The Ultimate Node.js Cheatsheet

~Developer Shaurya

Important Links

NodeJS Cheatsheet:

https://developershaurya.com/ultimate-node-js-cheatsheet/

Website: https://developershaurya.com/

All CheatSheets: https://developershaurya.com/cheat-sheets/

Youtube: https://www.youtube.com/@DeveloperShaurya

Developer Shaurya

Community: https://t.me/developer_shaurya (Telegram)

Community: https://whatsapp.com/channel/0029VaLpCKsI7Be6ByiErv35 (WhatsApp)

The Ultimate Node.js Cheatsheet (From Zero to Production)

Table of contents

What is Node.js?

Node.js is a runtime that lets you run JavaScript outside the browser.
It's built on the powerful **V8 Engine**, which makes it fast.

But the real magic is:

- Event-driven architecture
- Non-blocking I/O
- Asynchronous execution

This is why Node.js is perfect for APIs, real-time apps, and scalable backends.

Understanding the Core (Most Important)

Before writing code, understand this:

Node.js is:

- Single-threaded
- But handles multiple requests using the **Event Loop**

Event Loop Phases:

1. Timers (`setTimeout`)
2. Pending callbacks
3. Poll (I/O operations)
4. Check (`setImmediate`)
5. Close callbacks

```
setTimeout(() => console.log("Timeout"), 0);
setImmediate(() => console.log("Immediate"));
```

This is the heart of Node.js. If you understand this, you're ahead of most beginners.

Modules System (How Node Works Internally)

Everything in Node is modular.

CommonJS

```
const fs = require('fs');
module.exports = myFunction;
```

ES Modules

```
import fs from 'fs';
export default myFunction;
```

File System (fs Module)

Working with files is core backend work.

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

// Sync
const data = fs.readFileSync('file.txt');

// Async
fs.readFile('file.txt', 'utf8', (err, data) => {
  console.log(data);
});

// Streams (for large files)
fs.createReadStream('file.txt');
```

Always prefer async or streams in real apps.

Creating Your First Server

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

http.createServer((req, res) => {
  res.end("Hello World");
}).listen(3000);
```

Now open:
http://localhost:3000

Express.js (Real-World Backend)

Nobody builds production apps with raw HTTP anymore.
We use **Express.js**.

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

app.use(express.json());

app.get('/', (req, res) => {
  res.send("Home Page");
});

app.listen(3000);
```

Middleware (The Backbone of Express)

Middleware runs between request and response.

```
app.use((req, res, next) => {
  console.log("Request received");
  next();
});
```

Types:

- Built-in
- Custom
- Third-party
- Error middleware

Routing

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

Asynchronous Programming

Callback

```
fs.readFile('file.txt', cb);
```

Promise

```
fs.promises.readFile('file.txt');
```

Async/Await (Best)

```
const data = await fs.promises.readFile('file.txt');
```

Streams (Underrated Superpower)

```
const stream = fs.createReadStream('file.txt');

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

Streams help you handle large files efficiently.

Buffers

```
const buffer = Buffer.from("Hello");
console.log(buffer.toString());
```

Used for binary data (images, files, etc.)

Events

```
const EventEmitter = require('events');
const emitter = new EventEmitter();

emitter.on('start', () => console.log("Started"));
emitter.emit('start');
```

Environment Variables

Never hardcode secrets.

Using **dotenv**:

```
require('dotenv').config();
console.log(process.env.PORT);
```

Authentication (Real Apps)

JWT (Login System)

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

const token = jwt.sign({ id: 1 }, "secret");
```

Password Security

```
const bcrypt = require('bcrypt');
const hash = await bcrypt.hash("password", 10);
```

Databases

MongoDB

Using **MongoDB** + **Mongoose**

```
mongoose.connect("mongodb://localhost/test");
```

MySQL

Using **MySQL**

```
const mysql = require('mysql2');
```

Project Structure (MVC)

```
/models
/controllers
/routes
/middleware
```

This is how real backend apps are structured.

Testing

```
test("adds numbers", () => {
  expect(1 + 1).toBe(2);
});
```

Tools:

- Jest
- Mocha

Performance & Scaling

Clustering

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

Worker Threads

```
const { Worker } = require('worker_threads');
```

Security Essentials

- Use Helmet
- Enable CORS
- Rate limiting

```
const cors = require('cors');
app.use(cors());
```

npm (Advanced)

```
npm install
npm uninstall
npm update
npm audit
```

Tools Every Developer Uses

- **Nodemon** → Auto restart
- **Postman** → API testing
- **PM2** → Production

Deployment

- Use VPS or cloud
- Use NGINX
- Run app with PM2

Advanced Topics (When You Level Up)

- WebSockets (real-time apps)
- GraphQL
- Microservices
- Redis caching
- Message queues (Kafka, RabbitMQ)

Final Advice (From Me to You)

When I started, I made a mistake:

❌ Trying to memorize everything
❌ Watching tutorials without building

Instead, do this:

Learn → Build → Break → Fix → Repeat

That's how real developers grow.

Conclusion

This cheatsheet covers:

- Basics
- Intermediate
- Advanced
- Production concepts

It's not something to memorize.
It's something to **use while building real projects**.

If you're serious about backend:

Next step: Build a real API
Even better: Build your own project (like a test platform or dashboard)