REFERENSI CEPAT EXPRESS.JS

Routing, middleware, request, response, pola umum

Setup

Buat & Jalankan Server

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

Middleware Bawaan

```
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 Method

```
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));
```

Parameter Route

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

Query String

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

Middleware

Level Aplikasi

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

Level Route

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

Urutan Eksekusi

`app.use(fn)`	Berjalan pada setiap request (berurutan)
`app.use(path, fn)`	Berjalan hanya pada prefix path yang cocok
`next()`	Teruskan kontrol ke middleware berikutnya
`next(err)`	Langsung ke error handler

Request & Response

Objek Request

`req.params`	Parameter route (`/users/:id`)
`req.query`	Query string (`?key=val`)
`req.body`	Body request yang diparsing (butuh parser)
`req.headers`	Objek header request
`req.method`	HTTP method (GET, POST, ...)
`req.path`	Pathname URL
`req.cookies`	Cookie (butuh cookie-parser)

Objek Response

`res.json(obj)`	Kirim response JSON
`res.send(body)`	Kirim string/Buffer/objek
`res.status(code)`	Set HTTP status (bisa di-chain)
`res.redirect(url)`	Redirect 302 (atau sertakan status)
`res.sendFile(path)`	Kirim file sebagai response
`res.sendStatus(code)`	Kirim status dengan teks default
`res.set(header, val)`	Set header response

Penanganan Error

Middleware Error

```
// 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 });
});
```

Definisikan error handler setelah semua app.use() dan route lainnya

Error Async

```
// 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);
}));
```

File Statis

Sajikan Direktori Statis

```
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
```

Opsi

`dotfiles`	`'ignore'` | `'allow'` | `'deny'`
`maxAge`	Cache-Control max-age dalam ms
`index`	Nama file index (default: `index.html`)
`fallthrough`	Teruskan ke middleware berikutnya saat 404

Template

Setup View Engine

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

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

Engine Umum

`ejs`	Template JS embedded (`<%= val %>`)
`pug`	Berbasis indentasi (dulunya Jade)
`handlebars`	Gaya Mustache (`{{val}}`)

Router

Route Modular

```
// 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"
```

Method Router

`router.get/post/put/delete`	Handler HTTP method
`router.use(fn)`	Middleware level router
`router.param(name, fn)`	Pre-proses parameter route
`router.route(path)`	Chain method pada satu path

Pola Autentikasi

Middleware JWT

```
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();
};
```

Route Terproteksi

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

Pola Umum

CORS

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

Environment & Konfigurasi

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

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

Paket npm Berguna

`cors`	Cross-origin resource sharing
`helmet`	Header keamanan
`morgan`	Logger HTTP request
`cookie-parser`	Parse header Cookie
`dotenv`	Muat `.env` ke `process.env`
`multer`	Data form multipart (upload file)