**缓存**

expires （一个末期时刻）

缺陷：本地时间判断是否向服务器请求数据

1. 本地时间与服务器时间不一定相等。
2. 本地时间可以修改。

cache-control （最大有效时间，一个相对时间）

单位是秒,如 cache-control：

‘max-age=5’

no-cache: 直接访问服务器，判断文件是否有变化。（etag）

no-store ： 不允许浏览器与其它任何中间缓存 储存响应

public : 表示缓存可以被缓存

private：只能为单个用户缓存（浏览器），而不允许中间缓存 储存（如CDN）;

Last-modified 与 If-Modified-Since

返回头 请求头（上一次的返回头的Last-modified）

文件更新时间对比，相等则文件未改变，不相等则文件改变，重新拉取文件。

缺陷：由服务器渲染生成的页面会被判定为更改。返回的更新时间未渲染页面后的时间，也就是说缓存失效。

Etag令牌 与 If-None-Match

**我们唯一要做的就是确保服务器提供必要的 ETag 令牌。检查您的服务器文档中有无必要的配置标志**

返回头 请求头（上一次的返回头的Etag）

文件是否修改:文件修改标记。

注：

浏览器行为引起的缓存请求变化

1. 刷新网页（就是url地址 enter。而且是一个新的窗口，如果是当前网站里的地址栏enter，也会max-age=0）

浏览器（cache-control 、 expires）判断是否过期

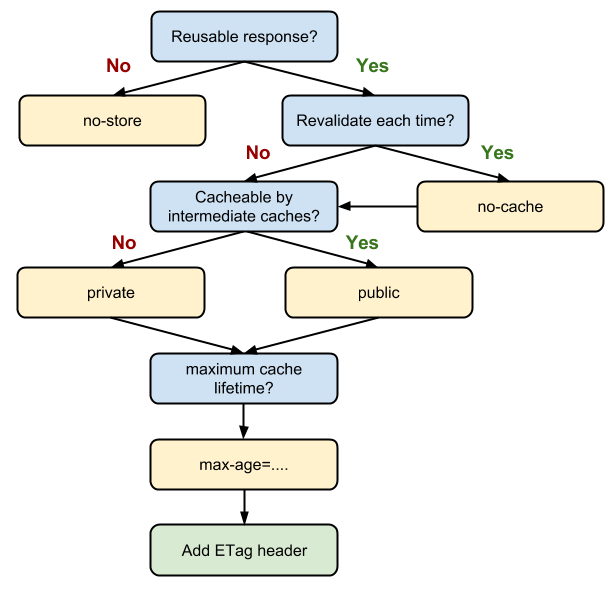
1. F5 或者鼠标右键刷新 ，浏览器认为缓存失效，询问服务器数据是否刷新。

请求头：cache-control：max-age=0

3. ctrl+f5 。强制刷新

请求头：Pragma：no-cache ; cache-control:no-cache

缓存策略决策树：



cache-control 默认值为private,且可以有多个值

在html文件返回加no-cache，检测在其中的文件是否有变更

如 a.a111.css

而将a.a111.css的max-age设为31536000

**关键的是：确定缓存层次结构**

**需要注意的是：**

**使用一致的网址**

**确保服务器提供验证令牌**

**确定中间缓存可以缓存那些资源**

**确定最适合的缓存层次结构**

**为每个资源确定最佳缓存周期  
 自大限度减少搅动（将文件中频繁更新的部分提取出来，制定新的缓存策略）**