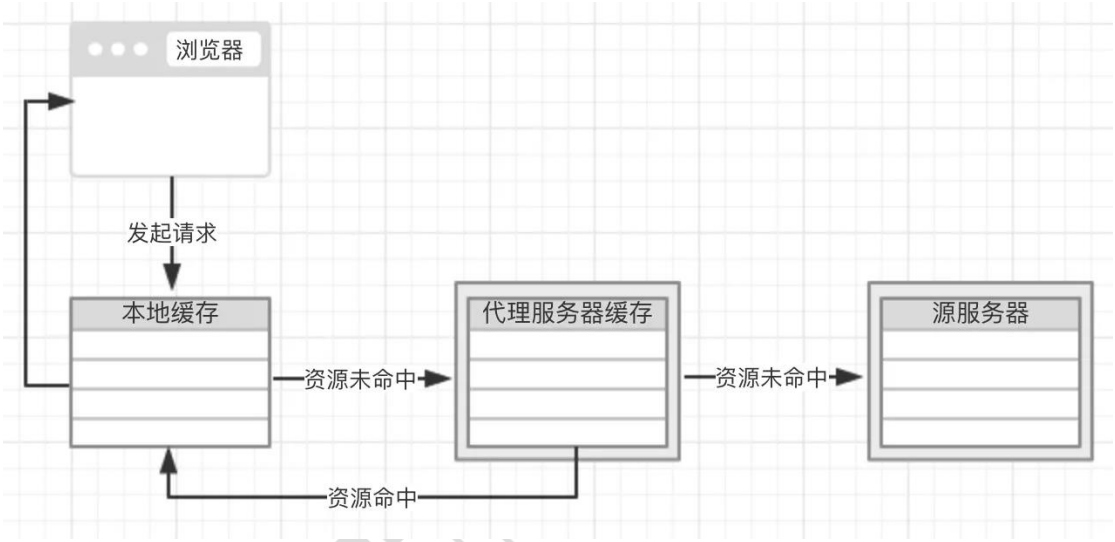
对于 HTTP 缓存你了解的哪些

重用已获取的资源能够有效的提升网站与应用的性能。Web 缓存能够减少延迟与网络阻塞，进而减少显示某个资源所用的时间。借助 HTTP 缓存，Web 站点变得更具有响应性。



1. 浏览器发起请求(携带 Cache-Control)，会先去本地缓存看看是否有缓存并且命中，假如有就直接返回缓存资源，反之则就转向代理服务器；
2. 代理服务器去查找相关的缓存设置，如 s-maxage，以及该资源是否有缓存，同样的会去检查是否命中缓存资源，假如有则会返回至本地缓存，反之则到达源服务器；
3. 到达源服务器后，源服务器会返回资源新文件，然后一步步返回。

这大概就是缓存最粗糙的一个基本流程，接下来我们来一步步的浅析缓存的原理。

Cache-Control

常见缓存请求指令	说明
no-cache	强制向服务器再次验证
no-store	不缓存请求或响应的任何内容
max-age=(秒)	响应的最大 Age 值
max-stale=[秒]	接收已过期的响应
min-fresh=(秒)	期望在指定时间内的响应仍有效

常见缓存响应指令	说明
public	可向任意方提供响应的缓存
private	仅向特定用户返回响应
no-cache	缓存前必须先确认其有效性
no-store	不缓存请求或响应的任何内容
max-age=(秒)	响应的最大 Age 值
s-maxage=(秒)	公共缓存服务器响应的最大 Age 值
must-revalidate	可缓存但必须再向源服务器进行确认

Cache-Control 缓存特性

我们一开始开到表格估计会吓一跳，仅仅只是一个 Cache-Control 就几乎有那么多指令。但实际上我们把它分为特性模块来看，我们自然而然就会清晰很多。

可缓存性

- public：就是该 HTTP 请求所请求的内容，无论是经过**代理服务器**还是客户端，都可以对该请求进行缓存操作；
- private：只有发起请求的浏览器才可以进行缓存，而**代理服务器**则不可以；
- no-cache：这里的意思是可以缓存，但是在缓存之前不管过没过期，都需要向源服务器进行资源有效性校验；

过期性

- max-age：该缓存什么时候到期；
- s-maxage：在**代理服务器**中，如果我们同时设置了 max-age 以及 s-maxage，那么**代理服务器**会读取 s-maxage，因为该指令是专门为代理服务器而存在的；

重新验证

- must-revalidate：假如还没到过期时间，那么可以使用缓存资源；反之就必须到源服务器进行有效检验；
- proxy-revalidate：同 must-revalidate，只是 proxy-revalidate 用在缓存服务器；

Expires

除了 Cache-Control 可以控制资源的缓存状态之外，还有 Expires，它是 HTTP 1.0 的产物，但是还是有很多地方会有到它。它跟 Cache-Control 中的 max-age 有什么区别呢？

- 表达方式：Expires 是绝对时间，如 Expires: Tue Jul 09 2019 23:13:28 GMT+0800，而 max-age 是相对时间，如 max-age=3600；
- 协议版本：Expires 是 HTTP 1.0 版本的首部字段，而 max-age 是 HTTP 1.1 版本及其之后的首部字段；
- 优先级：当请求协议版本为 HTTP 1.0 时，同时存在 Expires 和 max-age 会**无视** max-age，而当请求协议版本为 HTTP 1.1 则会**优先处理** max-age 指令；

除此之外，它们的使用方法时一样的，因此我们就不再实战演示了，它们都是用来校验**强缓存**的标识。

缓存校验 Last-Modified & ETag

Last-Modified

顾名思义，上次修改时间。主要配合 If-Modified-Since 或者 If-Unmodified-Sice。

基本流程：

1. 首次请求资源，服务器返回资源时带上实体首部字段 Last-Modified；
2. 当我们再次请求该资源时，浏览器会自动在请求头带上首部字段 If-Modified-Since，此时的 If-Modified-Since 等于 Last-Modified ；
3. 服务器接收到请求后，会根据 If-Modified-Since 配合 Last-Modified 来判断资源在该日期之后是否发生过变化；
4. 如果发生修改了，则返回新的资源并返回新的 Last-Modified，反之则返回状态码 304 Not Modified，这个过程称为**协商缓存**。

ETag

相对于 Last-Modified，ETag 是一个更加严格的验证，它主要是通过数字签名表示资源的**唯一性**，但当该资源发生修改，那么该签名也会随之变化，但是无论如何都会保证它的**唯一性**。所以根据它的**唯一性**，就可以 If-Match 或者 If-Non-Match 知道资源有没有发生修改。

基础流程： 同 Last-Modified，只是把 Last-Modified 换成 ETag， If-Modified-Since 换成 If-Match 。但是假如 Last-Modified 以及 ETag 同时存在，则后者 ETag 的优先级比较高。

强缓存 & 协商缓存

在进行最后一个实战模拟之前，要先说下这两个十分重要的概念：**强缓存**以及**协商缓存**。

强缓存

简单粗暴来讲，就是**客户端知道资源过期时间后，由客户端来决定要不要缓存。**那么怎么知道资源的过期时间呢？由谁来决定它们的过期时间呢？就是由我们上文提到的 Expires 以及 Cache-Control: max-age。

协商缓存

跟**强缓存**相反，是由**服务器来决定客户端要不要使用缓存**。在有 ETag 以及 Last-Modified 响应首部字段的情况下，客户端会向服务器发起资源的缓存校验，然后服务器会告知客户端是使用缓存(304)还是返回一个全新的资源，表面上看都是会发起一个请求，但是响应的时候则是不是一个完整的响应则看是否需要缓存。

还记得 Cache-Control 的指令 no-cache 和 no-store 吗？这时候应该就清楚了两者的区别了。no-cache 就是直接跳过**强缓存**进入**协商缓存**。而 no-store 则是**不缓存**，效果等同于 Chrome 浏览器的 Disable Cache，仔细观察，你会发现请求首部字段是不会携带关于缓存的任何首部字段。

总结

这下总算知道为什么有时候 Chrome 浏览器会展示 disk cache/memory cache 了吧，它跟 304 Not Modified 同样都是被缓存的意思，这是方式不一样。由此可见，合理的使用缓存是多么的重要，它可以使我们**减少无所谓的请求、避免资源文件的重复传输、减少对源服务器的资源占用**等等好处，但也不能滥用。