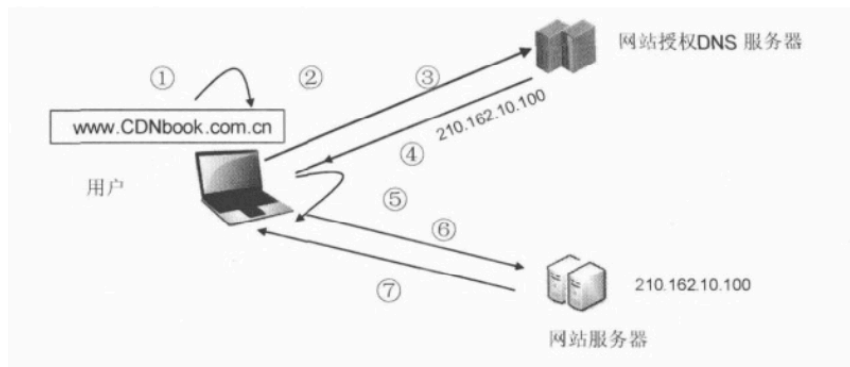


# 前端缓存

什么是缓存？缓存分类？缓存工作机制？

## 浏览器请求资源过程

浏览器发起请求到收到服务器响应的过程：



上图说明了一个浏览器向服务端请求资源的大概过程，但是有些步骤还是可以细分的：

- 1.客户端：输入URL，enter
- 2.DNS解析URL得到主机IP
- 3.向主机发起TCP连接请求（三次握手）
- 4.建立了TCP连接，客户端发起HTTP请求
- 5.服务器监听到请求，开始处理请求
- 6.服务器返回响应
- 7.客户端接收到响应, 处理响应
- 8.渲染

缓存存在于以上哪些步骤？

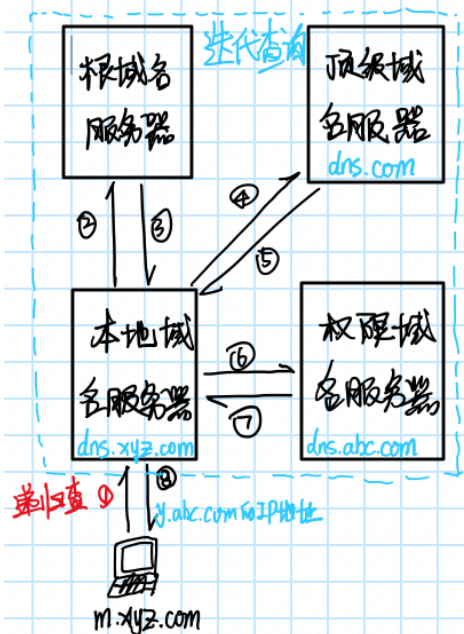
## DNS缓存

DNS解析过程

- 1.查找浏览器自身缓存
- 2.查找主机host文件
- 3.本地DNS服务器（路由器DNS缓存）
- 4.根域名DNS服务器，全球13台固定IP

## 解析过程:

- ① 主机向本地域名服务器的查询 **使用递归**
- ② 本地域名服务器向根域名服务器 **使用迭代**



- ① 主机 m.xyz.com 向本地域名服务器 dns.xyz.com 进行递归
- ② 本地域名服务器采用迭代查询, 先向一个根域名服务器查询
- ③ 根域名服务器告诉本地域名服务器, 下次应查询的
- ④ 本地域名服务器向顶级域名服务器 dns.com 进行查询.
- ⑤ 顶级域名服务器告诉本地域名服务器, 下次应查询的权威域
- ⑥ 本地域名服务器向权威域名服务器 dns.abc.com 进行查询.
- ⑦ 权威域名服务器 dns.abc.com 告诉本地域名服务器, 所查询的主
- ⑧ 本地域名服务器最后把结果告诉主机 m.xyz.com

**DNS 优化:** 浏览器缓存, 系统缓存, 路由器缓存,

## 浏览器缓存处理步骤

我们通常接触最多的缓存

1. 接收, 缓存从网络中读取抵达的请求报文
2. 解析, 解析报文, 提取出 URL 和各种首部
3. 查找, 缓存查看有本地副本可用, 没有就获取一份, 并保存在本地
4. 新鲜度检测, 本地有缓存就开始新鲜度校验
  - 强缓存
  - 协商缓存
5. 创建响应, 缓存用新的首部和已缓存的主体来构建一条响应报文
6. 发送, 缓存通过网络将响应返回给客户端
7. 日志, 可选得建一个日志条目来描述这个事务

## 强缓存

请求发出, 浏览器先检查本地缓存, 若有, 检查新鲜度, 新鲜则命中缓存, 否则执行协商缓存

缓存控制:

1. 第一次发起 GET 请求, 服务器返回文档, 浏览器缓存该文档 (由服务器控制)

HTTP1.1服务器返回响应头Cache-Control: max-age=18000, 则浏览器缓存该响应, 新鲜度18000s, 过程如下:

```
1 request:
2 GET index.html HTTP/1.1
3 Host:www.abc.com
4 Accept-Language: zh-CN;
5
6 response:
7 HTTP/1.1 200 OK
8 Content-Type: application/json; charset=UTF-8
9 Cache-Control: max-age=18000
10 Last-Modified: Thu,04 Nov 2021 07:24:38 GMT
11 Date: ...
```

HTTP1.0服务器可能返回Expires响应头, 表示该缓存在未来某个时间节点不再新鲜:

```
1 Expires: Fri, 04 Nov 2021 07:24:38 GMT
```

至此, 浏览器缓存了一个文档, 等待下一次请求, 缓存是否生效。

2.第二次发起同样的请求, 浏览器缓存检测到并解析该请求, 然后在浏览器缓存中找到了这个请求的历史响应。

开始检查缓存的新鲜度:

- 若缓存由Cache-Control控制, 检查max-age与当前日期减去上次响应创建时间(Date响应头)大小
- 若缓存由Expires控制, 检查系统时间

3.若缓存新鲜, 命中缓存, 否则请求转发给服务器, 进行协商缓存校验

## 协商缓存

缓存过了expires或者max-age时间, 不代表该文档有改动, 需要服务器进一步验证

服务器再验证有两种方式:

1.发起一条GET请求, 该请求包含校验新鲜度的请求头If-Modified-Since (IMS请求), 询问从某个时间节点(上次请求的Last Modified Date)以后文件是否发生改变, 服务器需要回答这个问题。

```
1 浏览器发起GET请求
2 If-Modified-Since: Thu,04 Nov 2021 07:24:38 GMT
3
4 服务器响应:
5 1.文档没有修改,返回如下响应头, 没有响应体
6 HTTP/1.1 304 NOT Modified
7 Date:...
8 Cache-Control: max-age=18000
9 2.文档修改过, 返回新的文档
10 HTTP1.1 200 OK
11 Date:...
12 Content-type: text/plain
```

```
13 Content-length: 124
14 Cache-Control: max-age=18000
15
16 this is a new doc.
```

2.文件被重写，不代表内容变化，校验文件是否发生改动，使用If-None-Match首部。

服务器第一次返回响应带上文档的实体标签Etag，浏览器第二次请求带上If-None-Match首部检验：

```
1 第一次服务器响应
2 HTTP/1.1 200 OK
3 Date:...
4 Etag:"v3.1"
5
6 浏览器第二次请求：
7 GET index.html HTTP/1.1
8 If-None-Match:"3.1"
9
10 服务器响应：
11 1.没有改动，返回304
12 2.返回新的文档和Etag
```

若服务器返回了一个实体标签，HTTP/1.1客户端必须使用实体标签验证器。

If-Modified-Since 和 If-None-Match同时存在，优先校验Etag。

## 启发式缓存、试探性过期

1.先小结一下前面提到的知识，浏览器缓存处理的流程如下：

发起请求=>缓存接受请求，解析

=>查找缓存：

无缓存=>向服务器请求新的副本并缓存

有缓存=>新鲜度校验？

2.服务器没有返回明确的缓存策略，既没有返回Expires首部，也没有Cache-Control首部，缓存会计算出一个 **试探性** 最大使用期。常用的算法是LM-Factor算法，该算法利用Date首部和Last Modified首部：

$$\text{factor} * (\text{Date} - \text{Last Modified})$$

factor大于0小于1通常取值0.2或0.1。

缓存没有更改的两种情况：

- 很久没有访问，即Date比较前，那么再进行访问，缓存很可能失效，请求将从服务器返回新的副本
- 最近才访问，Date比较新，再次进行访问，缓存可能还在新鲜期，那么请求由缓存处理，此时即使服务器更新资源，客户端提供的仍然是旧的副本

# 浏览器缓存位置

打开Chrome devtool，在网络tab下能看到以下几种资源访问情况：

Type	Initiator	Cookies	Size	Time	W:
png	style.css	0	(memory cache)	0 ms	
png	app.fe950d8....js:1	0	(memory cache)	0 ms	
png	app.fe950d8....js:1	0	(memory cache)	0 ms	
png	app.fe950d8....js:1	0	(memory cache)	Pend...	
png	style.css	0	(memory cache)	Pend...	
png	vendor.8535812....js:12	0	(memory cache)	1 ms	
xhr	vendor.8535812....js:12	0	31.4 kB	77 ms	
png	style.css	0	(memory cache)	0 ms	

Type	Initiator	Cookies	Size	Time	W:
script	style.css	0	(disk cache)	0 ms	
stylesh...	(index)	0	(disk cache)	4 ms	
script	(index)	0	(disk cache)	5 ms	
script	(index)	0	(disk cache)	15 ms	
script	(index)	0	(disk cache)	8 ms	
png	style.css	0	(disk cache)	1 ms	
png	vendor.8535812....js:12	0	(disk cache)	1 ms	
png	vendor.8535812....js:12	0	(disk cache)	1 ms	
xhr	vendor.8535812....js:12	0	1.8 kB	21 ms	

1.一种是存放在内存中的缓存，memory cahche:

浏览器获取到副本以后，会保存在内存中一段时间，再次访问直接读取内存，关闭tab会释放资源。

2.一种是存放在磁盘的缓存，disk cache:

浏览器第一次获取到服务器副本，会保存在本地（根据服务器和自己的计算策略），是最主要的缓存来源。

3.上面图片Size除了memory cache、disk cache，还有一种数值，如31.4kb，这种一般是从服务器返回来的资源大小，如果是304状态码，则表示报文大小。

disk cache既然是存放在磁盘，那就应该可以查看这些缓存副本:

1

//mac存放位置

2

~/资源库/Caches/Google/Chrome/Default/Cache

可以看到，副本被浏览器编码保存。怎么查看？在windows下通过工具ChromCacheView工具可以查看缓存：



文件名	URL网址	内容类型	文件大小	最后访问	服务器时间	上次修改的服务器	过期时间	服务器名	服务器响应	Web Site	Frame	内容编码	缓存	缓存控制	ETag	服务器IP...	URL长度	Deleted File
zh...	http://pip...	image/x-ic...	3,774	2021/11/16 15:0...	2021/11/1...	2021/11/5 14:2...	2021/1...	nginx	HTTP/1.1 200	http://cmbchina.cn	http://cmbchina.cn		data_2 [6324224]		'618...	55.13.21...	57	No
qrc...	http://pip...	image/svg...	1,049	2021/11/16 15:0...	2021/11/1...	2021/11/5 14:2...	2021/1...	nginx	HTTP/1.1 200	http://cmbchina.cn	http://cmbchina.cn		data_2 [6320128]	public	W/"6...	55.13.21...	60	No
a3...	http://pip...	applicatio...	287	2021/11/16 15:0...	2021/11/1...			nginx	HTTP/1.1 200	http://cmbchina.cn	http://cmbchina.cn	gzip	data_1 [4656640]			55.13.21...	108	No
ale...	http://pip...	image/svg...	3,748	2021/11/16 15:0...	2021/11/1...	2021/11/5 14:2...	2021/1...	nginx	HTTP/1.1 200	http://cmbchina.cn	http://cmbchina.cn		data_2 [6316032]	public	W/"6...	55.13.21...	56	No
tru...	http://pip...	applicatio...	2	2021/11/16 15:0...	2021/11/1...			nginx	HTTP/1.1 200	http://cmbchina.cn	http://cmbchina.cn		data_1 [4658944]			55.13.21...	121	No
na...	http://pip...	image/svg...	4,531	2021/11/16 15:0...	2021/11/1...	2021/11/5 14:2...	2021/1...	nginx	HTTP/1.1 200	http://cmbchina.cn	http://cmbchina.cn		data_3 [11866592]	public	W/"6...	55.13.21...	54	No
bo...	http://pip...	image/svg...	3,584	2021/11/16 15:0...	2021/11/1...	2021/11/5 14:2...	2021/1...	nginx	HTTP/1.1 200	http://cmbchina.cn	http://cmbchina.cn		data_2 [22835200]	public	W/"6...	55.13.21...	54	No
no...	http://pip...	image/png	51,079	2021/11/16 15:0...	2021/11/1...	2021/11/5 14:2...	2021/1...	nginx	HTTP/1.1 200	http://cmbchina.cn	http://cmbchina.cn	gzip	f_00612a	public	W/"6...	55.13.21...	62	No
cm...	http://pip...	image/svg...	10,152	2021/11/16 15:0...	2021/11/1...	2021/11/5 14:2...	2021/1...	nginx	HTTP/1.1 200	http://cmbchina.cn	http://cmbchina.cn	gzip	data_3 [11870208]	public	W/"6...	55.13.219.26	64	No
roc...	http://pip...	image/svg...	2,581	2021/11/16 15:0...	2021/11/1...	2021/11/5 14:2...	2021/1...	nginx	HTTP/1.1 200	http://cmbchina.cn	http://cmbchina.cn	gzip	data_2 [22839296]	public	W/"6...	55.13.21...	64	No
8-e...	http://pip...	applicatio...	984,000	2021/11/16 15:0...	2021/11/1...	2021/11/5 14:2...	2021/1...	nginx	HTTP/1.1 200	http://cmbchina.cn	http://cmbchina.cn	gzip	f_006129	public	W/"6...	55.13.21...	67	No
5-e...	http://pip...	applicatio...	6,808	2021/11/16 15:0...	2021/11/1...	2021/11/5 14:2...	2021/1...	nginx	HTTP/1.1 200	http://cmbchina.cn	http://cmbchina.cn	gzip	data_3 [11862016]	public	W/"6...	55.13.21...	67	No
3-e...	http://pip...	applicatio...	193,107	2021/11/16 15:0...	2021/11/1...	2021/11/5 14:2...	2021/1...	nginx	HTTP/1.1 200	http://cmbchina.cn	http://cmbchina.cn	gzip	f_006128	public	W/"6...	55.13.21...	67	No
4-e...	http://pip...	applicatio...	27,838	2021/11/16 15:0...	2021/11/1...	2021/11/5 14:2...	2021/1...	nginx	HTTP/1.1 200	http://cmbchina.cn	http://cmbchina.cn	gzip	f_006125	public	W/"6...	55.13.21...	67	No
1-e...	http://pip...	applicatio...	17,106	2021/11/16 15:0...	2021/11/1...	2021/11/5 14:2...	2021/1...	nginx	HTTP/1.1 200	http://cmbchina.cn	http://cmbchina.cn	gzip	f_006122	public	W/"6...	55.13.21...	67	No
1-e...	http://pip...	applicatio...	80,918	2021/11/16 15:0...	2021/11/1...	2021/11/5 14:2...	2021/1...	nginx	HTTP/1.1 200	http://cmbchina.cn	http://cmbchina.cn	gzip	f_006124	public	W/"6...	55.13.21...	67	No

## 4.APP端缓存存在哪里

data/cache

# 缓存控制

怎么启用、禁止缓存？缓存验证策略？

1.使用HTTP首部，以下首部在服务器原始请求中添加：

- 1 Cache-Control:no-store; //禁止缓存复制响应
- 2 Cache-Control:no-cache; //缓存可以存储响应，但在提供给客户端使用之前必须先向服务端验证新鲜度
- 3 Cache-Control:max-age=3600; //新鲜度1小时
- 4 Expires:Date //指明过期时间（不推荐）
- 5 Cache-Control:must-revalidate; //可以通过配置缓存，使其提供一些过期的对象以提高性能，若服务器希望客户端严格遵守过期信息，可以在响应加上这个请求头告诉缓存，没有根服务器验证之前不能提供对象的陈旧副本，对新鲜的副本则没有限制。

2.服务器既没有返回Expires首部，也没有返回Cache-Control首部，缓存会尝试计算出一个试探性最大使用期。可以使用任意算法

3.通过HTTP-EQUIV控制HTML缓存

- 1 HTML2.0定义了<META HTTP-EQUIV>标签
- 2 <html>
- 3 <head>
- 4 <title>doc</title>
- 5 <META HTTP-EQUIV="Cache-control" CONTENT="no-cache">

4.其他

在文档后面加上不同的版本号以获取最新文档而非缓存：

- 1 http://www.abc.com/index.css?v=sdfadf

浏览器刷新按钮，会自动添加Cache-Control首部

## CDN

什么是CDN？CDN的作用、为什么要用它？怎么使用CDN？简单了解一下原理。

1.CDN指的是内容分发网络（由一组分布在不同地方的主机构成），能够将服务器资源发布到边缘节点。

- 缓存服务器资源
- 将离用户最近的资源分发给用户

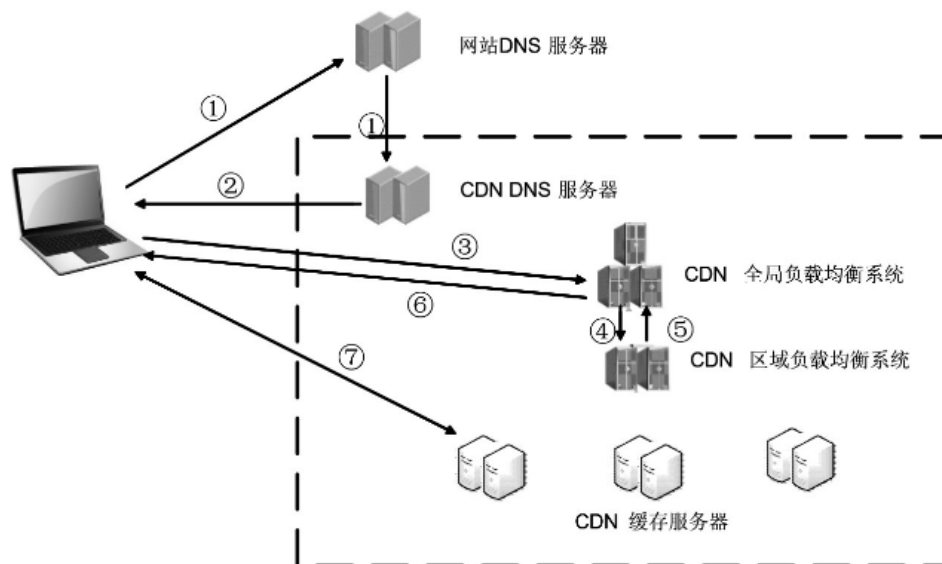
2.作用：

- 加速，提升资源访问速度
- 实现跨运营商、跨地域的全网覆盖
- 容错，健壮

3.架构

- DNS解析
- 负载均衡，合理分配
- 内容分发

4.调度



1.用户输入URL，经本地DNS系统解析，DNS系统会将域名的解析权交给CNAME指向的CDN专用DNS服务器

2.CDN服务器将CDN的全局负载均衡设备IP地址返回给用户

3.用户向CDN的全局负载均衡设备发起内容URL访问

4.全局负载均衡设备根据用户IP地址和请求内容URL，选择一台用户所属区域的负载均衡设备

5.区域负载均衡设备为用户选择一台合适的缓存服务器提供服务（考虑因素：离用户最近，是否有该URL资源）

6.全局负载均衡设备把区域负载均衡选择的服务器IP返回给用户

7.用户向该缓存服务器IP发起请求，并得到响应

如果缓存服务器的副本过期了，或者根本就没有副本，该缓存服务器会一层一层往上查找缓存，若整个CDN没有该副本，会追溯到网站的源服务器拉取副本，即回源。

5.源服务器更新资源，CDN怎么更新？

# Webview中H5缓存

项目是webview嵌入H5，这种情况缓存会存入APP的data目录

APP端webview原理是使用手机浏览器内核渲染H5，手机浏览器是对HTTP协议的标准实现，上述浏览器缓存同样适用APP端，另外，可能会出现其他缓存方式。

1.协议缓存，就是上面提到的强缓存和协商缓存

2.H5应用程序缓存机制，开发者需要提供一个cache manifest文件，这个文件列出了所有需要在离线状态下使用的资源，浏览器会把这些资源缓存到本地。

```
1  <!-- calender.html -->
2  <!DOCTYPE HTML>
3  <html manifest="calender.manifest">
4  <head>
5      <title>calender</title>
6      <script src="calender.js"></script>
7      <link rel="stylesheet" href="calender.css">
8  </head>
9  <body>
10     <p>The time is: <output id="calender"></output></p>
11 </body>
12 </html>
13
14 其对应的 calender.manifest代码
15  CACHE MANIFEST
16  calender.html
17  calender.css
18  calender.js
```

## H5的应用缓存

HTML5 提供一种应用程序缓存机制，使得基于web的应用程序可以离线运行。为了能够让用户在离线状态下继续访问 Web 应用，开发者需要提供一个 cache manifest 文件。这个文件中列出了所有需要在离线状态下使用的资源，浏览器会把这些资源缓存到本地。

```
1  CACHE MANIFEST
2  calender.html
3  calender.css
4  calender.js
```

## 缓存带来的问题

1.好的缓存机制能够吸收大部分的流量，不利于服务器统计访问量。



2.使用不当，可能出现服务端更新了资源，用户访问到的还是旧的资源。

## 缓存最佳实践

说了那么多，实际中怎么使用缓存呢？

根据项目架构来选择缓存策略。

1.静态资源，图片、三方库js、css等存放在CDN（假如有）

2.Vue项目只有一个HTML文件，其他文件都是在里面引用，为了保证用户能获取最新的文件，HTML采用协商缓存，JS、CSS等采用强缓存，并且更新的时候文件名带上hash值，保证能够及时得到更新。

- 缓存控制在nginx根据资源类型配置相关首部
- 若使用Etag，nginx配置即可
- 文件打包，webpack启用hash命名

```
1  server {
2      location ~* \.(html)$ {
3          access_log off;
4          add_header Cache-Control max-age=no-cache;
5      }
6
7      location ~* \.(css|js|png|jpg|jpeg|gif|gz|svg|mp4|ogg|ogv|webm|htc|xml|woff)$ {
8          # 同上，通配所有以.css/.js/...结尾的请求
9          access_log off;
10         add_header Cache-Control max-age=360000;
11     }
12 }
```

nginx除了通过配置响应首部做前端缓存，也可以通过proxy\_cache做后端缓存

总结：在做前端缓存时，我们尽可能设置长时间的强缓存，通过文件名加hash的方式来做版本更新。

参考：

《HTTP权威指南》

《CDN技术详解》

<https://juejin.cn/post/6844903737538920462>

<https://zhuanlan.zhihu.com/p/27456323>