🚀 Advanced Node.js, Express & MongoDB: A Developer's Guide

✅ Table of Contents

---

1. Advanced Node.js

✅ **Event Loop & Async Patterns**

- The **Event Loop** handles async callbacks.
- Use **Promises**, **Async/Await**, and util.promisify for cleaner async code.
- Avoid **Callback Hell** by chaining Promises or using Async/Await.

✅ **Streams & Buffers**

- Streams handle large data efficiently.
- **Example:** Reading a file stream:

```
js
CopyEdit
const fs = require('fs');
const readStream = fs.createReadStream('file.txt', 'utf8');

readStream.on('data', chunk => {
  console.log(chunk);
});
```

- Types: Readable, Writable, Duplex, Transform.

✅ **Clustering & Child Processes**

- **Cluster Module:** Use multiple CPU cores.

```
js
CopyEdit
const cluster = require('cluster');
const http = require('http');
const numCPUs = require('os').cpus().length;

if (cluster.isMaster) {
  for (let i = 0; i < numCPUs; i++) {
    cluster.fork();
  }
} else {
```

```
 http.createServer((req, res) => {
   res.end('Handled by worker');
 }).listen(3000);
}
```

✅ **Environment Variables & Config**

- Store sensitive data outside code:

```
js
CopyEdit
require('dotenv').config();
console.log(process.env.DB_URI);
```

✅ **Error Handling & Logging**

- Use try/catch with async/await.
- Global error handler middleware in Express.
- Use logging libraries like **winston** for better logs.

🚀 **2️⃣ Advanced Express**

✅ **Middleware Deep Dive**

- Middleware functions execute in sequence.
- Example: Custom middleware

```
js
CopyEdit
app.use((req, res, next) => {
 console.log(`${req.method} ${req.url}`);
 next();
});
```

✅ **Routers & Modularization**

- Use express.Router() for clean, modular routes.

```
js
CopyEdit
const router = require('express').Router();
router.get('/users', ...);
app.use('/api', router);
```

✅ **Authentication & Authorization**

- **JWT:** Secure token-based auth.
- Use passport.js for OAuth, social logins.
- Protect routes:

```
js
CopyEdit
const jwt = require('jsonwebtoken');
const verifyToken = (req, res, next) => {
  const token = req.headers['authorization'];
  if (!token) return res.sendStatus(403);
  jwt.verify(token, process.env.JWT_SECRET, (err, user) => {
    if (err) return res.sendStatus(403);
    req.user = user;
    next();
  });
};
```

✅ **Validating Input Data**

- Use **express-validator** or **Joi** to validate & sanitize inputs.

```
js
CopyEdit
const { body, validationResult } = require('express-validator');

app.post('/user',
  body('email').isEmail(),
```

```
 (req, res) => {
  const errors = validationResult(req);
  if (!errors.isEmpty()) {
   return res.status(400).json({ errors: errors.array() });
  }
 });
```

✅ **Rate Limiting & Security**

- Use **helmet** for secure headers.
- Use **cors** for cross-origin requests.
- Use **express-rate-limit** to prevent abuse.

🗂 **3️⃣ Advanced MongoDB**

✅ **Data Modeling & Relationships**

- Use **ObjectId** to reference related collections.
- Example: One-to-many relationship (User & Posts).

✅ **Aggregation Framework**

- Powerful for analytics:

```
js
CopyEdit
db.orders.aggregate([
 { $match: { status: "active" } },
 { $group: { _id: "$customerId", total: { $sum: "$amount" } } }
]);
```

✅ **Indexing & Performance**

- Use .explain() to analyze query performance.
- Create indexes to speed up queries:

```
js
```

CopyEdit

```
db.users.createIndex({ email: 1 }, { unique: true });
```

✅ **Transactions**

- For multi-document ACID transactions:

js

CopyEdit

```
const session = await mongoose.startSession();
session.startTransaction();
try {
  await User.create([{ name: 'Alice' }], { session });
  await Post.create([{ title: 'Post 1' }], { session });
  await session.commitTransaction();
} catch (err) {
  await session.abortTransaction();
}
session.endSession();
```

✅ **Backup & Security**

- Use MongoDB Atlas backup tools.
- Use **roles** and **IP Whitelisting** for production databases.

🚀 **4️⃣ Best Practices & Deployment**

✅ Use .env files for secrets (never hard-code!).
✅ Use **morgan** for HTTP request logging.
✅ Use **Jest/Supertest** for API testing.
✅ Use **API Versioning** (/api/v1/...).
✅ Deploy on Heroku, Vercel, AWS, or DigitalOcean.

🗄️ **5️⃣ Helpful Resources**

- Node.js Docs: https://nodejs.org/en/docs/
- Express Docs: https://expressjs.com/

- MongoDB Docs: https://www.mongodb.com/docs/
- Mongoose Docs: https://mongoosejs.com/
- OWASP Node.js Security: https://owasp.org/www-project-nodejs-goat/

✨ Final Tips

✅ Write clean, modular code.
✅ Always validate and sanitize inputs.
✅ Keep dependencies updated.
✅ Learn from real projects and refactor often.