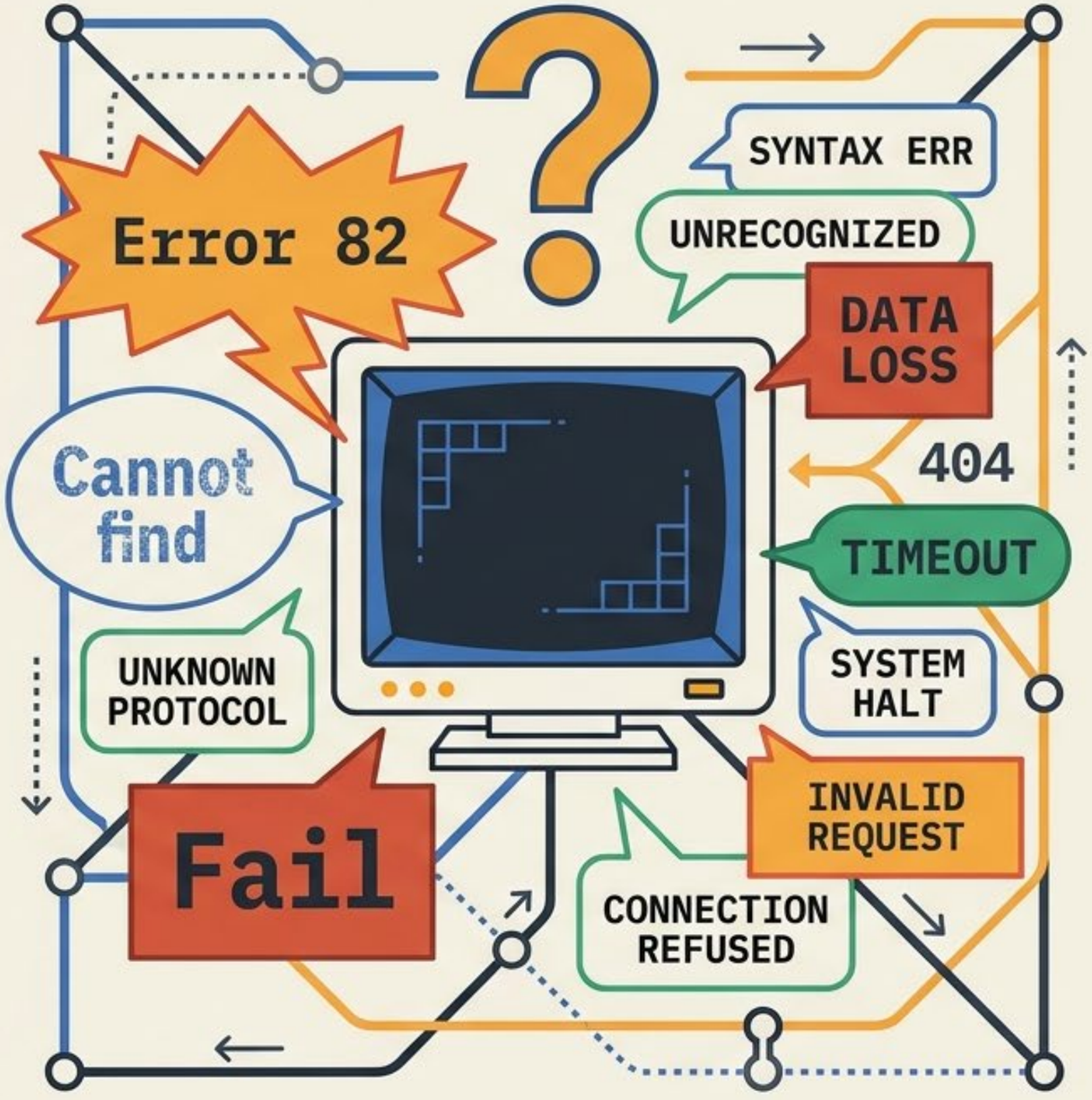
The Universal Language of the Web

Browsers, mobile apps, and servers need a way to talk to each other without reading large amounts of data. HTTP Status Codes are standardized messages that act as a quick summary of a request's outcome. They are the immediate answer from a server explaining exactly what happened.

The Pre-Standardization Tower of Babel

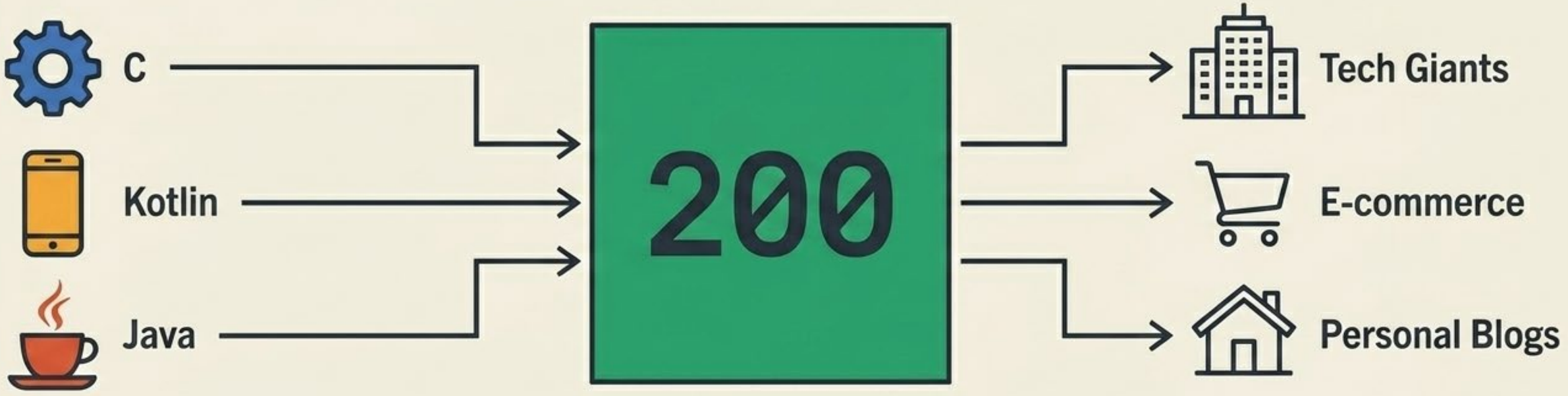
In the early days of computer networking, systems communicated using custom formats. Humans could read these messages, but computers could not reliably interpret them.

If every website used different words, a browser would have to guess whether a page was missing, if a user needed to log in, or if the server had crashed.



A Protocol Built on Universal Numbers

Instead of arbitrary text, servers return standardized numeric codes understood by any software.



Universal

"200" means exactly the same thing in English, Spanish, or Japanese.

Fast

Software evaluates "404" instantly without parsing complex text.

Consistent

A massive streaming platform and a tiny blog use the exact same logic.

The Web's Digital Traffic Signals

Green Light

The request succeeded perfectly.

Yellow Light

The client needs to take an additional action or detour.

Red Light

The request has a problem, usually caused by the client.

Emergency Signal

The server itself is broken or unavailable.

The Five Architectural Categories

The first digit dictates the response category.

1xx

Informational

Processing continues. The server received the request and is working on it.

2xx

Success

Request succeeded. The server completed the desired action.

3xx

Redirection

Additional action required. The resource moved moved or the browser should check its cache.

4xx

Client Error

Problem with the request. The client sent bad data, lacked permissions, or asked for something missing.

5xx

Server Error

Problem on the server. The client's request was valid, but the server failed to fulfill it.

2xx Success — The Green Lights

200 OK

Everything worked successfully. Used for standard actions like loading user data, fetching products, or viewing a page.

201 Created

A new resource was successfully born. Used when creating a new user account, placing an order, or publishing a blog post.

204 No Content

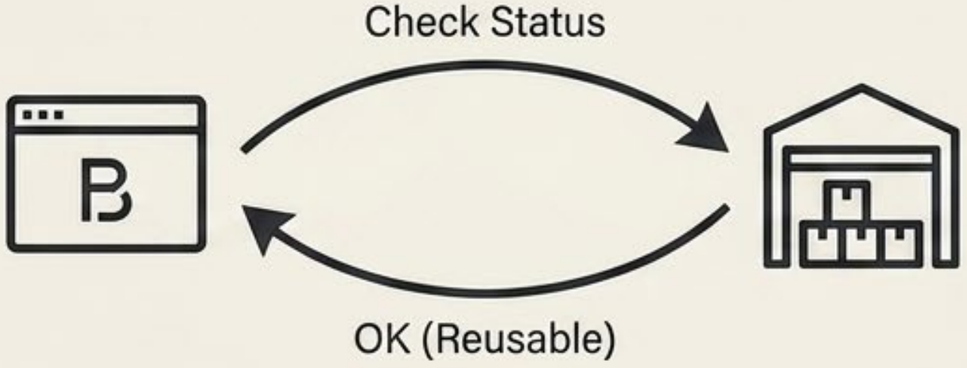
The action succeeded, but the server has no data to return. Most commonly used when successfully deleting a record.

3xx Redirection — Detours and Caching

301 Moved Permanently

302 Found

304 Not Modified



“Nothing changed.” A massive performance booster. The browser reuses its cached copy of CSS or images to save bandwidth and speed up page loads.

Identity vs Permission

The two most frequently confused client errors.

401 Unauthorized

“Who are you?” Authentication is required.

Occurs when a login token is missing or expired.

403 Forbidden

“I know who you are, but you can’t do this.”

The user is authenticated, but lacks permission to view restricted pages.

Missing vs Deleted Resources

404 Not Found

- The requested resource does not exist.
- The most common error on the web.
- Caused by a wrong URL, invalid ID, or searching for a record that was never there.
- “I can’t find it.”

410 Gone

- The resource existed in the past but has been permanently removed.
- Used deliberately for API deprecations or deleted pages.
- Tells clients to stop requesting the old resource.
- “It definitely used to exist, but it’s gone.”

Data Conflicts and System Limits

409 Conflict

The request clashes with existing server data. Common when trying to register a duplicate email or unique username.

422 Unprocessable Entity

The format is technically valid, but the data fails business validation rules.

429 Too Many Requests

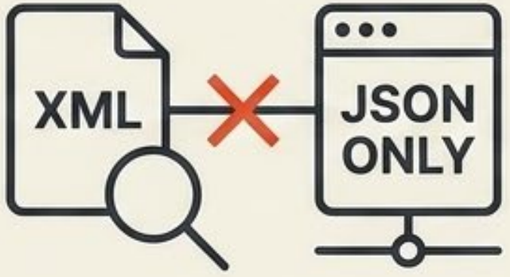
The client is sending too much data, too fast. Essential for API rate limits and spam protection.

Wrong Action, Wrong Format

405 Method Not Allowed

The destination exists, but the action is wrong (e.g., using DELETE on a read-only page).

406 Not Acceptable



The client asks for a specific format but the server only supports a different format.

415 Unsupported Media Type

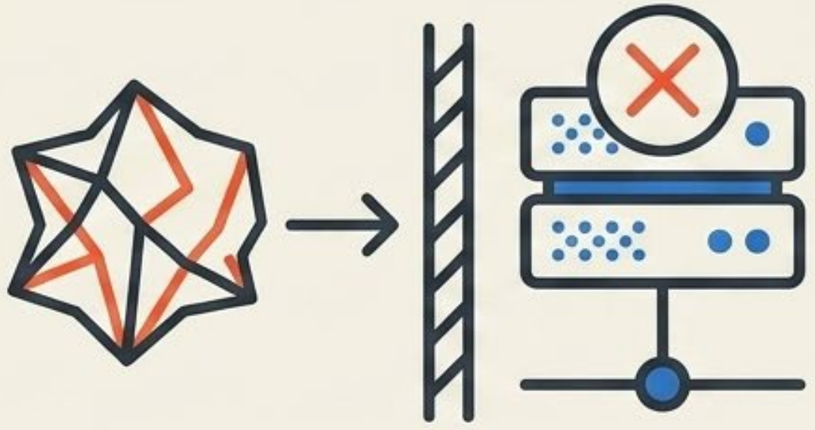
The client sends data in a format the server doesn't understand (unsupported file uploads).

413 Payload Too Large

Data exceeds server limits (e.g., a 500MB video sent to a 50MB endpoint).

Your Fault

400 Bad Request

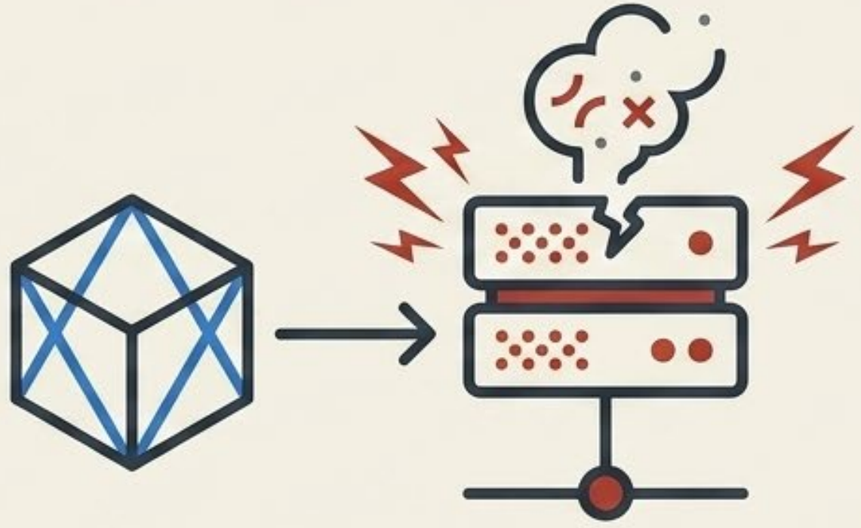


The general catch-all for invalid requests. The client sent missing fields, invalid input formats, or malformed JSON.

The server says: Fix your request.

My Fault

500 Internal Server Error

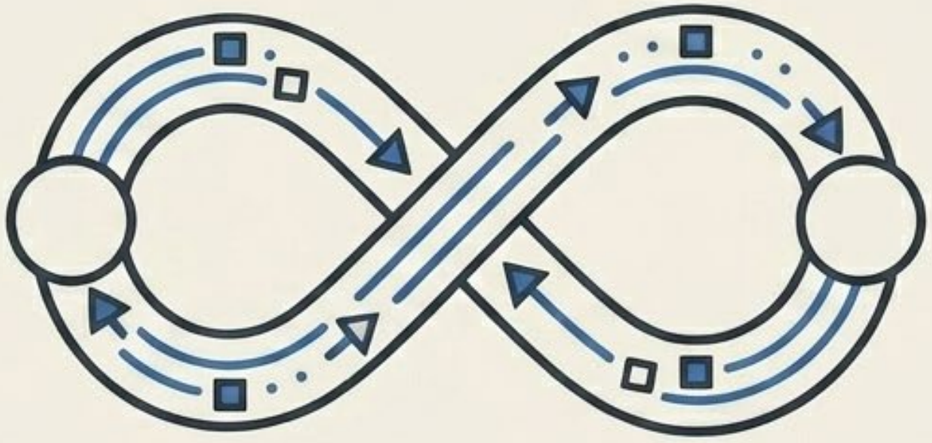


The universal emergency signal. The client did everything right, but a fatal problem occurred.

The server says:
Something went wrong on my side.

Asynchronous Processing & Persistent Connections

101 Switching Protocols



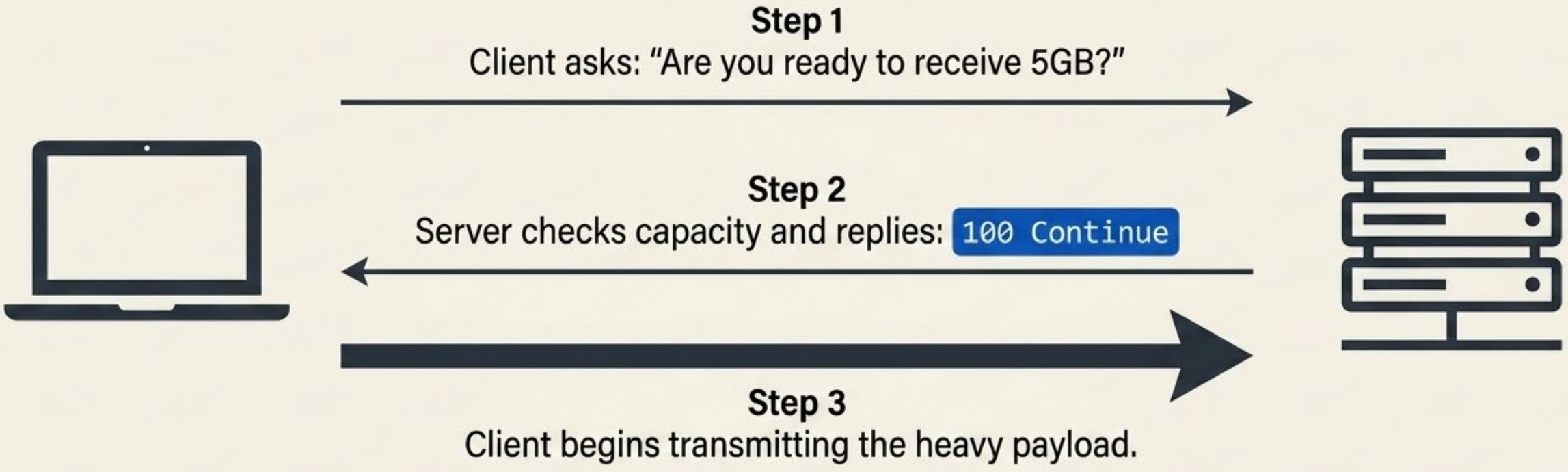
Moving from standard request-response HTTP to a continuous connection, like a WebSocket. Essential for real-time chat apps, multiplayer games, and live updates.

202 Accepted

“I got your request. I’ll process it later.” The server accepted the job but hasn’t finished it. Used for heavy background tasks like generating massive PDFs, rendering video, or training AI models.

The Bandwidth-Saving Handshake

Used for massive payloads to avoid wasting bandwidth.



Browsers and HTTP libraries handle this rare code automatically behind the scenes.

The Debugging Decision Tree

Don’t memorize every code. Look at the category first.

The network is fine.
The issue is in your local application logic.

Check your own work.

- Verify the URL (404)
- Check the access token (401)
- Validate the request payload (400)

Check the infrastructure.

- Investigate server logs
- Check database connections
- Review backend code