03 | HTTP请求流程：为什么很多站点第二次打开速度会很快？

2019-08-10 李兵

《浏览器工作原理与实践》 课程介绍 >

讲述：李兵

时长 16:16 大小 14.91M

在上一篇文章中我介绍了 TCP 协议是如何保证数据完整传输的，相信你还记得，一个 TCP 连接过程包括了建立连接、传输数据和断开连接三个阶段。

而 HTTP 协议，正是建立在 TCP 连接基础之上的。**HTTP 是一种允许浏览器向服务器获取资源的协议，是 Web 的基础**，通常由浏览器发起请求，用来获取不同类型的文件，例如 HTML 文件、CSS 文件、JavaScript 文件、图片、视频等。此外，**HTTP 也是浏览器使用最广的协议**，所以要想学好浏览器，就要先深入了解 HTTP。

不知道你是否有过下面这些疑问：

1. 为什么通常在第一次访问一个站点时，打开速度很慢，当再次访问这个站点时，速度就很快了？
2. 当登录过一个网站之后，下次再访问该站点，就已经处于登录状态了，这是怎么做到的呢？

这一切的秘密都隐藏在 HTTP 的请求过程中。所以，在今天这篇文章中，我将通过分析一个 HTTP 请求过程中每一步的状态来带你了解完整的 HTTP 请求过程，希望你看完这篇文章后，能够对 HTTP 协议有个全新的认识。

浏览器端发起 HTTP 请求流程

如果你在浏览器地址栏里键入极客时间网站的地址：http://time.geekbang.org/index.html，那么接下来，浏览器会完成哪些动作呢？下面我们就一步一步详细“追踪”下。

1. 构建请求

首先，浏览器构建**请求行**信息（如下所示），构建好后，浏览器准备发起网络请求。

```
GET /index.html HTTP1.1
```

2. 查找缓存

在真正发起网络请求之前，浏览器会先在浏览器缓存中查询是否有要请求的文件。其中，**浏览器缓存是一种在本地保存资源副本，以供下次请求时直接使用的技术**。

当浏览器发现请求的资源已经在浏览器缓存中存有副本，它会拦截请求，返回该资源的副本，并直接结束请求，而不会再去源服务器重新下载。这样做的好处有：

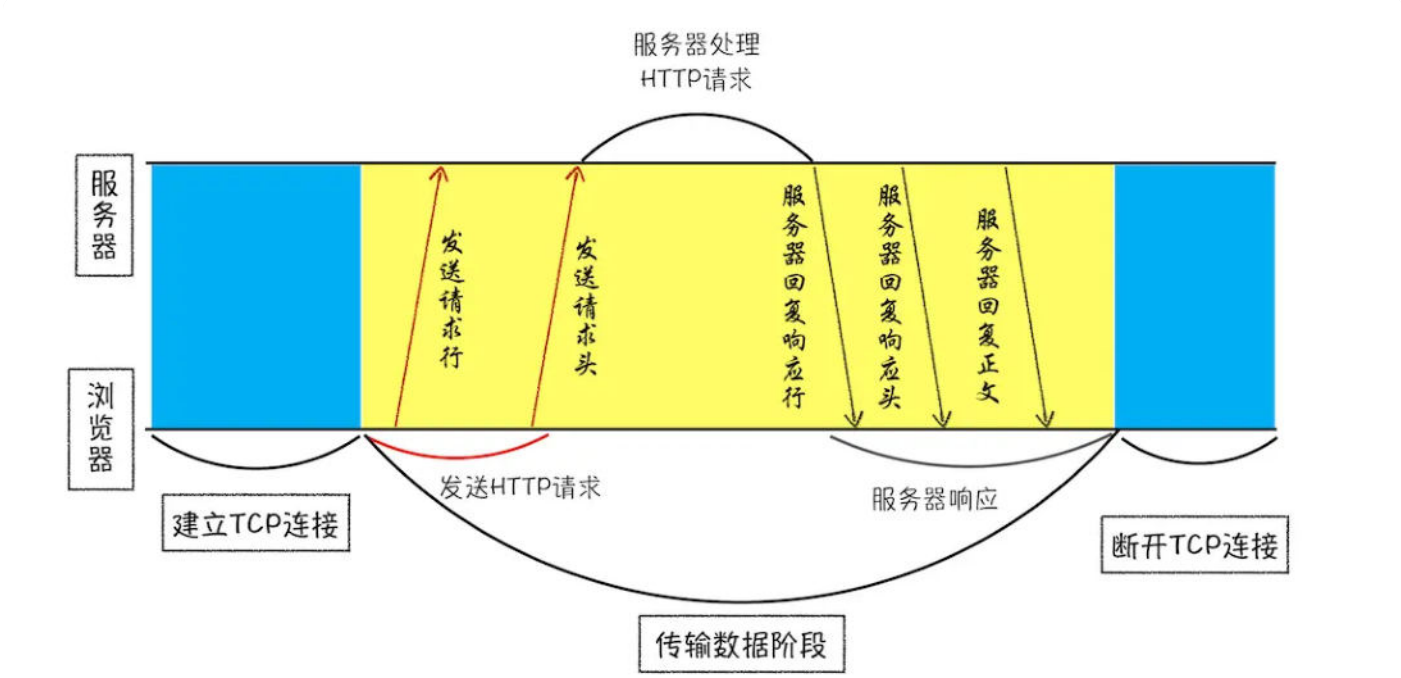
- 缓解服务器端压力，提升性能（获取资源的耗时更短了）；
- 对于网站来说，缓存是实现快速资源加载的重要组成部分。

当然，如果缓存查找失败，就会进入网络请求过程了。

3. 准备 IP 地址和端口

不过，先不急，在了解网络请求之前，我们需要先看看 HTTP 和 TCP 的关系。因为浏览器使用 **HTTP 协议作为应用层协议**，用来封装请求的文本信息；并使用 **TCP/IP 作传输层协议**将它发到网络上，所以在 HTTP 工作开始之前，浏览器需要通过 TCP 与服务器建立连接。也就

是说 **HTTP 的内容是通过 TCP 的传输数据阶段来实现的**，你可以结合下图更好地理解这二者的关系。



TCP 和 HTTP 的关系示意图

那接下来你可以思考这么“一连串”问题：

- HTTP 网络请求的第一步是做什么呢？结合上图看，是和服务器建立 TCP 连接。
- 那建立连接的信息都有了吗？上一篇文章中，我们讲到建立 TCP 连接的第一步就是需要准备 IP 地址和端口号。
- 那怎么获取 IP 地址和端口号呢？这得看看我们现在有什么，我们有一个 URL 地址，那么是否可以利用 URL 地址来获取 IP 和端口信息呢？

在上一篇文章中，我们介绍过数据包都是通过 IP 地址传输给接收方的。由于 IP 地址是数字标识，比如极客时间网站的 IP 是 39.106.233.176，难以记忆，但使用极客时间的域名（time.geekbang.org）就好记多了，所以基于这个需求又出现了一个服务，负责把域名和 IP 地址做一一映射关系。这套域名映射为 IP 的系统就叫做“**域名系统**”，简称 **DNS**（Domain Name System）。

所以，这样一路推导下来，你会发现在**第一步浏览器会请求 DNS 返回域名对应的 IP**。当然浏览器还提供了 **DNS 数据缓存服务**，如果某个域名已经解析过了，那么浏览器会缓存解析的结

果，以供下次查询时直接使用，这样也会减少一次网络请求。

拿到 IP 之后，接下来就需要获取端口号了。通常情况下，如果 URL 没有特别指明端口号，那么 HTTP 协议默认是 80 端口。

4. 等待 TCP 队列

现在已经把端口和 IP 地址都准备好了，那么下一步是不是可以建立 TCP 连接了呢?

答案依然是“不行”。Chrome 有个机制，同一个域名同时最多只能建立 6 个 TCP 连接，如果在同一个域名下同时有 10 个请求发生，那么其中 4 个请求会进入排队等待状态，直至进行中的请求完成。

当然，如果当前请求数量少于 6，会直接进入下一步，建立 TCP 连接。

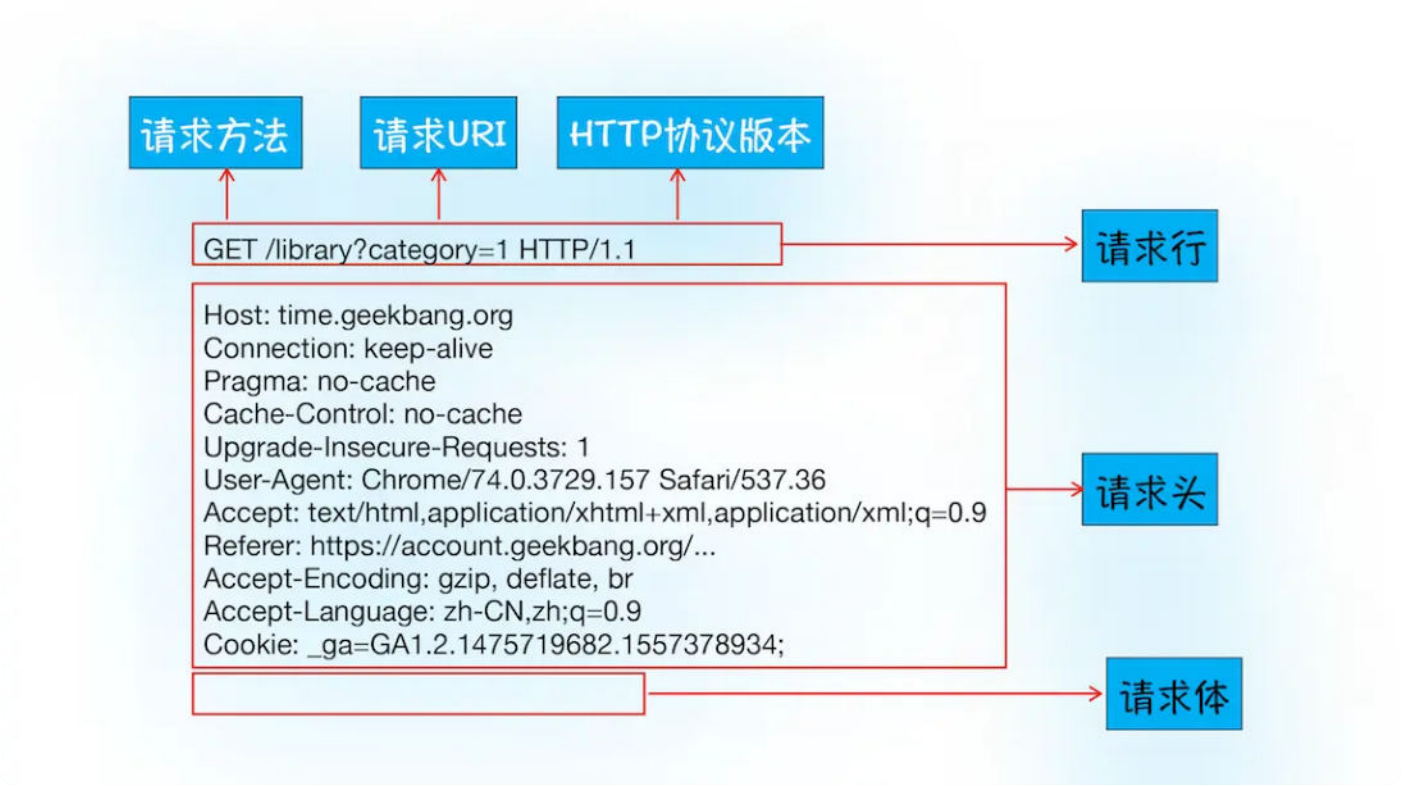
5. 建立 TCP 连接

排队等待结束之后，终于可以快乐地和服务器握手了，在 HTTP 工作开始之前，浏览器通过 TCP 与服务器建立连接。而 TCP 的工作方式，我在上一篇文章中已经做过详细介绍了，如果有必要，你可以自行回顾下，这里我就不再重复讲述了。

6. 发送 HTTP 请求

一旦建立了 TCP 连接，浏览器就可以和服务器进行通信了。而 HTTP 中的数据正是在这个通信过程中传输的。

你可以结合下图来理解，浏览器是如何发送请求信息给服务器的。



HTTP 请求数据格式

首先浏览器会向服务器发送**请求行**，它包括了**请求方法、请求 URI（Uniform Resource Identifier）和 HTTP 版本协议**。

发送请求行，就是告诉服务器浏览器需要什么资源，最常用的请求方法是 **Get**。比如，直接在浏览器地址栏键入极客时间的域名（time.geekbang.org），这就是告诉服务器要 Get 它的首页资源。

另外一个常用的请求方法是 **POST**，它用于发送一些数据给服务器，比如登录一个网站，就需要通过 POST 方法把用户信息发送给服务器。如果使用 POST 方法，那么浏览器还要准备数据给服务器，这里准备的数据是通过**请求体**来发送。

在浏览器发送请求行命令之后，还要以**请求头**形式发送其他一些信息，把浏览器的一些基础信息告诉服务器。比如包含了浏览器所使用的操作系统、浏览器内核等信息，以及当前请求的域名信息、浏览器端的 Cookie 信息，等等。

服务器端处理 HTTP 请求流程

历经千辛万苦，HTTP 的请求信息终于被送达了服务器。接下来，服务器会根据浏览器的请求信息来准备相应的内容。

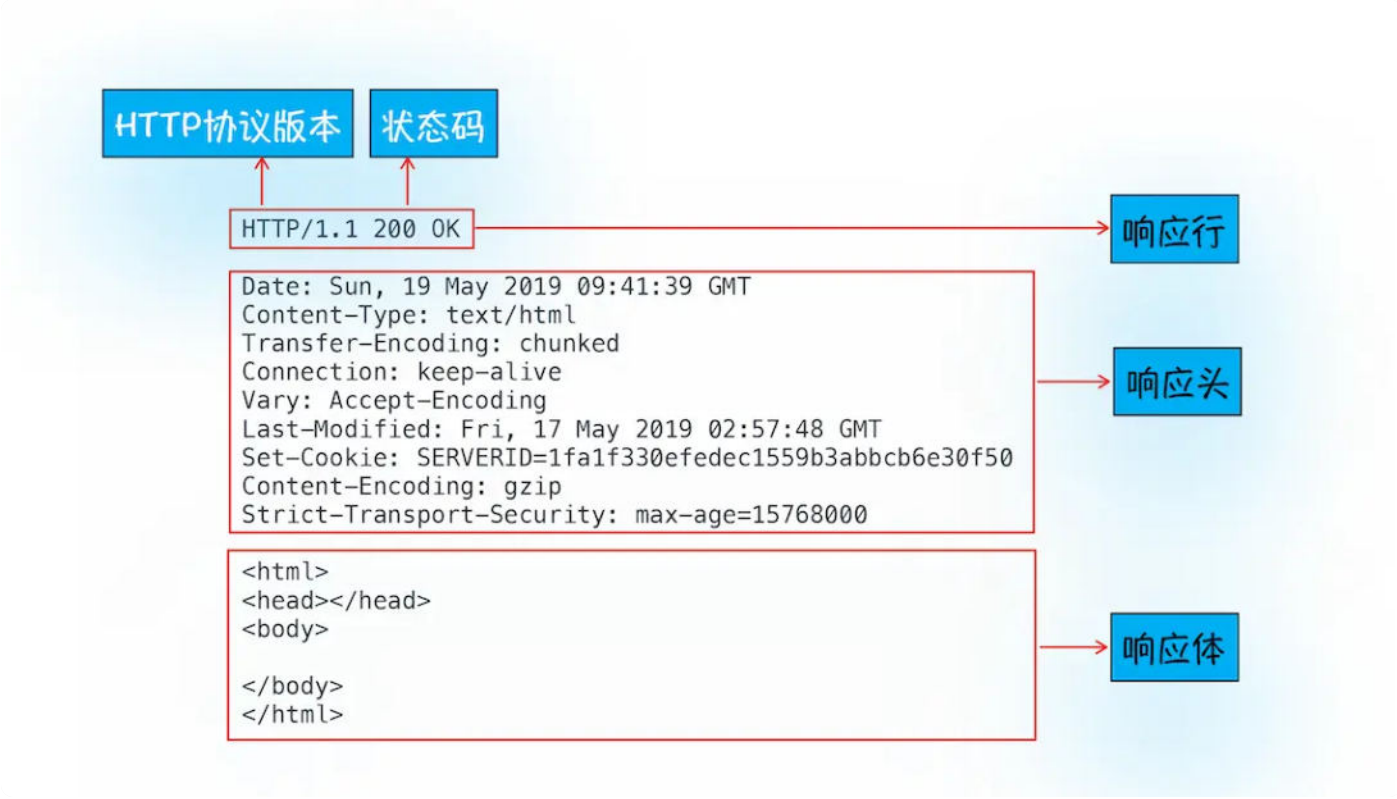
1. 返回请求

一旦服务器处理结束，便可以返回数据给浏览器了。你可以通过工具软件 curl 来查看返回请求数据，具体使用方法是在命令行中输入以下命令：

```
curl -i  https://time.geekbang.org/
```

复制代码

注意这里加上了-i是为了返回响应行、响应头和响应体的数据，返回的结果如下图所示，你可以结合这些数据来理解服务器是如何响应浏览器的。



服务器响应的数据格式

首先服务器会返回**响应行**，包括协议版本和状态码。

但并不是所有的请求都可以被服务器处理的，那么一些无法处理或者处理出错的信息，怎么办呢？服务器会通过请求行的**状态码**来告诉浏览器它的处理结果，比如：

- 最常用的状态码是 200，表示处理成功；
- 如果没有找到页面，则会返回 404。

状态码类型很多，这里我就不过多介绍了，网上有很多资料，你可以自行查询和学习。

随后，正如浏览器会随同请求发送请求头一样，服务器也会随同响应向浏览器发送**响应头**。响应头包含了服务器自身的一些信息，比如服务器生成返回数据的时间、返回的数据类型（JSON、HTML、流媒体等类型），以及服务器要在客户端保存的 Cookie 等信息。

发送完响应头后，服务器就可以继续发送**响应体**的数据，通常，响应体就包含了 HTML 的实际内容。

以上这些就是服务器响应浏览器的具体过程。

2. 断开连接

通常情况下，一旦服务器向客户端返回了请求数据，它就要关闭 TCP 连接。不过如果浏览器或者服务器在其头信息中加入了：

```
Connection:Keep-Alive
```

那么 TCP 连接在发送后将仍然保持打开状态，这样浏览器就可以继续通过同一个 TCP 连接发送请求。**保持 TCP 连接可以省去下次请求时需要建立连接的时间，提升资源加载速度**。比如，一个 Web 页面中内嵌的图片就都来自同一个 Web 站点，如果初始化了一个持久连接，你就可以复用该连接，以请求其他资源，而不需要重新再建立新的 TCP 连接。

3. 重定向

到这里似乎请求流程快结束了，不过还有一种情况你需要了解下，比如当你在浏览器中打开 geekbang.org 后，你会发现最终打开的页面地址是 https://www.geekbang.org。

这两个 URL 之所以不一样，是因为涉及到了一个**重定向操作**。跟前面一样，你依然可以使用 curl 来查看下请求 geekbang.org 会返回什么内容?

在控制台输入如下命令：

```
curl -I geekbang.org
```

注意这里输入的参数是-I，和-i不一样，-I表示只需要获取响应头和响应行数据，而不需要获取响应体的数据，最终返回的数据如下图所示：

服务器返回响应行和响应头（含重定向格式）

从图中你可以看到，响应行返回的状态码是 301，状态 301 就是告诉浏览器，我需要重定向到另外一个网址，而需要重定向的网址正是包含在响应头的 Location 字段中，接下来，浏览器获取 Location 字段中的地址，并使用该地址重新导航，这就是一个完整重定向的执行流程。这也就解释了为什么输入的是 geekbang.org，最终打开的却是 https://www.geekbang.org 了。

不过也不要认为这种跳转是必然的。如果你打开 https://12306.cn，你会发现这个站点是打不开的。这是因为 12306 的服务器并没有处理跳转，所以必须要手动输入完整的 https://www.12306.cn 才能打开页面。

问题解答

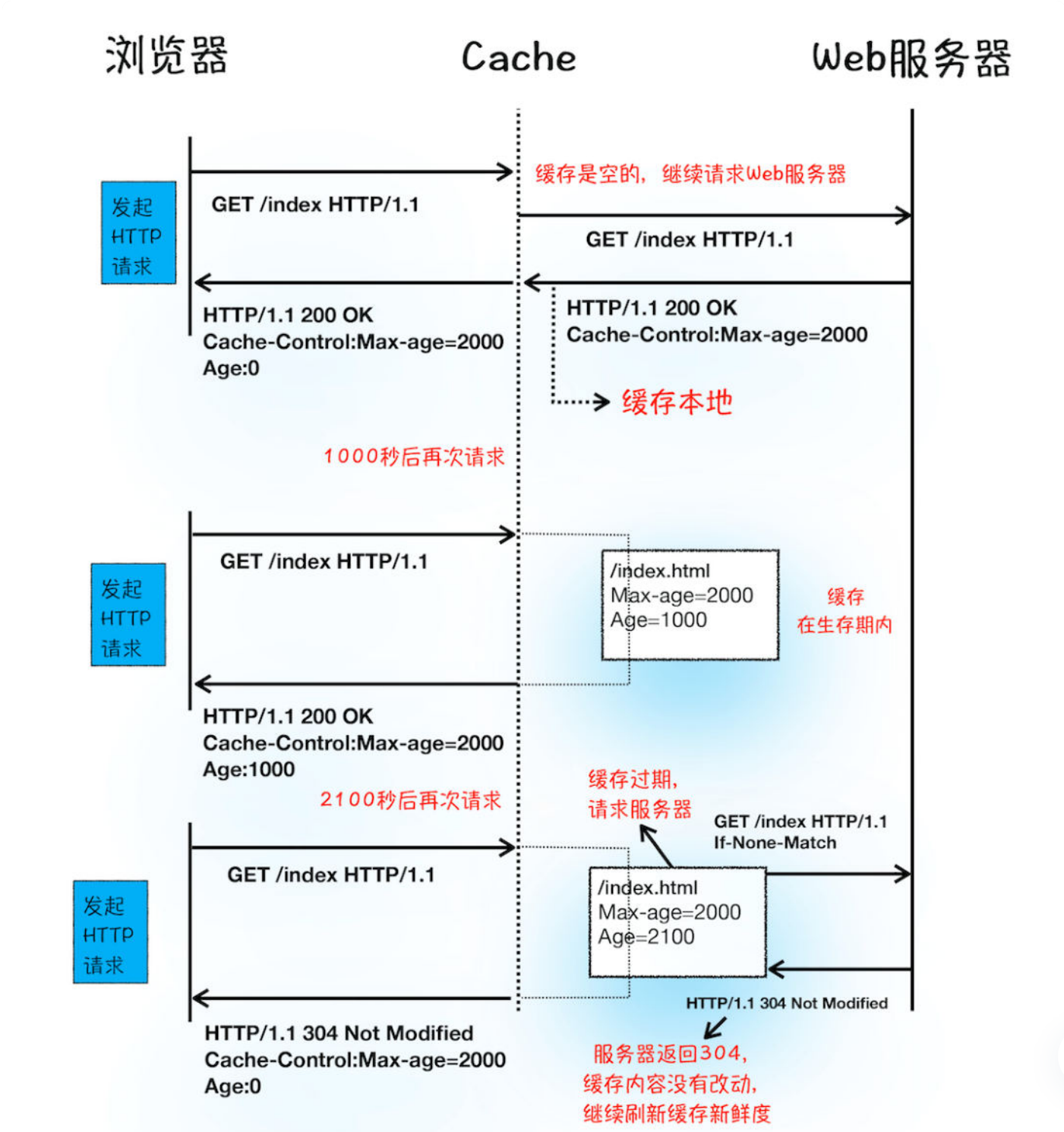
说了这么多，相信你现在已经了解了 HTTP 的请求流程，那现在我们再回过头来看看文章开头提出的问题。

1. 为什么很多站点第二次打开速度会很快？

如果第二次页面打开很快，主要原因是第一次加载页面过程中，缓存了一些耗时的数据。

那么，哪些数据会被缓存呢？从上面介绍的核心请求路径可以发现，**DNS 缓存**和**页面资源缓存**这两块数据是会被浏览器缓存的。其中，DNS 缓存比较简单，它主要就是在浏览器本地把对应的 IP 和域名关联起来，这里就不做过多分析了。

我们重点看下浏览器资源缓存，下面是缓存处理的过程：



缓存查找流程示意图

首先，我们看下服务器是通过什么方式让浏览器缓存数据的?

从上图的第一次请求可以看出，当服务器返回 **HTTP 响应头**给浏览器时，浏览器是**通过响应头中的 Cache-Control 字段来设置是否缓存该资源**。通常，我们还需要为这个资源设置一个缓存过期时长，而这个时长是通过 Cache-Control 中的 Max-age 参数来设置的，比如上图设置的缓存过期时间是 2000 秒。

```
Cache-Control:Max-age=2000
```

这也就意味着，在该缓存资源还未过期的情况下, 如果再次请求该资源，会直接返回缓存中的资源给浏览器。

但如果缓存过期了，浏览器则会继续发起网络请求，并且在 **HTTP 请求头**中带上：

```
If-None-Match:"4f80f-13c-3a1xb12a"
```

服务器收到请求头后，会根据 If-None-Match 的值来判断请求的资源是否有更新。

- 如果没有更新，就返回 304 状态码，相当于服务器告诉浏览器：“这个缓存可以继续使用，这次就不重复发送数据给你了。”
- 如果资源有更新，服务器就直接返回最新资源给浏览器。

关于缓存的细节内容特别多，具体细节你可以参考这篇 HTTP 缓存，在这里我就不赘述了。

简要来说，很多网站第二次访问能够秒开，是因为这些网站把很多资源都缓存在了本地，浏览器缓存直接使用本地副本来回应请求，而不会产生真实的网络请求，从而节省了时间。同时，DNS 数据也被浏览器缓存了，这又省去了 DNS 查询环节。

2. 登录状态是如何保持的?

通过上面的介绍，你已经了解了缓存是如何工作的。下面我们再一起看下登录状态是如何保持的。

- 用户打开登录页面，在登录框里填入用户名和密码，点击确定按钮。点击按钮会触发页面脚本生成用户登录信息，然后调用 POST 方法提交用户登录信息给服务器。
- 服务器接收到浏览器提交的信息之后，查询后台，验证用户登录信息是否正确，如果正确的话，会生成一段表示用户身份的字符串，并把该字符串写到响应头的 Set-Cookie 字段里，如下所示，然后把响应头发送给浏览器。

```
Set-Cookie: UID=3431uad;
```

- 浏览器在接收到服务器的响应头后，开始解析响应头，如果遇到响应头里含有 Set-Cookie 字段的情况，浏览器就会把这个字段信息保存到本地。比如把`UID=3431uad`保持到本地。
- 当用户再次访问时，浏览器会发起 HTTP 请求，但在发起请求之前，浏览器会读取之前保存的 Cookie 数据，并把数据写进请求头里的 Cookie 字段里（如下所示），然后浏览器再将请求头发送给服务器。

```
Cookie: UID=3431uad;
```

- 服务器在收到 HTTP 请求头数据之后，就会查找请求头里面的“Cookie”字段信息，当查找到包含`UID=3431uad`的信息时，服务器查询后台，并判断该用户是已登录状态，然后生成含有该用户信息的页面数据，并把生成的数据发送给浏览器。
- 浏览器在接收到该含有当前用户的页面数据后，就可以正确展示用户登录的状态信息了。

好了，通过这个流程你可以知道浏览器页面状态是通过使用 Cookie 来实现的。Cookie 流程可以参考下图：

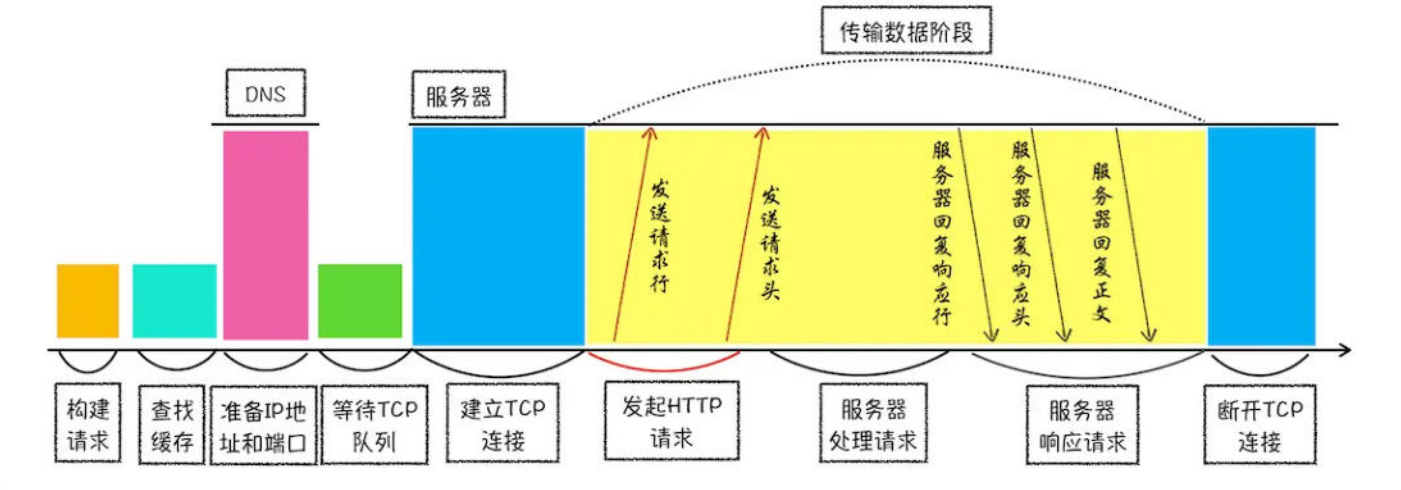
Cookie 流程图

简单地说，如果服务器端发送的响应头内有 Set-Cookie 的字段，那么浏览器就会将该字段的内容保持到本地。当下次客户端再往该服务器发送请求时，客户端会自动在请求头中加入 Cookie 值后再发送出去。服务器端发现客户端发送过来的 Cookie 后，会去检查究竟是从哪一个客户端发来的连接请求，然后对比服务器上的记录，最后得到该用户的状态信息。

总结

本篇文章的内容比较多、比较碎，但是非常重要，所以我先来总结下今天的主要内容。

为了便于你理解，我画了下面这张详细的“HTTP 请求示意图”，用来展现浏览器中的 HTTP 请求所经历的各个阶段。



HTTP 请求流程示意图

从图中可以看到，浏览器中的 HTTP 请求从发起到结束一共经历了如下八个阶段：构建请求、查找缓存、准备 IP 和端口、等待 TCP 队列、建立 TCP 连接、发起 HTTP 请求、服务器处理请求、服务器返回请求和断开连接。

然后我还通过 HTTP 请求路径解答了两个经常会碰到的问题，一个涉及到了 Cache 流程，另外一个涉及到如何使用 Cookie 来进行状态管理。

通过今天系统的讲解，想必你已经了解了一个 HTTP 完整的工作流程，相信这些知识点之于你以后的学习或工作会很有帮助。

另外，你应该也看出来了本篇文章是有很多分析问题的思路在里面的。所以在学习过程中，你也要学会提问，通过最终要做什么和现在有什么，去一步步分析并提出一些问题，让疑问带领着你去学习，抓住几个本质的问题就可以学透相关知识点，让你能站在更高维度去查看整体框架。希望它能成为你的一个学习技巧吧！

思考时间

最后，还是留给你个思考题：结合今天所讲 HTTP 请求的各个阶段，如果一个页面的网络加载时间过久，你是如何分析卡在哪个阶段的？

欢迎在留言区与我分享你的想法，也欢迎你在留言区记录你的思考过程。感谢阅读，如果你觉得这篇文章对你有帮助的话，也欢迎把它分享给更多的朋友。

上一篇　02 | TCP协议：如何保证页面文件能被完整送达浏览器?

下一篇　04 | 导航流程：从输入URL到页面展示，这中间发生了什么?

精选留言 (141)

写留言

mfist

2019-08-10

1 首先猜测最可能的出问题的地方，网络传输丢包比较严重，需要不断重传。然后通过ping c url看看对应的时延高不高。

2 然后通过wireshake看看具体哪里出了问题。

3 假如别人访问很快，自己电脑很慢，就要看看自己客户端是否有问题了。

感觉平常碰到很多http问题，基本都能通过上面方式搞定

作者回复: 👍

共 7 条评论>

99

一步

2019-08-10

对于浏览器缓存地方的选择一直搞不明白其中的原理，在浏览器中访问的时候打开network面板，发现缓存的来源有的from disk有的是from memory。对于资源什么情况下缓存到硬盘什么时候缓存到内存，想请教一下老师

作者回复: 这是浏览器的三级缓存机制，使用memory cache比disk cache 的访问速度要快，但是具体什么规则等我回头看下源码再来回答你了。

还有另外一种cache，是service worker的cache。

共 25 条评论>

69

宇宙全栈

2019-08-16

老师好，请教下TCP连接如何断开的问题：

1、没有 keep-alive 时，http数据传输完成后，是由浏览器主动发起断开TCP连接，还是由服务器主动发起断开 TCP 连接?

2、设置了 keep-alive 时，当关闭页面时，浏览器会发起断开 TCP 连接吗？如果不关闭页面，浏览器会一直保持这个 TCP 连接吗?

3、设置了 keep-alive 时，如果浏览器出现故障时了（挂掉了），此时服务器保持的 TCP 连接多久会释放?

共 8 条评论>

58

Hurry

2019-08-13

使用 chrome network 面板，看那个 瀑布图 中每个阶段的含义，就可以判断问题出现在那个方向了，每个阶段的含义，https://developers.google.com/web/tools/chrome-devtools/network/reference#timing-explanation 举个例子 Content Download 如果太长，很有可能是下载的资源太大，但也有可能是网络慢导致的下载太慢，简单计算一下，在例如 Waiting (TTFB) 这个太长的话，有可能是网络慢，或者就是 后端处理时间过长导致的，至少可以排查掉前端原因，还有很多，例如 DNS lookup 等，但是最终要确认具体哪里慢，最好是结合系统日志去分析

作者回复: 结合网络模块来分析，总结的挺好

共 5 条评论>

42

XWL

2019-08-12

浏览器刷新操作，ctrl+F5和F5有什么区别

作者回复: 一个是强制刷新，也就是资源都走网络。

一个是正常处理流程。

比如通过网络面板，打开一个站点，再使用强制刷新，可以看到如下信息
176 requests
3.1 MB transferred
3.5 MB resources
Finish: 26.30 s
DOMContentLoaded: 5.04 s
Load: 14.88 s

如果使用正常的刷新，看到的信息如下：
171 requests
419 KB transferred
3.2 MB resources
Finish: 25.09 s
DOMContentLoaded: 1.41 s
Load: 6.24 s

其中的transferred是真正的网络传输的数据，使用强制刷新，传输的数据体积就大多了，而且请求时间也变得更长了。

共 4 条评论>

36

Shopee内推码：NTAGx...

2019-08-12

set cookie 会不会有安全问题，麻烦老师指导下

作者回复: cookie是不安全的，比如黑客可以通过一些手段插入一些脚本到你的页面里面（具体一些途径我们浏览器安全篇再讲），通过脚本获取到你的cookie数据，然后就可以利用cookie做一些坏事了。

当然也有一些方法规避，常用的一个是将部分cookie设置成httponly的属性，设置了httponly属性后cookie，就无法通过js脚本来读取了，只是在发送http请求时候会被带上！

当然还有一些其他防范的方法，这个同样咱们后面在聊！

共 3 条评论>

35

月隐千山

2019-08-10

老师，在做前端页面的时候，是否可以设置当前页面是否可以被缓存，以及哪些部分可以被缓存？还是说整个缓存机制都是由浏览器自己控制的？

作者回复: 是没有办法通过前端代码来控制缓存的，缓存是后端或者部署的同学来控制的，但是前端同学应该知道那些内容要被缓存，和后端或者部署的同学配合来打！

共 9 条评论>

32

CMS

2019-11-07

Chrome 有个机制，同一个域名同时最多只能建立 6 个 TCP 连接，如果在同一个域名下同时有 10 个请求发生，那么其中 4 个请求会进入排队等待状态，直至进行中的请求完成。 是指同一个域名下的6个并发请求么。我理解建立一个tcp连接，可以处理多个请求吧？
这一块蒙了，看好多人问这个

作者回复: http/1.1 一个tcp同时只能处理一个请求，浏览器会为每个域名维护6个tcp连接！

但是每个tcp连接是可以复用的，也就是处理完一个请求之后，不断开这个tcp连接，可以用来处理下个http请求！

不过http2是可以并行请求资源的，所以如果使用http2，浏览器只会为每个域名维护一个tcp连接

共 8 条评论>

30

Geek_0d3179

2019-11-03

老师您好~有个问题困扰我了。希望您能解答我，十分感谢。http2同个域名只能维持一个长连接。那我现在打开了一个域名下的a页面，然后又打开了这个域名的b页面，那这个b页面是新开一个tcp长连接吗？还是会用a页面的长连接？换句话说，维持一个tcp长连接，指的是一个页面维持一个？还是整个浏览器维持一个？还是同一个渲染进程维持一个？

作者回复: 浏览器为用同一个域名只维护一个TCP连接。

你从通过Chrome打开chrome://net-export/ 这个地址，然后记录网络过程。

最后在使用这个https://netlog-viewer.appspot.com/ 打开你日志文件，就能看到h2的详细信息了。

共 2 条评论>

25

liu_xm

2019-08-10

同域名只能建立6个tcp链接的话，那加载大量图片或者其他资源的时候不是很卡呢?

作者回复: 是的，通常如图片这种静态资源都是直接配置到cdn上的

共 11 条评论>

23

houqx

2019-08-23

问一个今天遇到的跟本篇文章没有太大关系的问题：
问题描述：前端有一按钮，点击可以签到(比如调用的接口名为doSign)，签到成功后置灰，不能再点击。但是看服务端日志存在同一时间同一用户该接口的多次调用(通常为3次，间隔10ms以内)。
我想问的问题是：
1：这里的重复调用是哪里引入的(用户手抖多次点击？)
2：浏览器在发出一个网络请求后，当这个请求还没返回的时候，会再重复发起相同的请求吗？(用户快速点击)

作者回复: 用户快速点击也有可能的，通常，前端要处理防止多次点击的情况。

共 8 条评论>

15

nissan

2019-08-10

请问老师，使用者的资讯（如UID=3431uad）是放在cookie还是localstorage中好呢？我的理解是存不存cookie是后端决定（要求set-cookie)存localstorage是前端程式控制的。是这样吗?

作者回复: 是否要使用localstorage，还是要看具体应用场景。其实使用cookie会很方便，因为它会随着http请求头把cookie内容发送服务器，用localstorage需要重新实现数据上传和下载。

15

peter
2019-08-12

请问：请求行和请求头是发送两次吗？ 从文章的文字来看，是发送两次。 但我感觉是发送一次，即发送一次请求，该请求包含请求行和请求头。

作者回复: 对，只发送一次。

问下文章什么地方让你感觉是发送了两次啊?

我检查下。

共 17 条评论>

13

Louis
2019-08-13

请问老师http的keep alive和http2中的信道服用有什么区别呢

作者回复: 一个http中的keep-alive是排队请求，也就是一个http请求完成之后才能继续请求下一个，而http2中请求是并发的，可以同时处理很多请求!

10

aaron
2019-08-10

老师，请问https为什么能防止网络劫持?

作者回复: http在传输过程中是明文的，所以数据在传输过程中是能够被截获或者修改的，比如谁在你电脑上安装了一个网络拦截软件，或者你的路由器被谁安装了监听软件，甚至在网络服务提供商都有可能修改你页面的内容，基于这些原因，我们需要在传输过程中加密数据，这就是https出现的原因，即便你拦截到了请求，获取的只是加密后的数据，拿到也没有什么用。

这块在浏览器安全篇会系统介绍。

共 3 条评论>

10

weineel
2019-08-29

老是你好，有个疑问：在我理解中的 TCP 连接需要四元组(client ip, client port, server ip, server port)保证连接的唯一性，那浏览器是怎么对同一服务器(server ip, server port 不变)创

建多个 TCP 连接的?

浏览器只有一个网络进程且 client ip 不变，是不是在创建连接的时候，会监听多个 client port 以保证4元组的唯一?

共 4 条评论 >　　8

行云流水
2019-08-22

1.按照老师说的,至少在浏览器中 http是在 tcp的传输阶段。不是每一层独立完成后交给下一层。OSI七层模型只是一个参考,浏览器只是参考OSI模型实现,不一定遵守OSI的层级关系?
2. 上一节有一个同学提问说的，针对数据包实时渲染问题。老师说在TCP层就解决了丢包和顺序问题。但现在HTTP是在TCP层的传输层中使用。不明白怎么解决丢包和顺序问题?

共 1 条评论 >　　7

stanlee
2019-10-25

老师，同一个域名同时最多只能建立 6 个 TCP 连接 是不是意思是统一域名同时只能发送6个AJAX请求吗， TCP连接和AJAX请求有什么关系吗

> 作者回复: 首先回答第一个问题：
> ”同一个域名同时最多只能建立 6 个 TCP 连接“ 指的不光是指Ajax，还包括页面中的资源加载，只要是一个域名下的资源，浏览器同一时刻最多只支持6个并行请求。
>
> 不过这是HTTP/1.1的规则，HTTP/2已经不用这套规则了，而且HTTP/2也很成熟了，有条件可以考虑切换到HTTP/2.
>
> Ajax其实就是HTTP请求，包括了XMLHttpRequest和Fetch，HTTP请求是建立在TCP协议之上的。

共 3 条评论 >　　6

houqx
2019-08-23

不接CDN与接入CDN，DNS解析过程一样吗? 知道 A记录与cname，但是这里串不起来，还希望老师讲解一下

共 1 条评论 >　　5

sunshineling
2019-08-12

老师，你说chrome一次最多只能建立6个tcp连接，有点不理解，这是说只能支持6个用户并发吗?

作者回复: 是同一个域名下面，同一时间只能有6个并发请求，超过六个以上的需要排队！

共 5 条评论 >

5