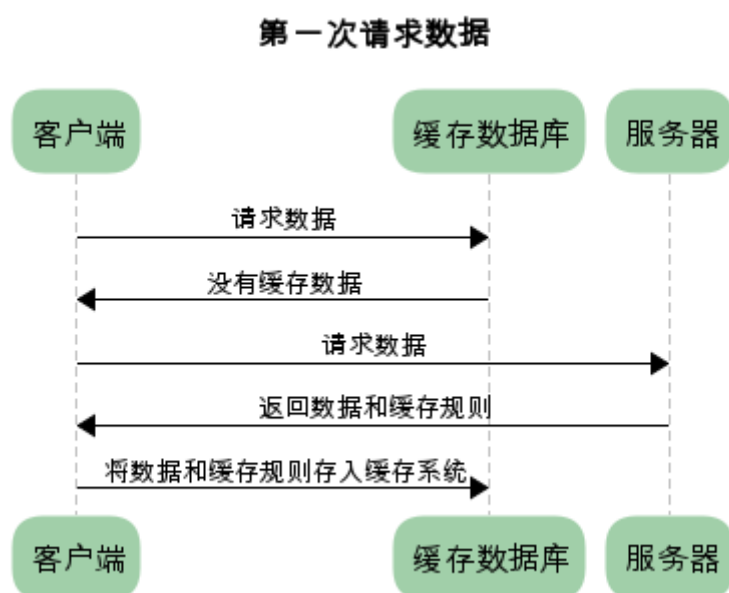


包含属性的首部(**header**)-----附加信息(**cookie**, 缓存信息等)与缓存相关的规则信息, 均包含在 **header** 中

缓存规则解析

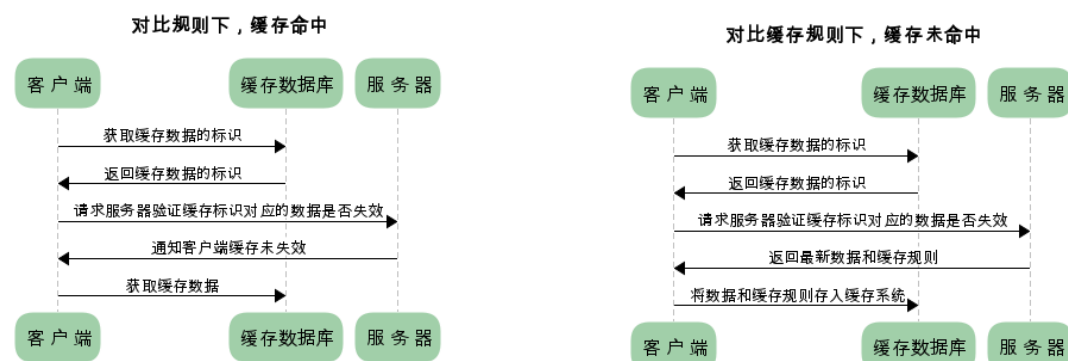
为方便大家理解, 我们认为浏览器存在一个缓存数据库, 用于存储缓存信息。在客户端第一次请求数据时, 此时缓存数据库中并没有对应的缓存数据, 需要请求服务器, 服务器返回后, 将数据存储至缓存数据库中。



HTTP 缓存有多种规则, 根据是否需要重新向服务器发起请求来分类, 我将其分为两大类(强制缓存, 对比缓存)

在详细介绍这两种规则之前, 先通过时序图的方式, 让大家对这两种规则有个简单了解。

已存在缓存数据时, 仅基于强制缓存, 请求数据的流程如下
已存在缓存数据时, 仅基于对比缓存, 请求数据的流程如下



强制缓存如果生效，不需要再和服务器发生交互，而对比缓存不管是否生效，都需要与服务端发生交互。两类缓存规则可以同时存在，**强制缓存优先级高于对比缓存**，也就是说，当执行强制缓存的规则时，如果缓存生效，直接使用缓存，不再执行对比缓存规则。

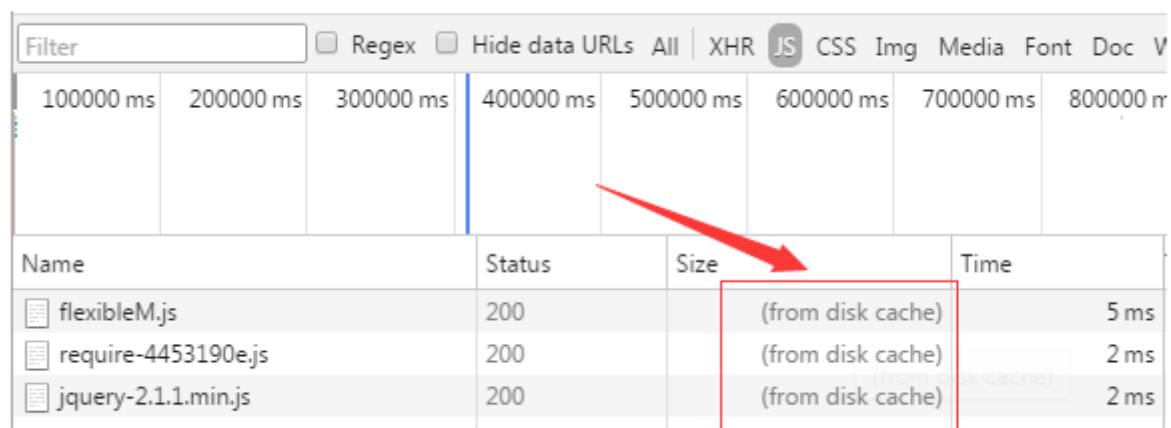
强制缓存

强制缓存，在缓存数据未失效的情况下，可以直接使用缓存数据，那么浏览器是如何判断缓存数据是否失效呢？

我们知道，在没有缓存数据的时候，浏览器向服务器请求数据时，服务器会将数据和缓存规则一并返回，缓存规则信息包含在响应 **header** 中。

对于强制缓存来说，响应 **header** 中会有两个字段来标明失效规则

（Expires/Cache-Control）使用 **chrome** 的开发者工具，可以很明显的看到对于强制缓存生效时，网络请求的情况



The screenshot shows the Chrome DevTools Network tab with the 'JS' filter selected. A table lists network requests. A red arrow points to the 'Size' column, which contains the text '(from disk cache)' for three files: 'flexibleM.js', 'require-4453190e.js', and 'jquery-2.1.1.min.js'. A red box highlights these three rows in the 'Size' column.

Name	Status	Size	Time
flexibleM.js	200	(from disk cache)	5 ms
require-4453190e.js	200	(from disk cache)	2 ms
jquery-2.1.1.min.js	200	(from disk cache)	2 ms

Expires

Expires 的值为服务端返回的到期时间，即下一次请求时，请求时间小于服务端返回的到期时间，直接使用缓存数据。

不过 Expires 是 HTTP 1.0 的东西，现在默认浏览器均默认使用 HTTP 1.1，所以它的作用基本忽略。

另一个问题是，到期时间是由服务端生成的，但是客户端时间可能跟服务端时间有误差，这就会导致缓存命中的误差。

所以 HTTP 1.1 的版本，使用 Cache-Control 替代。

Cache-Control

Cache-Control 是最重要的规则。常见的取值有 private、public、no-cache、max-age，no-store，默认为 private。

private: 客户端可以缓存

public: 客户端和代理服务器都可缓存（前端的同学，可以认为

public 和 private 是一样的)

max-age=xxx: 缓存的内容将在 xxx 秒后失效

no-cache: 需要使用对比缓存来验证缓存数据 (后面介绍)

no-store: 所有内容都不会缓存, 强制缓存, 对比缓存都不会触发
(对于前端开发来说, 缓存越多越好, so...基本上和它说 886)

举个板栗

```
▼ Response Headers    view source
Cache-Control: max-age=31536000
Connection: keep-alive
Content-Encoding: gzip
Content-Type: application/javascript
Date: Tue, 24 Jan 2017 02:21:12 GMT
ETag: W/"58847adf-110d2d"
Last-Modified: Sun, 22 Jan 2017 09:26:55 GMT
```

图中 Cache-Control 仅指定了 max-age, 所以默认为 private, 缓存时间为 31536000 秒 (365 天)




也就是说, 在 365 天内再次请求这条数据, 都会直接获取缓存数据库中的数据, 直接使用。

对比缓存




对比缓存, 顾名思义, 需要进行比较判断是否可以使用缓存。浏览器第一次请求数据时, 服务器会将缓存标识与数据一起返回给客户端, 客户端将二者备份至缓存数据库中。

再次请求数据时, 客户端将备份的缓存标识发送给服务器, 服务器根据缓存标识进行判断, 判断成功后, 返回 304 状态码, 通知客户端比较成功, 可以使用缓存数据。

第一次访问:

Name	Status	Size	Time
 flexible-c207ebf8.js	200	1.3 KB	103 ms
 base-5c8a15c8.js	200	228 KB	429 ms
 index-6c1a969b.js	200	35.2 KB	151 ms

再次访问:

Name	Status	Size	Time
 flexible-c207ebf8.js	304	206 B	28 ms
 base-5c8a15c8.js	304	209 B	66 ms
 index-6c1a969b.js	304	208 B	37 ms

通过两图的对比, 我们可以很清楚的发现, 在对比缓存生效时, 状态码为 304, 并且报文大小和请求时间大大减少。

原因是，服务端在进行标识比较后，只返回 **header** 部分，通过状态码通知客户端使用缓存，不再需要将报文主体部分返回给客户端。

对于对比缓存来说，缓存标识的传递是我们着重需要理解的，它在请求 **header** 和响应 **header** 间进行传递，
一共分为两种标识传递，接下来，我们分开介绍。

Last-Modified / If-Modified-Since

Last-Modified:

服务器在响应请求时，告诉浏览器资源的最后修改时间。

```
▼ Response Headers    view source
Cache-Control: max-age=31536000
Connection: keep-alive
Content-Encoding: gzip
Content-Type: application/javascript
Date: Tue, 24 Jan 2017 07:26:54 GMT
ETag: W/"5886c231-8d9"
Last-Modified: Tue, 24 Jan 2017 02:55:45 GMT
Server: TGWEB
Transfer-Encoding: chunked
Vary: Accept-Encoding
```

第一次请求时，服务器返回的资源最后修改时间

If-Modified-Since:

再次请求服务器时，通过此字段通知服务器上次请求时，服务器返回的资源最后修改时间。服务器收到请求后发现有无 **If-Modified-Since** 则与被请求资源的最后修改时间进行比对。若资源的最后修改时间大于 **If-Modified-Since**，说明资源又被改动过，则响应整片资源内容，返回状态码 **200**；若资源的最后修改时间小于或等于 **If-Modified-Since**，说明资源无新修改，则响应 **HTTP 304**，告知浏览器继续使用所保存的 **cache**。

```
▼ Request Headers    view source
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate, sdch
Accept-Language: zh-CN,zh;q=0.8
Cache-Control: max-age=0
Connection: keep-alive
Host: m.51tiangou.com
If-Modified-Since: Tue, 24 Jan 2017 02:55:45 GMT
If-None-Match: W/"5886c231-8d9"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0
```

再次请求时，浏览器通知服务器，上次请求时返回的资源最后修改时间

Etag / If-None-Match（优先级高于 Last-Modified / If-Modified-Since）

Etag:

服务器响应请求时，告诉浏览器当前资源在服务器的唯一标识（生成规则由服务器决定）。

```
▼ Response Headers view source
Cache-Control: max-age=31536000
Connection: keep-alive
Content-Encoding: gzip
Content-Type: application/javascript
Date: Tue, 24 Jan 2017 07:26:54 GMT
ETag: W/"5886c231-8d9"
Last-Modified: Tue, 24 Jan 2017 02:55:45 GMT
Server: TGWEB
Transfer-Encoding: chunked
Vary: Accept-Encoding
```

第一次请求时，服务器返回的资源唯一标识

If-None-Match:

再次请求服务器时，通过此字段通知服务器客户端缓存数据的唯一标识。服务器收到请求后发现头 **If-None-Match** 则与被请求资源的唯一标识进行比对，
不同，说明资源又被改动过，则响应整片资源内容，返回状态码 **200**；
相同，说明资源无新修改，则响应 **HTTP 304**，告知浏览器继续使用所保存的 cache。

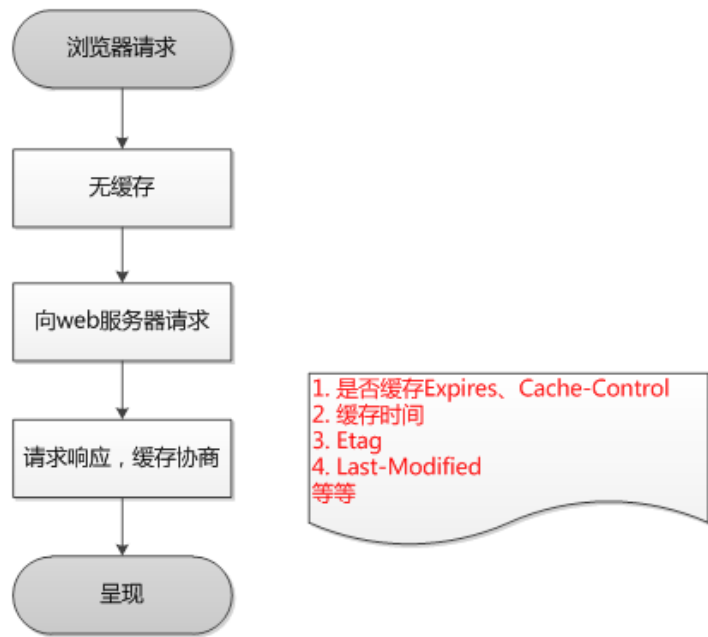
```
▼ Request Headers view source
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate, sdch
Accept-Language: zh-CN,zh;q=0.8
Cache-Control: max-age=0
Connection: keep-alive
Host: m.51tiangou.com
If-Modified-Since: Tue, 24 Jan 2017 02:55:45 GMT
If-None-Match: W/"5886c231-8d9"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0
```

再次请求时，浏览器通知服务器上次返回的资源唯一标识

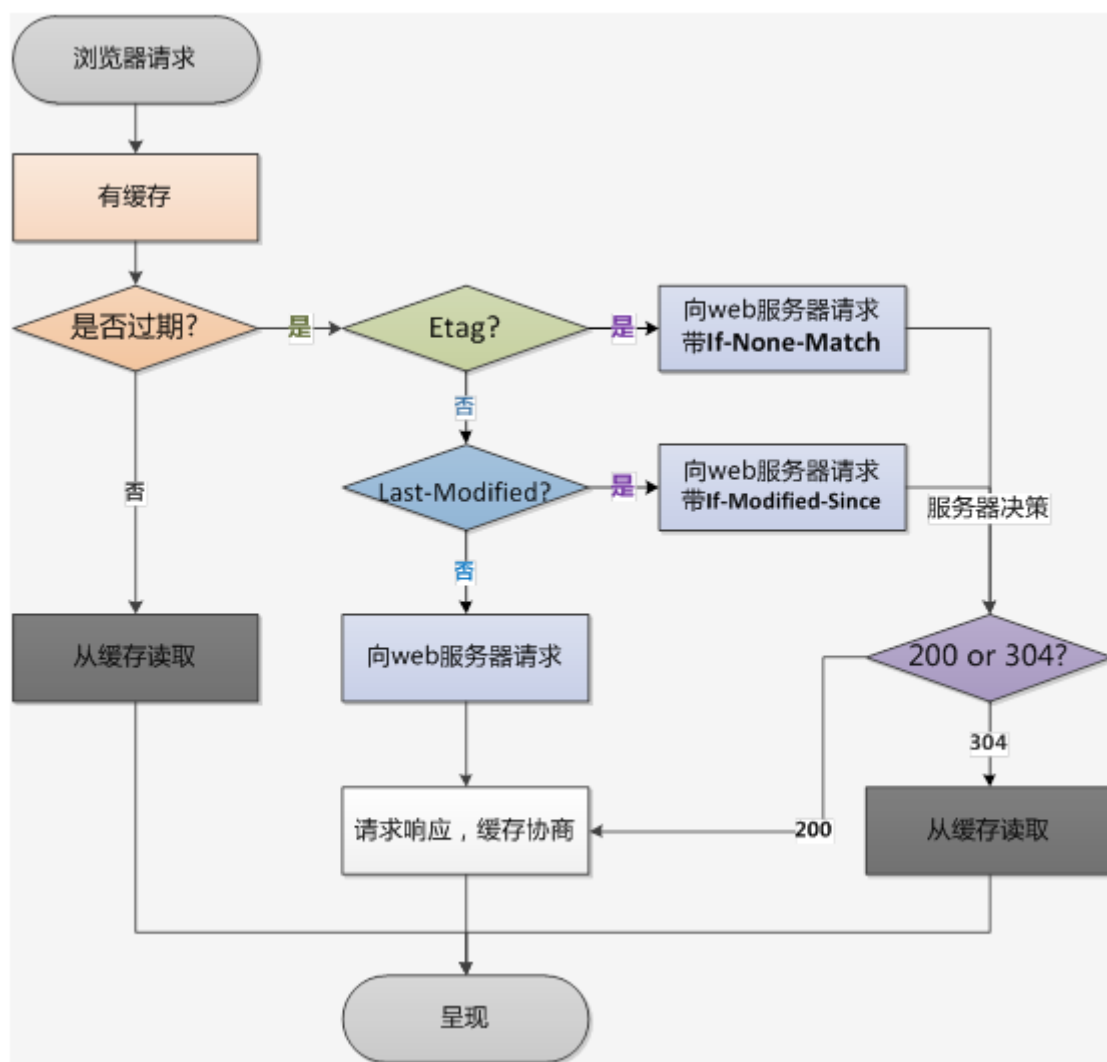
总结

对于强制缓存，服务器通知浏览器一个缓存时间，在缓存时间内，下次请求，直接用缓存，不在时间内，执行比较缓存策略。
对于比较缓存，将缓存信息中的 **Etag** 和 **Last-Modified** 通过请求发送给服务器，由服务器校验，返回 **304** 状态码时，浏览器直接使用缓存。

浏览器第一次请求：



浏览器再次请求时：



文中如果出现错误，希望小伙伴们可以谅解，更希望可以给予指正