$ cd reference / HTTP headers

HTTP headers: caching, cookies, CORS, security

Header names are case-insensitive; HTTP/2 and HTTP/3 send them in lowercase. The meanings below follow RFC 9110 (semantics), RFC 9111 (caching), RFC 6265 (cookies) with its revision draft 6265bis for SameSite and the cookie prefixes, and the Fetch standard (CORS).

[01] Caching

HEADER	EXAMPLE	MEANING
Cache-Control	`max-age=3600`	the response is fresh for 3600 seconds; after that it must be revalidated or fetched again
Cache-Control	`no-cache`	may be stored, but must be revalidated with the origin before every use
Cache-Control	`no-store`	must not be stored by any cache at all
Cache-Control	`private`	only the browser's own cache may store it, not a shared cache
Cache-Control	`public`	any cache may store it, even when it would otherwise not be cacheable, such as a response to a request with credentials
Cache-Control	`s-maxage=600`	overrides max-age for shared caches only
Cache-Control	`must-revalidate`	once stale, the response may not be used without a successful revalidation
Cache-Control	`immutable`	the content will not change while fresh, so browsers that support it skip revalidation even on reload
Cache-Control	`stale-while-revalidate=60`	a stale copy may be served for 60 seconds while a fresh one is fetched in the background (RFC 5861)
Expires	`Wed, 07 Oct 2026 08:00:00 GMT`	older absolute expiry; ignored when max-age is present
ETag	`"v42"` or `W/"v42"`	a version identifier; the W/ prefix marks a weak validator
Last-Modified	`Wed, 07 Oct 2026 08:00:00 GMT`	a date validator, coarser than an ETag (one-second resolution)
If-None-Match	`"v42"`	request: send the body only if the ETag differs; otherwise the answer is 304 Not Modified

HEADER	EXAMPLE	MEANING
If-Modified-Since	`Wed, 07 Oct 2026 08:00:00 GMT`	request: the date-based version of the same check; ignored when If-None-Match is present
Vary	`Accept-Encoding`	the response depends on these request headers, so caches must store one copy per value
Age	`120`	set by caches: the estimated seconds since the response was generated or revalidated at the origin

`no-cache` does not mean "do not cache"; `no-store` does. For files with a content hash in their name, `Cache-Control: max-age=31536000, immutable` is the usual choice. For HTML pages, `no-cache` together with an ETag gives fast 304 responses without serving stale pages.

[02] Content and negotiation

HEADER	EXAMPLE	MEANING
Content-Type	`text/html; charset=utf-8`	media type of the body; for text, include the charset
Content-Length	`3495`	body size in bytes
Content-Encoding	`br`, `gzip`, `zstd`	the body is compressed; the client decompresses it transparently
Transfer-Encoding	`chunked`	HTTP/1.1 only: the body arrives in chunks of announced size; not used in HTTP/2 or HTTP/3
Content-Disposition	`attachment; filename="report.pdf"`	ask the browser to download rather than display; `filename*=UTF-8''r%C3%A9sum%C3%A9.pdf` for non-ASCII names
Content-Language	`en`	language of the content
Accept	`application/json, text/html;q=0.8`	request: media types the client prefers; q weights from 0 to 1, default 1
Accept-Encoding	`gzip, br, zstd`	request: compression formats the client can decode
Accept-Language	`en-GB, en;q=0.9, de;q=0.5`	request: preferred languages

`X-Content-Type-Options: nosniff` stops browsers from guessing a different type than `Content-Type` says. Without it, a text upload served as `text/plain` can be treated as a script or a stylesheet in some contexts.

[03] Ranges and resumed downloads

HEADER	EXAMPLE	MEANING
Accept-Ranges	`bytes`	response: byte ranges are supported
Range	`bytes=0-499`, `bytes=500-`, `bytes=-200`	request: first 500 bytes, everything from byte 500, the last 200 bytes
Content-Range	`bytes 0-499/8000`	response to a range request, sent with status 206
If-Range	`"v42"`	request: send the range only if the resource is unchanged, otherwise the whole body with 200

A range that lies entirely outside the resource gets 416 Range Not Satisfiable, with `Content-Range: bytes */8000` giving the actual length.

TASK	COMMAND
Fetch the first 100 bytes	`curl -s -r 0-99 -o part.bin "$URL"`
Resume an interrupted download	`curl -C - -O "$URL"`

[04] Conditional writes

HEADER	EXAMPLE	MEANING
If-Match	`"v42"`	perform the PUT or DELETE only if the current ETag is still "v42"; otherwise 412 Precondition Failed
If-Unmodified-Since	`Wed, 07 Oct 2026 08:00:00 GMT`	the date-based version of If-Match
If-None-Match	`*`	on a PUT: create only if nothing exists yet

Sending the ETag you read back in `If-Match` turns a read-modify-write cycle into optimistic locking: if someone else saved in between, your write fails with 412 instead of silently overwriting theirs.

[05] Authentication and redirects

HEADER	EXAMPLE	MEANING
Authorization	`Bearer eyJhbGciOi...`	request: credentials; Basic, Bearer and other schemes

HEADER	EXAMPLE	MEANING
WWW-Authenticate	`Bearer realm="api", error="invalid_token"`	required in a 401 response: which scheme to use and why it failed
Location	`/reports/42`	the target of a 3xx redirect, or the new resource after 201 Created; may be relative
Retry-After	`120` or an HTTP date	with 429 or 503: when to try again
Allow	`GET, HEAD, OPTIONS`	required in a 405 response: the methods that do work

[06] Cookies

ATTRIBUTE	EFFECT
Max-Age=3600	lifetime in seconds; wins over Expires; without either, the cookie lasts until the browser session ends
Expires=Wed, 07 Oct 2026 08:00:00 GMT	absolute expiry
Path=/	sent only for request paths under this prefix
Domain=example.com	also sent to subdomains; without Domain, only to the exact host that set it
Secure	sent only over https
HttpOnly	not readable from JavaScript through document.cookie
SameSite=Lax	not sent on cross-site subrequests, but sent on top-level navigations with GET; Chrome treats a cookie without SameSite as Lax
SameSite=Strict	never sent with cross-site requests, not even when following a link
SameSite=None	sent with cross-site requests; requires Secure

A cookie whose name starts with `__Host-` is accepted only with `Secure`, `Path=/` and no `Domain`, so it cannot be set or overwritten for the host by a subdomain. `__Secure-` only requires `Secure`.

[07] CORS

HEADER	EXAMPLE	MEANING
Origin	`https://example.com`	request: scheme, host and port of the page making a cross-origin request
Access-Control-Allow-Origin	`https://example.com` or `*`	which origin may read the response

HEADER	EXAMPLE	MEANING
Access-Control-Allow-Credentials	`true`	the response may be read even though the request carried cookies or HTTP authentication; not allowed together with *
Access-Control-Allow-Methods	`GET, POST, PUT`	preflight response: methods allowed for the actual request
Access-Control-Allow-Headers	`Content-Type, Authorization`	preflight response: request headers allowed
Access-Control-Expose-Headers	`ETag, Location`	response headers that scripts may read beyond the default safe list
Access-Control-Max-Age	`600`	seconds a preflight result may be cached; browsers cap it (Chrome at two hours)

A preflight is an `OPTIONS` request the browser sends before a cross-origin request that uses a method other than GET, HEAD or POST, a custom header, or a `Content-Type` other than the two form encodings and `text/plain`. JSON requests sent with `Content-Type: application/json` therefore always trigger one. When the allowed origin depends on the request, the response must also carry `Vary: Origin`, or a cache may hand one origin's answer to another.

CORS is enforced by browsers only. It does not protect an API from scripts, `curl` or other non-browser clients.

[08] Security headers

HEADER	EXAMPLE	EFFECT
Content-Security-Policy	`default-src 'self'; frame-ancestors 'none'`	which sources scripts, styles, images, connections and frames may come from
Strict-Transport-Security	`max-age=63072000; includeSubDomains`	the browser uses only https for this host for the given time; ignored when received over plain http
X-Content-Type-Options	`nosniff`	no content-type guessing
X-Frame-Options	`DENY`	older way to forbid framing; CSP frame-ancestors replaces it and takes precedence where supported
Referrer-Policy	`strict-origin-when-cross-origin`	how much of the current URL is sent as Referer; this value is also the default in current browsers
Permissions-Policy	`camera=(), geolocation=()`	turns off browser features for the page and its frames
Cross-Origin-Opener-Policy	`same-origin`	puts the page in its own browsing context group, cut off from cross-origin popups

[09] Looking at headers

TASK	COMMAND
Response headers only (HEAD request)	`curl -sI "$URL"`
Request and response headers of a GET	`curl -sv -o /dev/null "$URL" 2>&1 | grep -E "^[<>]"`
Send a conditional request	`curl -sI -H 'If-None-Match: "v42"' "$URL"`
Check what a preflight returns	`curl -si -X OPTIONS -H "Origin: https://example.com" -H "Access-Control-Request-Method: PUT" "$URL"`

[10] Sources

[1] RFC 9110: HTTP semantics

[2] RFC 9111: HTTP caching

[3] RFC 5861: stale-while-revalidate

[4] RFC 8246: immutable

[5] RFC 6265: cookies

[6] draft-ietf-httpbis-rfc6265bis: SameSite and cookie prefixes (Internet-Draft)

[7] Fetch standard: the CORS protocol

[8] RFC 6797: Strict-Transport-Security

[9] Content Security Policy Level 3