面试题总结——上官青云

**http状态码**

HTTP状态码（HTTP Status Code）是用以表示网页[服务器](http://baike.baidu.com/view/899.htm" \t "http://baike.baidu.com/_blank)HTTP响应状态的3位数字代码。它由 RFC 2616 规范定义的，并得到RFC 2518、RFC 2817、RFC 2295、RFC 2774、RFC 4918等规范扩展。

分为以下几个大类：

[消息（1字头）](http://baike.baidu.com/link?url=Y9vC1THZOCsr7WhYRPfxguy6fr70Vr-QcPsHTQltbXuUVYXAn2YC-bFYuiaGgTrsFt6k137xoN_K2TxQb31rg_" \l "2)

[成功（2字头）](http://baike.baidu.com/link?url=Y9vC1THZOCsr7WhYRPfxguy6fr70Vr-QcPsHTQltbXuUVYXAn2YC-bFYuiaGgTrsFt6k137xoN_K2TxQb31rg_" \l "3)

[重定向（3字头）](http://baike.baidu.com/link?url=Y9vC1THZOCsr7WhYRPfxguy6fr70Vr-QcPsHTQltbXuUVYXAn2YC-bFYuiaGgTrsFt6k137xoN_K2TxQb31rg_" \l "4)

[请求错误（4字头）](http://baike.baidu.com/link?url=Y9vC1THZOCsr7WhYRPfxguy6fr70Vr-QcPsHTQltbXuUVYXAn2YC-bFYuiaGgTrsFt6k137xoN_K2TxQb31rg_" \l "5)

[服务器错误（5、6字头）](http://baike.baidu.com/link?url=Y9vC1THZOCsr7WhYRPfxguy6fr70Vr-QcPsHTQltbXuUVYXAn2YC-bFYuiaGgTrsFt6k137xoN_K2TxQb31rg_" \l "6)

补充：一般用于爬虫软件爬取网站信息，搜索引擎的网站排名（通过爬虫软件每天访问网站并通过获取http状态码，如果是报错等不好的状态码则会降低网站排名）

详细内容：

|  |  |
| --- | --- |
| **状态码** | **含义** |
| 100 | 客户端应当继续发送请求。这个临时响应是用来通知客户端它的部分请求已经被服务器接收，且仍未被拒绝。客户端应当继续发送请求的剩余部分，或者如果请求已经完成，忽略这个响应。服务器必须在请求完成后向客户端发送一个最终响应。 |
| 101 | 服务器已经理解了客户端的请求，并将通过Upgrade 消息头通知客户端采用不同的协议来完成这个请求。在发送完这个响应最后的空行后，服务器将会切换到在Upgrade 消息头中定义的那些协议。 　　只有在切换新的协议更有好处的时候才应该采取类似措施。例如，切换到新的HTTP 版本比旧版本更有优势，或者切换到一个实时且同步的协议以传送利用此类特性的资源。 |
| 102 | 由WebDAV（RFC 2518）扩展的状态码，代表处理将被继续执行。 |
| 200 | 请求已成功，请求所希望的响应头或数据体将随此响应返回。 |
| 201 | 请求已经被实现，而且有一个新的资源已经依据请求的需要而建立，且其 URI 已经随Location 头信息返回。假如需要的资源无法及时建立的话，应当返回 '202 Accepted'。 |
| 202 | 服务器已接受请求，但尚未处理。正如它可能被拒绝一样，最终该请求可能会也可能不会被执行。在异步操作的场合下，没有比发送这个状态码更方便 的做法了。 　　返回202状态码的响应的目的是允许服务器接受其他过程的请求（例如某个每天只执行一次的基于批处理的操作），而不必让客户端一直保持与服务器的连接 直到批处理操作全部完成。在接受请求处理并返回202状态码的响应应当在返回的实体中包含一些指示处理当前状态的信息，以及指向处理状态监视器或状态预测 的指针，以便用户能够估计操作是否已经完成。 |
| 203 | 服务器已成功处理了请求，但返回的实体头部元信息不是在原始服务器上有效的确定集合，而是来自本地或者第三方的拷贝。当前的信息可能是原始版 本的子集或者超集。例如，包含资源的元数据可能导致原始服务器知道元信息的超级。使用此状态码不是必须的，而且只有在响应不使用此状态码便会返回200 OK的情况下才是合适的。 |
| 204 | 服务器成功处理了请求，但没有返回任何内容。 |
| 205 | 服务器成功处理了请求，且没有返回任何内容。但是与204响应不同，返回此状态码的响应要求请求者重置文档视图。该响应主要是被用于接受用户输入后，立即重置表单，以便用户能够轻松地开始另一次输入。 　　与204响应一样，该响应也被禁止包含任何消息体，且以消息头后的第一个空行结束。 |
| 206 | 服务器已经成功处理了部分 GET 请求。类似于 FlashGet 或者迅雷这类的 HTTP 下载工具都是使用此类响应实现断点续传或者将一个大文档分解为多个下载段同时下载。 　　该请求必须包含 Range 头信息来指示客户端希望得到的内容范围，并且可能包含 If-Range 来作为请求条件。 　　响应必须包含如下的头部域： 　　Content-Range 用以指示本次响应中返回的内容的范围；如果是 Content-Type 为 multipart/byteranges 的多段下载，则每一 multipart 段中都应包含 Content-Range 域用以指示本段的内容范围。假如响应中包含 Content-Length，那么它的数值必须匹配它返回的内容范围的真实字节数。 　　Date 　　ETag 和/或 Content-Location，假如同样的请求本应该返回200响应。 　　Expires, Cache-Control，和/或 Vary，假如其值可能与之前相同变量的其他响应对应的值不同的话。 　　假如本响应请求使用了 If-Range 强缓存验证，那么本次响应不应该包含其他实体头；假如本响应的请求使用了 If-Range 弱缓存验证，那么本次响应禁止包含其他实体头；这避免了缓存的实体内容和更新了的实体头信息之间的不一致。否则，本响应就应当包含所有本应该返回200响 应中应当返回的所有实体头部域。 　　假如 ETag 或 Last-Modified 头部不能精确匹配的话，则客户端缓存应禁止将206响应返回的内容与之前任何缓存过的内容组合在一起。 　　任何不支持 Range 以及 Content-Range 头的缓存都禁止缓存206响应返回的内容。 |
| 207 | 由WebDAV(RFC 2518)扩展的状态码，代表之后的消息体将是一个XML消息，并且可能依照之前子请求数量的不同，包含一系列独立的响应代码。 |
| 300 | 被请求的资源有一系列可供选择的回馈信息，每个都有自己特定的地址和浏览器驱动的商议信息。用户或浏览器能够自行选择一个首选的地址进行重定 向。 　　除非这是一个 HEAD 请求，否则该响应应当包括一个资源特性及地址的列表的实体，以便用户或浏览器从中选择最合适的重定向地址。这个实体的格式由 Content-Type 定义的格式所决定。浏览器可能根据响应的格式以及浏览器自身能力，自动作出最合适的选择。当然，RFC 2616规范并没有规定这样的自动选择该如何进行。 　　如果服务器本身已经有了首选的回馈选择，那么在 Location 中应当指明这个回馈的 URI；浏览器可能会将这个 Location 值作为自动重定向的地址。此外，除非额外指定，否则这个响应也是可缓存的。 |
| 301 | 请求的网页已永久移动到新位置。当URLs发生变化时，使用301代码。搜索引擎索引中保存新的URL。 |
| 302 | 请求的网页临时移动到新位置。搜索引擎索引中保存原来的URL。 |
| 303 | 当请求者应对不同的位置进行单独的 GET 请求以检索响应时，服务器会返回此代码。对于除 HEAD 请求之外的所有请求，服务器会自动转到其他位置。 |
| 304 | 自从上次请求后，请求的网页未被修改过。服务器返回此响应时，不会返回网页内容。 |
| 305 | 被请求的资源必须通过指定的代理才能被访问。Location 域中将给出指定的代理所在的 URI 信息，接收者需要重复发送一个单独的请求，通过这个代理才能访问相应资源。只有原始服务器才能建立305响应。 　　注意：RFC 2068中没有明确305响应是为了重定向一个单独的请求，而且只能被原始服务器建立。忽视这些限制可能导致严重的安全后果。 |
| 306 | 在最新版的规范中，306状态码已经不再被使用。 |
| 307 | 请求的资源现在临时从不同的URI 响应请求。由于这样的重定向是临时的，客户端应当继续向原有地址发送以后的请求。只有在Cache-Control或Expires中进行了指定的情况 下，这个响应才是可缓存的。 　　新的临时性的URI 应当在响应的 Location 域中返回。除非这是一个HEAD 请求，否则响应的实体中应当包含指向新的URI 的超链接及简短说明。因为部分浏览器不能识别307响应，因此需要添加上述必要信息以便用户能够理解并向新的 URI 发出访问请求。 　　如果这不是一个GET 或者 HEAD 请求，那么浏览器禁止自动进行重定向，除非得到用户的确认，因为请求的条件可能因此发生变化。 |
| 400 | 1、语义有误，当前请求无法被服务器理解。除非进行修改，否则客户端不应该重复提交这个请求。 　　2、请求参数有误。 |
| 401 | 当前请求需要用户验证。该响应必须包含一个适用于被请求资源的 WWW-Authenticate 信息头用以询问用户信息。客户端可以重复提交一个包含恰当的 Authorization 头信息的请求。如果当前请求已经包含了 Authorization 证书，那么401响应代表着服务器验证已经拒绝了那些证书。如果401响应包含了与前一个响应相同的身份验证询问，且浏览器已经至少尝试了一次验证，那么 浏览器应当向用户展示响应中包含的实体信息，因为这个实体信息中可能包含了相关诊断信息。参见RFC 2617。 |
| 402 | 该状态码是为了将来可能的需求而预留的。 |
| 403 | 服务器已经理解请求，但是拒绝执行它。与401响应不同的是，身份验证并不能提供任何帮助，而且这个请求也不应该被重复提交。如果这不是一个 HEAD 请求，而且服务器希望能够讲清楚为何请求不能被执行，那么就应该在实体内描述拒绝的原因。当然服务器也可以返回一个404响应，假如它不希望让客户端获得 任何信息。 |
| 404 | 请求失败，请求所希望得到的资源未被在服务器上发现。没有信息能够告诉用户这个状况到底是暂时的还是永久的。假如服务器知道情况的话，应当使 用410状态码来告知旧资源因为某些内部的配置机制问题，已经永久的不可用，而且没有任何可以跳转的地址。404这个状态码被广泛应用于当服务器不想揭示 到底为何请求被拒绝或者没有其他适合的响应可用的情况下。 |
| 405 | 请求行中指定的请求方法不能被用于请求相应的资源。该响应必须返回一个Allow 头信息用以表示出当前资源能够接受的请求方法的列表。 　　鉴于 PUT，DELETE 方法会对服务器上的资源进行写操作，因而绝大部分的网页服务器都不支持或者在默认配置下不允许上述请求方法，对于此类请求均会返回405错误。 |
| 406 | 请求的资源的内容特性无法满足请求头中的条件，因而无法生成响应实体。 　　除非这是一个 HEAD 请求，否则该响应就应当返回一个包含可以让用户或者浏览器从中选择最合适的实体特性以及地址列表的实体。实体的格式由 Content-Type 头中定义的媒体类型决定。浏览器可以根据格式及自身能力自行作出最佳选择。但是，规范中并没有定义任何作出此类自动选择的标准。 |
| 407 | 与401响应类似，只不过客户端必须在代理服务器上进行身份验证。代理服务器必须返回一个 Proxy-Authenticate 用以进行身份询问。客户端可以返回一个 Proxy-Authorization 信息头用以验证。参见RFC 2617。 |
| 408 | 请求超时。客户端没有在服务器预备等待的时间内完成一个请求的发送。客户端可以随时再次提交这一请求而无需进行任何更改。 |
| 409 | 由于和被请求的资源的当前状态之间存在冲突，请求无法完成。这个代码只允许用在这样的情况下才能被使用：用户被认为能够解决冲突，并且会重新 提交新的请求。该响应应当包含足够的信息以便用户发现冲突的源头。 　　冲突通常发生于对 PUT 请求的处理中。例如，在采用版本检查的环境下，某次 PUT 提交的对特定资源的修改请求所附带的版本信息与之前的某个（第三方）请求向冲突，那么此时服务器就应该返回一个409错误，告知用户请求无法完成。此时， 响应实体中很可能会包含两个冲突版本之间的差异比较，以便用户重新提交归并以后的新版本。 |
| 410 | 被请求的资源在服务器上已经不再可用，而且没有任何已知的转发地址。这样的状况应当被认为是永久性的。如果可能，拥有链接编辑功能的客户端应 当在获得用户许可后删除所有指向这个地址的引用。如果服务器不知道或者无法确定这个状况是否是永久的，那么就应该使用404状态码。除非额外说明，否则这 个响应是可缓存的。 　　410响应的目的主要是帮助网站管理员维护网站，通知用户该资源已经不再可用，并且服务器拥有者希望所有指向这个资源的远端连接也被删除。这类事件在 限时、增值服务中很普遍。同样，410响应也被用于通知客户端在当前服务器站点上，原本属于某个个人的资源已经不再可用。当然，是否需要把所有永久不可用 的资源标记为'410 Gone'，以及是否需要保持此标记多长时间，完全取决于服务器拥有者。 |
| 411 | 服务器拒绝在没有定义 Content-Length 头的情况下接受请求。在添加了表明请求消息体长度的有效 Content-Length 头之后，客户端可以再次提交该请求。 |
| 412 | 服务器在验证在请求的头字段中给出先决条件时，没能满足其中的一个或多个。这个状态码允许客户端在获取资源时在请求的元信息（请求头字段数据）中设置先决条件，以此避免该请求方法被应用到其希望的内容以外的资源上。 |
| 413 | 服务器拒绝处理当前请求，因为该请求提交的实体数据大小超过了服务器愿意或者能够处理的范围。此种情况下，服务器可以关闭连接以免客户端继续发送此请求。 　　如果这个状况是临时的，服务器应当返回一个 Retry-After 的响应头，以告知客户端可以在多少时间以后重新尝试。 |
| 414 | 请求的URI 长度超过了服务器能够解释的长度，因此服务器拒绝对该请求提供服务。这比较少见，通常的情况包括： 　　本应使用POST方法的表单提交变成了GET方法，导致查询字符串（Query String）过长。 　　重定向URI “黑洞”，例如每次重定向把旧的 URI 作为新的 URI 的一部分，导致在若干次重定向后 URI 超长。 　　客户端正在尝试利用某些服务器中存在的安全漏洞攻击服务器。这类服务器使用固定长度的缓冲读取或操作请求的 URI，当 GET 后的参数超过某个数值后，可能会产生缓冲区溢出，导致任意代码被执行[1]。没有此类漏洞的服务器，应当返回414状态码。 |
| 415 | 对于当前请求的方法和所请求的资源，请求中提交的实体并不是服务器中所支持的格式，因此请求被拒绝。 |
| 416 | 如果请求中包含了 Range 请求头，并且 Range 中指定的任何数据范围都与当前资源的可用范围不重合，同时请求中又没有定义 If-Range 请求头，那么服务器就应当返回416状态码。 　　假如 Range 使用的是字节范围，那么这种情况就是指请求指定的所有数据范围的首字节位置都超过了当前资源的长度。服务器也应当在返回416状态码的同时，包含一个 Content-Range 实体头，用以指明当前资源的长度。这个响应也被禁止使用 multipart/byteranges 作为其 Content-Type。 |
| 417 | 在请求头 Expect 中指定的预期内容无法被服务器满足，或者这个服务器是一个代理服务器，它有明显的证据证明在当前路由的下一个节点上，Expect 的内容无法被满足。 |
| 421 | 从当前客户端所在的IP地址到服务器的连接数超过了服务器许可的最大范围。通常，这里的IP地址指的是从服务器上看到的客户端地址（比如用户的网关或者代理服务器地址）。在这种情况下，连接数的计算可能涉及到不止一个终端用户。 |
| 422 | 从当前客户端所在的IP地址到服务器的连接数超过了服务器许可的最大范围。通常，这里的IP地址指的是从服务器上看到的客户端地址（比如用户的网关或者代理服务器地址）。在这种情况下，连接数的计算可能涉及到不止一个终端用户。 |
| 422 | 请求格式正确，但是由于含有语义错误，无法响应。（RFC 4918 WebDAV）423 Locked 　　当前资源被锁定。（RFC 4918 WebDAV） |
| 424 | 由于之前的某个请求发生的错误，导致当前请求失败，例如 PROPPATCH。（RFC 4918 WebDAV） |
| 425 | 在WebDav Advanced Collections 草案中定义，但是未出现在《WebDAV 顺序集协议》（RFC 3658）中。 |
| 426 | 客户端应当切换到TLS/1.0。（RFC 2817） |
| 449 | 由微软扩展，代表请求应当在执行完适当的操作后进行重试。 |
| 500 | 服务器遇到了一个未曾预料的状况，导致了它无法完成对请求的处理。一般来说，这个问题都会在服务器的程序码出错时出现。 |
| 501 | 服务器不支持当前请求所需要的某个功能。当服务器无法识别请求的方法，并且无法支持其对任何资源的请求。 |
| 502 | 作为网关或者代理工作的服务器尝试执行请求时，从上游服务器接收到无效的响应。 |
| 503 | 由于临时的服务器维护或者过载，服务器当前无法处理请求。这个状况是临时的，并且将在一段时间以后恢复。如果能够预计延迟时间，那么响应中可 以包含一个 Retry-After 头用以标明这个延迟时间。如果没有给出这个 Retry-After 信息，那么客户端应当以处理500响应的方式处理它。 　　注意：503状态码的存在并不意味着服务器在过载的时候必须使用它。某些服务器只不过是希望拒绝客户端的连接。 |
| 504 | 作为网关或者代理工作的服务器尝试执行请求时，未能及时从上游服务器（URI标识出的服务器，例如HTTP、FTP、LDAP）或者辅助服务器（例如DNS）收到响应。 　　注意：某些代理服务器在DNS查询超时时会返回400或者500错误 |
| 505 | 服务器不支持，或者拒绝支持在请求中使用的 HTTP 版本。这暗示着服务器不能或不愿使用与客户端相同的版本。响应中应当包含一个描述了为何版本不被支持以及服务器支持哪些协议的实体。 |
| 506 | 由《透明内容协商协议》（RFC 2295）扩展，代表服务器存在内部配置错误：被请求的协商变元资源被配置为在透明内容协商中使用自己，因此在一个协商处理中不是一个合适的重点。 |
| 507 | 服务器无法存储完成请求所必须的内容。这个状况被认为是临时的。WebDAV (RFC 4918) |
| 509 | 服务器达到带宽限制。这不是一个官方的状态码，但是仍被广泛使用。 |
| 510 | 获取资源所需要的策略并没有没满足。（RFC 2774） |

如果没有响应体,则在大多数场景下,这两种响应码完全等效,但有一种情况下,HTTP/204响应会让浏览器有不同的表现.这种情况就是当用户在浏览器窗口window或者frame/iframe框架中导航的时候.

如果导航到的URL返回了一个没有响应体的HTTP/200响应,则页面将会显示一个空白文档(就是一片白色).页面的URL地址也会变成新指定的URL.

如果服务器返回的是一个HTTP/204响应,当前页面不会有任何变化,就好像根本没有进行导航操作一样.页面的URL地址也保持不变.

**Cookie**

Cookie是由服务器端生成，发送给User-Agent（一般是浏览器），浏览器会将Cookie的key/value保存到某个目录下的文本文件内，下次请求同一网站时就发送该Cookie给服务器（前提是浏览器设置为启用cookie）。Cookie名称[和值](http://baike.so.com/doc/6323305.html" \t "http://baike.so.com/doc/_blank)可以由[服务器](http://baike.so.com/doc/4487696.html" \t "http://baike.so.com/doc/_blank)端开发自己定义，对于JSP而言也可以直接写入jsessionid，这样服务器可以知道该用户是否合法用户以及是否需要重新登录等。

用途

1. 根据过去的登录信息，来自动识别用户，而无需登录
2. 利用cookies跟踪统计用户访问该网站的习惯，一方面是可以为用户提供个性化的服务，另一方面，也可以作为了解所有用户行为的工具，对于网站经营策略的改进有一定参考价值。

注意：cookies与用户隐私权的问题并没有相关法律约束，很多网站利用cookie跟踪用户行为，有些程序要求用户必须开启cookie才能正常应用。

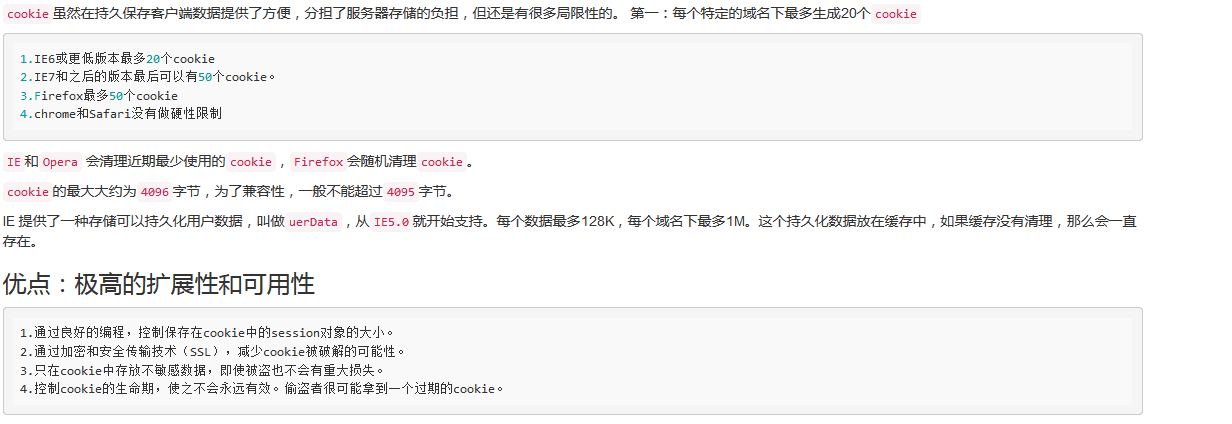
浏览器用户可以通过“隐私”选项中的隐私设置的高低来决定是否允许网站利用cookie跟踪自己的信息，从全部限制到全部允许。

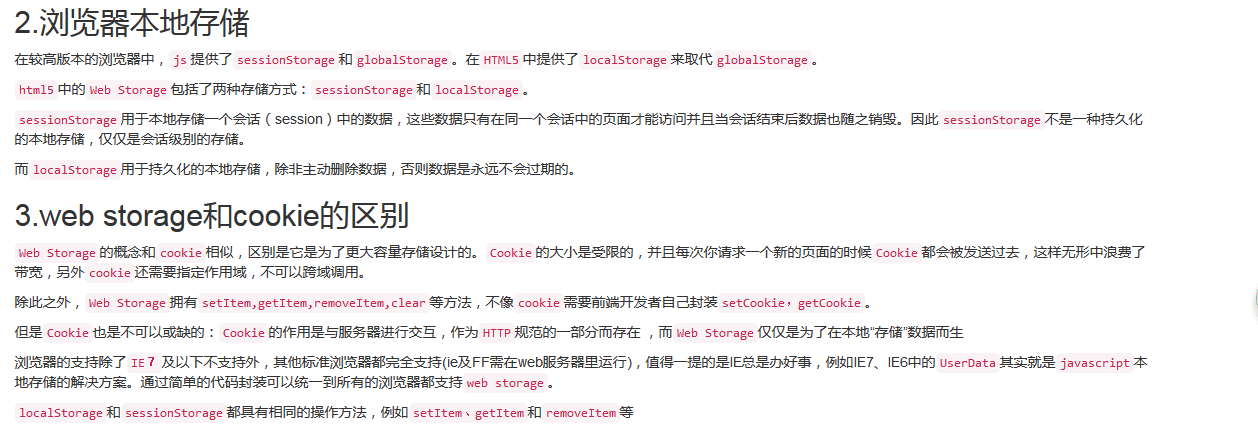
生存周期—Expire值

Cookie在生成时就会被指定一个Expire值，这就是Cookie的生存周期，超出周期Cookie就会被清除。有些页面将Cookie的生存周期设置为“0”或负值，这样在关闭浏览器时，就马上清除Cookie，不会记录用户信息，更加安全。

识别功能

cookie中有用户信息，还包含计算机和浏览器的信息





**https**

**含义**：HTTPS（全称：Hyper Text Transfer Protocol over Secure Socket Layer），是以安全为目标的[HTTP](http://baike.baidu.com/view/9472.htm" \t "http://baike.baidu.com/_blank)通道，简单讲是HTTP的安全版。

**HTTPS和HTTP的区别**：https协议是安全的，需要到ca（Certification Authority 认证机构）申请证书，在http上加了SSL(Secure Sockets Layer [安全套接层](http://baike.baidu.com/view/525499.htm" \t "http://baike.baidu.com/_blank))，具有[安全性](http://baike.baidu.com/view/421194.htm" \t "http://baike.baidu.com/_blank)的[ssl](http://baike.baidu.com/view/16147.htm" \t "http://baike.baidu.com/_blank)加密传输协议。端口也不再是80而是443。此外在OSI 网络模型中，HTTP工作于应用层，而HTTPS 工作在传输层

TLS（Transport Layer Security，SSL的继承者）与SSL在[传输层](http://baike.baidu.com/view/239605.htm" \t "http://baike.baidu.com/_blank)对网络连接进行加密

**Https的工作原理**

HTTPS在传输数据之前需要客户端（浏览器）与服务端（网站）之间进行一次握手，在握手过程中将确立双方加密传输数据的密码信息。

这里浏览器与网站互相发送加密的握手消息并验证，目的是为了保证双方都获得了一致的密码，并且可以正常的加密解密数据，为后续真正数据的传输做一次测试。

**\*补充：**

**握手的过程：**

1.浏览器告诉网站自己的加密规则

2.网站告诉客户端自己的加密算法和hash算法，还有身份信息

3.获得网站证书之后浏览器要做以下工作：

a) 验证证书的合法性

b) 如果证书受信任，或者是用户接受了不受信的证书，浏览器会生成一串随机数的密码，并用证书中提供的公钥加密。

c) 使用约定好的HASH计算握手消息，并使用生成的随机数对消息进行加密，最后将之前生成的所有信息发送给网站。

4.网站接收浏览器发来的数据之后要做以下的操作：

a) 使用自己的私钥将信息解密取出密码，使用密码解密浏览器发来的握手消息，并验证HASH是否与浏览器发来的一致。

b) 使用密码加密一段握手消息，发送给浏览器。

5.浏览器解密并计算握手消息的HASH，如果与服务端发来的HASH一致，此时握手过程结束，之后所有的通信数据将由之前浏览器生成的随机密码并利用对称加密算法进行加密。

另外，HTTPS一般使用的加密与HASH算法如下：

非对称加密算法：RSA，DSA/DSS

对称加密算法：AES，RC4，3DES

HASH算法：MD5，SHA1，SHA256

其中非对称加密算法用于在握手过程中加密生成的密码，对称加密算法用于对真正传输的数据进行加密，而HASH算法用于验证数据的完整性。由于浏览器生成的密码是整个数据加密的关键，因此在传输的时候使用了非对称加密算法对其加密。非对称加密算法会生成公钥和私钥，公钥只能用于加密数据，因此可以随意传输，而网站的私钥用于对数据进行解密，所以网站都会非常小心的保管自己的私钥，防止泄漏。

TLS握手过程中如果有任何错误，都会使加密连接断开，从而阻止了隐私信息的传输。

\*再次补充：

**Http的请求方式:**

HTTP1.1支持7种请求方法：GET、POST、HEAD、OPTIONS、PUT、DELETE、CONNECT和TARCE。在Internet应用中，最常用的方法是GET和POST。

[OPTIONS](https://www.baidu.com/s?wd=OPTIONS&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3PWT1mWTkPjb3PhwBPADk0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHDLrHfzPWm4" \t "http://zhidao.baidu.com/_blank) 返回服务器针对特定资源所支持的[HTTP](https://www.baidu.com/s?wd=HTTP&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3PWT1mWTkPjb3PhwBPADk0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHDLrHfzPWm4" \t "http://zhidao.baidu.com/_blank)请求方法。也可以利用向Web服务器发送'\*'的请求来测试服务器的功能性。　这个方法很有趣，但极少使用。它用于获取当前URL所支持的方法。若请求成功，则它会在HTTP头中包含一个名为“Allow”的头，值是所支持的方法，如“GET, POST”。

[HEAD](https://www.baidu.com/s?wd=HEAD&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3PWT1mWTkPjb3PhwBPADk0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHDLrHfzPWm4" \t "http://zhidao.baidu.com/_blank) 向服务器索要与[GET](https://www.baidu.com/s?wd=GET&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3PWT1mWTkPjb3PhwBPADk0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHDLrHfzPWm4" \t "http://zhidao.baidu.com/_blank)请求相一致的响应，只不过响应体将不会被返回。这一方法可以在不必传输整个响应内容的情况下，就可以获取包含在响应消息头中的元信息。　HEAD和GET本质是一样的，区别在于HEAD不含有呈现数据，而仅仅是HTTP头信息。有的人可能觉得这个方法没什么用，其实不是这样的。想象一个业务情景：欲判断某个资源是否存在，我们通常使用GET，但这里用HEAD则意义更加明确。

[GET](https://www.baidu.com/s?wd=GET&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3PWT1mWTkPjb3PhwBPADk0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHDLrHfzPWm4" \t "http://zhidao.baidu.com/_blank) 请求指定的页面信息，并返回实体主体。本质就是发送一个请求来取得服务器上的某一资源。资源通过一组HTTP头和呈现数据（如HTML文本，或者图片或者视频等）返回给客户端。GET请求中，永远不会包含呈现数据。注意：[GET](https://www.baidu.com/s?wd=GET&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3PWT1mWTkPjb3PhwBPADk0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHDLrHfzPWm4" \t "http://zhidao.baidu.com/_blank)方法不应当被用于产生“副作用”的操作中，例如在web app.中。其中一个原因是GET可能会被网络蜘蛛等随意访问。

[POST](https://www.baidu.com/s?wd=POST&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3PWT1mWTkPjb3PhwBPADk0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHDLrHfzPWm4" \t "http://zhidao.baidu.com/_blank) 向指定资源提交数据进行处理请求（例如提交表单或者上传文件）。数据被包含在请求体中。[POST](https://www.baidu.com/s?wd=POST&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3PWT1mWTkPjb3PhwBPADk0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHDLrHfzPWm4" \t "http://zhidao.baidu.com/_blank)请求可能会导致新的资源的建立和/或已有资源的修改。

[PUT](https://www.baidu.com/s?wd=PUT&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3PWT1mWTkPjb3PhwBPADk0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHDLrHfzPWm4" \t "http://zhidao.baidu.com/_blank) 向指定资源位置上传其最新内容。HTML表单也不支持这个。本质上来讲， PUT和POST极为相似，都是向服务器发送数据，但它们之间有一个重要区别，PUT通常指定了资源的存放位置，而POST则没有，POST的数据存放位置由服务器自己决定。

[DELETE](https://www.baidu.com/s?wd=DELETE&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3PWT1mWTkPjb3PhwBPADk0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHDLrHfzPWm4" \t "http://zhidao.baidu.com/_blank) 请求服务器删除Request-URI所标识的资源。基本上这个也很少见，不过还是有一些地方比如amazon的S3云服务里面就用的这个方法来删除资源。

[TRACE](https://www.baidu.com/s?wd=TRACE&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3PWT1mWTkPjb3PhwBPADk0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHDLrHfzPWm4" \t "http://zhidao.baidu.com/_blank) 回显服务器收到的请求，主要用于测试或诊断。

[CONNECT](https://www.baidu.com/s?wd=CONNECT&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3PWT1mWTkPjb3PhwBPADk0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHDLrHfzPWm4" \t "http://zhidao.baidu.com/_blank) [HTTP](https://www.baidu.com/s?wd=HTTP&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3PWT1mWTkPjb3PhwBPADk0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHDLrHfzPWm4" \t "http://zhidao.baidu.com/_blank)/1.1协议中预留给能够将连接改为管道方式的代理服务器。

**Http的版本：**

0.9　已过时。只接受 GET 一种请求方法，没有在通讯中指定版本号，且不支持请求头。由于该版本不支持 POST 方法，所以客户端无法向服务器传递太多信息。

HTTP/1.0　这是第一个在通讯中指定版本号的HTTP 协议版本，至今仍被广泛采用，特别是在[代理服务器](http://baike.baidu.com/view/751.htm" \t "http://baike.baidu.com/_blank)中。

HTTP/1.1　当前版本。持久连接被默认采用，并能很好地配合代理服务器工作。还支持以管道方式同时发送多个请求，以便降低线路负载，提高传输速度。

HTTP/1.1相较于 HTTP/1.0 协议的区别主要体现在：

1 缓存处理

2 带宽优化及网络连接的使用

3 错误通知的管理

4 消息在网络中的发送

5 互联网地址的维护

6 安全性及完整性

HTTP 1.0规定浏览器与服务器只保持短暂的连接，浏览器的每次请求都需要与服务器建立一个TCP连接，服务器完成请求处理后立即断开TCP连接，服务器不跟踪每个客户也不记录过去的请求。

HTTP 1.1支持持久连接，在一个TCP连接上可以传送多个HTTP请求和响应，减少了建立和关闭连接的消耗和延迟。

**http的请求和响应格式：**

**HTTP请求报文格式:**

http请求由三部分组成，分别是：请求行、消息报头、请求正文



HTTP请求格式:

<request-line>  
<headers>  
<blank line>  
[<request-body>]

说明:第一行必须是一个请求行(request-line),由3部分组成，分别为：请求方法、URL以及协议版本，之间由空格分隔.

请求方法包括GET、HEAD、PUT、POST、TRACE、OPTIONS、DELETE以及扩展方法，当然并不是所有的服务器都实现了所有的方法，部分方法即便支持，处于安全性的考虑也是不可用的

协议版本的格式为：HTTP/主版本号.次版本号，常用的有HTTP/1.0和HTTP/1.1

紧接着是一个首部(header)小节,请求头部为请求报文添加了一些附加信息，由“名/值”对组成，每行一对，名和值之间使用冒号分隔

常见请求头如下：

|  |  |
| --- | --- |
| **请求头** | **说明** |
| **Host** | 接受请求的服务器地址，可以是IP:端口号，也可以是域名 |
| **User-Agent** | 发送请求的应用程序名称 |
| **Connection** | 指定与连接相关的属性，如Connection:Keep-Alive |
| **Accept-Charset** | 通知服务端可以发送的编码格式 |
| **Accept-Encoding** | 通知服务端可以发送的数据压缩格式 |
| **Accept-Language** | 通知服务端可以发送的语言 |

之后是一个空行.

再后面可以添加任意的其他数据[称之为主体(body)].

**HTTP响应报文格式:**

HTTP响应报文主要由状态行、响应头部、响应正文3部分组成



**1.状态行**

由3部分组成，分别为：协议版本，状态码，状态码描述，之间由空格分隔

**2.响应头部**

与请求头部类似，为响应报文添加了一些附加信息

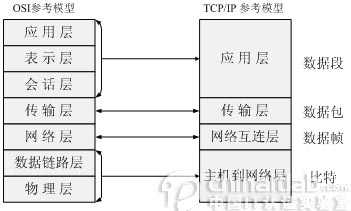
常见响应头部如下：

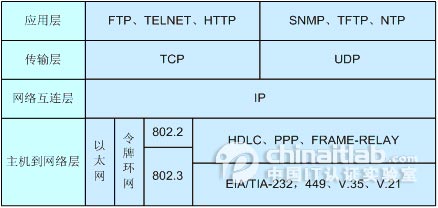
|  |  |
| --- | --- |
| **响应头** | **说明** |
| **Server** | 服务器应用程序软件的名称和版本 |
| **Content-Type** | 响应正文的类型（是图片还是二进制字符串） |
| **Content-Length** | 响应正文长度 |
| **Content-Charset** | 响应正文使用的编码 |
| **Content-Encoding** | 响应正文使用的数据压缩格式 |
| **Content-Language** | 响应正文使用的语言 |

**TCP/IP协议：**

Transmission Control Protocol/Internet Protocol的简写，中译名为传输控制协议/[因特网](http://baike.baidu.com/view/1706.htm" \t "http://baike.baidu.com/_blank)互联协议，又名网络[通讯协议](http://baike.baidu.com/view/278358.htm" \t "http://baike.baidu.com/_blank)，是Internet最基本的协议、Internet国际[互联网](http://baike.baidu.com/view/6825.htm" \t "http://baike.baidu.com/_blank)络的基础，由[网络层](http://baike.baidu.com/view/239600.htm" \t "http://baike.baidu.com/_blank)的IP协议和[传输层](http://baike.baidu.com/view/239605.htm" \t "http://baike.baidu.com/_blank)的TCP协议组成。

TCP/IP不是一个协议，而是一个协议族的统称。里面包括了IP协议，IMCP协议，TCP协议，以及我们更加熟悉的http、ftp、pop3协议等等。





TCP/IP协议族按照层次由上到下，层层包装。

最上面的就是应用层了，这里面有http 超文本传输协议，Telnet [远程登录](http://baike.baidu.com/view/59099.htm" \t "http://baike.baidu.com/_blank)， FTP [文件传输协议](http://baike.baidu.com/view/103832.htm" \t "http://baike.baidu.com/_blank)，TFTP简单文件传输协议，SMTP 简单邮件传送协议，SNMP 简单网络治理协议，域名系统(DNS),该系统用于在internet中将域名及其公共广播的网络节点转换成IP地址.

第二层则是传输层，著名的TCP和UDP协议就在这个层次。

第三层是网络层，网络层协议包括IP协议（[网际协议](http://baike.baidu.com/view/229604.htm" \t "http://baike.baidu.com/_blank)），ICMP协议（internet互联网控制[报文](http://baike.baidu.com/view/175122.htm" \t "http://baike.baidu.com/_blank)协议），以及IGMP协议（internet组治理协议），它负责对数据加上IP地址和其他的数据（后面会讲到）以确定传输的目标。地址解析协议(ARP)，反向地址解析协议(RARP)

第四层是叫网络访问层:网络访问层又称作主机到网络层(host-to-network).网络访问层的功能包括IP地址与物理地址硬件的映射,以及将IP封装成帧.基于不同硬件类型的网络接口,网络访问层定义了和物理介质的连接.

**TCP与UDP:**

[TCP](https://www.baidu.com/s?wd=TCP&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLnycsnynYmWIhn1nzmhnv0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnWDvPjDYnHcs" \t "http://zhidao.baidu.com/_blank)（Transmission Control Protocol，传输控制协议）是基于连接的协议，也就是说，在正式收发数据前，必须和对方建立可靠的连接。

一个[TCP](https://www.baidu.com/s?wd=TCP&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLnycsnynYmWIhn1nzmhnv0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnWDvPjDYnHcs" \t "http://zhidao.baidu.com/_blank)连接必须要经过三次“对话”才能建立起来。

[UDP](https://www.baidu.com/s?wd=UDP&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLnycsnynYmWIhn1nzmhnv0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnWDvPjDYnHcs" \t "http://zhidao.baidu.com/_blank)（User Data Protocol，用户数据报协议）是与[TCP](https://www.baidu.com/s?wd=TCP&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLnycsnynYmWIhn1nzmhnv0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnWDvPjDYnHcs" \t "http://zhidao.baidu.com/_blank)相对应的协议。它是面向非连接的协议，它不与对方建立连接，而是直接就把数据包发送过去！[UDP](https://www.baidu.com/s?wd=UDP&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLnycsnynYmWIhn1nzmhnv0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnWDvPjDYnHcs" \t "http://zhidao.baidu.com/_blank)适用于一次只传送少量数据、对可靠性要求不高的应用环境。

“ping”命令就是向对方主机发送[UDP](https://www.baidu.com/s?wd=UDP&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLnycsnynYmWIhn1nzmhnv0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnWDvPjDYnHcs" \t "http://zhidao.baidu.com/_blank)数据包，然后对方主机确认收到数据包，如果数据包是否到达的消息及时反馈回来，那么网络就是通的。

TCP UDP   
是否连接 面向连接 面向非连接   
传输可靠性 可靠 不可靠   
应用场合 传输大量数据 少量数据   
速度 慢 快

**三次握手（简单通俗版）**：

其中的过程非常复杂，我们这里只做简单、形象的介绍，你只要做到能够理解这个过程即可。我们来看看这三次对话的简单过程：

主机A向主机B发出连接请求数据包：“我想给你发数据，可以吗？”，这是第一次对话；主机B向主机A发送同意连接和要求同步（同步就是两台主机一个在发送，一个在接收，协调工作）的数据包：“可以，你什么时候发？”，这是第二次对话；主机A再发出一个数据包确认主机B的要求同步：“我现在就发，你接着吧！”，这是第三次对话。三次“对话”的目的是使数据包的发送和接收同步，经过三次“对话”之后，主机A才向主机B正式发送数据。

**TCP三次握手过程**  
1 主机A通过向主机B 发送一个含有同步序列号的标志位的数据段给主机B ,向主机B 请求建立连接,通过这个数据段,

主机A告诉主机B 两件事:我想要和你通信;你可以用哪个序列号作为起始数据段来回应我.

2 主机B 收到主机A的请求后,用一个带有确认应答(ACK)和同步序列号(SYN)标志位的数据段响应主机A,也告诉主机A两件事:

我已经收到你的请求了,你可以传输数据了;你要用哪佧序列号作为起始数据段来回应我

3 主机A收到这个数据段后,再发送一个确认应答,确认已收到主机B 的数据段:"我已收到回复,我现在要开始传输实际数据了

这样3次握手就完成了,主机A和主机B 就可以传输数据了.

3次握手的特点  
 没有应用层的数据

SYN这个标志位只有在TCP建产连接时才会被置1  
 握手完成后SYN标志位被置0

### TCP 连接断开过程-四次挥手

由于TCP连接是全双工的，因此每个方向都必须单独进行关闭。这个原则是当一方完成它的数据发送任务后就能发送一个 FIN来终止这个方向的连接。收到一个 FIN只意味着这一方向上没有数据流动，一个TCP连接在收到一个FIN后仍能发送数据。首先进行关闭的一方将执行主动关闭，而另一方执行被动关闭。

 TCP的连接的拆除需要发送四个包，因此称为四次挥手(four-way handshake)。客户端或服务器均可主动发起挥手动作，在socket编程中，任何一方执行close()操作即可产生挥手操作。

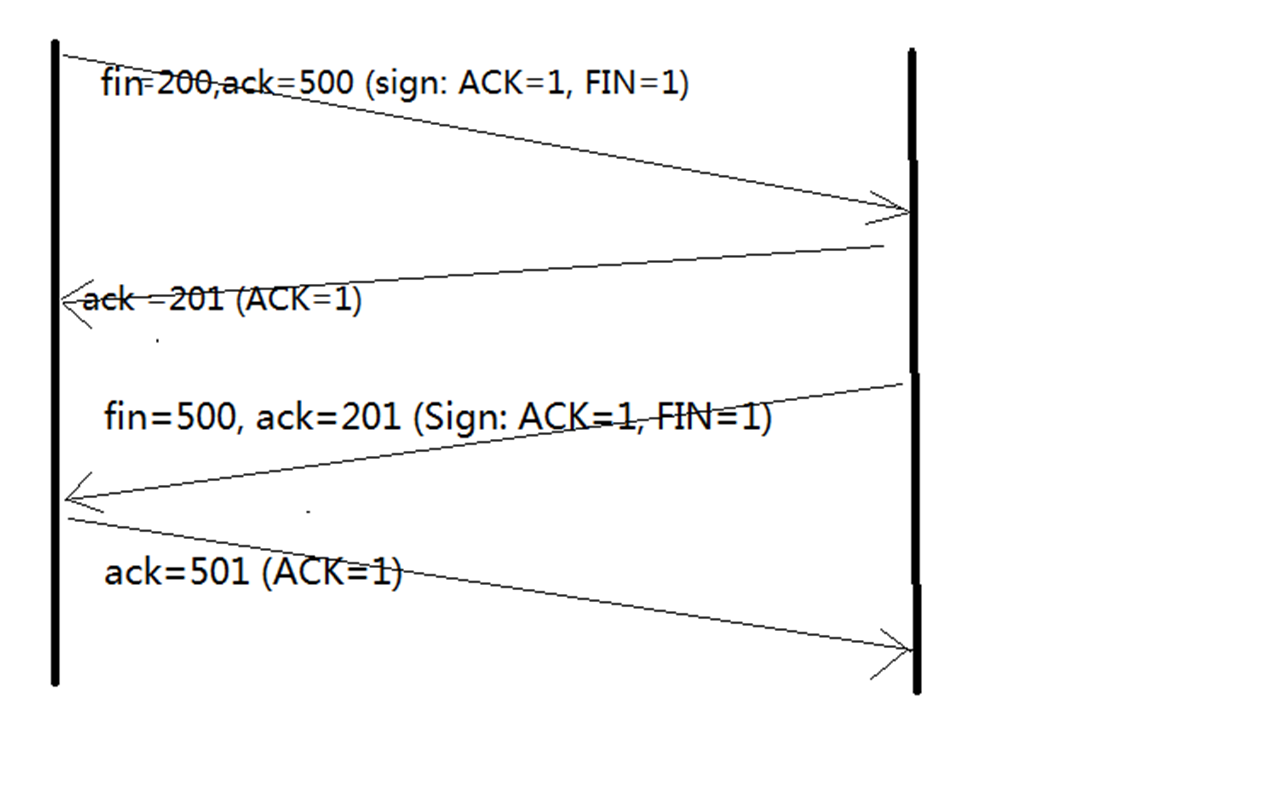
（1）客户端A发送一个FIN，用来关闭客户A到服务器B的数据传送。

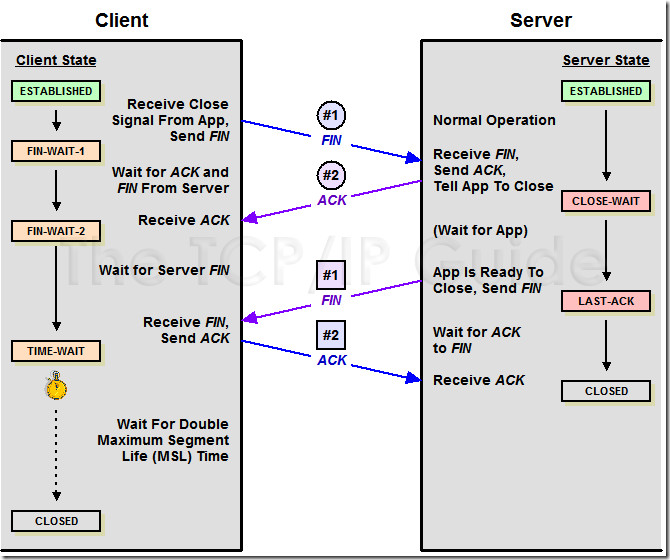
（2）服务器B收到这个FIN，它发回一个ACK，确认序号为收到的序号加1。和SYN一样，一个FIN将占用一个序号。

（3）服务器B关闭与客户端A的连接，发送一个FIN给客户端A。

（4）客户端A发回ACK报文确认，并将确认序号设置为收到序号加1。

TCP采用四次挥手关闭连接如图2所示。





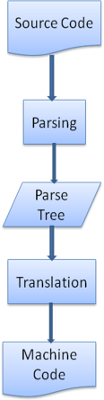
**[TCP的三次握手(建立连接）和四次挥手(关闭连接）](http://www.cnblogs.com/Jessy/p/3535612.html)以及深入解析：**

**http://www.cnblogs.com/Jessy/p/3535612.html**

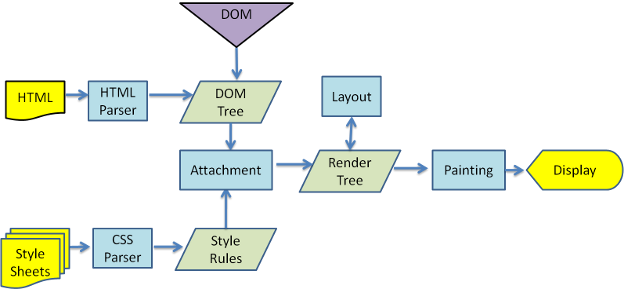
**浏览器的渲染过程**

关于加载顺序：当浏览器获得一个html文件时，会”自上而下“加载，并在加载过程中进行解析渲染。

解析html以构建dom树 -> 构建render树 -> 布局render树 -> 绘制render树

  
图 7：编译过程（源码，解析，解析树，转换，机器码）(最后才转换为机械码)

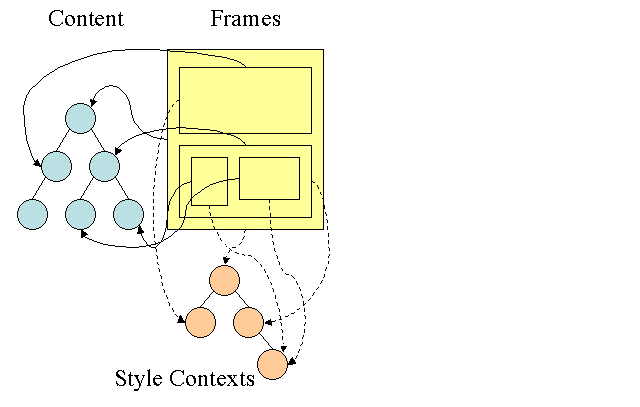
为了更好的用户体验，渲染引擎将会尽可能早的将内容呈现到屏幕上，并不会等到所有的html都解析完成之后再去构建和布局render树。解析完一部分内容就显示一部分内容，同时，可能还在通过网络下载其余内容。从MVC的角度来说，可以将render树看成是V，dom树看成是M，C则是具体的调度者，比HTMLDocumentParser等。



**渲染过程如下：**

1，浏览器解析html源码，然后创建一个 DOM树。  
在DOM树中，每一个HTML标签都有一个对应的节点，并且每一个文本也都会有一个对应的文本节点。  
DOM树的根节点就是 documentElement，对应的是html标签。  
  
2，浏览器解析CSS代码，计算出最终的样式数据，生成CSS 规则树。  
对CSS代码中非法的语法她会直接忽略掉。  
解析CSS的时候会按照如下顺序来定义优先级：浏览器默认设置，用户设置，外链样式，内联样式，html中的style。  
  
3，构建出DOM树，并且计算出样式数据后，下一步就是构建一个 渲染树（rendering tree）。  
渲染树和DOM树有点像，但是是有区别的。DOM树完全和html标签一一对应，但是渲染树会忽略掉不需要渲染的元素，比如head、display:none的元素等。  
而且一大段文本中的每一个行在渲染树中都是独立的一个节点。  
渲染树中的每一个节点都存储有对应的css属性。

具体效果图如下：（firefox的Frames对应renderers,content对应dom）



具体显示的时候，每一个renderer体现了一个矩形区块的东西，即我们常说的CSS盒子模型的概念，它本身包含了一些几何学相关的属性，如宽度 width，高度height，位置position等。每一个renderer还有一个很重要的属性，就是如何显示它，display。

4，一旦渲染树创建好了，浏览器就可以根据渲染树直接把页面绘制到屏幕上。

一个渲染过程的例子

例如有下面这样一段HTML代码：  
<html>  
<head>  
  <title>Beautiful page</title>  
</head>  
<body>  
  <p>  
    Once upon a time there was   
    a looong paragraph...  
  </p>  
  <div style="display: none">  
    Secret message  
  </div>  
    
  <div><img src="..." /></div>  
  ...  
   
</body>  
</html>  
  
  
那么DOM树是完全和HTML标签一一对应的，如下所示：  
documentElement (html)  
    head  
        title  
    body  
        p  
            [text node]    
        div   
            [text node]  
        div  
            img  
        ...  
而渲染树就不同了，她只有哪些需要绘制出来的元素，所以head 以及被隐藏的div都不会出现在渲染树中。  
root (RenderView)  
    body  
        p  
            line 1  
            line 2  
            line 3  
            ...  
        div  
            img  
        ...

重绘和重排（repaints and reflows）

每个页面至少在初始化的时候会有一次重排操作。任何对渲染树的修改都有可能会导致下面两种操作：  
1.重排  
 就是渲染树的一部分必须要更新 并且节点的尺寸发生了变化。这就会触发重排操作。  
2.重绘  
 部分节点需要更新，但是没有改变他的集合形状，比如改变了背景颜色，这就会触发重绘。