DEVELOPER REFERENCE · 2026

# 🔢 The HTTP Status Code Cheat Sheet

Every status code that actually matters — 1xx to 5xx — color-coded by category, with what it means and what your client should do about it.

By Veeresh Bashetti | veereshbashetti.com | Companion to the full guide on the blog

| 1xx Informational | 2xx Success | 3xx Redirection | 4xx Client Error | 5xx Server Error |
|---|---|---|---|---|

## 1xx · Informational

*The ones you never see — the server saying "keep going"*

| | | | |
|---|---|---|---|
| 100 | **Continue** | Server got the headers, keep sending the body — used before large uploads. | *None, automatic* |
| 101 | **Switching Protocols** | Confirms the switch to the protocol requested, e.g. a WebSocket handshake. | *None, automatic* |

## 2xx · Success

*It worked — but there's more nuance than just "200 good"*

| | | | |
|---|---|---|---|
| 200 | **OK** | Standard success for GET/PUT and most requests. Body contains the resource. | *Use the response* |
| 201 | **Created** | Successful creation, typically after a POST — a new user, order, or row. | *Use new resource's data* |
| 202 | **Accepted** | Accepted for processing, not finished yet — async jobs, queued emails. | *Poll or await callback* |
| 204 | **No Content** | Succeeded, nothing to send back — perfect for a successful DELETE. | *Treat as success* |

## 3xx · Redirection

*Matters more for SEO and API design than most developers realize*

| | | | |
|---|---|---|---|
| 301 | **Moved Permanently** | New home for good. Search engines update the index and pass ranking value. | *Update links to new URL* |
| 302 | **Found (Temporary)** | Not permanent. Search engines keep the original URL indexed, expecting a return. | *Follow, keep old URL* |
| 304 | **Not Modified** | Cached version is still valid — a quiet, effective bandwidth saver. | *Use cached copy* |
| 307/308 | **Temp/Permanent (strict)** | Like 302/301, but guarantee the method & body are preserved on redirect. | *Re-send same method/body* |

## 4xx · Client Error

*Where most day-to-day debugging happens, and where precision matters*

| | | | |
|---|---|---|---|
| 400 | **Bad Request** | Malformed or invalid request — bad JSON, missing field, wrong type. | *Fix request format* |
| 401 | **Unauthorized** | Server doesn't know who's asking. No/expired credentials. A login problem. | *Log in / refresh token* |
| 403 | **Forbidden** | Server knows exactly who's asking, and the answer is no. A permissions problem. | *Request proper access* |
| 404 | **Not Found** | The resource doesn't exist at this URL, full stop. | *Check URL / ID* |
| 405 | **Method Not Allowed** | URL exists, but this HTTP method isn't supported there. | *Use supported method* |
| 409 | **Conflict** | Conflicts with current resource state — e.g. a uniqueness violation. | *Resolve, then retry* |
| 422 | **Unprocessable Entity** | Syntactically valid, semantically wrong — e.g. an end date before start date. | *Fix data / business rule* |
| 429 | **Too Many Requests** | Rate limiting — sending too fast. Should be paired with Retry-After. | *Wait, then retry* |

## 5xx · Server Error

*The request was fine — the server is the one that failed*

| | | | |
|---|---|---|---|
| 500 | **Internal Server Error** | Deliberately vague catch-all. Real diagnosis always comes from server logs. | *Check logs, retry w/ backoff* |
| 502 | **Bad Gateway** | A proxy/gateway got an invalid response from an upstream server. | *Retry with backoff* |
| 503 | **Service Unavailable** | Server temporarily unable to handle requests — overload or maintenance. | *Retry / check status page* |
| 504 | **Gateway Timeout** | A gateway or proxy timed out waiting on an upstream server. | *Retry with backoff* |

## The Two Mix-Ups That Cause the Most Bugs

### 401 Unauthorized

"I don't recognize you — do you have an invitation?"

A **login** problem. Fully fixable by sending valid credentials or refreshing a token.

### 403 Forbidden

"I know exactly who you are — you're just not on the list tonight."

A **permissions** problem. No amount of re-logging in fixes it.

KNOW WHO'S ASKING VS. KNOW WHAT THEY'RE ALLOWED TO DO

### 301 Moved Permanently

"I sold the apartment — forward my mail to the new address for good."

Search engines update the index and transfer ranking value to the new URL.

### 302 Found

"I'm just subletting for now — keep my old address on file."

Search engines keep the original URL indexed, expecting things to revert.

PERMANENT MOVE VS. TEMPORARY DETOUR

## Which Errors Are Safe to Retry?

### ✓ Usually safe to retry (with backoff)

**5xx Server Errors** — often temporary: a restart, a database hiccup, a brief overload. Exponential backoff (1s, 2s, 4s, 8s...) gives the server room to recover.

**429 Too Many Requests** — designed to be retried, but only after the duration in its **Retry-After** header.

### ✗ Not safe to retry as-is

**Most 4xx Client Errors** — the request itself is the problem. Sending the exact same broken request again just produces the exact same failure. Fix the request first: the URL, the credentials, the payload, or the business rule it violated.

## Quick Debugging Rule

Check the **first digit** before anything else — it tells you who's at fault before you read another line of the response.
**4xx** means something about the request needs to change. **5xx** means the server failed to handle a request that was otherwise fine. That single rule is the fastest debugging shortcut in web development.