen(3000, () => {
  console.log('Server running on port 3000');
});
```

🌡️ Express Application Structure

```
project/
├── controllers/
├── routes/
├── models/
├── middleware/
├── config/
├── app.js
└── server.js
```

🎒Routing in Express

```
app.get('/users', (req, res) => {
  res.send('Get all users');
});

app.post('/users', (req, res) => {
  res.send('Create user');
});
```

Route Parameters

```
app.get('/users/:id', (req, res) => {
  res.send(req.params.id);
});
```

🎓Request & Response Objects

- `req.params` – URL params
- `req.query` – Query string
- `req.body` – Request body

```
app.post('/login', (req, res) => {
  res.json(req.body);
});
```

🎩Middleware in Express

- Middleware runs **before route handlers**.
- Has access to `req`, `res`, `next`.

```
app.use((req, res, next) => {
  console.log('Middleware executed');
  next();
});
```

🎽 Built-in Middleware

```
app.use(express.json());
app.use(express.urlencoded({ extended: true }));
```

🕐Custom Middleware

```
function authMiddleware(req, res, next) {
  const token = req.headers.authorization;
  if (!token) return res.status(401).send('Unauthorized');
  next();
}

app.use(authMiddleware);
```

1✂️Error Handling Middleware

```
app.use((err, req, res, next) => {
  res.status(500).json({ message: err.message });
});
```

1✒️Serving Static Files

```
app.use(express.static('public'));
```

1🛏️Handling Forms & JSON Data

```
app.post('/submit', (req, res) => {
  console.log(req.body);
  res.send('Form submitted');
});
```

1🌂REST API Design with Express

```
app.get('/api/products', getProducts);
app.post('/api/products', createProduct);
app.put('/api/products/:id', updateProduct);
app.delete('/api/products/:id', deleteProduct);
```

---

1 🌡️ MVC Pattern in Express

- **Model** – Database logic
- **View** – UI / frontend
- **Controller** – Business logic

```
router.get('/users', userController.getUsers);
```

1🎒Connecting Express with MongoDB

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

mongoose.connect(process.env.MONGO_URI)
  .then(() => console.log('DB Connected'))
  .catch(err => console.log(err));
```

1🎓Authentication Basics in Express

- Use JWT for authentication
- Use bcrypt for password hashing

```
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');
```

1🎩Environment Variables in Express

```
npm install dotenv
```

```
require('dotenv').config();
const PORT = process.env.PORT || 3000;
```

---

1 🎽 Security Best Practices

- Use `helmet`
- Validate user input
- Never expose secrets
- Use HTTPS

```
npm install helmet
```

---

2 🏮 Production Best Practices

- Use PM2
- Enable logging
- Centralized error handling
- Proper folder structure

```
npm install pm2 -g
pm2 start server.js
```

🔗 Summary

Express.js simplifies backend development by providing: * Clean routing * Powerful middleware * Scalable architecture

Perfect for MERN stack and REST APIs 🚀