# Express.js Quick Reference

Routing, middleware, requests, responses, patterns

## Setup

### Create & Start Server

```
const express = require("express");
const app = express();
app.listen(3000, () => console.log("Running on :3000"));
```

### Built-in Middleware

```
app.use(express.json());          // parse JSON bodies
app.use(express.urlencoded({ extended: true })); // form data
app.use(express.static("public")); // serve static files
```

## Routing

### HTTP Methods

```
app.get("/users", (req, res) => res.json(users));
app.post("/users", (req, res) => res.status(201).json(req.body));
app.put("/users/:id", (req, res) => res.json(updated));
app.delete("/users/:id", (req, res) => res.sendStatus(204));
```

### Route Parameters

```
app.get("/users/:id", (req, res) => {
  const { id } = req.params;
  res.json({ id });
});
```

### Query Strings

```
// GET /search?q=express&page=2
app.get("/search", (req, res) => {
  const { q, page } = req.query;
  res.json({ q, page });
});
```

## Middleware

### Application-Level

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

### Route-Level

```
const auth = (req, res, next) => {
  if (!req.headers.authorization) return res.sendStatus(401);
  next();
};
app.get("/secret", auth, (req, res) => res.json({ ok: true }));
```

### Execution Order

| | |
|---|---|
| **`app.use(fn)`** | Runs on every request (in order) |
| **`app.use(path, fn)`** | Runs only on matching path prefix |
| **`next()`** | Passes control to next middleware |
| **`next(err)`** | Skips to error handler |

## Request & Response

### Request Object

| | |
|---|---|
| **`req.params`** | Route parameters (**`/users/:id`**) |
| **`req.query`** | Query string (**`?key=val`**) |
| **`req.body`** | Parsed request body (needs parser) |
| **`req.headers`** | Request headers object |
| **`req.method`** | HTTP method (GET, POST, ...) |
| **`req.path`** | URL pathname |
| **`req.cookies`** | Cookies (needs cookie-parser) |

### Response Object

| | |
|---|---|
| **`res.json(obj)`** | Send JSON response |
| **`res.send(body)`** | Send string/Buffer/object |
| **`res.status(code)`** | Set HTTP status (chainable) |
| **`res.redirect(url)`** | 302 redirect (or pass status) |
| **`res.sendFile(path)`** | Send a file as response |
| **`res.sendStatus(code)`** | Send status with default text |
| **`res.set(header, val)`** | Set response header |

## Error Handling

### Error Middleware

```
// Must have 4 parameters — Express recognizes it as error handler
app.use((err, req, res, next) => {
  console.error(err.stack);
  res.status(err.status || 500).json({ error: err.message });
});
```

Define error handlers after all other app.use() and routes

### Async Errors

```
// Wrap async route handlers to catch rejections
const wrap = (fn) => (req, res, next) =>
  Promise.resolve(fn(req, res, next)).catch(next);

app.get("/data", wrap(async (req, res) => {
  const data = await fetchData();
  res.json(data);
}));
```

## Static Files

### Serve Static Directory

```
app.use(express.static("public"));
// serves public/style.css at /style.css

// With virtual path prefix
app.use("/assets", express.static("public"));
// serves public/style.css at /assets/style.css
```

### Options

| | |
|---|---|
| **`dotfiles`** | **`'ignore'`** \| **`'allow'`** \| **`'deny'`** |
| **`maxAge`** | Cache-Control max-age in ms |
| **`index`** | Index file name (default: **`index.html`**) |
| **`fallthrough`** | Pass to next middleware on 404 |

## Templates

### View Engine Setup

```
app.set("view engine", "ejs");
app.set("views", "./views");

app.get("/", (req, res) => {
  res.render("index", { title: "Home", items: [1, 2, 3] });
});
```

### Common Engines

| | |
|---|---|
| **`ejs`** | Embedded JS templates (**`<%= val %>`**) |
| **`pug`** | Indentation-based (formerly Jade) |
| **`handlebars`** | Mustache-style (**`{{val}}`**) |

## Router

### Modular Routes

```
// routes/users.js
const router = require("express").Router();
router.get("/", (req, res) => res.json(users));
router.get("/:id", (req, res) => res.json(user));
module.exports = router;
```

### Mount Router

```
const usersRouter = require("./routes/users");
app.use("/api/users", usersRouter);
// GET /api/users -> router's "/"
// GET /api/users/5 -> router's "/:id"
```

### Router Methods

| | |
|---|---|
| **`router.get/post/put/delete`** | HTTP method handlers |
| **`router.use(fn)`** | Router-level middleware |
| **`router.param(name, fn)`** | Pre-process route param |
| **`router.route(path)`** | Chain methods on one path |

## Authentication Patterns

### JWT Middleware

```
const jwt = require("jsonwebtoken");
const auth = (req, res, next) => {
  const token = req.headers.authorization?.split(" ")[1];
  if (!token) return res.sendStatus(401);
  req.user = jwt.verify(token, process.env.SECRET);
  next();
};
```

### Protected Routes

```
app.get("/profile", auth, (req, res) => {
  res.json({ user: req.user });
});
app.use("/api/admin", auth, adminRouter);
```

## Common Patterns

### CORS

```
const cors = require("cors");
app.use(cors());                          // allow all origins
app.use(cors({ origin: "https://example.com" })); // restrict
```

### Environment & Config

```
const port = process.env.PORT || 3000;
app.listen(port);

// Access env in routes
if (app.get("env") === "production") {
  app.use(helmet());
}
```

### Useful npm Packages

| | |
|---|---|
| **`cors`** | Cross-origin resource sharing |
| **`helmet`** | Security headers |
| **`morgan`** | HTTP request logger |
| **`cookie-parser`** | Parse Cookie header |
| **`dotenv`** | Load **`.env`** into **`process.env`** |
| **`multer`** | Multipart form data (file uploads) |