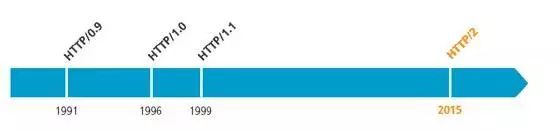
Http协议，对于拥有丰富开发经验的程序员来说简直是信手拈来，家常便饭。虽然天天见，但是对于http协议的问题，可能很多人在没有积极准备的情况下，不一定能很好的回答出来。

今天，我们就来讲讲http 2.0和之前版本的区别。

# 一、HTTP定义

HTTP协议（HyperTextTransferProtocol，超文本传输协议）是用于从WWW服务器传输超文本到本地浏览器的传输协议。

# HTTP发展史



# 三、HTTP 2.0 vs 1.0 性能能

HTTP 2.0 的出现，相比于 HTTP 1.x ，大幅度的提升了 web 性能。

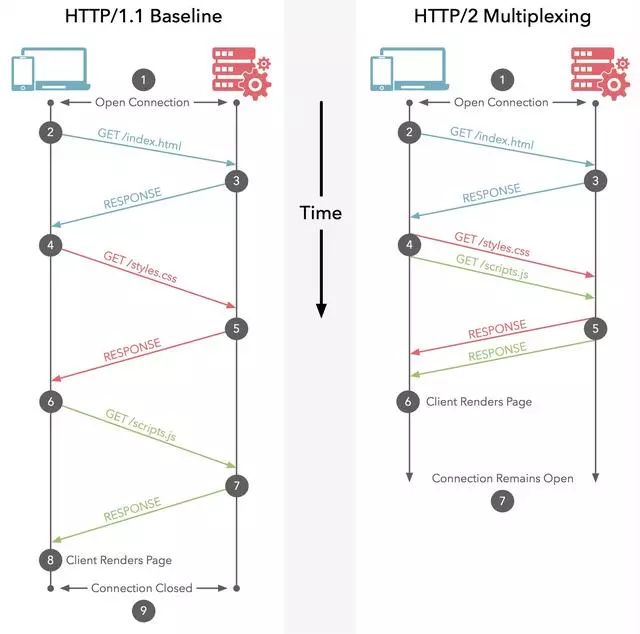


这是 Akamai 公司建立的一个官方的演示，用以说明 HTTP/2 相比于之前的 HTTP/1.1 在性能上的大幅度提升。 同时请求 379 张图片，从Load time 的对比可以看出 HTTP/2 在速度上的优势。

# 四、HTTP 2.0 和 1.1 区别别

后面我们将通过几个方面来说说HTTP 2.0 和 HTTP1.1 区别，并且和你解释下其中的原理。

## 区别一：多路复用



多路复用允许单一的 HTTP/2 连接同时发起多重的请求-响应消息。看个例子：

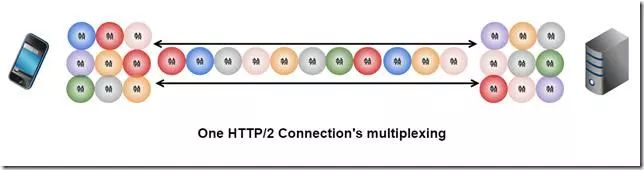
整个访问流程第一次请求index.html页面,之后浏览器会去请求style.css和scripts.js的文件。左边的图是顺序加载两个个文件的，右边则是并行加载两个文件。

我们知道HTTP底层其实依赖的是TCP协议，那问题是在同一个连接里面同时发生两个请求响应着是怎么做到的？

首先你要知道，TCP连接相当于两根管道（一个用于服务器到客户端，一个用于客户端到服务器），管道里面数据传输是通过字节码传输，传输是有序的，每个字节都是一个一个来传输。

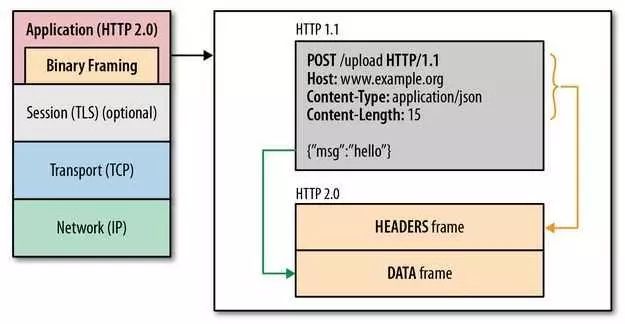
例如客户端要向服务器发送Hello、World两个单词，只能是先发送Hello再发送World，没办法同时发送这两个单词。不然服务器收到的可能就是HWeolrllod（注意是穿插着发过去了，但是顺序还是不会乱）。这样服务器就懵b了。

接上面的问题，能否同时发送Hello和World两个单词能，当然也是可以的，可以将数据拆成包，给每个包打上标签。发的时候是这样的①H ②W ①e ②o ①l ②r ①l ②l ①o ②d。这样到了服务器，服务器根据标签把两个单词区分开来。实际的发送效果如下图：



要实现上面的效果我们引入一个新的概念就是：二进制分帧。

二进制分帧层 在 应用层(HTTP/2)和传输层(TCP or UDP)之间。HTTP/2并没有去修改TCP协议而是尽可能的利用TCP的特性。

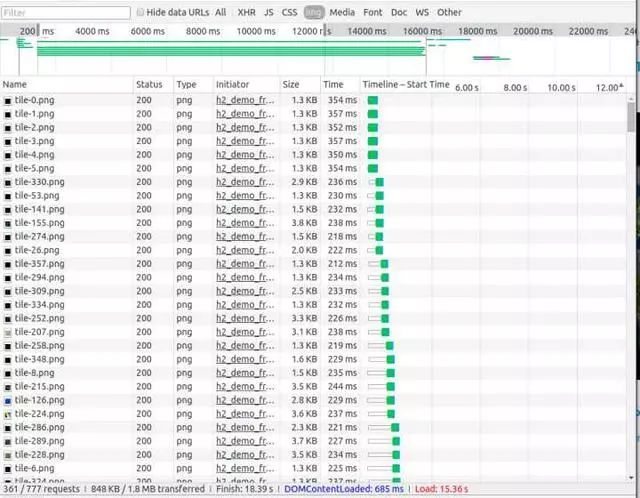


在二进制分帧层中， HTTP/2 会将所有传输的信息分割为帧（frame）,并对它们采用二进制格式的编码 ，其中 首部信息会被封装到 HEADER frame，而相应的 Request Body 则封装到 DATA frame 里面。掌握这 11 个方法论，搞定一场完美技术面试！

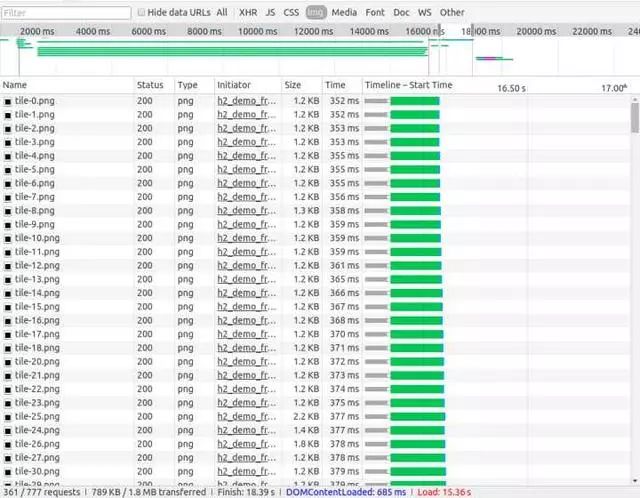
HTTP 性能优化的关键并不在于高带宽，而是低延迟。TCP 连接会随着时间进行自我「调谐」，起初会限制连接的最大速度，如果数据成功传输，会随着时间的推移提高传输的速度。这种调谐则被称为 TCP 慢启动。由于这种原因，让原本就具有突发性和短时性的 HTTP 连接变的十分低效。

HTTP/2 通过让所有数据流共用同一个连接，可以更有效地使用 TCP 连接，让高带宽也能真正的服务于 HTTP 的性能提升。

通过下面两张图，我们可以更加深入的认识多路复用：



HTTP/1



HTTP/2

总结下：多路复用技术：单连接多资源的方式，减少服务端的链接压力,内存占用更少,连接吞吐量更大；由于减少TCP 慢启动时间，提高传输的速度

## 区别二：首部压缩

为什么要压缩？在 HTTP/1 中，HTTP 请求和响应都是由「状态行、请求 / 响应头部、消息主体」三部分组成。一般而言，消息主体都会经过 gzip 压缩，或者本身传输的就是压缩过后的二进制文件（例如图片、音频），但状态行和头部却没有经过任何压缩，直接以纯文本传输。

随着 Web 功能越来越复杂，每个页面产生的请求数也越来越多，导致消耗在头部的流量越来越多，尤其是每次都要传输 UserAgent、Cookie 这类不会频繁变动的内容，完全是一种浪费。掌握这 11 个方法论，搞定一场完美技术面试！

我们再用通俗的语言解释下，压缩的原理。头部压缩需要在支持 HTTP/2 的浏览器和服务端之间。

维护一份相同的静态字典（Static Table），包含常见的头部名称，以及特别常见的头部名称与值的组合；

维护一份相同的动态字典（Dynamic Table），可以动态的添加内容；

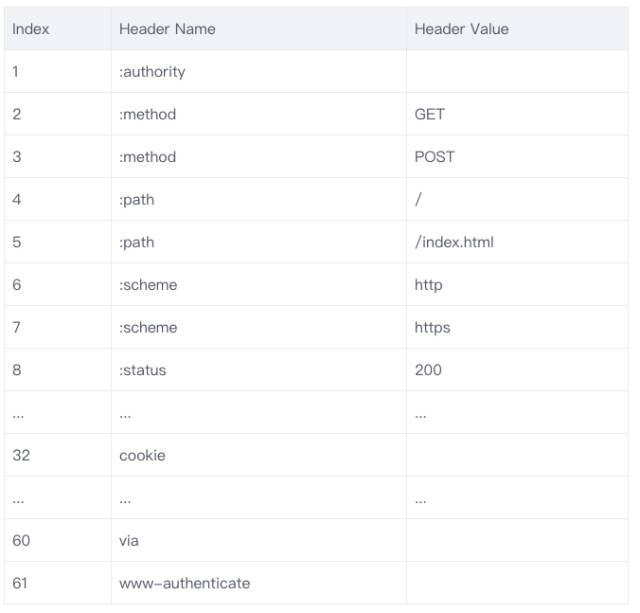
支持基于静态哈夫曼码表的哈夫曼编码（Huffman Coding）；

静态字典的作用有两个：

1）对于完全匹配的头部键值对，例如 “:method :GET”，可以直接使用一个字符表示；

2）对于头部名称可以匹配的键值对，例如 “cookie :xxxxxxx”，可以将名称使用一个字符表示。

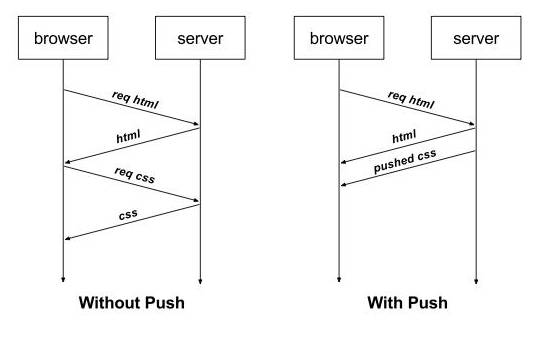
HTTP/2 中的静态字典如下（以下只截取了部分，完整表格在这里）：



同时，浏览器和服务端都可以向动态字典中添加键值对，之后这个键值对就可以使用一个字符表示了。需要注意的是，动态字典上下文有关，需要为每个 HTTP/2 连接维护不同的字典。在传输过程中使用，使用字符代替键值对大大减少传输的数据量。

## 区别三：HTTP2支持服务器推送

服务端推送是一种在客户端请求之前发送数据的机制。当代网页使用了许多资源:HTML、样式表、脚本、图片等等。在HTTP/1.x中这些资源每一个都必须明确地请求。这可能是一个很慢的过程。浏览器从获取HTML开始，然后在它解析和评估页面的时候，增量地获取更多的资源。因为服务器必须等待浏览器做每一个请求，网络经常是空闲的和未充分使用的。



为了改善延迟，HTTP/2引入了server push，它允许服务端推送资源给浏览器，在浏览器明确地请求之前。一个服务器经常知道一个页面需要很多附加资源，在它响应浏览器第一个请求的时候，可以开始推送这些资源。这允许服务端去完全充分地利用一个可能空闲的网络，改善页面加载时间。