Carson
a coder 👨‍💻
Follow

HTTP Caching — How does it work step by step?

Carson Mar 4 · 8 min read

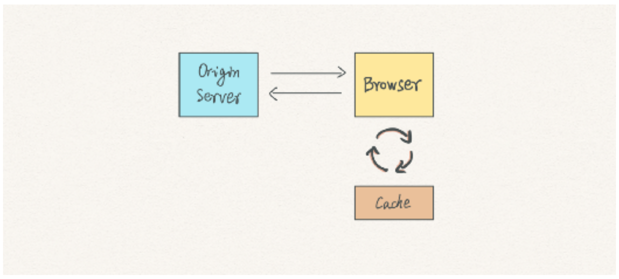
An HTTP request is expensive.

Whenever possible, we reach out to cache, which keeps (almost) the latest resource.

Cache lives at every node of the network, including

- clients (in our context, browsers),
- proxy servers, and
- origin servers

Two-players: browser and origin server



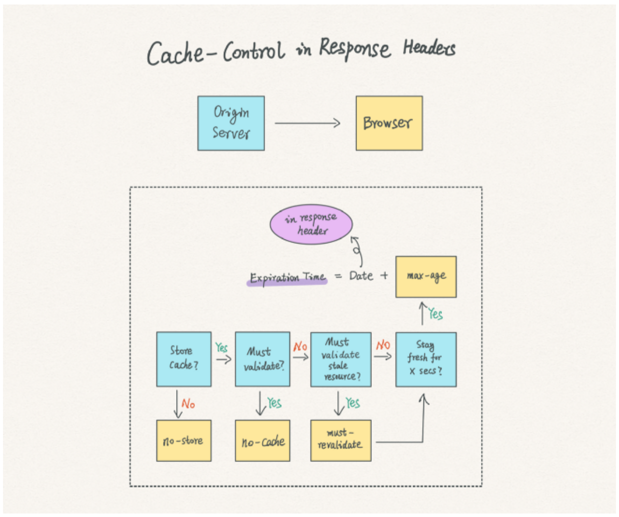
Cache-Control from the server

If you have saved an edit in a Medium post, you want other visitors to see the updated one.

Which player knows whether the content is the latest one?

A browser doesn't know without sending a request. The server holding resources is the one knowing it.

With the `Cache-Control` header, the server informs the browser when to use cache or request new resources.



Many values are available for the header, but four of them are most common:

- `no-store` prohibits the browser store any cache. It is useful for continually changing sources.
- `no-cache` has a misleading name. It actually allows cache, but validation with the origin server is mandatory before using it.

Carson

a coder

Follow

2

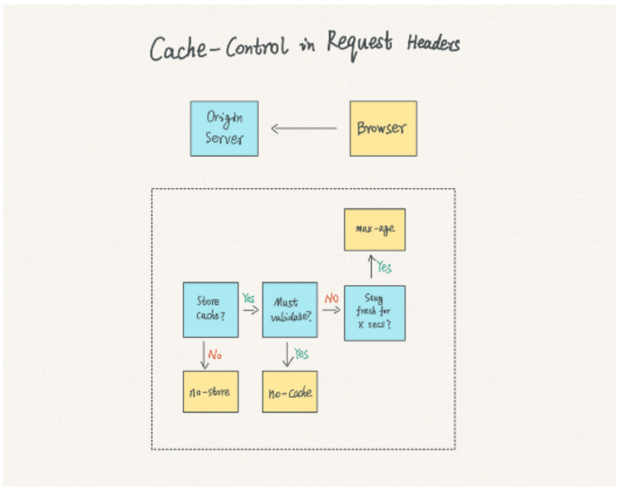
- `max-age` helps our browser decide if a resource is considered fresh.
- `must-revalidate` instructs the browser to validate a resource on the origin server when it is stale. It usually uses with `max-age` .

Unlike `Expires` in a cookie, `max-age` is relative to the time of the response generated on the origin server, which is stated in the `Date` response header.

Let’s say a server sets the `max-age` to 10 seconds, and it takes 1 second to reach the browser. The remaining 9 seconds are the resource's "shelf life."

What if we set the `max-age` to 0? With it, we guarantee that the resource becomes stale as soon as it reaches the browser.

Cache-Control from the browser



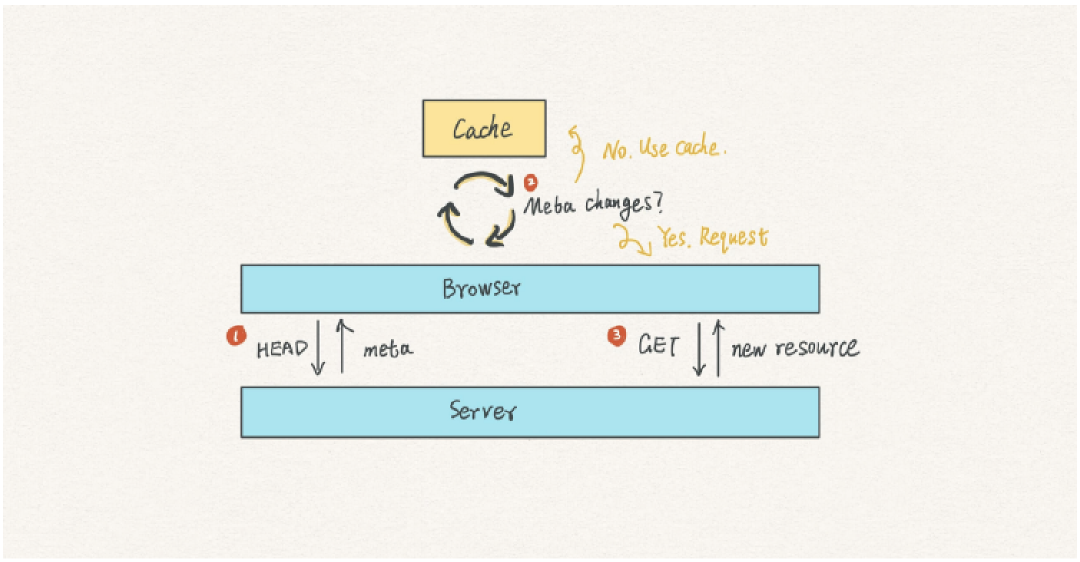
HTTP is about the negotiation between two players. A browser also has the flexibility to determine what it wants.

For example, by setting `Cache-Control: no-cache` in its request header, a browser reaches out to the server instead of using cache.

This is also what happens when you hard-reload a webpage. The browser adds the header to its request.

To use a cache from your disk, click the back or the forward button on your browser. In this way, the browser initializes a request without a `Cache-Control` header and receives a cached resource. Therefore, the process is super fast.

What if a browser wants to validate a resource?



An option is initializing a validation call with a `HEAD` method. The server responds with meta information.

Next, the browser can do the math and see if its cache is still fresh. If not, it requests the latest resource in another call.

Carson
a coder 👾
Follow

2

Two calls sound a lot of work, and it is. Can we save a trip?

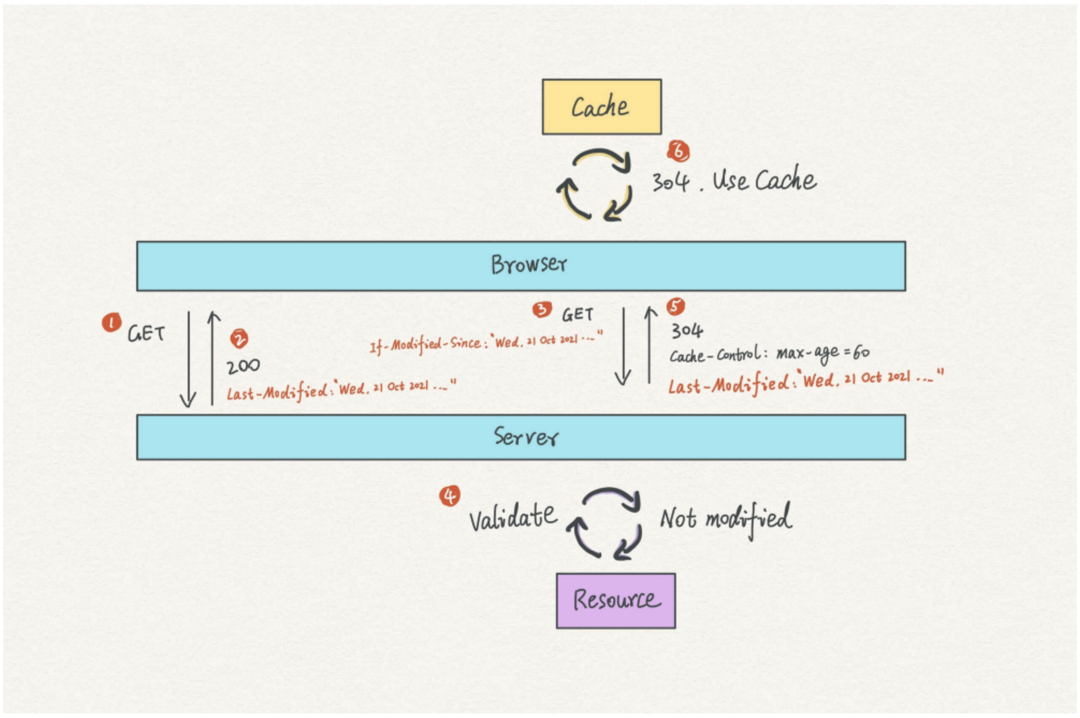
Conditional requests from the browser

HTTP introduces a group of headers to facilitate the validation and request the new resources in one call.

These are conditional request headers. All are prefixed with `If-`.

Among them, the most popular two are `If-Modified-Since` and `If-None-Match`.

If-Modified-Since


Cache
6 304 . Use Cache
Browser
1 GET
2 200
Last-Modified: "Wed, 21 Oct 2021 ..."
3 GET
If-Modified-Since: "Wed, 21 Oct 2021 ..."
5 304
Cache-Control: max-age=60
Last-Modified: "Wed, 21 Oct 2021 ..."
Server
4 Validate
Not modified
Resource

From its name, you can tell it is about expiration time.

The completed steps with `If-Modified-Since` are the following:

1. The browser requests a resource for the first time.
2. The server response includes a `Last-Modified` header.
3. When the browser requests the resource again, it attaches the `If-Modified-Since` header automatically.
4. The server validates the time and notices no modifications.
5. Since there is no modification, the server responds with a 304 status code and other headers, telling the browser to use cache.
6. When seeing 304, the browser knows it is safe to use cache.

However, `If-Modified-Since` has its restrictions — the shortest time unit is seconds.

If changes happen multiple times within a second, a server won't respond with the latest resource. In this case, the browser uses its cache when it should not.

Another restriction is related to the modification check.

Some changes don't modify the actual content but update the last-modified time. In this case, the browser will send a request when it should read from the cache.

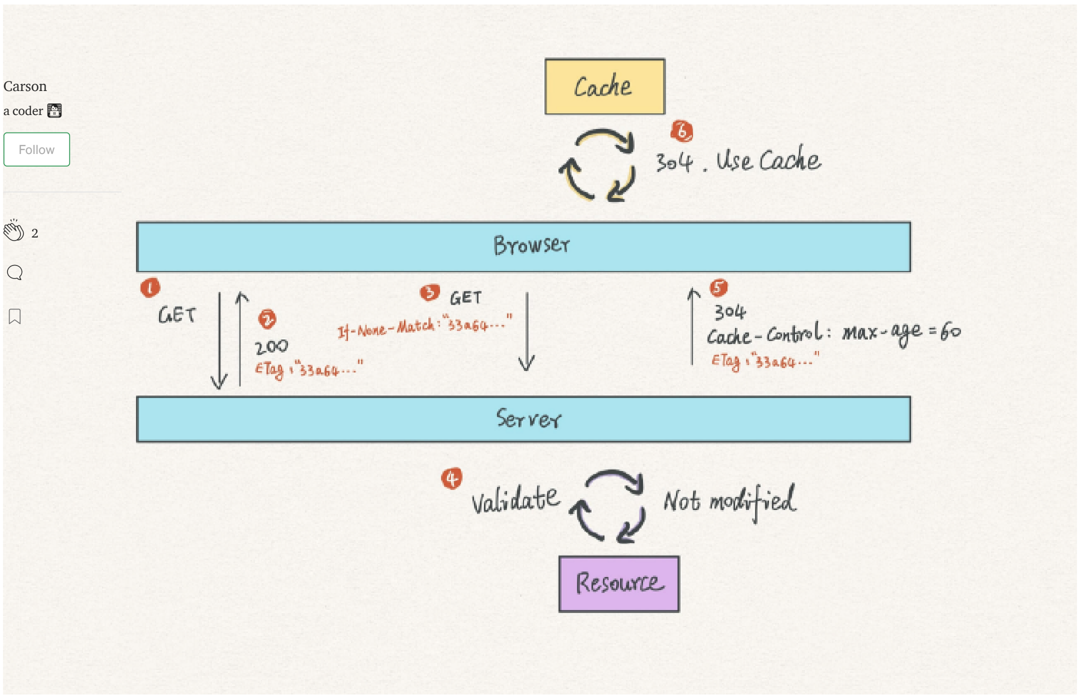
To take care of both cases, we need `ETag` and `If-None-Match`.

If-None-Match

Carson
a coder

Follow

2



Usually, a resource is a large string, and comparing two are expensive.

To speed up the comparison, we hash the resource to a relatively short one. Then, we set the shorten representation in a response header called `ETag` .

The steps are the following:

1. The browser sends the first request.
2. The server responds with the resources. An `ETag` header is added to the response.
3. When the browser requests the same resource, an `If-None-Match` header is attached, along with `ETag` 's value received in the previous response.
4. The server validates the `ETag` and detects if the resource has been modified.
5. If the resource remains the same, the server sends a 304 status code response with other headers.
6. The browser knows that it is safe to use its cache.

There are two types of `ETag` for your choice: strong and weak.

The strong `ETag` taking more server resources to generate guarantees two representations of the same resources are byte-for-byte identical, including the body and the headers.

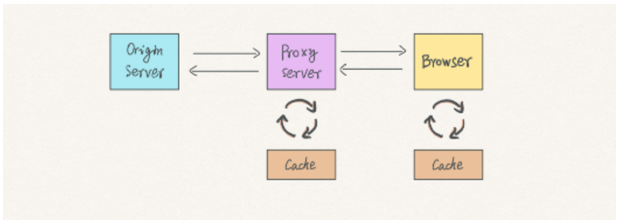
By contrast, the weak `ETag` is easier to create but can only ensure the body is semantically equivalent.

Here are examples of a strong and a weak one.

```
# Strong ETag
"33a64c34d8d3832b4ac257297d23d99"

# Weak ETag
W/"33a64c34d8d3832b4ac257297d23d99"
```

Three-players: Browser, proxy server, and origin server



Ready for something complex?

Carson

a coder

Follow

2

This time, a proxy server join the party, and we have three players in the cache chain.

What is the role a proxy server plays here?

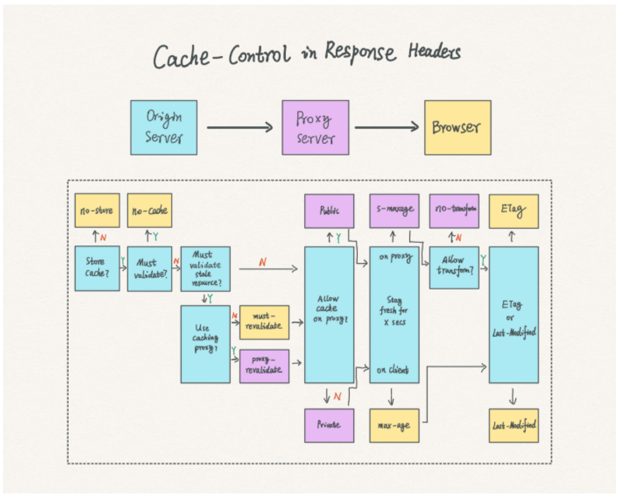
It is a “client” because it sends a request to the origin server.

It is a “server” because it sends a response to clients.

Therefore, all cache-control options we mentioned are available on it.

Other than the shared headers, it has its unique ones as a middleman.

Cache-Control from the origin server



More `Cache-Control` values are introduced to instruct how a proxy server responds with its cache.

A proxy could respond to multiple browsers. A `Cache-Control` with the `public` means the resource can be used by everyone. By contrast, `private` means the resource, well, is private to the specific client. A client can store the resource, but a proxy should not.

For example, the Medium site’s CSS file is public. Your medium login token in a `Set-Cookie` header should be private. The proxy should not send your token to other browsers.

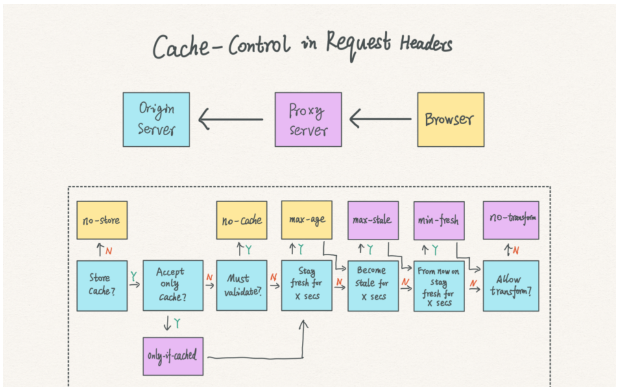
`proxy-revalidate` asks the proxy to validate its cache when it expires. It applies to public resources only.

A public resource on a proxy can have a fresh time with `s-maxage`, a header designed for the proxy only. Clients still follow the instruction in the `max-age`.

Last but not least, the `no-transform` prohibits the middleman modifies the response body.

Sometimes, a proxy edits the response body. For example, it could optimize an image resource by offering a light-weight alternative like the `webp` format.

Cache-Control from the browser



Carson

a coder

Follow

2

The browser also introduces new tools to negotiate the cache-control with a proxy. Besides the `no-transform`, we have three new options.

With the `only-if-cached`, the browser announces that it only accepts the cache from a proxy, not the origin server.

In this case, when the proxy's cache becomes stale, the browser could receive a 504 status code, indicating the timeout from the origin server.

`max-stale` extends the cache lifetime. The fresh time equals the sum of `max-age` and `max-stale` values.

For example,

- max-age=10
- max-stale=10

The cache is still considered fresh at the 20th second.

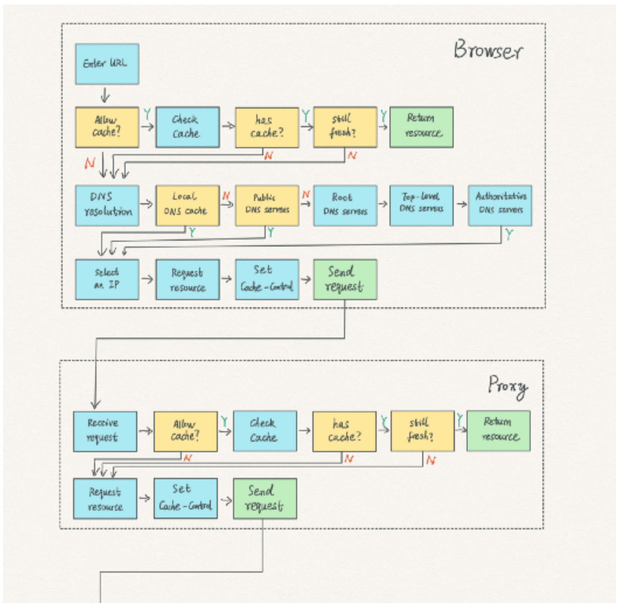
By contrast, `min-fresh` reduces the cache lifetime. The fresh time equals the difference between `max-age` and `min-fresh` values.

For example,

- max-age=10
- min-fresh=5

The cache becomes stale after 5 seconds.

Let's put everything together



When you enter an URL, the browser checks if the resource is allowed to be cached. The cached resource will be returned if it is permitted, existed, and still fresh. Otherwise, the browser needs to request an update.

The request starts from the DNS resolution. The browser checks the local DNS cache, then public DNS servers. If no IP is found, it starts from a root DNS server to a top-level DNS server and then to an authoritative DNS server. Finally, a series of IP addresses are returned to the browser.

If you are curious about how the DNS resolution works, check this post.

Among all IPs, the browser picks an appropriate one based on various criteria.

Next, it creates a request with the `Cache-Control` header and sends the request.

A proxy receives the request. If the cache is conceded, existed and fresh, it returns the cached resource to the browser. Otherwise, it needs to request an update from the origin server.

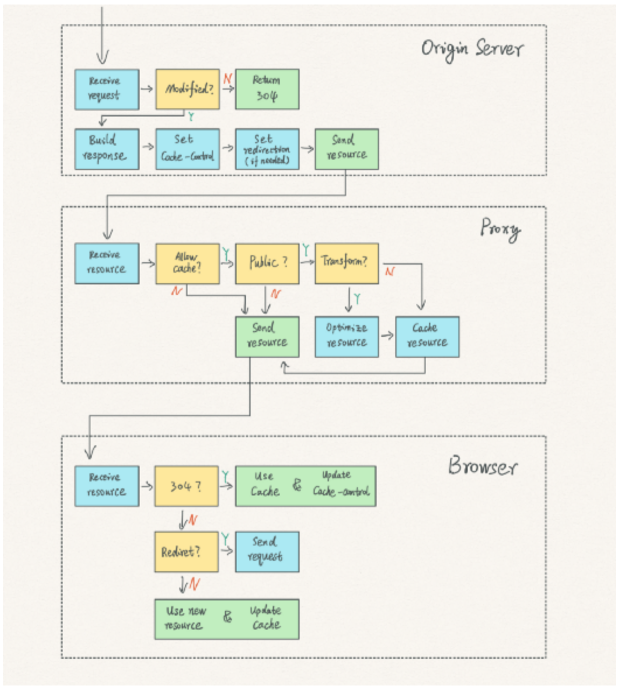
Carson

a coder 👨‍💻

Follow

👏 2

Same as the browser, the proxy creates a request with the appropriate `Cache-Control` header and sends it.



The origin server receives the request and checks if the source has been modified.

If not, a 304 status code will be returned. If the source is edited, it creates a response with a `Cache-Control` header and other headers like the `ETag` or `Last-Modified`.

If the redirection is required, 301 or 302 status codes will be included.

Eventually, the origin server sends the response.

The proxy receives the response and cache it if the cache is allowed and public. Otherwise, the resource will be sent to the browser directly.

Before caching, the proxy may optimize the resource if it is permitted.

Next, the browser receives the response.

Suppose it is a 304 status code. The browser uses its cache and update the cache-related variable correspondingly, such as the `max-age`.

If the redirection is indicated in the response, the browser will initialize a new request to the new location.

Lastly, if the resource is a new one, the browser will render it and update the cache.

Further reading

To learn more about HTTP Cache-Control header, take a look at https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Cache-Control

The `If-` headers include:

- If-Match
- If-Modified-Since
- If-None-Match
- If-Range
- If-Unmodified-Since

Use cases of HTTP conditional requests: https://developer.mozilla.org/en-US/docs/Web/HTTP/Conditional_requests#use_cases