# HTTP协议

把专栏的学习由“被动地听”，转变为“主动地学”，实现“学以致用

URI：即统一资源标识符，作为互联网上资源的唯一身份；

HTML：即超文本标记语言，描述超文本文档；

HTTP：即超文本传输协议，用来传输超文本。

<https://www.ruanyifeng.com/blog/2016/08/http.html>

# 历史[推动HTTP发展的原动力：用户需求推动技术发展]

* HTTP 协议始于三十年前蒂姆·伯纳斯 - 李的一篇论文；
* HTTP/0.9 是个简单的文本协议，只能获取文本资源；
* HTTP/1.0 确立了大部分现在使用的技术，但它不是正式标准；
* HTTP/1.1 是目前互联网上使用最广泛的协议，功能也非常完善；
* HTTP/2 基于 Google 的 SPDY 协议，注重性能改善，但还未普及；HTTP/3 基于 Google 的 QUIC 协议，是将来的发展方向。

HTTP/0.9

* 只允许用“GET”
* 服务器只能回应HTML格式的字符串，不能回应别的格式。
* 服务器发送完毕，就关闭TCP连接。

HTTP/1.0

* 增加了 HEAD、POST 等新方法；
* 增加了响应状态码，标记可能的错误原因；
* 引入了协议版本号概念；
* 引入了 HTTP Header（头部）的概念，
* 让 HTTP 处理请求和响应更加灵活；传输的数据不再仅限于文本。

HTTP /1.1

* 增加了 PUT、DELETE 等新的方法；增加了缓存管理和控制；
* 明确了连接管理，允许持久连接；
* 允许响应数据分块（chunked），利于传输大文件；
* 强制要求 Host 头，让互联网主机托管成为可能。

HTTP/2.0

* 二进制协议，不再是纯文本；
* 可发起多个请求，废弃了 1.1 里的管道；
* 使用专用算法压缩头部，减少数据传输量；
* 允许服务器主动向客户端推送数据；
* 增强了安全性，“事实上”要求加密通信。

HTTP/3

Google 又发明了一个新的协议，叫做 QUIC

# HTTP是什么

**定义：HTTP 是一个在计算机世界里专门在两点之间传输文字、图片、音频、视频等超文本数据的约定和规范”。**

**超文本**：超文本”，就是“超越了普通文本的文本”，它是文字、图片、音频和视频等的混合体，最关键的是含有“超链接”，能够从一个“超文本”跳跃到另一个“超文本”，形成复杂的非线性、网状的结构关系。HTML 了，它本身只是纯文字文件，但内部用很多标签定义了对图片、音频、视频等的链接，再经过浏览器的解释，呈现在我们面前的就是一个含有多种视听信息的页面。

传输 ：

* HTTP 协议是一个“双向协议”。就是把一堆东西从 A 点搬到 B 点，或者从 B 点搬到 A 点，即“A<===>B
* 数据虽然是在 A 和 B 之间传输，但并没有限制只有 A 和 B 这两个角色，允许中间有“中转”或者“接力”。传输方式就从“A<===>B”，变成了“A<=>X<=>Y<=>Z<=>B

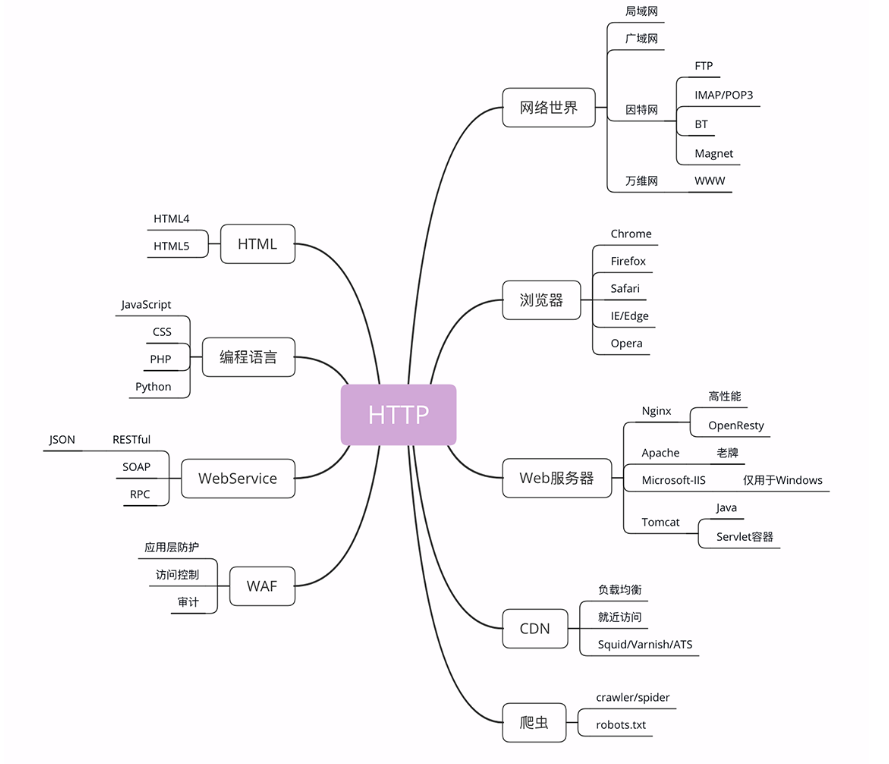
协议（多个参与者之间的约定）

* HTTP 是一个用在计算机世界里的协议，它确立了一种计算机之间交流通信的规范，以及相关的各种控制和错误处理方式。
* HTTP 专门用来在两点之间传输数据，不能用于广播、寻址或路由。
* HTTP 传输的是文字、图片、音频、视频等超文本数据。
* HTTP 是构建互联网的重要基础技术，它没有实体，依赖许多其他的技术来实现，但同时许多技术也都依赖于它。

【Question】HTTP不是什么？

* HTTP 不是互联网。
* HTTP 不是编程语言。
* HTTP 不是 HTML（HTML 是超文本的载体，是一种标记语言）
* HTTP 不是一个孤立的协议。
  + 在互联网世界里，HTTP 通常跑在 **TCP/IP** 协议栈之上，
  + 依靠 **IP** 协议实现寻址和路由、
  + TCP 协议实现可靠数据传输、
  + DNS 协议实现域名查找、
  + SSL/TLS 协议实现安全通信。
  + 此外，还有一些协议依赖于 HTTP，例如 WebSocket、HTTPDNS 等。
  + 这些协议相互交织，构成了一个协议网，而 HTTP 则处于中心地位。

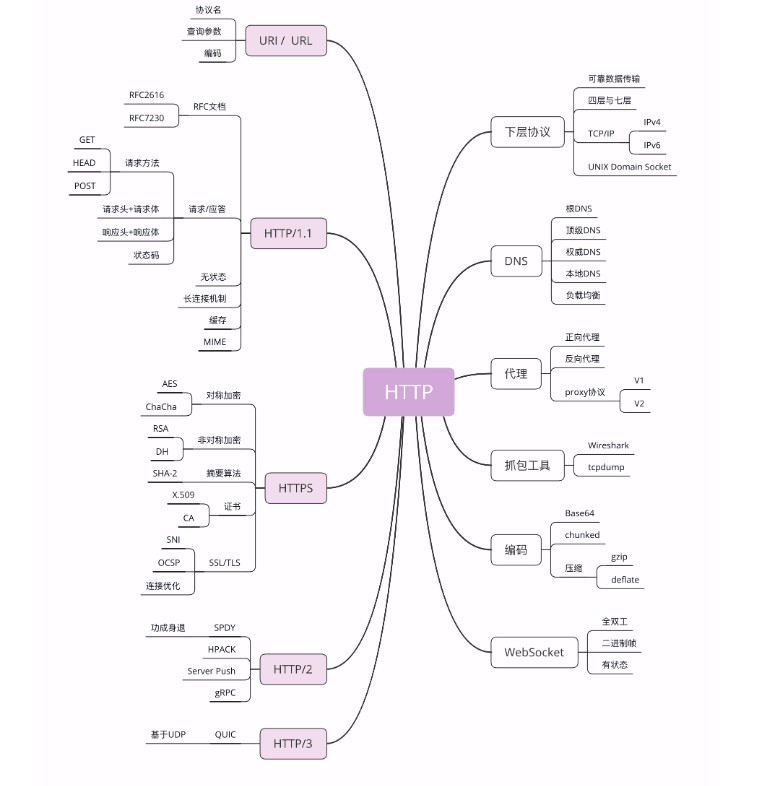
# HTTP相关概念



* 互联网上绝大部分资源都使用 HTTP 协议传输；
* 浏览器是 HTTP 协议里的请求方，即 User Agent；
* 服务器是 HTTP 协议里的应答方，常用的有 Apache 和 Nginx；
* CDN 位于浏览器和服务器之间，主要起到缓存加速的作用；

CDN，全称是“Content Delivery Network”，翻译过来就是“内容分发网络”。它应用了 HTTP 协议里的缓存和代理技术，代替源站响应客户端的请求。

* 爬虫是另一类 User Agent，是自动访问网络资源的程序。



* TCP/IP 是网络世界最常用的协议，HTTP 通常运行在 TCP/IP 提供的可靠传输基础上；
  + 四层模型：应用层、传输层、网际层、链接层
  + TCP/IP 协议实际上是一系列网络通信协议的统称
  + IP 协议是“Internet Protocol”的缩写，主要目的是解决寻址和路由问题，以及如何在两点间传送数据包。
* DNS 域名是 IP 地址的等价替代，需要用域名解析实现到 IP 地址的映射；URI 是用来标记互联网上资源的一个名字，由“协议名 + 主机名 + 路径”构成，俗称 URL；
  + 级别从左到右逐级升高
  + URI（Uniform Resource Identifier），中文名称是 统一资源标识符，使用它就能够唯一地标记互联网上资源
  + URL（Uniform Resource Locator）， 统一资源定位符，也就是我们俗称的“网址”，它实际上是 URI 的一个子集
  + <http://nginx.org/en/download.html>

你可以看到，URI 主要有三个基本的部分构成：

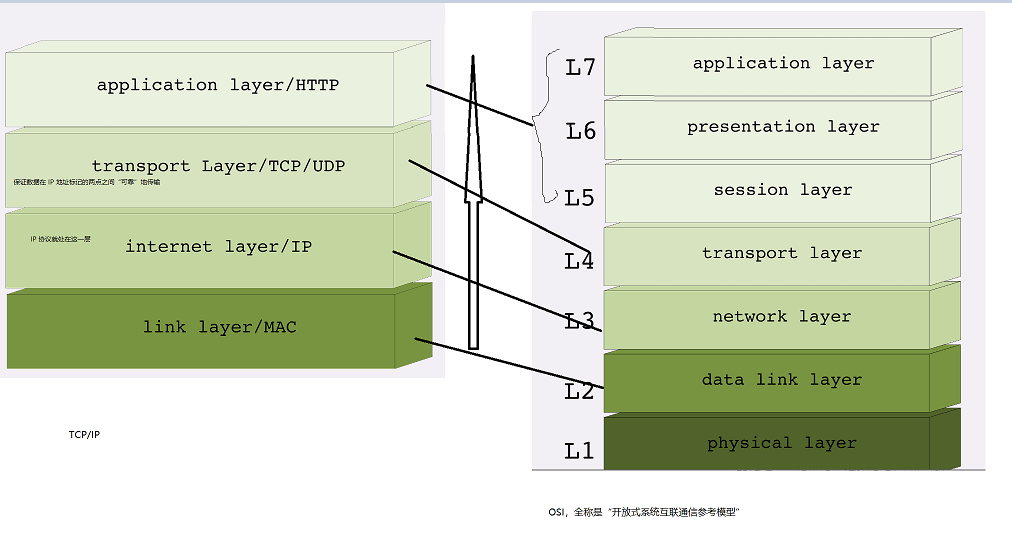
* + - 协议名：即访问该资源应当使用的协议，在这里是“http”；
    - 主机名：即互联网上主机的标记，可以是域名或 IP 地址，在这里是“nginx.org”；
    - 路径：即资源在主机上的位置，使用“/”分隔多级目录，在这里是“/en/download.html”。
* HTTPS 相当于“HTTP+SSL/TLS+TCP/IP”，为 HTTP 套了一个安全的外壳；
  + SSL/TLS它是一个负责加密通信的安全协议
  + HTTPS： “HTTP over SSL/TLS”，也就是运行在 SSL/TLS 协议上的 HTTP
  + SSL 的全称是“Secure Socket Layer”，由网景公司发明，当发展到 3.0 时被标准化，改名为 TLS，即“Transport Layer Security”，但由于历史的原因还是有很多人称之为 SSL/TLS，或者直接简称为 SSL。
* 代理是 HTTP 传输过程中的“中转站”，可以实现缓存加速、负载均衡等功能。
  + CDN，实际上就是一种代理，它代替源站服务器响应客户端的请求，通常扮演着透明代理和反向代理的角色。

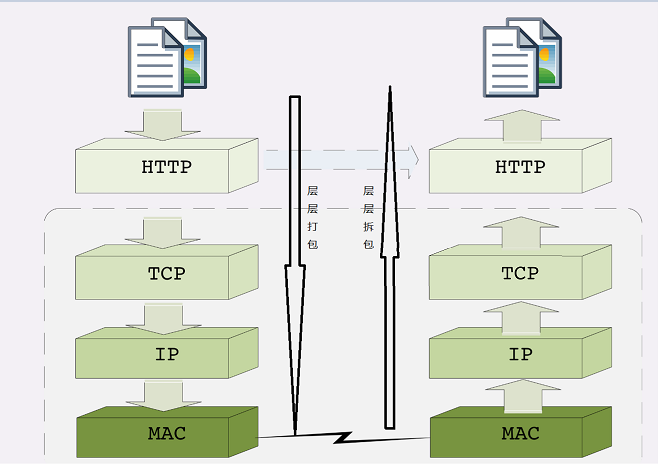
# TCP/IP

它是 HTTP 协议的下层协议，负责具体的数据传输工作。并且还特别说了，TCP/IP 协议是一个“有层次的协议栈”

* TCP/IP 分为四层，核心是二层的 IP 和三层的 TCP，HTTP 在第四层；
* OSI 分为七层，基本对应 TCP/IP，TCP 在第四层，HTTP 在第七层；
* OSI 可以映射到 TCP/IP，但这期间一、五、六层消失了；
* 日常交流的时候我们通常使用 OSI 模型，用四层、七层等术语；
* HTTP 利用 TCP/IP 协议栈逐层打包再拆包，实现了数据传输，但下面的细节并不可见。

网络模型图





# HTTP报文



HTTP 协议的请求报文和响应报文的结构基本相同，由三大部分组成：

起始行（start line）：描述请求或响应的基本信息；

头部字段集合（header）：使用 key-value 形式更详细地说明报文；

消息正文（entity）：实际传输的数据，它不一定是纯文本，可以是图片、视频等二进制数据。

这其中前两部分起始行和头部字段经常又合称为“请求头”或“响应头”，消息正文又称为“实体”，但与“header”对应，很多时候就直接称为“body”。

请求行：请求方法 + 请求目标 + 版本号

这三个部分通常使用空格（space）来分隔，最后要用 CRLF 换行表示结束。



状态行：版本号+ 状态码+ 原因（解释文字）

HTTP/1.1 里唯一要求必须提供的头字段是 **Host**，它必须出现在请求头里，标记虚拟主机名。

请求方法：

GET：获取资源，可以理解为读取或者下载数据；

HEAD：获取资源的元信息；

POST：向资源提交数据，相当于写入或上传数据；

PUT：类似 POST；

DELETE：删除资源；

CONNECT：建立特殊的连接隧道；

OPTIONS：列出可对资源实行的方法；

TRACE：追踪请求 - 响应的传输路径。

安全： 在 HTTP 协议里，所谓的“安全”是指请求方法不会“破坏”服务器上的资源，即不会对服务器上的资源造成实质的修改。（GET/HEAD安全 PUT和POST不安全）

幂等：所谓的“幂等”实际上是一个数学用语，被借用到了 HTTP 协议里，意思是多次执行相同的操作，结果也都是相同的，即多次“幂”后结果“相等”

GET/HEAD 幂等 SQL select

DELETE 幂等 SQL delete

POST SQL insert 不幂等

PUT SQL update 幂等

# URI

URI 本质上是一个字符串，这个字符串的作用是唯一地标记资源的位置或者名字。



scheme: “方案名”或者“协议名”，表示资源应该使用哪种协议来访问。

在“://”之后，是被称为“authority”的部分，表示资源所在的主机名，通常的形式是“host:port”，即主机名加端口号。

path 标记资源所在的位置；

query 表示对资源附加的额外要求；

在 URI 里对“@&/”等特殊字符和汉字必须要做编码，否则服务器收到 HTTP 报文后会无法正确处理。

http://nginx.org

http://www.chrono.com:8080/11-1

https://tools.ietf.org/html/rfc7230

<file:///D:/http_study/www/>

# 状态码



它是一个十进制数字，以代码的形式表示服务器对请求的处理结果

1××：提示信息，表示目前是协议处理的中间状态，还需要后续的操作；

101 Switching Protocols 客户端使用Upgrade头字段，要求在HTTP协议的基础上改用其他协议继续通信，如果服务器同意变更协议，则会发送状态码101，此后的数据传输协议就不再是HTTP了

2××：成功，服务器收到并成功处理了客户的请求，报文已经收到并被正确处理；

200 OK 最常见的成功状态码，服务器返回了客户端期望的处理结果，若不是HEAD请求，响应头后会有body数据

204 No Content, 与200 OK基本相同，但响应头后没有body数据。

206 Partial Content 是HTTP分块下载或者断电续传的基础，在客户端发送“范围请求”，要求获取资源的部分数据时出现，与200一样，也是服务器成功处理了请求，但是body里面的数据不是资源的全部，而是其中的一部分

状态码 206 通常还会伴随着头字段“**Content-Range**”，表示响应报文里 body 数据的具体范围，供客户端确认，例如“Content-Range: bytes 0-99/2000”，意思是此次获取的是总计 2000 个字节的前 100 个字节。

3××：重定向，资源位置发生变动，需要客户端重新发送请求；

301 Moved Permanently， 俗称“永久重定向，此次请求的资源已经不存在了，需要改用改用新的 URI 再次访问。

302 Found， 俗称“临时重定向”，意思是请求的资源还在，但需要暂时用另一个 URI 来访问

“304 Not Modified” 是一个比较有意思的状态码，它用于 If-Modified-Since 等条件请求，表示资源未修改，用于缓存控制。它不具有通常的跳转含义，但可以理解成“重定向已到缓存的文件”（即“缓存重定向”）。

使用场景：

比如，你的网站升级到了 HTTPS，原来的 HTTP 不打算用了，这就是“永久”的，所以要配置 301 跳转，把所有的 HTTP 流量都切换到 HTTPS。再比如，今天夜里网站后台要系统维护，服务暂时不可用，这就属于“临时”的，可以配置成 302 跳转，把流量临时切换到一个静态通知页面，浏览器看到这个 302 就知道这只是暂时的情况，不会做缓存优化，第二天还会访问原来的地址。

4××：客户端错误，请求报文有误，服务器无法处理；

“400 Bad Request”是一个通用的错误码，**表示请求报文有错误**，但具体是数据格式错误、缺少请求头还是 URI 超长它没有明确说，只是一个笼统的错误，客户端看到 400 只会是“一头雾水”“不知所措”。所以，在开发 Web 应用时应当尽量避免给客户端返回 400，而是要用其他更有明确含义的状态码。

“403 Forbidden”实际上不是客户端的请求出错，而是表示**服务器禁止访问资源**。原因可能多种多样，例如信息敏感、法律禁止等，如果服务器友好一点，可以在 body 里详细说明拒绝请求的原因，不过现实中通常都是直接给一个“闭门羹”。

“404 Not Found”可能是我们最常看见也是最不愿意看到的一个状态码，它的原意是资源在本服务器上未找到，所以无法提供给客户端。但现在已经被“用滥了”，只要服务器“不高兴”就可以给出个 404，而我们也无从得知后面到底是真的未找到，还是有什么别的原因，某种程度上它比 403 还要令人讨厌。

4××里剩下的一些代码较明确地说明了错误的原因，都很好理解，开发中常用的有：

405 Method Not Allowed：不允许使用某些方法操作资源，例如不允许 POST 只能 GET；406 Not Acceptable：资源无法满足客户端请求的条件，例如请求中文但只有英文；

408 Request Timeout：请求超时，服务器等待了过长的时间；

409 Conflict：多个请求发生了冲突，可以理解为多线程并发时的竞态；

413 Request Entity Too Large：请求报文里的 body 太大；

414 Request-URI Too Long：请求行里的 URI 太大；

429 Too Many Requests：客户端发送了太多的请求，通常是由于服务器的限连策略；

431 Request Header Fields Too Large：请求头某个字段或总体太大；

5××：服务器错误，服务器在处理请求时内部发生了错误。

“500 Internal Server Error”与 400 类似，也是一个通用的错误码，服务器究竟发生了什么错误我们是不知道的。不过对于服务器来说这应该算是好事，通常不应该把服务器内部的详细信息，例如出错的函数调用栈告诉外界。虽然不利于调试，但能够防止黑客的窥探或者分析。

“501 Not Implemented”表示客户端请求的功能还不支持，这个错误码比 500 要“温和”一些，和“即将开业，敬请期待”的意思差不多，不过具体什么时候“开业”就不好说了。

“502 Bad Gateway”通常是服务器作为网关或者代理时返回的错误码，表示服务器自身工作正常，访问后端服务器时发生了错误，但具体的错误原因也是不知道的。

“503 Service Unavailable”表示服务器当前很忙，暂时无法响应服务，我们上网时有时候遇到的“网络服务正忙，请稍后重试”的提示信息就是状态码 503。

# HTTP优缺点（HTTP1）

HTTP 最大的优点是简单、灵活和易于扩展；

HTTP 拥有成熟的软硬件环境，应用的非常广泛，是互联网的基础设施；

HTTP 是无状态的，可以轻松实现集群化，扩展性能，但有时也需要用 Cookie 技术来实现“有状态”；

HTTP 是明文传输，数据完全肉眼可见，能够方便地研究分析，但也容易被窃听；

HTTP 是不安全的，无法验证通信双方的身份，也不能判断报文是否被窜改；

HTTP 的性能不算差，但不完全适应现在的互联网，还有很大的提升空间。