HTTP Caching

▼ **What is HTTP Caching?**

The HTTP cache stores a response associated with a request and reuses the stored response for subsequent requests.

There are several advantages to reusability. First, since there is no need to deliver the request to the origin server, then the closer the client and cache are, the faster the response will be. The most typical example is when the browser itself stores a cache for browser requests.

Also, when a response is reusable, the origin server does not need to process the request — so it does not need to parse and route the request, restore the session based on the cookie, query the DB for results, or render the template engine. That reduces the load on the server.

Source: MDN

▼ **Purpose**

- **Speed Improvement:** Reduces load times by serving cached resources quickly.
- **Reduced Traffic:** Reduces the amount of data transmitted over the network.
- **Reduced Server Load:** Lessens the burden on origin servers by serving cached content.
- **Improved Availability:** Increases resilience against network issues by providing local copies of resources.
- **Faster Load Times:** Cached resources are served more quickly than fetching them from the origin server.
- **Lower Latency:** Reduces the time taken for requests to travel across the network.

▼ Resources

1. Static assets like CSS, JavaScript files, images, fonts, and HTML pages.
2. API responses

▼ HTTP Headers

▼ `Cache-Control`

The `Cache-Control` HTTP header field holds *directives* (instructions) — in both requests and responses — that control caching in browsers and shared caches

The directives are as follows:

1. **public**: the 'public' directive means any cache can store the resource.
2. **private:** A response with a 'private' directive can only be cached by the client and never by an intermediary agent, such as a CDN or a proxy. These are often resources containing private data, such as a website displaying a user's personal information.
3. **max-age=seconds**: Specifies the maximum amount of time a resource is considered valid. After this period, the resource must be fetched again.
4. **no-store:** A response with a 'no-store' directive cannot be cached anywhere, ever. This means, that every time a user requests this data, a request must be sent to the origin server for a fresh copy.
5. **no-cache:** Forces caches to submit the request to the origin server for validation before releasing a cached copy. This revalidation is typically done using ETag which contains a token unique to the version of the resource at the time it was requested. This token is changed on the origin server whenever the resource is updated. The client connects to the origin server to verify the ETag, and if the ETags are identical, then the cached resource is utilised, otherwise, that means the resource is updated and hence new data is fetched
6. **must-revalidate**: Indicates that the cache must verify the status of the stale resources before using it and expired resources should not be used

```
// Example usage
res.setHeader('Cache-Control', 'public, max-age=86400');
```

▼ `Expires`

The `Expires` HTTP header contains the date/time after which the response is considered expired.

```
// Example usage
res.setHeader('Expires', 'Sat, 23 Dec 2023 11:20:39 GMT')
```

▼ `Last-Modified`

The `Last-Modified` response HTTP header contains a date and time when the origin server believes the resource was last modified. It is used as a validator to determine if the resource is the same as the previously stored one.

```
// Example usage
res.setHeader('Last-Modified', 'Sat, 23 Dec 2023 11:20:39
```

▼ Etag

ETag headers identify whether the version of a resource cached in the browser is the same as the resource at the web server. A visitor's browser stores ETags. When a visitor revisits a site, the browser compares each ETag to the one it stored. Matching values cause a `304 Not-Modified` `HTTP` response that indicates the cached resource version is current.

```
// Example usage
res.setHeader('ETag', 'dj3958ehcxvj69237dh59')
```

▼ How to avoid resources being fetched from cache?

Cache Busting: Change the resource URL whenever the content changes, ensuring a fresh fetch (e.g., `image.gif?hash=abcdef`).

`no-store`: Prevents caching of the resource entirely.

`max-age=0`: Indicates that the resource is stale immediately.