

浏览器缓存知识

1.强缓存原理

当浏览器对某个资源的请求命中了强缓存时，返回的 http 状态为 200，在 chrome 的开发者工具的 network 里面 size 会显示为 from cache。

Name	Status	Type	Initiator	Size	Time
item2.png	200	png	(index):542	5.7 KB	15 ms
g-merge.gif	200	gif	(index):721	18.8 KB	17 ms
tangram.js?v=37768233.js	200	script	share.js?v=89860593.js..	35.4 KB	20 ms
api_base.js	200	script	share.js?v=89860593.js..	956 B	11 ms
view_base.js	200	script	share.js?v=89860593.js..	1.1 KB	21 ms
partners.js?v=911c4302.js	200	script	share.js?v=89860593.js..	1.2 KB	9 ms
share_style0_24.css	200	stylesheet	share.js?v=89860593.js..	1.2 KB	15 ms
logger.js?v=d16ec0e3.js	200	script	share.js?v=89860593.js..	(from cache)	3 ms
v.gif?pid=307&type=3071&sign=&desturl=&u...	200	gif	Other	289 B	50 ms
v.gif?l=http%3A%2F%2Fwww.ci123.com%2Fs...	200	gif	Other	198 B	62 ms

57 requests | 1.0 MB transferred | Finish: 3.79 s | DOMContentLoaded: 668 ms | Load: 1.40 s

强缓存是利用 Expires 或者 Cache-Control 这两个 http response header 实现的，它们都用来表示资源在客户端缓存的有效期。

1.1Expires 实现强缓存

Expires 是 http1.0 提出的一个表示资源过期时间的 header，它描述的是一个绝对时间，由服务器返回，用 GMT 格式的字符串表示，如：Expires:Thu, 31 Dec 2037 23:55:55 GMT，它的缓存原理是：

1) 浏览器第一次跟服务器请求一个资源，服务器在返回这个资源的同时，在 response 的 header 加上 Expires 的 header，如：

```
Expires: Tue, 27 Sep 2016 07:54:48 GMT
```

2) 浏览器在接收到这个资源后，会把这个资源连同所有 response header 一起缓存下来(所以缓存命中的请求返回的 header 并不是来自服务器，而是来自之前缓存的 header)；

3) 浏览器再请求这个资源时，先从缓存中寻找，找到这个资源后，拿出它的 Expires 跟当

前的请求时间比较,如果请求时间在 Expires 指定的时间之前,就能命中缓存,否则就不行。

4) 如果缓存没有命中,浏览器直接从服务器加载资源时,Expires Header 在重新加载的时候会被更新。

1.2 Cache-Control 实现强缓存

在 http1.1 的时候,提出了一个新的 header,就是 Cache-Control,这是一个相对时间,在配置缓存的时候,以秒为单位,用数值表示,如:

Cache-Control:max-age=315360000,它的缓存原理是:

1) 浏览器第一次跟服务器请求一个资源,服务器在返回这个资源的同时,在 response 的 header 加上 Cache-Control 的 header,如:

```
Cache-Control: max-age=31536000
```

2) 浏览器接收到这个资源后,会把这个资源连同所有 response header 一起缓存下来;

3) 浏览器再请求这个资源时,先从缓存中寻找,找到这个资源后,根据它第一次的请求时间和 Cache-Control 设定的有效期,计算出一个资源过期时间,再拿这个过期时间跟当前的请求时间比较,如果请求时间在过期时间之前,就能命中缓存,否则就不行。

4) 如果缓存没有命中,浏览器直接从服务器加载资源时,Cache-Control Header 在重新加载的时候会被更新。

1.3 比较

Cache-Control 描述的是一个相对时间,在进行缓存命中的时候,都是利用客户端时间进行判断,所以相比较 Expires,Cache-Control 的缓存管理更有效,安全一些。

这两个 header 可以只启用一个，也可以同时启用，当 response header 中，Expires 和 Cache-Control 同时存在时，Cache-Control 优先级高于 Expires。

2.强缓存管理

2.1 设置

在实际应用中我们会碰到需要强缓存的场景和不需要强缓存的场景，通常在服务器设置是否启用强缓存。



2.2 清除

由于在开发的时候不会专门去配置强缓存，而浏览器又默认会缓存图片，css 和 js 等静态资源，所以开发环境下经常会因为强缓存导致资源没有及时更新而看不到最新的效果，解决这个问题的方法有很多，常用的有以下几种：

- 直接 ctrl+f5，这个办法能解决页面直接引用的资源更新的问题；
- 使用浏览器的隐私模式开发；
- 如果用的是 chrome，可以在 network 那里把缓存给禁掉
- 如果缓存问题出现在 ajax 请求中，最有效的解决办法就是 ajax 的请求地址追加随机数

3.协商缓存原理

当浏览器对某个资源的请求没有命中强缓存，就会发一个请求到服务器，验证协商缓存是否命中，如果协商缓存命中，请求响应返回的 http 状态为 304 并且会显示一个 Not Modified 的字符串；

 style1.css	304
 style_sub1.css	304

协商缓存从客户端缓存中加载的，而不是服务器最新的资源：

协商缓存是利用的是【Last-Modified ,If-Modified-Since】和【ETag、If-None-Match】这两对 Header 来管理的。

3.1 【Last-Modified , If-Modified-Since】实现协商缓存

浏览器第一次跟服务器请求一个资源，服务器在返回这个资源的同时，在 response 的 header 加上 Last-Modified 的 header，这个 header 表示这个资源在服务器上的最后修改时间：

`Last-Modified: Tue, 15 Mar 2016 08:37:11 GMT`

浏览器再次跟服务器请求这个资源时，在 request 的 header 上加上 If-Modified-Since 的 header，这个 header 的值就是上一次请求时返回的 Last-Modified 的值：

服务器再次收到资源请求时，根据浏览器传过来 If-Modified-Since 和资源在服务器上的最后修改时间判断资源是否有变化，如果没有变化则返回 304 NotModified，但是不会返回资源内容；如果有变化，就正常返回资源内容。当服务器返回 304 Not Modified 的响应时，response header 中不会再添加 Last-Modified 的 header，因为既然资源没有变化，那么 Last-Modified 也就不会改变，这是服务器返回 304 时的 response header

浏览器收到 304 的响应后，就会从缓存中加载资源。

如果协商缓存没有命中，浏览器直接从服务器加载资源时，Last-Modified Header 在重新加载的时候会被更新，下次请求时，If-Modified-Since 会启用上次返回的 Last-Modified 值。

3.2 【ETag、If-None-Match】实现协商缓存

浏览器第一次跟服务器请求一个资源，服务器在返回这个资源的同时，在 response 的 header 加上 ETag 的 header，这个 header 是服务器根据当前请求的资源生成的一个唯一

标识，这个唯一标识是一个字符串，只要资源有变化这个串就不同，跟最后修改时间没有关系，所以能很好的补充 Last-Modified 的问题。

浏览器再次跟服务器请求这个资源时，在 request 的 header 上加上 If-None-Match 的 header，这个 header 的值就是上一次请求时返回的 ETag 的值：

服务器再次收到资源请求时，根据浏览器传过来 If-None-Match 和然后再根据资源生成一个新的 ETag，如果这两个值相同就说明资源没有变化，否则就是有变化；如果没有变化则返回 304 Not Modified，但是不会返回资源内容；如果有变化，就正常返回资源内容。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：

浏览器收到 304 的响应后，就会从缓存中加载资源。

4.总结

- 共同点是：如果命中，都是从客户端缓存中加载资源，而不是从服务器加载资源数据
- 区别是：强缓存不发请求到服务器，所以有时候资源更新了浏览器还不知道；协商缓存会发请求到服务器，资源是否更新，服务器肯定知道。
- 大部分 web 服务器都默认开启协商缓存，而且是同时启用【Last-Modified，If-Modified-Since】和【ETag、If-None-Match】。
- 如果没有协商缓存，每个到服务器的请求，就都得返回资源内容，这样服务器的性能会极差。