浏览器的协商缓存与强缓存

做前端有两个比较令人头痛的事，一个是命名，另一个就是缓存了。缓存的问题在移动端上尤其严重，因为手机随时随地会缓存你的资源，要想清缓存，不像 PC 使用强制刷新，还要手动找到浏览器的缓存，有时候还要重启等。下面这篇文章清晰的讲解关注浏览器的缓存，值得看看。

什么是浏览器缓存

浏览器缓存(Brower Caching)是浏览器在本地磁盘对用户最近请求过的文档进行存储，当访问者再次访问同一页面时，浏览器就可以直接从本地磁盘加载文档。

浏览器缓存的优点有：

1. 减少了冗余的数据传输，节省了网费
2. 减少了服务器的负担，大大提升了网站的性能
3. 加快了客户端加载网页的速度

在前端开发面试中，浏览器缓存是 web 性能优化面试题中很重要的一个知识点，从而说明浏览器缓存是提升 web 性能的一大利器，但是浏览器缓存如果使用不当，也会产生很多问题。

浏览器缓存的分类

浏览器缓存主要有两类：缓存协商和彻底缓存，也有称之为协商缓存和强缓存。

浏览器在第一次请求发生后，再次请求时：

1. 浏览器会先获取该资源缓存的 header 信息，根据其中的 expires 和 cahe-control 判断是否命中强缓存，若命中则直接从缓存中获取资源，包括缓存的 header 信息，本次请求不会与服务器进行通信；

2. 如果没有命中强缓存，浏览器会发送请求到服务器，该请求会携带第一次请求返回的有关缓存的 header 字段信息（Last-Modified/IF-Modified-Since、Etag/IF-None-Match），由服务器根据请求中的相关 header 信息来对比结果是否命中协商缓存，若命中，则服务器返回新的响应 header 信息更新缓存中的对应 header 信息，但是并不返回资源内容，它会告知浏览器可以直接从缓存获取；否则返回最新的资源内容

强缓存

强缓存是利用 http 的返回头中的 `Expires` 或者 `Cache-Control` 两个字段来控制的，用来表示资源的缓存时间。

Expires

该字段是 http1.0 时的规范，它的值为一个绝对时间的 GMT 格式的时间字符串，比如 `Expires:Mon,18 Oct 2066 23:59:59` GMT。这个时间代表着这个资源的失效时间，在此时间之前，即命中缓存。这种方式有一个明显的缺点，由于失效时间是一个绝对时间，所以当服务器与客户端时间偏差较大时，就会导致缓存混乱。所以 HTTP 1.1 的版本，使用 Cache-Control 替代。

Cache-Control

Cache-Control 是 http1.1 时出现的 header 信息，主要是利用该字段的 `max-age` 值来进行判断，它是一个相对时间，例如 Cache-Control:max-age=3600，代表着资源的有效期是 3600 秒。cache-control 除了该字段外，还有下面几个比较常用的设置值：

- no-cache：不使用本地缓存。需要使用缓存协商，先与服务器确认返回的响应是否被更改，如果之前的响应中存在 ETag，那么请求的时候会与服务端验证，如果资源未被更改，则可以避免重新下载。

- no-store：直接禁止游览器缓存数据，每次用户请求该资源，都会向服务器发送一个请求，每次都会下载完整的资源。
- public：可以被所有的用户缓存，包括终端用户和 CDN 等中间代理服务器。
- private：只能被终端用户的浏览器缓存，不允许 CDN 等中继缓存服务器对其缓存。

Cache-Control 与 Expires 可以在服务端配置同时启用，同时启用的时候 Cache-Control 优先级高。

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：from disk cache

协商缓存

协商缓存就是由服务器来确定缓存资源是否可用，所以客户端与服务器端要通过某种标识来进行通信，从而让服务器判断请求资源是否可以缓存访问，这主要涉及到下面两组 header 字段，这两组搭档都是`成对`出现的，即第一次请求的响应头带上某个字段（`Last-Modified` 或者 `Etag`），则后续请求则会带上对应的请求字段（`If-Modified-Since` 或者 `If-None-Match`），若响应头没有 Last-Modified 或者 Etag 字段，则请求头也不会有对应的字段。

Last-Modify/If-Modify-Since

浏览器第一次请求一个资源的时候，服务器返回的 header 中会加上 Last-Modify，Last-modify 是一个时间标识该资源的最后修改时间，例如 Last-Modify: Thu,31 Dec 2037 23:59:59 GMT。

当浏览器再次请求该资源时，request 的请求头中会包含 If-Modify-Since，该值为缓存之前返回的 Last-Modify。服务器收到 If-Modify-Since 后，根据资源的最后修改时间判断是否命中缓存。

如果命中缓存，则返回 304，并且不会返回资源内容，并且不会返回 Last-Modify。

ETag/If-None-Match

与 Last-Modify/If-Modify-Since 不同的是，Etag/If-None-Match 返回的是一个校验码。ETag 可以保证每一个资源是唯一的，资源变化都会导致 ETag 变化。服务器根据浏览器上送的 If-None-Match 值来判断是否命中缓存。

与 Last-Modified 不一样的是，当服务器返回 304 Not Modified 的响应时，由于 ETag 重新生成过，response header 中还会把这个 ETag 返回，即使这个 ETag 跟之前的没有变化。

为什么要有 Etag

你可能会觉得使用 Last-Modified 已经足以让浏览器知道本地的缓存副本是否足够新，为什么还需要 Etag 呢？HTTP1.1 中 Etag 的出现主要是为了解决几个 Last-Modified 比较难解决的问题：

- 一些文件也许会周期性的更改，但是他的内容并不改变(仅仅改变的修改时间)，这个时候我们并不希望客户端认为这个文件被修改了，而重新 GET；
- 某些文件修改非常频繁，比如在秒以下的时间内进行修改，(比方说 1s 内修改了 N 次)，If-Modified-Since 能检查到的粒度是 s 级的，这种修改无法判断(或者说 UNIX 记录 MTIME 只能精确到秒)；

- 某些服务器不能精确的得到文件的最后修改时间。

Last-Modified 与 ETag 是可以一起使用的，服务器会优先验证 ETag，一致的情况下，才会继续比对 Last-Modified，最后才决定是否返回 304。

强缓存与协商缓存的区别可以用下表来表示：

缓存类型	获取资源形式	状态码	发送请求到服务器
强缓存	从缓存取	200（from cache）	否，直接从缓存取
协商缓存	从缓存取	304（Not Modified）	否，通过服务器来告知缓存是否可用

用户行为对缓存的影响

用户操作	Expires/Cache-Control	Last-Modied/Etag
地址栏回车	有效	有效
页面链接跳转	有效	有效
新开窗口	有效	有效
前进回退	有效	有效
F5 刷新	无效	有效
Ctrl+F5 强制刷新	无效	无效

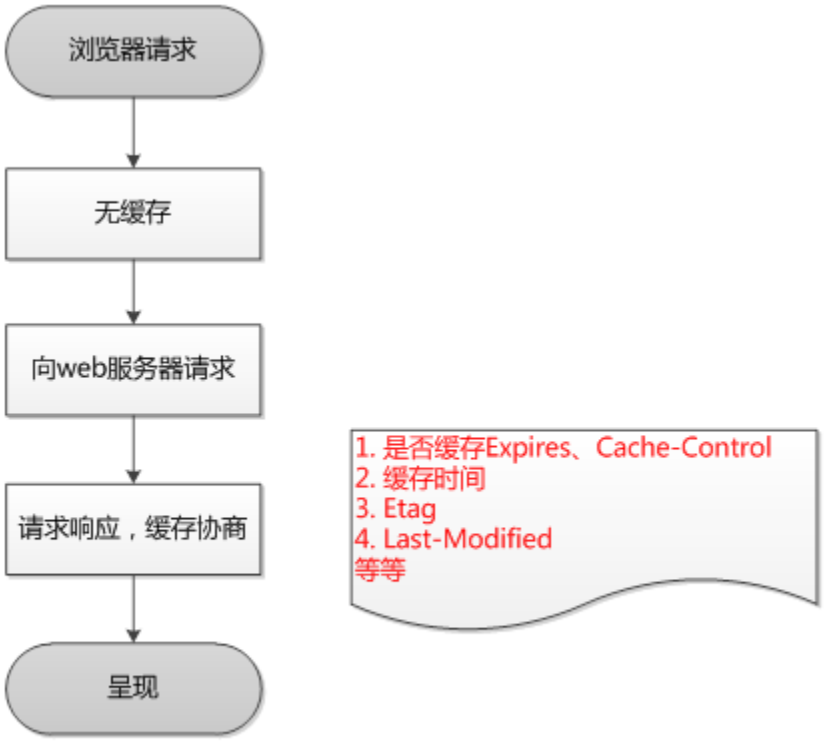
实际问题分析

如文章开头所属，代码更新到线上后用户浏览器不能自行更新，我们不能要求客户在系统更新后都进行一次缓存清理的操作。

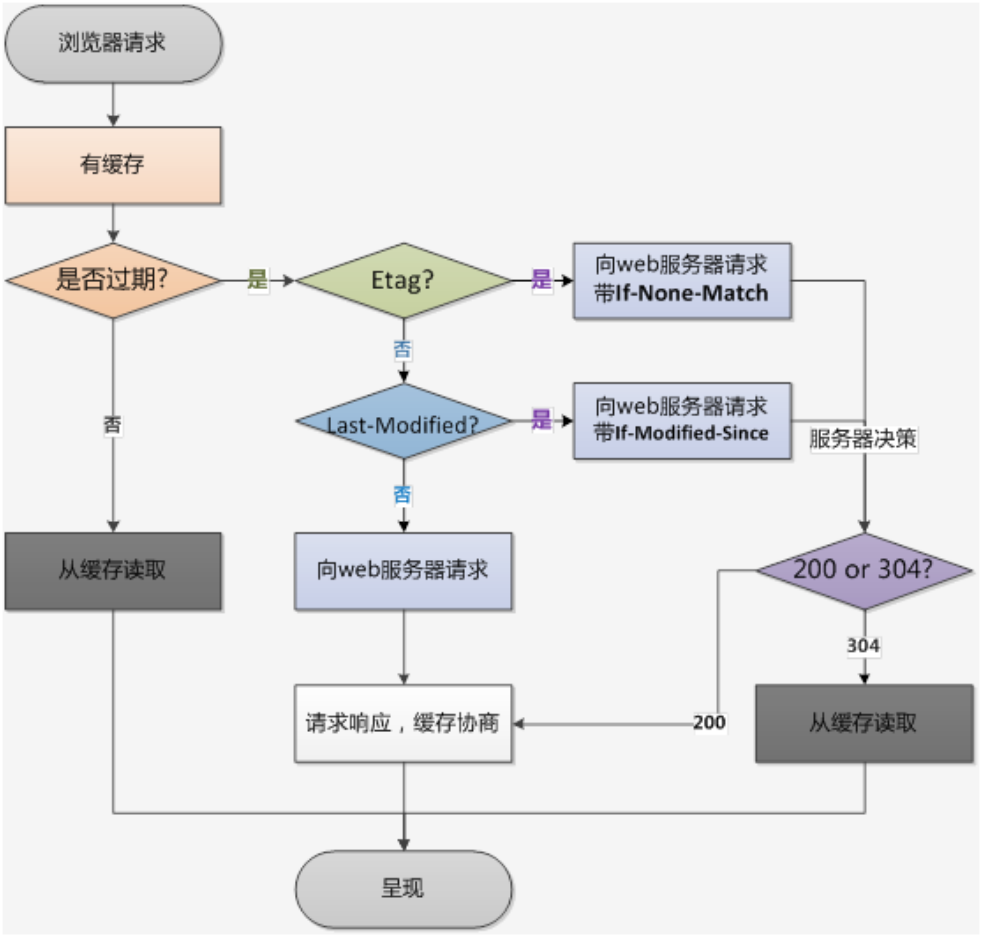
到底该如何解决呢?

在资源请求的 URL 中增加一个参数，比如：js/mian.js?ver=0.7.1。这个参数是一个版本号，每一次部署的时候变更一下，当这个参数变化的时候，强缓存都会失效并重新加载。这样一来，静态资源，部署以后就需要重新加载。这样就比较完美的解决了问题。

第一次请求：


浏览器请求
无缓存
向web服务器请求
请求响应，缓存协商
呈现
1. 是否缓存Expires、Cache-Control
2. 缓存时间
3. Etag
4. Last-Modified
等等

第二次请求


浏览器请求
有缓存
是否过期?
是
Etag?
是
向web服务器请求
带If-None-Match
否
Last-Modified?
是
向web服务器请求
带If-Modified-Since
服务器决策
否
否
从缓存读取
向web服务器请求
200 or 304?
304
请求响应，缓存协商
200
从缓存读取
呈现

html 页面不缓存处理方法

```
<meta http-equiv="Pragma" content="no-cache">

<meta http-equiv="Cache-Control" content="no-cache">

<meta http-equiv="Expires" content="0">
```