FZG 01.A // SYSTEM BOOT DIAGNOSTIC

0x00FF.A1 // PROTOCOL STAGE

DATA COLUMN 01.A

COURSE:
ADVANCED BACKEND SYSTEMS

INSTRUCTOR:
PROFESSOR SOLO

OBJECTIVE:
CLIENT-SIDE SCRIPTING >> SERVER-SIDE ARCHITECTURE

STATUS:
INITIALIZING...
[FLASHING INDICATOR]

PROTOCOL: NODE.JS // DAY 01

NODE.JS FOUNDATIONS

FIG 02.B // INSTRUCTOR AVATAR

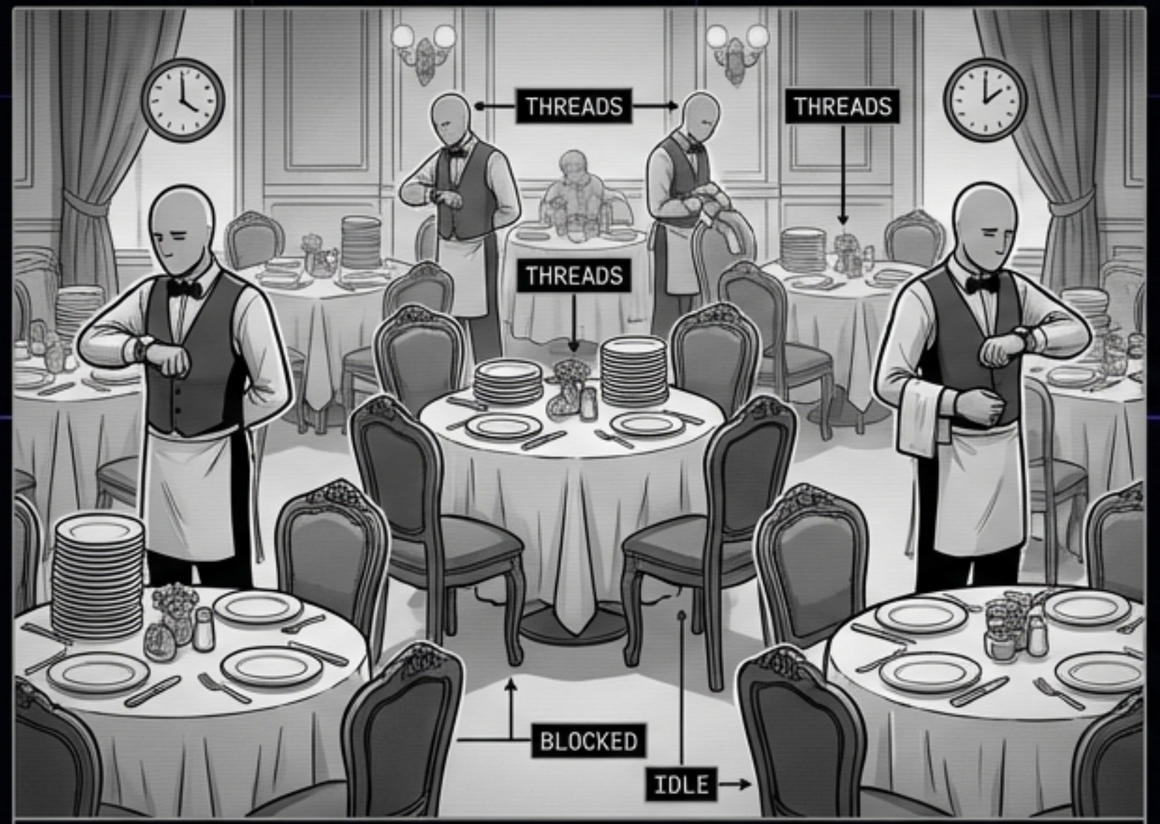
Buckle up. We're leaving the browser sandbox. **No lifeguards** where we're going.

PROGRESS: 10% // LOADING CORE MODULES...

FIG 01.A // COMPARATIVE DIAGNOSTIC

THE PROBLEM WITH THREADS (OR: WHY APACHE IS A SNOOZE)

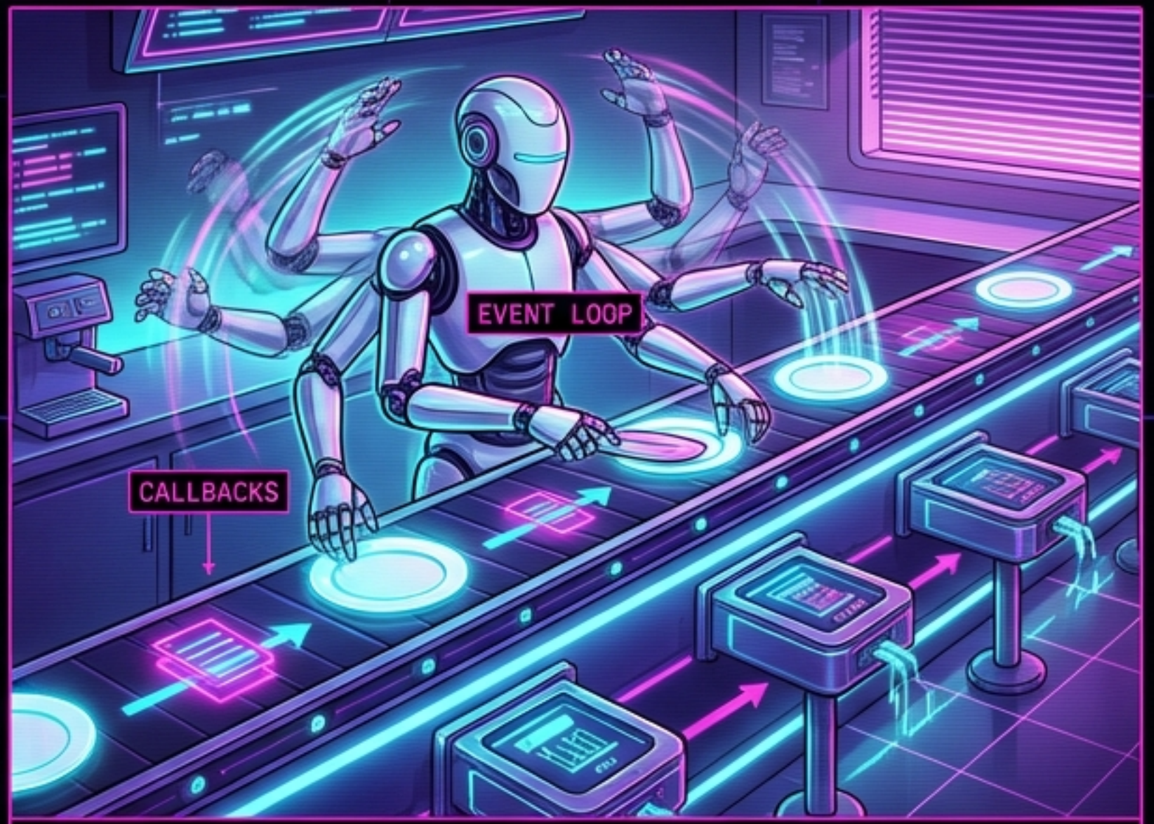
TRADITIONAL SERVERS: 1 REQUEST = 1 THREAD. BLOCKING I/O.



Waiters (Threads) are allocated one per customer (request). When a request waits for data (I/O), the entire thread halts, consuming resources without work. Inefficient at scale.

FIG 01.B // LEGACY ARCHITECTURE (DESATURATED)

NODE.JS: SINGLE THREAD. NON-BLOCKING I/O.



The single Event Loop handles all requests asynchronously. It delegates I/O tasks, then moves to the next request, processing callbacks only when data is ready. Maximizes throughput.

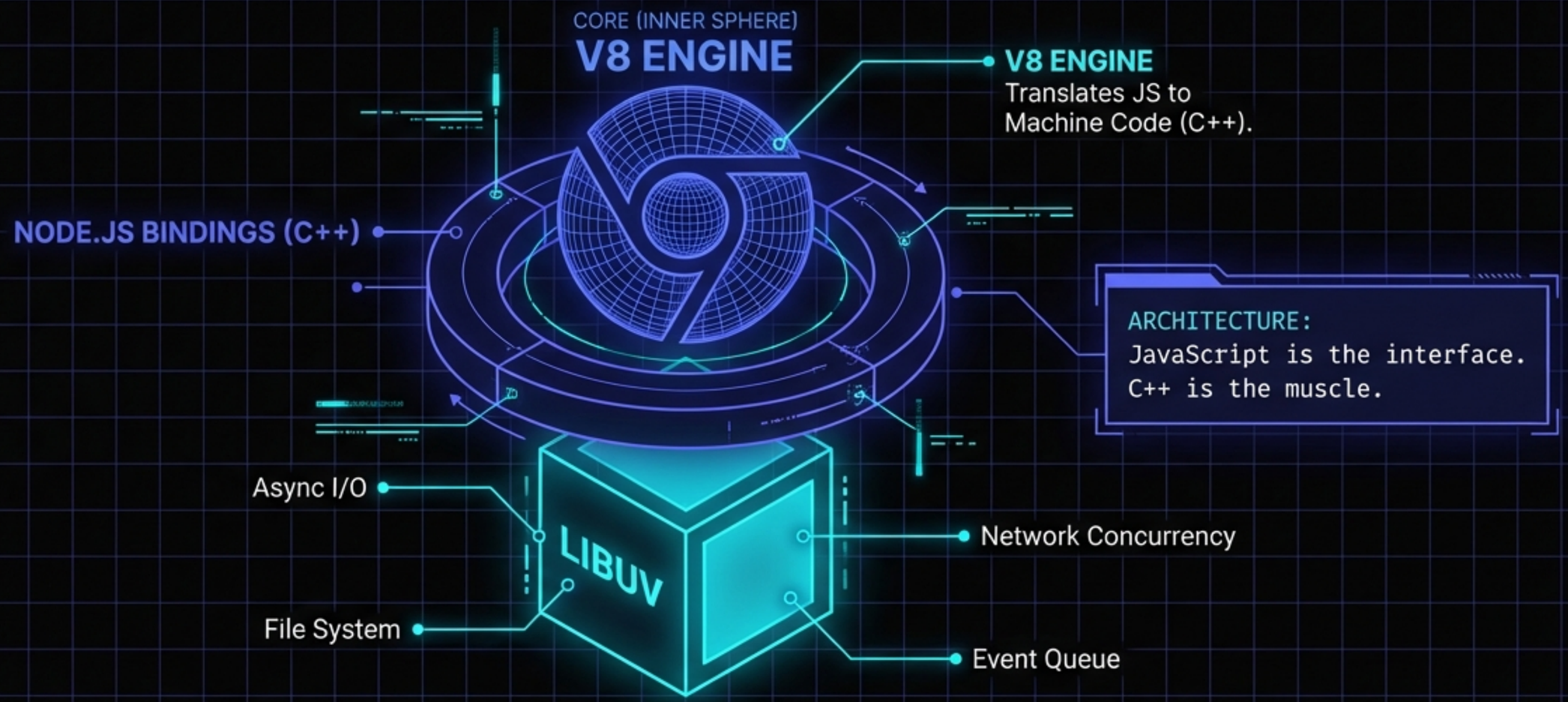
FIG 01.C // MODERN ASYNC FLOW (ACTIVE)

ONE FAST COOK BEATS A THOUSAND IDLE WAITERS.
THROUGHPUT > RAW CALCULATION.

The power of Node.js lies in handling concurrency efficiently, not in raw computational speed.

THE ENGINE ROOM: V8 & LIBUV

FIG 02.A // CORE ARCHITECTURE



⚠ **WARNING:** You are driving a C++ racecar. Respect the machinery.

FIG 02.B // SYSTEM DIAGNOSTIC (ACTIVE)

STATUS: OPTTMT7FD

YOU ARE NOT IN KANSAS (THE BROWSER) ANYMORE

FIG 03.A // CHARACTER PROFILE: BROWSER

THE BROWSER

AKA: THE SANDBOX

STRENGTH:
DOM Manipulation

WEAKNESS:
No FIle Access

GLOBAL:
`window`

STATUS:
LOCKED / SAFE

FIG 03.B // CHARACTER PROFILE: NODE.JS

NODE.JS

AKA: THE CONSTRUCTION SITE

STRENGTH:
Full System Access (FS, Network)

WEAKNESS:
No DOM (No HTML)

GLOBAL:
`global`

STATUS:
UNLOCKED / DANGEROUS

⚠ **STOP TRYING TO `document.getElementById`. THERE IS NO DOCUMENT. ONLY STREAMS.** ⚠

FIG 03.C // SYSTEM ADVISORY (CRITICAL)

SITE VITALS & THE GLOBAL OBJECT

LOCATION DATA

```
__dirname:  /usr/local/bin/project
__filename: /usr/local/bin/project/app.js
```

IDENTITY

```
process.pid: 1024
process.env: [SECRET KEYS HIDDEN]
```

FIG 04.A // CENTRAL COMMAND

CENTRAL COMMAND

```
global: [Object]
  setTimeout
  console
  Buffer
  process
}
```

⚠ SECURITY ALERT

Use **process.env** for secrets. Never hardcode API keys.

IGNITION SEQUENCE: REPL VS. SCRIPT

REPL // SANDBOX MODE

FIG 05.A

```
> 124 * 8
< 992
> const x = 'temp'
> // Session lost on exit
```

Like Snapchat. Once you close it, it's gone.

SCRIPT // PRODUCTION MODE

FIG 05.B

```
$ node --watch app.js
[INFO] Restarting...
[SUCCESS] App running.
```

Permanent Record. Auto-restarts on save.

SYSTEM DIAGNOSTIC

MODE SELECT

FIG 05.C // SYSTEM INITIALIZATION (READY)

STANDARD ISSUE GEAR (CORE MODULES)

File System R/W

Create Servers

System Diagnostics

Normalize Paths (Win/Mac)

FACTORY SETTINGS: Baked into the binary. No download required.

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

FIG 07.A // LEGACY PROTOCOL

THE CIVIL WAR: COMMONJS VS. ES MODULES

CJS (COMMONJS)

FIG 07.A

```
const data = require('./data');
```

> Synchronous
> Legacy Standard
> Dynamic Loading

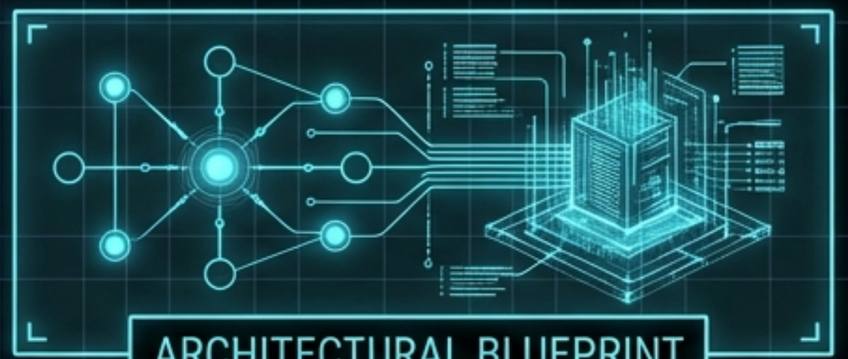
INDUSTRIAL SUPPLY CHAIN

ESM (ES MODULES)

FIG 07.B

```
import data from './data.js';
```

> Asynchronous
> Modern Standard
> Static Analysis



ARCHITECTURAL BLUEPRINT

SYSTEM: CLASH DIAGNOSTIC

MODE SELECT

⚠ **RULE:** New projects use ESM (**"type": 'module"**). Don't mix **require()** into ESM. ⚠

SYSTEM: CLASH DETECTED

STATUS: CRITICAL PATH ANALYSIS

SUPPLY CHAIN LOGISTICS (NPM)

NPM COMMAND PROTOCOLS

FIG 08.B

NPM INIT

The interview to create the manifest. Generates package.json through interactive prompts or default settings.

```
npm init -y
```

FIG 08.C

NPM INSTALL

The command to fill the warehouse. Downloads dependencies from the registry and stores them in node_modules.

```
npm install express
```

FIG 08.D

NPM RUN

The automation trigger. Executes scripts defined in the manifest, such as development servers, build processes, or tests.

```
npm run dev
```

BREAKING THE GLASS: THE FILE SYSTEM

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

async function readManifest() {
  try {
    const data = await fs.readFile('./secret.txt', 'utf-8');
    console.log('Manifest Decoded:', data);
  } catch (err) {
    console.error('Access Denied:', err);
  }
}

readManifest();
```

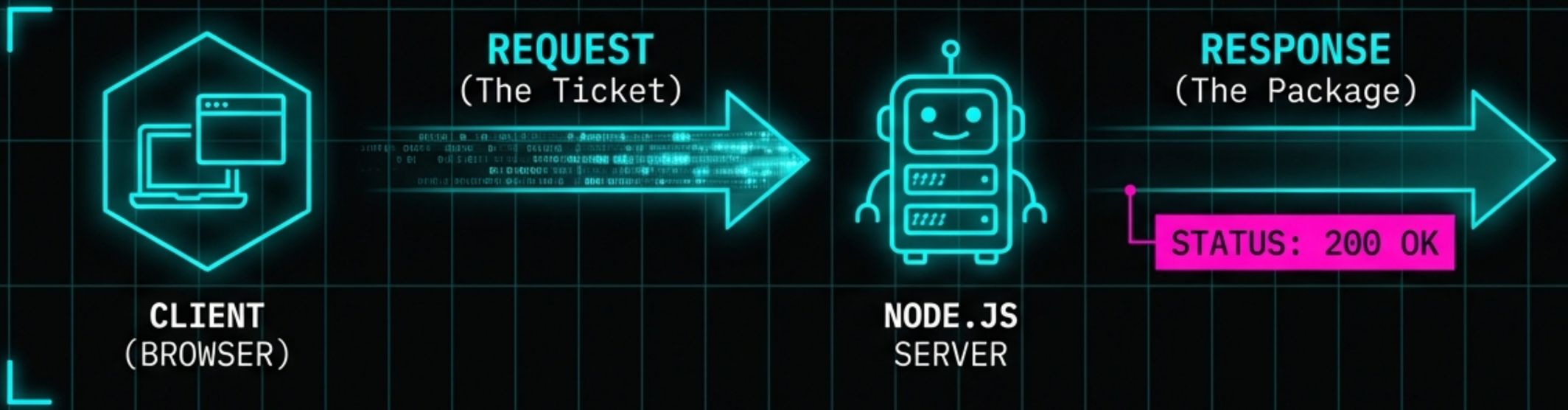
Use **.promises** to avoid **Callback Hell.**

Async/Await handles the non-blocking I/O.

Always handle errors. Don't crash silently.

FIG 10.A // REQUEST/RESPONSE CYCLE

HELLO WORLD? NO. HELLO SERVER.



REQ (Request)
Incoming URL, Headers.

RES (Response)
Outgoing Body, Status Code.

RULE
Always call res.end() **or the browser spins forever.**

SYSTEM: NODE.JS SERVER ONLINE

STATUS: CYCLE OPERATIONAL

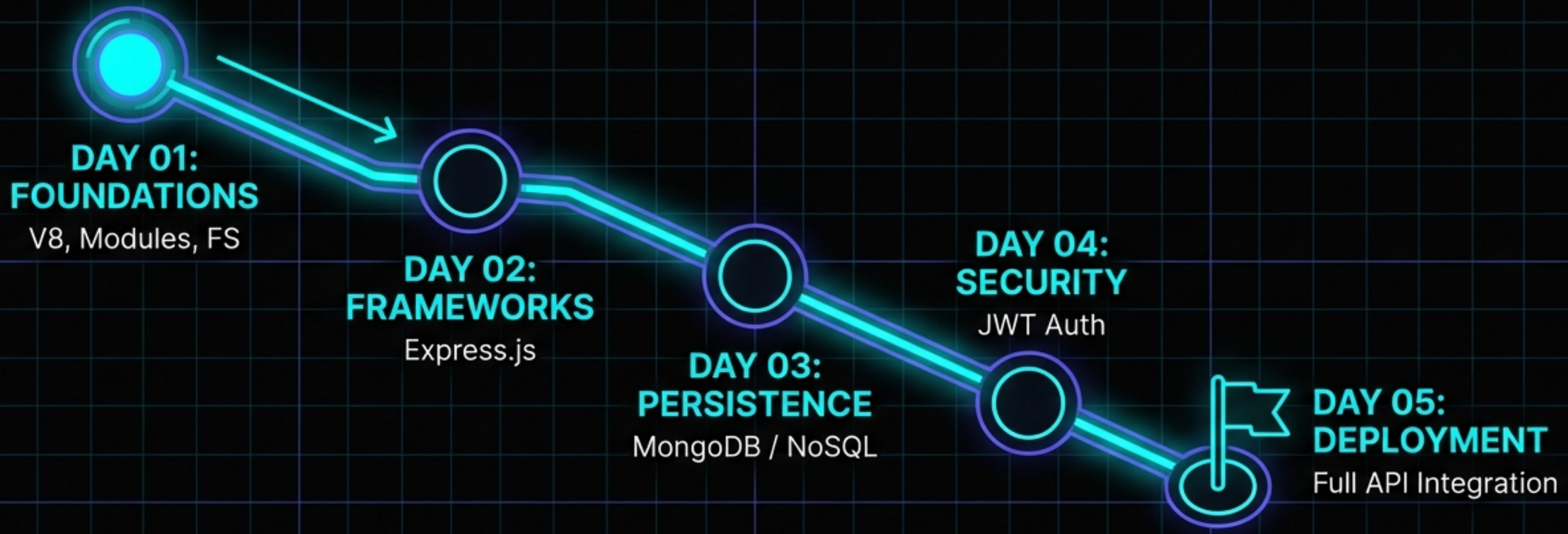
THE BLUEPRINT: YOUR FIRST API

```
import http from 'node:http';

const server = http.createServer((req, res) => {
  if (req.url === '/') {
    res.writeHead(200, { 'Content-Type': 'text/plain' });
    res.end('System Online: Node.js is active.');
  } else {
    res.writeHead(404);
    res.end('404: Sector Not Found');
  }
});

server.listen(3000, () => {
  console.log('Server listening on port 3000...');
});
```

CORE PATTERN: Every framework (Express, Nest) is just a wrapper around this raw HTTP module.

FIG 01.A // MISSION TRAJECTORY

THE MISSION TRAJECTORY



It gets harder before it gets easier. Trust the process.

SYSTEM: NODE.JS LEARNING PATH ONLINE

STATUS: TRAJECTORY ACTIVE

SHOW WHAT YOU NODE: AUDIT PHASE

- [ ] **VERIFY ENGINE:** Run `'node -v'` in terminal.
- [ ] **CREATE DNA:** Run `'npm init -y'` to generate `package.json`.
- [ ] **LAUNCH SERVER:** Create `'server.js'` on `Port 3000`
- [ ] **ACCESS HARDWARE:** Write a script to log `'os.uptime()'`.

MANDATORY

PROFESSOR SOLO: "I don't grade on effort. I grade on exit code 0."

SYSTEM STANDBY...

NEXT CLASS: EXPRESS.JS
REQUIRED READING: NODE.JS DOCS (FS & HTTP)
RESOURCE LINK: GITHUB.COM/SOLO/NODE-DAY-01

Go break something. Then fix it. Solo out.