INSTRUCTOR:
PROFESSOR SOLO _

MODULE 2: THE EXPRESS ENGINE

ARCHITECTURE, ROUTING, AND THE REQUEST-RESPONSE CYCLE

MISSION PROFILE:

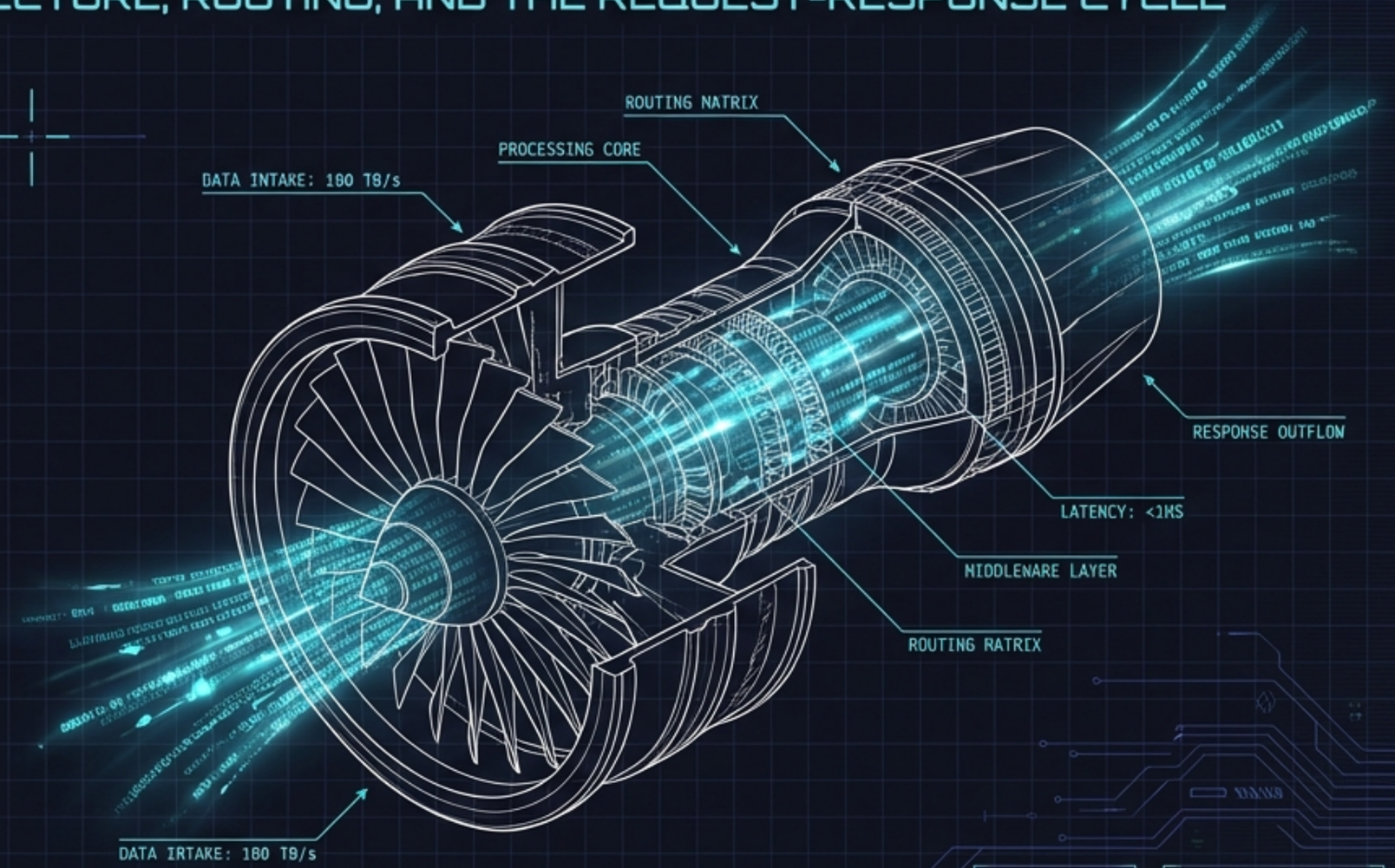
Upgrade operational capacity from raw Node.js machinery to the Express.js framework.

OBJECTIVES:

1. Maximize Throughput.
2. Ensure Maintainability.
3. Master the Request-Response Cycle.

STATUS: ⚠

Strap in. We are switching from manual assembly to automatic control.



THE UPGRADE: RAW NODE VS. EXPRESS

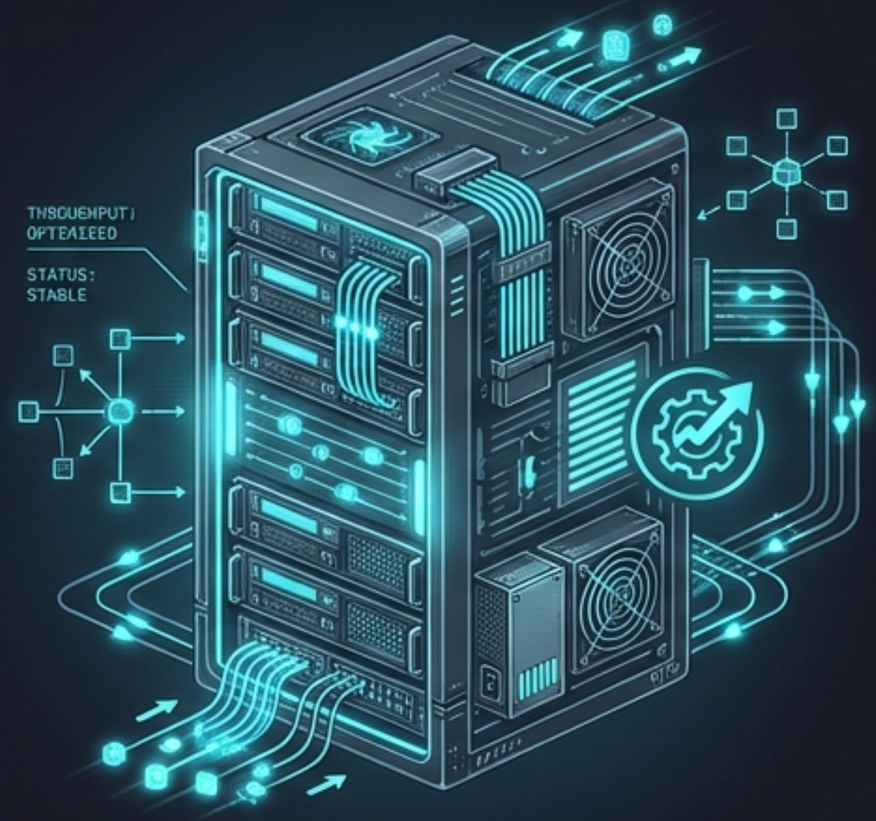
RAW NODE.JS

THE RAW MACHINERY

Powerful, but requires manual assembly for every route and error.

- High configuration cost.
- 'Re-inventing the wheel' risk.

EXPRESS FRAMEWORK



THE CONTROL PANEL

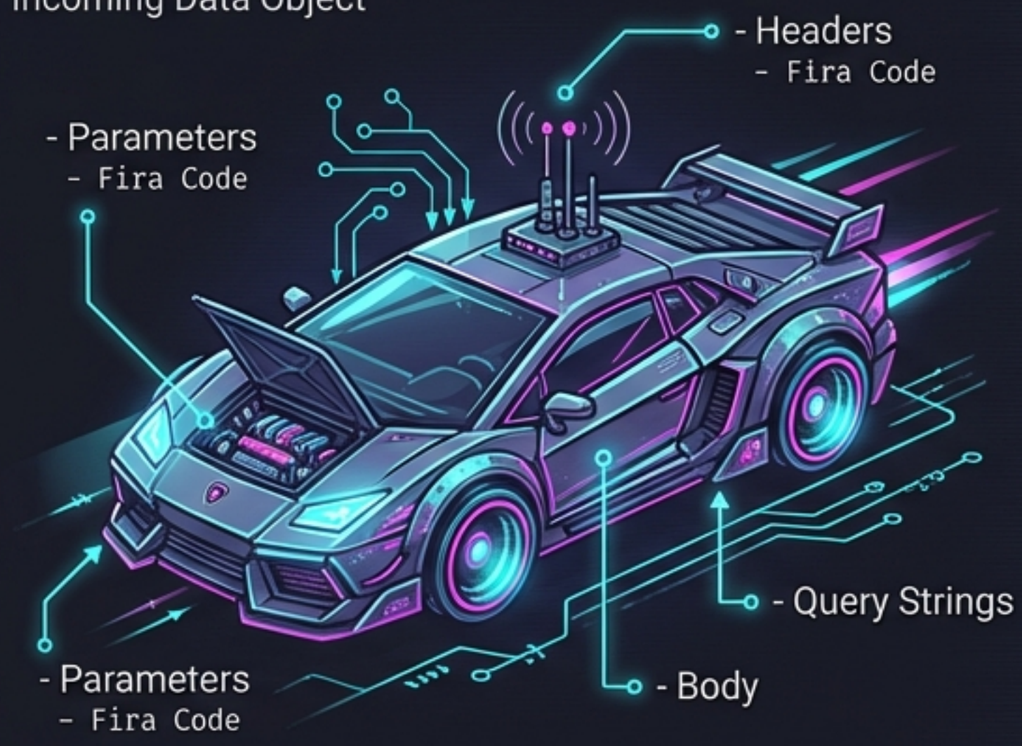
The 'Unopinionated' Minimalist Layer.

- The 'E' in MERN Stack.
- Organizes power without hiding it.
- **Solo's Directive:** "We install Express because we care about throughput."

ANATOMY OF TRAFFIC: REQ & RES

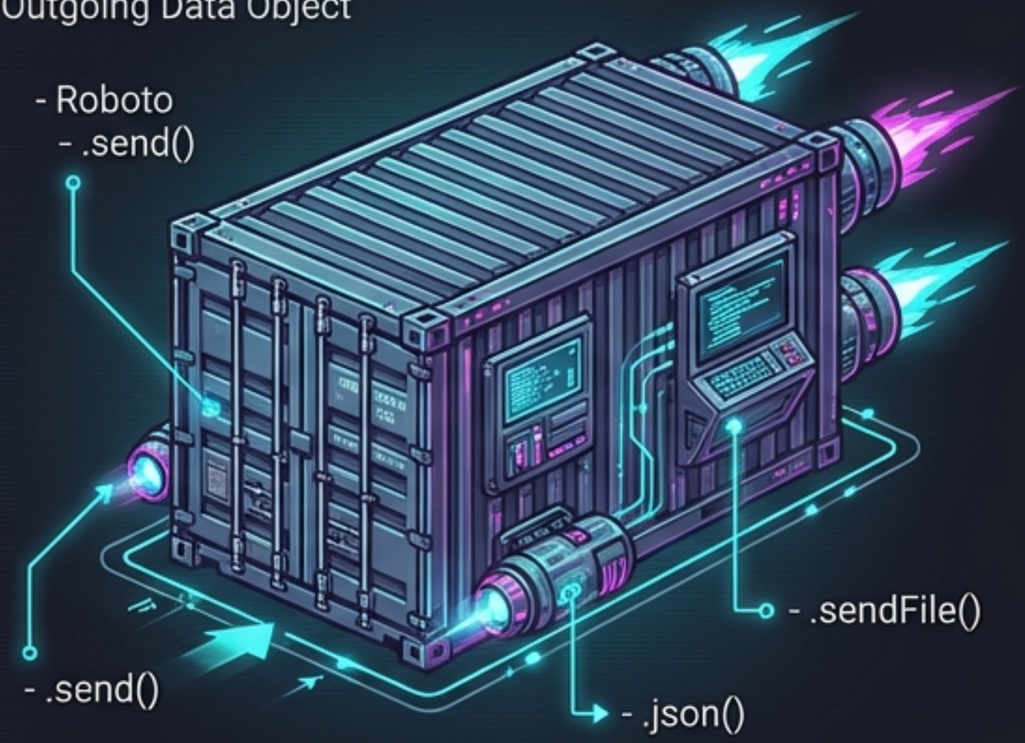
REQ (The Vehicle)

Incoming Data Object



RES (The Cargo)

Outgoing Data Object



THE GOLDEN RULE:

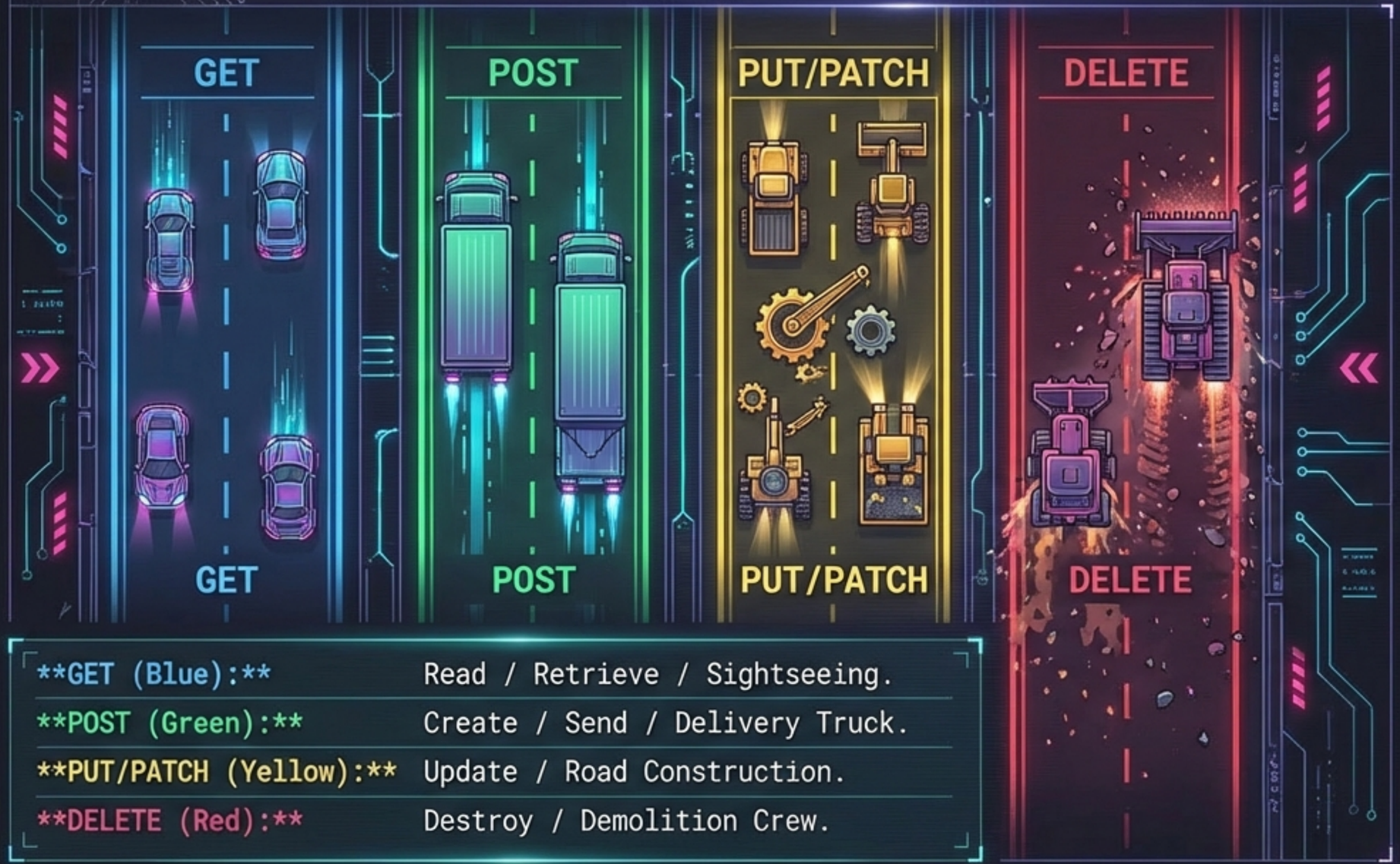
The cycle must close. Every request needs a response (`res`), or the connection hangs.

```
const express = require('express');
const app = express();
const PORT = 3000;

app.listen(PORT, () => console.log(`Driving
  on port ${PORT}`));
```

TRAFFIC CONTROL & LANES

Logic: app.METHOD(PATH, HANDLER)



GET (Blue):	Read / Retrieve / Sightseeing.
POST (Green):	Create / Send / Delivery Truck.
PUT/PATCH (Yellow):	Update / Road Construction.
DELETE (Red):	Destroy / Demolition Crew.

STRICT ENFORCEMENT

`app.get('/things')` and `app.post('/things')` are different roads.

NOTE: Browser HTML forms only support GET and POST natively.

DESIGNATED LANES VS. LUGGAGE

req.params (The Destination)

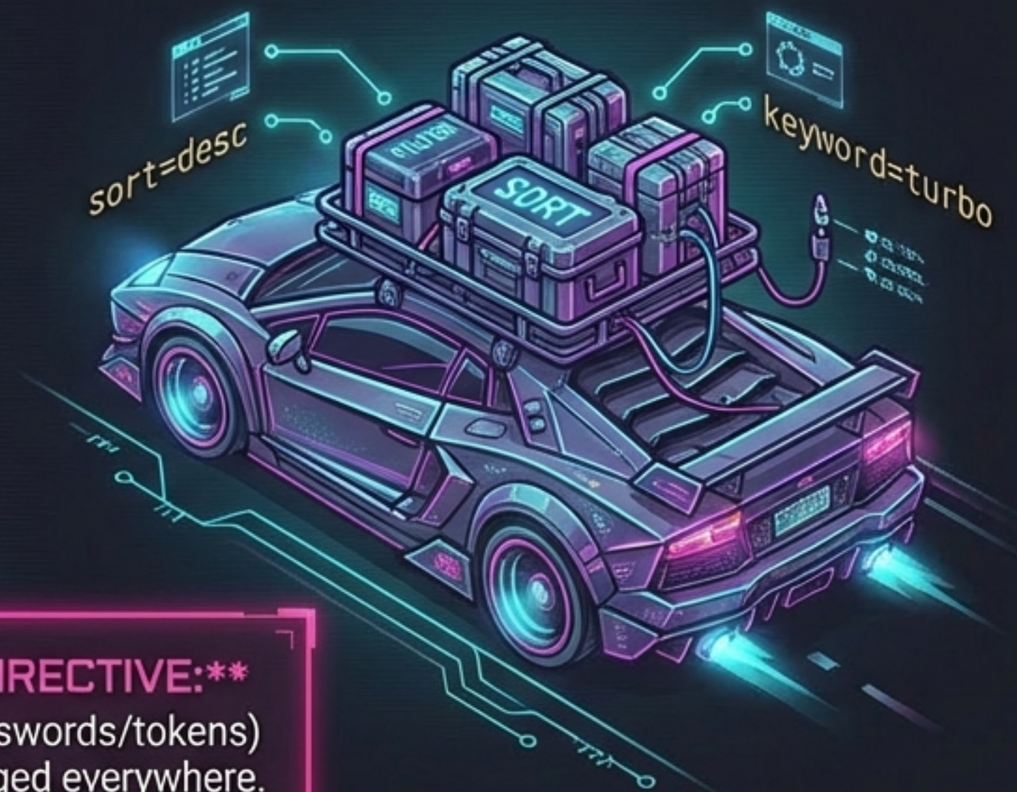
Dynamic segments for specific targets.

Code: `app.get('/things/:thingId', ...)`
Result: Captures `/things/42`

req.query (The Luggage)

Optional modifiers like filters or sorting.

Code: `/search?sort=desc&keyword=turbo`
Result: `{ sort: 'desc', keyword: 'turbo' }`

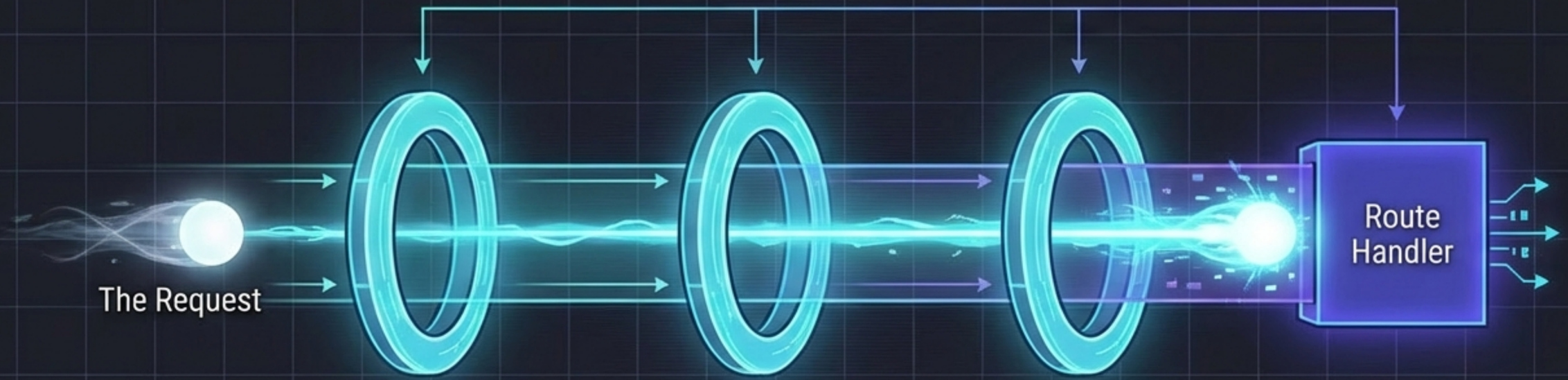


****SOLO'S SECURITY DIRECTIVE:****

Never put sensitive data (passwords/tokens) in query strings. URLs are logged everywhere.

THE SECURITY GAUNTLET (MIDDLEWARE)

Core Concept: Express is a series of functions the request passes through.

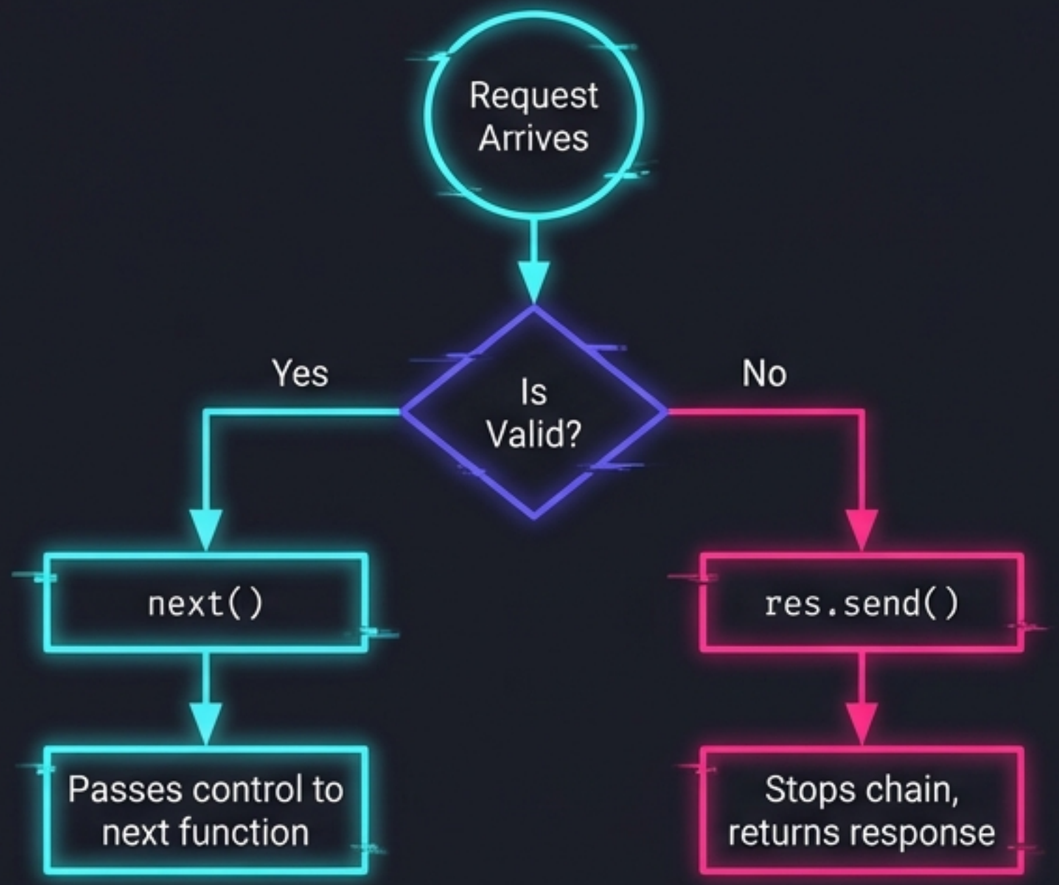


⚠ **SOLO'S ANALOGY:** ⚠

Think of it as a series of laser gates. You must pass every single one to get to the destination. Standardized equipment you can install anywhere.

CUSTOM CHECKPOINTS

The Signature: **(req, res, next)**




```
const authOfficer = (req, res, next) => {
  if (req.query.passport === 'valid') {
    // Wave them through
    next();
  } else {
    // Turn around
    res.status(403).send('Access Denied.');
  }
};
```

1. Welcome (Success): Send immediate response.

2. Access Denied (Failure): The chain stops here.

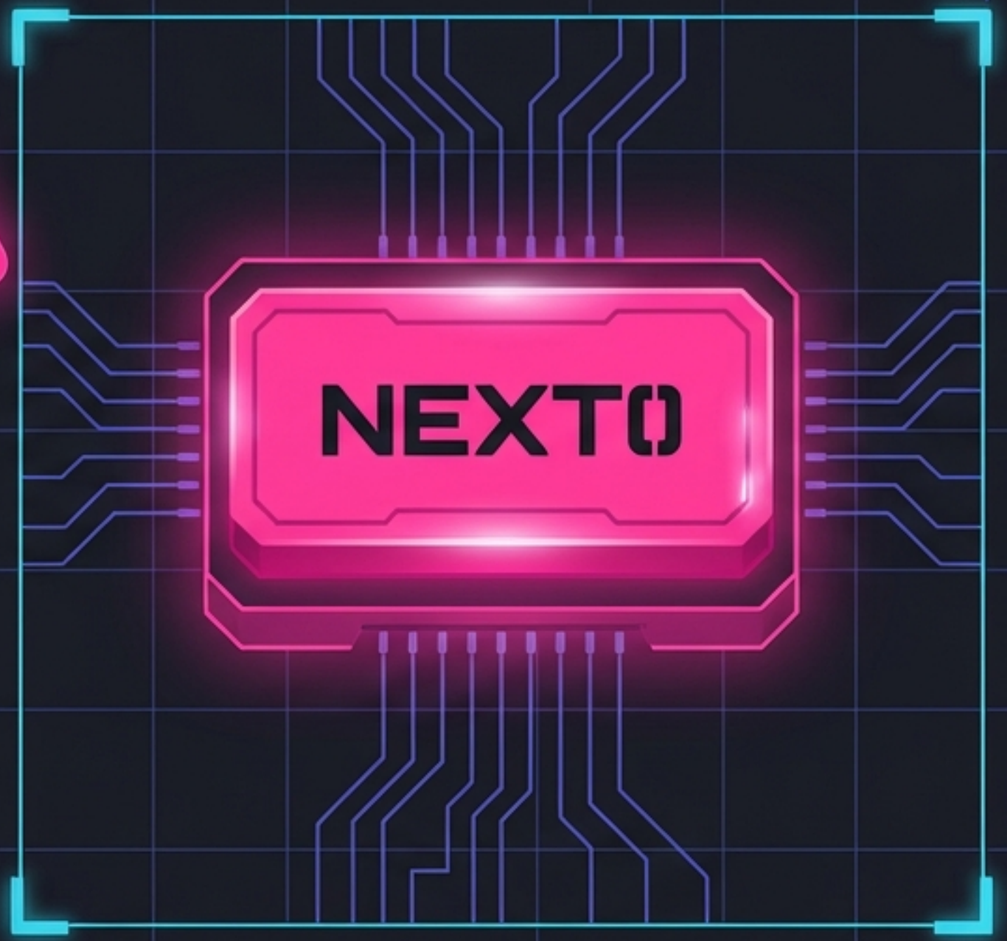
3. Inspect & Pass (next()): Open the gate.

THE “NEXT” FUNCTION

CRITICAL COMPONENT
A function that passes control to the next middleware in the stack.

THE SOLO SHORTCUT (WARNING):
If you don’t call `next()` or send a response (`res`), the request **hangs forever**. The driver sits there honking until the connection times out.

DIRECTIVE:
Don’t be that guy. Always close the loop.



AUTOMATED SURVEILLANCE (MORGAN)

Don't reinvent the wheel. Hire a professional.

TOOL: Morgan Middleware

FUNCTION: Analyzes the 'splatter' of HTTP requests.

OUTPUT: Method, URL, Status, Response Time.

```
npm install morgan
...
app.use(morgan('dev'));
```

```
GET /api/users 200 5.4ms - 120b
POST /api/login 200 12.3ms - 450b
GET /api/dashboard 404 2.1ms - 95b
PUT /api/settings 200 8.7ms - 210b
DELETE /api/account 500 15.6ms - 310b
GET /api/analytics 200 4.2ms - 80b
```

SOLO'S DIRECTIVE: Install this first. Driving blind is for stuntmen, not engineers.

THE SHIPPING MANIFEST (RESPONSES)

The Generalist

Quick text or HTML.

The Specialist

The Industry Standard for APIs.
Automatically sets headers.

Condition Report

200 Series: Cargo Loaded (OK).
400 Series: Driver Error (Bad Request/Forbidden).
500 Series: Engine Failure (Server Error).

Note: Express defaults to 200. If you crash, you must explicitly set `.status(500)`.

THE GIFT SHOP (STATIC FILES)

Self-service downloads.

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

MECHANISM:

Maps a folder to the root URL.
The URL does not include `/public`.
Express looks inside automatically.

SECURITY WARNING:

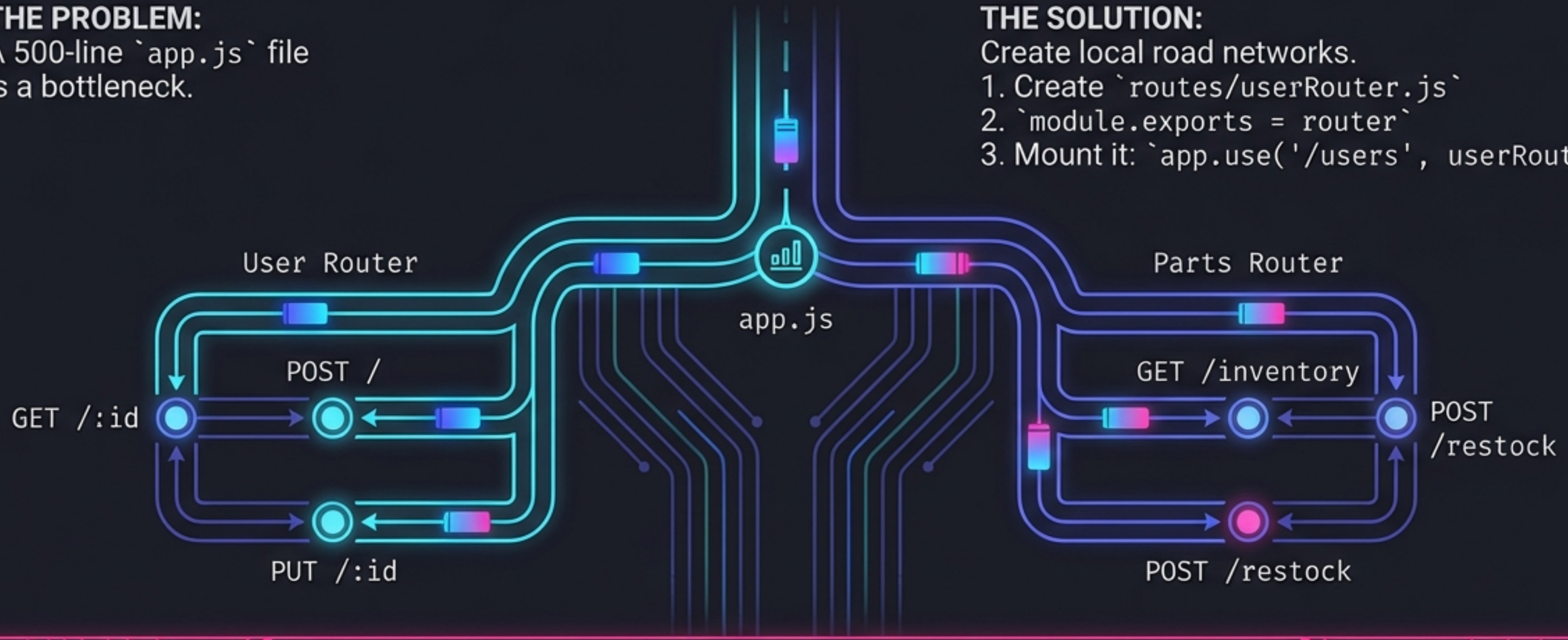
`express.static` opens the directory to the world. Never put `server.js` or `.env` files here.

MODULAR ROUTING NETWORKS

Managing the traffic jam.

THE PROBLEM:
A 500-line `app.js` file is a bottleneck.

THE SOLUTION:
Create local road networks.
1. Create `routes/userRouter.js`
2. `module.exports = router`
3. Mount it: `app.use('/users', userRoutor)`



***SOLO'S LOGIC:** We do this for developer sanity. Modular code enables teams to move fast without crashing into each other.

THE NAVIGATOR (PATH MODULE)

The smart GPS that calculates the correct route for the Operating System.


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

const fullPath = path.join(__dirname,
'data', 'file.json');
res.sendFile(fullPath);
```

- **`__dirname`:**
  The directory where THIS script lives.
- **`path.join`:**
  The smart GPS that calculates the correct route for the Operating System.

BULK DELIVERY:
Use `res.sendFile(absolutePath)` for sending entire HTML files.

WINDOWS:
`C:\path\to\file`

LINUX:
`/path/to/file`

NOTE: String concatenation is reckless driving. Use the navigator.

NOTE: String concatenation is reckless driving. Use the navigator.

DEAD ENDS (404 HANDLING)

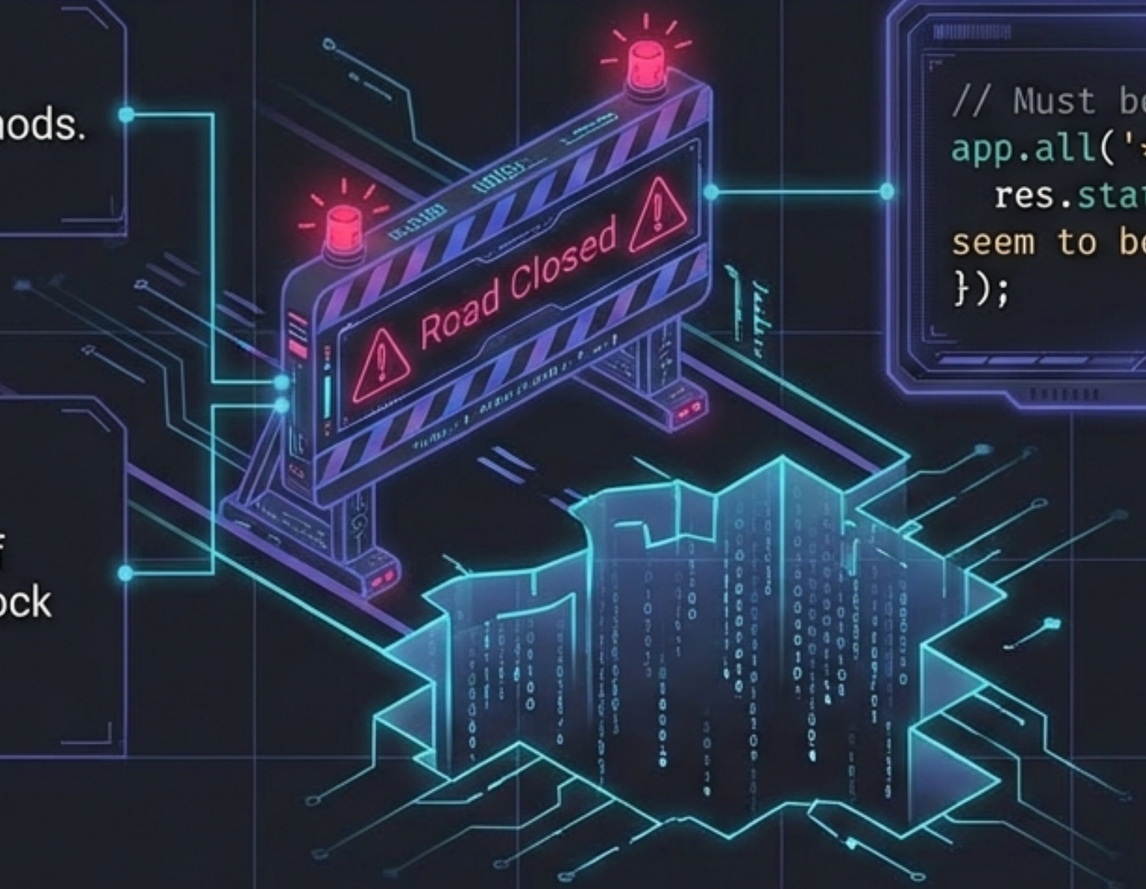
The Safety Net.

MECHANICS:

`app.all`: Matches ALL methods.
`*`: Matches ALL paths.

PLACEMENT RULE:

Express runs top-to-bottom. If you put this at the top, you block the whole highway. It is a roadblock, not a checkpoint.




```
// Must be placed LAST
app.all('*', (req, res) => {
  res.status(404).send('You
seem to be lost.');
});
```

SOLO'S DIAGNOSTICS & PITFALLS

Holographic Diagnostic Checklist

1. The Hanging Request
Forgot `next()` or `res.send`. Infinite spinning.

2. The Double Dip
Trying to send a response *after* already sending one.
Error: `Headers already sent`.

3. The Ghost Route
Defining a specific route (`/things/:id`) *after* a catch-all.

4. The Silent Failure
Sending an error message without `.status(404)`. Browser thinks it is a success (200 OK).

FINAL DIRECTIVE: SPEED IS NOTHING WITHOUT CONTROL.
BUILD IT CLEAN. KEEP THE PIPELINE FLOWING.