一、HTTP和HTTPS协议

1. get和post的区别
2. 常见的HTTP请求头和响应头

常见的HTTP请求方法：

3. 讲讲304
4. 说一下http和https
   - **（1）http和https的基本概念**
   - **（2）http 和 https的区别**
   - （3）https工作原理
5. HTTP1.0，1.1，2.0

   HTTP1.0比较1.1

   HTTP1.1比较2.0
6. HTTP状态码
   - （1）1XX 通知
   - （2）2XX Success 成功状态码
   - （3）3XX 重定向状态码
   - （4）4XX 客户端错误状态码
   - （5）5XX 服务器端错误状态码
7. PUT和POST都是给服务器发送新增资源，有什么区别?

二、DNS协议

1. DNS协议是什么?
2. DNS同时使用TCP和UDP协议?
3. DNS完整的查询过程
4. 迭代查询和递归查询

三、TCP与UDP

1. TCP和UDP的区别
2. TCP的三次握手和四次挥手

   三次握手

   为什么要三次握手？两次不行吗?

   四次挥手
3. 递归查询和迭代查询

一、HTTP和HTTPS协议

面试官（9）：可能是全网最全的http面试答案

1. get和post的区别

- **应用场景**：GET请求一般用于对服务器资源不会产生影响的场景。post一般用于对服务器产生影响的场景。
- **是否缓存**：浏览器一般会对get请求缓存，很少对post请求缓存
- **数据传输方式不同**：get通过url传输数据，post的数据通过请求体传输。
- **安全性**：post比get相对安全，post的数据因为在请求主体内，所以有一定的安全性保证，而get的数据在url中，通过历史记录，缓存很容易查到数据信息。
- **请求长度**：浏览器由于对url长度的限制，会影响get请求发送数据时的长度。
- **参数类型**：post的参数传递支持更多的数据类型。
- get无害：刷新、后退等浏览器操作GET请求是无害的，post可能重复提交表单。

2. 常见的HTTP请求头和响应头

HTTP Request Header常见的请求头：

```
Accept: 浏览器能够处理的内容类型
Accept-Charset: 浏览器能够显示的字符集·
Accept-Encoding: 浏览器能够处理的压缩编码
Accept-Language: 浏览器当前设置的语言
content-type: 表示后面的文档属于什么MIME类型
Connection: 浏览器与服务器之间的连接类型
Cookie: 当前页面设置的任何Cookie
Host: 发出请求的页面所在域
Referer: 发出请求的页面的URL
User-Agent: 浏览器用户代理字符串
if-None-Match: 上一次资源返回的ETag值
if-Modified-Since: 上一次资源返回的Last-Modified值
```

HTTP Responses Header 常见的响应头：

```
Date: 表示消息发送的时间，时间的描述格式由rfc822定义
server: 服务器名称
Connection: 浏览器与服务器之间连接的类型
Cache-Control: 控制http缓存
content-type: 表示后面的文档属于什么MIME类型
Set-Cookie: 设置cookie
Location: 告诉浏览器加载一个页面
expires: 资源过期时间
ETag: 资源经哈希运算生成的字符串
Last-Modified: 资源上次修改的时间
Access-Control-Allow-Origin: 允许跨域的源地址
Access-Control-Allow-Credientials: 值为true表示CORS下允许携带cookie
```

Content-Type属性值:

1. `application/x-www-form-urlencoded`：浏览器原生form表单，如果不设置enctype属性，最终会以这种方式提交数据。数据放在body中，按照key1=val1&key2=val2的方式进行编码，其中key和val都进行了URL转码
2. `multipart/form-data`：也是一种常见的POST提交方式，通常表单上传文件时使用
3. `application/json`：服务器消息主体是序列化后的JSON字符串
4. `text/xml`：主要用来提交XML格式的数据。

常见的HTTP请求方法：

- GET：向服务器获取数据
- POST：将实体提交到指定的资源，通常会造成服务器资源的修改
- PUT：上传文件，更新数据
- DELETE：删除服务器上的对象
- HEAD：获取报文首部，与GET 相比，不返回报文的主体部分
- OPTIONS：询问支持的请求方法，用来跨域请求
- CONNECT：要求在与代理服务器通信时建立隧道，使用隧道进行TCP通信
- TRACE：回显服务器收到的请求，主要用于检测或诊断。

3. 讲讲304

304 ： 如果客户端发送了一个带条件的get请求且该请求已被允许，而文档的内容（自上次访问以来或者根据请求的条件）并没有改变，则服务器应当返回这个304 状态码。

通俗来说就是：服务器为了提高网站的访问速度，对之前访问的部分页面指定缓存机制，当客户端在此对这些页面进行请求，服务器会根据缓存内容判断页面与之前是否相同，若相同便直接返回304，在此客户端调用缓存内容，不必进行二次下载。

状态码304不应该认为是一种错误，而是对客户端 **有缓存情况下** 服务端的一种响应。

产生较多304状态码的原因:

- 页面更新周期长或不更新
- 纯静态页面或强制生成静态HTML

304状态码出现过多会造成以下问题:

- 网站快照停止
- 收录减少
- 权重下降

4. 说一下http和https

> https 的 SSL加密是在传输层实现的。

（1）http和https的基本概念

`http`：超文本传输协议，是互联网上应用最为广泛的一种网络协议，是一个客户端和服务端请求和应答的标准（TCP），用于从WWW服务器传输超文本到本地浏览器的传输协议，它可以使浏览器更加高效，使网络传输减少。

`https`：是以安全为目标的HTTP通道，简单讲是http的安全版，即http下加入SSL层，https的安全基础是SSL，因此加密的详细内容就需要SSL。https协议的主要作用是，建立一个信息安全通道，来确保数组的传输，确保网站的真实性。

（2）http 和 https的区别

http传输的数据都是未加密的，也就是明文的，网景公司设置了SSL协议来对http协议传输的数据进行加密处理，简单来说https是由http和SSL协议构建的可进行加密传输和身份认证的网络协议，比http的安全性更高。

- https协议需要ca证书，费用较高
- http协议是超文本传输协议，信息是明文传输，https则是具有安全性的SSL加密传输协议
- 使用不同的连接方式，端口也不同，http端口是80，https端口是443
- http的连接很简单，是无状态的；https协议是由SSL+http协议构建的可进行加密传输、身份认证的网络协议，比http安全。

（3）https工作原理

1. 客户端使用https url 访问服务器，则要求web服务器建立ssl连接
2. web服务器接收到客户端的请求之后，会将网站的证书（证书中包含了公钥），返回或者说传输给客户端。
3. 客户端和web服务器端开始协商SSL链接的安全等级，也就是加密等级
4. 客户端浏览器通过双方协商一致的安全等级，建立会话秘钥，然后通过网站的公钥来加密会话秘钥，并传送给网站。
5. web服务器通过自己的私钥解密出会话秘钥
6. web服务器通过会话秘钥加密与客户端之间的通信

5. HTTP1.0 ， 1.1， 2.0

HTTP1.0比较1.1

- **连接方面**：http1.0默认使用非持久连接，http1.1默认使用持久连接。http1.1通过使用持久连接来使多个http请求复用同一个TCP连接，以此来避免使用非持久连接时每次需要建立连接的时延。
- **资源请求方面**：http1.0存在带宽浪费现象，比如客户端只是需要某个对象的一部分，而服务器却将整个对象送过来了，并且不支持断点功能。http1.1则在请求头引入了range头域，它允许只请求资源的某个部分，即返回码是206，便于充分利用带宽和连接。
- **缓存方面**：在http1.0中主要使用 header里的 `if-Modified-Since`、`Expires`来作为缓存判断标准，在http1.1则引入了更多的缓存控制策略，例如 `Etag`、`If-Unmodified-Since`、`If-Match`、`If-None-Match`等更多可供选择的缓存头来控制缓存策略。
- http1.0不支持Host头部，Http1.1支持，可以实现虚拟主机
- http1.1新增了很多的请求方法，如PUT、HEAD。还新增了24个错误状态响应码，如409（conflict）表示请求的资源与资源当前状态发生冲突。

HTTP1.1比较2.0

- **二进制协议**：HTTP2.0是一个二进制协议。在http1.1中，报文的头信息必须是文本（ASCLL编码），数据体可以是文本，也可以是二进制。HTTP2则是一个彻底的二进制协议，头信息和数据体都是二进制，并且统称为“帧”，可以分为头信息帧和数据帧。
- **多路复用**：http2实现了多路复用，http2仍然复用TCP连接，但是在一个连接里，客户端和服务器都可以同时发送多个请求或回应，而且不用按照顺序一一发送。就是说在同一时间里可以发送多个请求。
- **头部信息压缩**：对报文的头部进行压缩，在客户端和服务器端都维护着一份字典记录着头部对应的索引。
- **服务器推送**：HTTP2允许服务器未经请求，主动向客户端发送资源，这叫服务器推送。推送的是静态资源。

6. HTTP状态码

状态码的类别：

类别	原因	描述
1XX	Informational（信息性状态码）	接收的请求正在处理
2XX	Success（成功状态码）	请求正常处理完毕
3XX	Redirection（重定向状态码）	需要进行附加操作---完成请求
4XX	Client Error（客户端错误状态码）	服务器无法处理请求
5XX	Server Error（服务器错误状态码）	服务器请求出现错误

（1）1XX 通知

- 100 Continue 客户端重新发请求
- 101 更改协议 http，https，http1.0,1.1，2.0

（2）2XX Success 成功状态码

- **200 OK 请求成功。一般用于GET与POST请求**
- **201 Created 已创建。成功请求并创建了新资源**
- **202 Accepted 已接受。已经接收请求，但未处理完成**
- **204 No Content 无内容。但是报文不含实体的主体部分**
- **205 Reset Content 重置内容。服务器处理成功，用户终端应重置文档视图**
- 206 Partial Content 部分内容。服务器成功处理了部分GET请求

（3）3XX 重定向状态码

- 300 Multiple Choices 多种选择。请求的资源可包括多个位置，相应可返回一个资源特征与地址的列表用户终端选择
- **301 Moved Permanently 永久性重定向（移动）**
- **302 Found 临时重定向（移动）**
- **303 See Other 查看其它地址。资源存在另一个URL**
- **304 Not Modified 未修改。允许访问资源，但你实体为空**
- 305 Use Proxy 使用代理。
- 306 Unused 已经被废弃的HTTP状态码
- 307 Temporary Redirect 临时重定向

（4）4XX 客户端错误状态码

- **400 Bad Request 请求报文语法错误**
- **401 Unauthorized 请求要求用户的身份认证**
- **402 Payment Required 保留，将来使用**
- **403 Forbidden 服务器理解请求客户端的请求，但是拒绝执行此请求**
- **404 Not Found 找不到资源**
- **405 Method Not Allowed 不支持的请求方法**

（5）5XX 服务器端错误状态码

- **500 Internal server error 服务器在执行请求时发生了错误**
- 501 Not Implemented 服务器不支持请求的方法
- **502 Bad Gateway （代理服务器与上行服务器出现问题）作为网关或者代理工作的服务器执行请求时，从远程服务器接收到了一个无效的响应。**
- **503 Service unavailable 服务器暂时超负荷或正在停机处理，无法处理请求**（服务器不可用）

7. PUT和POST都是给服务器发送新增资源，有什么区别?

PUT和POST方法的区别是，PUT方法是幂等的：连续调用一次或者多次的效果相同（无副作用），而POST方法是等幂的。

除此之外，PUT的URI指向是具体单一资源，而POST可以指向资源集合。

比如说，在开发一个博客系统，当我们要创建一篇文章的时候往往用 `POST https://www.jianshu.com/articles`，这个请求的语义是，在article的资源集合下创建一篇新的文章，如果我们多次提交这个请求会创建多个文章，这是非幂等的。

而 `PUT https://www.jianshu.com/articles/89890798` 的语义是更新对应文章下的资源（比如修改作者的名字），这个URI指向的就是单一资源，而且是幂等的，比如把“aaaa”改成“bbbb”，提交多少次都是修改成“bbbb”。

二、DNS协议

1. DNS协议是什么?

DNS（Domain Name System）是域名系统的缩写，**提供一种主机名到IP地址的转换服务。**它是由一个分层的DNS服务器组成的分布式数据库，是定义了主机如何查询这个分布式数据库的方式的应用层协议。能够方便人访问互联网，而不用去记住能够被机器直接读取的IP数串。

作用：将域名解析为IP地址，客户端向DNS服务器（DNS服务器有自己的IP地址）发送域名查询请求，DNS服务器告知客户机Web服务器的IP地址。

2. DNS同时使用TCP和UDP协议?

DNS占用53端口，同时使用TCP和UDP协议。

（1）在区域传输的时候使用TCP协议

- 辅域名服务器会定时（一般3小时）向主域名服务器进行查询以便了解数据是否有变动。如果有变动，会执行一次区域传送，进行数据同步。区域传送使用TCP而不是UDP，因为数据同步传送的数据量比一个请求应答的数据量要多得多。
- TCP是一种可靠连接，保证了数据的准确性。

（2）在域名解析的时候使用UDP协议

- 客户端向DNS服务器查询域名，一般返回的内容都不超过512字节，用UDP传输即可。不用经过三次握手，这样DNS服务器负载更低，响应更快。理论上说客户端也可以指定向DNS服务器查询时用TCP，但事实上，很多DNS服务器进行配置的时候，仅支持UDP查询。

3. DNS完整的查询过程

DNS服务器解析域名的过程：

- 首先会在**浏览器的缓存**中查找对应的IP地址，如果差查找到直接返回，若找不到继续下一步
- 将请求发送给 **本地DNS服务器**，在本地域名（DNS）服务器缓存中查询，如果查找到直接将结果返回，若找不到继续下一步
- 本地域名服务器向 **根域名服务器** 发送请求，根域名服务器会返回一个所查询域的顶级域名服务器地址
- 本地域名服务器向 **顶级域名服务器** 发送请求，接受请求的服务查询自己的缓存，如果有记录，就返回查询结果，如果没有就返回相关的下一级的权威域名服务器的地址
- 本地域名服务器向**权限域名服务器**发送请求，域名服务器返回对应的结果
- 本地域名服务器将返回结果保存在缓存中，以便下一次使用
- 本地DNS服务器将返回结果返回给浏览器

4. 迭代查询和递归查询

实际上，DNS解析是一个包含迭代查询和递归查询的过程。

- **递归查询**指的是查询请求发出后，域名服务器代为向下一级域名服务器发出请求，最后向用户返回查询的最终结果。只用递归查询，用户只需要发出一次查询请求。
- **迭代查询**指的是查询请求后，域名服务器返回单次查询的结果。下一级查询由用户自己请求。使用迭代查询，用户需要发出多次的查询请求。

三、TCP与UDP

> TCP是传输控制协议，UDP是用户数据报协议。都是基于传输层的通信协议。

1. TCP和UDP的区别

1. TCP是面向连接的，UDP是无连接的即发送数据前不需要先建立连接。
2. TCP提供可靠的服务，也就是说，提供TCP连接的数据，无差错，不丢失，不重复，且按序到达；UDP尽最大努力交付，即不保证可靠交付。
3. TCP是面向字节流，UDP面向报文，并且网络出现拥塞不会使得发送速率降低
4. TCP只能是一对一，UDP支持1对1,1对多，多对多
5. TCP是面向连接的可靠传输，UDP是不可靠的。
6. TCP头部较大为20字节，而UDP只有8字节。

2. TCP的三次握手和四次挥手

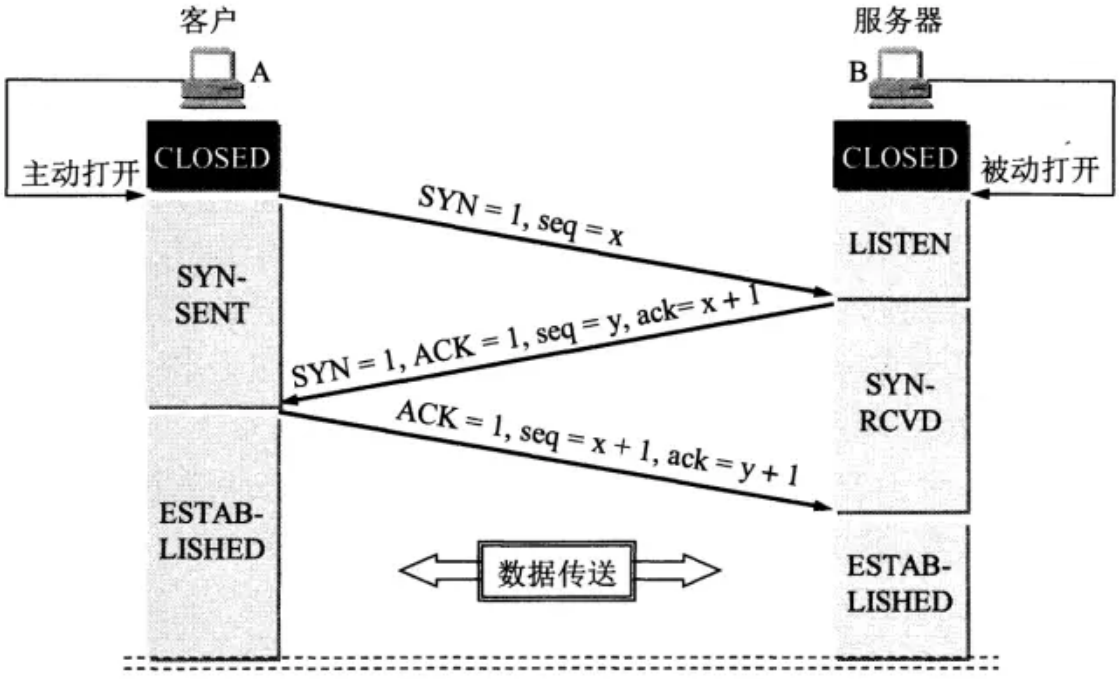


图 5-28 用三报文握手建立 TCP 连接

三次握手

三次握手其实就是指建立一个TCP连接，需要客户端和服务器总共发送3个包。进行三次握手的主要作用是为了确认双方的接收能力和发送能力是否正常、指定自己的初始化序列号为后面的可靠性传送做准备。实质上其实就是连接服务器指定端口，建立TCP连接，并同步连接双方的序列号和确认号，交换TCP窗口大小信息。

刚开始客户端处于closed的状态，服务端处于listen状态：

（1）**第一次握手**：客户端给服务器发送一个SYN报文，服务器知道自己可以接收到客户端发送的报文

`该报文的SYN=1，初始序号seq=x。`（SYN=1的报文段不能携带数据，但要消耗一个序号）

（2）**第二次握手**：服务器向客户端发送**报文**，客户端知道自己可以接收服务端发送的报文，知道自己发送的报文能被服务端接收。

`该报文的SYN = 1，ACK=1，确认号ack = x+1，初始序号 seq = y。`

（3）**第三次握手**：客户端给服务端发送报文，服务端知道自己发送的报文能被客户端接收。此时，客户端处于Establish状态，服务端收到ACK报文后，也处于establish状态，此时，双方已经建立了连接。

`该报文的ACK=1，确认号ack=y+1，序号seq=x+1`（初始为seq=x，第二个报文段所以要+1，这个是针对客户端来说的）ACK报文段可以携带数据，不携带数据则不消耗序号。

> 注：SYN / ACK / FIN 为TCP报文头部的一个标识，seq为报文的序列号,ack为报文的确认号（并不是ACK）。
>
> SYN = 1，seq = x对应的是ACK = 1，ack = x+1。
>
> - `SYN`：同步位。发送连接请求或连接请求确认时才为1，其余状态为0。
> - `seq`：序号位，刚开始发送时随机的
> - `ACK`：确认位。
> - `ack`：确认号。ack = seq（上一个报文信息的seq）+1 。是同一端发出的哦。

为什么要三次握手？两次不行吗？

> 采用三次握手主要是为了防止已经失效的连接请求报文突然又传送到了服务器，因而产生错误。为了确认双方的接收能力和发送能力都正常。

如果采用两次握手会出现下面这种情况：

假设有一种场景，客户端发送的第一个请求连接报文并且没有丢失，只是被滞留的时间太长了。由于TCP的客户端没有迟迟没有收到确认报文，以为服务器没有收到，此时重新向服务器发送报文，收到了服务器的确认，建立了连接，数据传输完毕后，就释放了连接。而现在第一个请求到达了服务端，这个请求已经报废了，但是又会建立新连接，此时客户端忽略服务端发来的确认，也不发送数据，则服务器端一直等到客户端发送数据，浪费资源。

如果采用的是三次握手，就算那一次是失效的报文传输过来了，服务端接受到了那条失效报文并且回复了确认报文，但是客户端不会再次发出确认，由于服务器收不到确认，就知道客户端并没有请求连接。

为什么不四次握手?

既然三次握手已经足够了，那多一次就是浪费资源。

四次挥手

> TCP使用四次挥手的原因是因为TCP的连接是全双工的，所以需要双方分别释放到对方的连接，单独一方连接释放，只代表不能再向对方发送数据，连接处于的是半释放状态。
>
> 最后一次挥手中，客户端会等待一段时间再关闭的原因，是为了防止发送给服务器的确认报文段丢失或者出错，从而导致服务器端不能正常关闭。

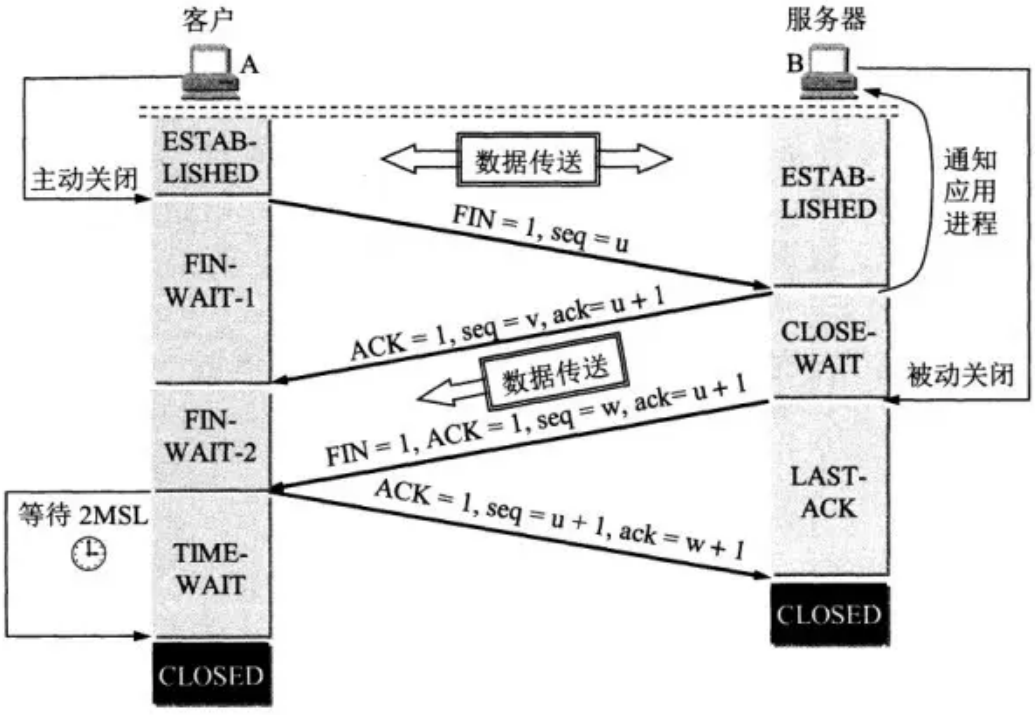


图 5-29　TCP 连接释放的过程

简单来说就是一下四步：

1. **第一次挥手**：若客户端 A 认为数据发送完成，则它需要向服务器 B 发送连接释放请求。

2. **第二次挥手**：服务端收到连接释放请求后，会告诉应用层要释放TCP连接。然后会发送ACK包，并进入 **CLOSE_WAIT**状态，表示客户端到服务端的连接已经释放，不再接受客户端发送的数据。但是因为T**CP连接是双向的**，所以服务器仍然可以发送数据给客户端。
3. **第三次挥手**：服务端如果此时还有没有发送完的数据会继续发送，完毕后会向客户端发送链接释放请求，然后服务端便进入 **LAST-ACK** 状态。
    - PS：通过延迟确认技术（通常有时间限制，否则对方会误以为需要重传），可以将第二次和第三次握手合并，延迟ACK包的发送。
4. **第四次握手**：客户端收到释放请求后，向服务端发送确认应答，此时客户端进入 **TIME-WAIT** 状态。该状态会持续2MSL（最大生存期，指报文段在网络中生存的时间，超时会被抛弃）时间，若该时间段内没有服务端的重发请求的话，就进入 **CLOSE**状态。当服务器收到确认应答后，也便进入**CLOSE**状态。

3. 递归查询和迭代查询

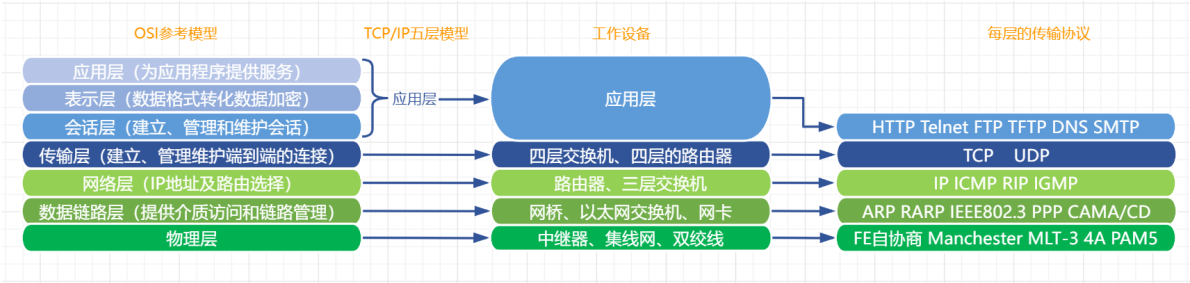
实际上，DNS解析是一个包含迭代查询和递归查询的过程。

- **递归查询**指的是查询请求发出后，域名服务器代为向下一级域名服务器发出请求，最后向用户返回查询的最终结果。使用递归查询，用户只需要发出一次查询请求。
- **迭代查询**指的是查询请求后，域名服务器返回单次查询结果。下一级的查询由用户自己请求。使用迭代查询，用户需要发出多次的查询请求。

一般我们向本地DNS服务器发送请求的方式是递归查询，因为我们只需要发出一次请求，然后本地DNS服务器返回给我们最终的请求结果。

而本地DNS服务器向其他域名服务器请求的过程是迭代查询的过程，因为每一次域名服务器只返回单次查询的结果，下一级的查询由本地DNS服务器自己进行。

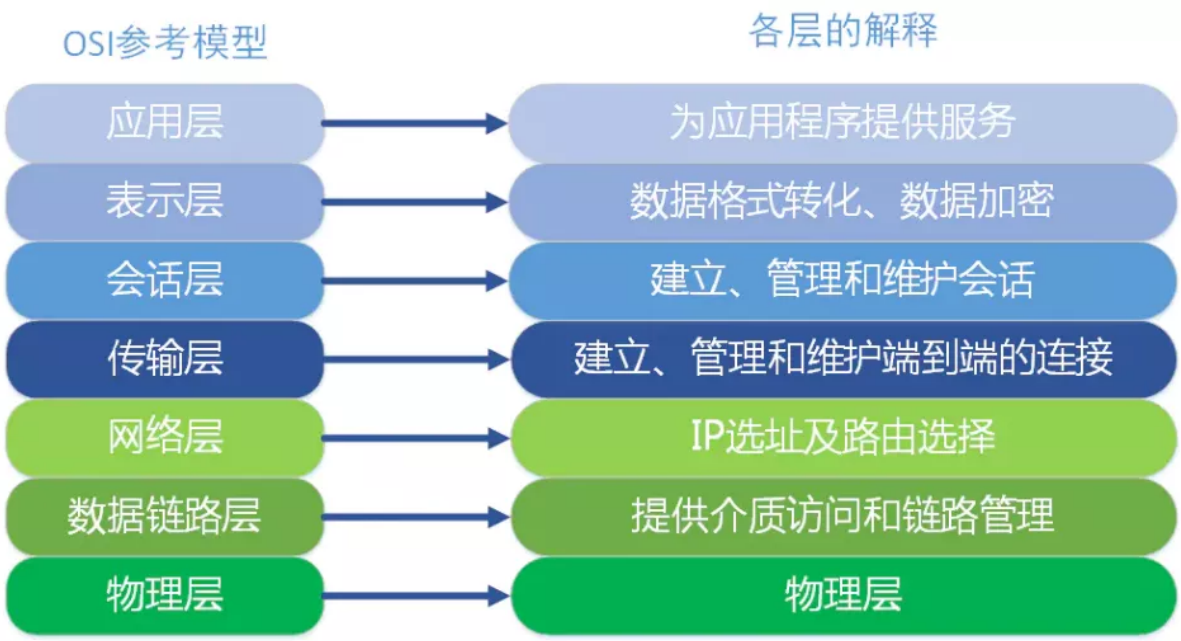
四、网络模型



OSI七层网络模型	TCP/IP四层概念模型	对应网络协议
应用层（Application）	应用层	HTTP、TFTP, FTP, NFS, WAIS、SMTP
表示层（Presentation）		Telnet, Rlogin, SNMP, Gopher
会话层（Session）		SMTP, DNS
传输层（Transport）	传输层	TCP, UDP
网络层（Network）	网络层	IP, ICMP, ARP, RARP, AKP, UUCP
数据链路层（Data Link）	数据链路层	FDDI, Ethernet, Arpanet, PDN, SLIP, PPP
物理层（Physical）		IEEE 802.1A, IEEE 802.2到IEEE 802.11

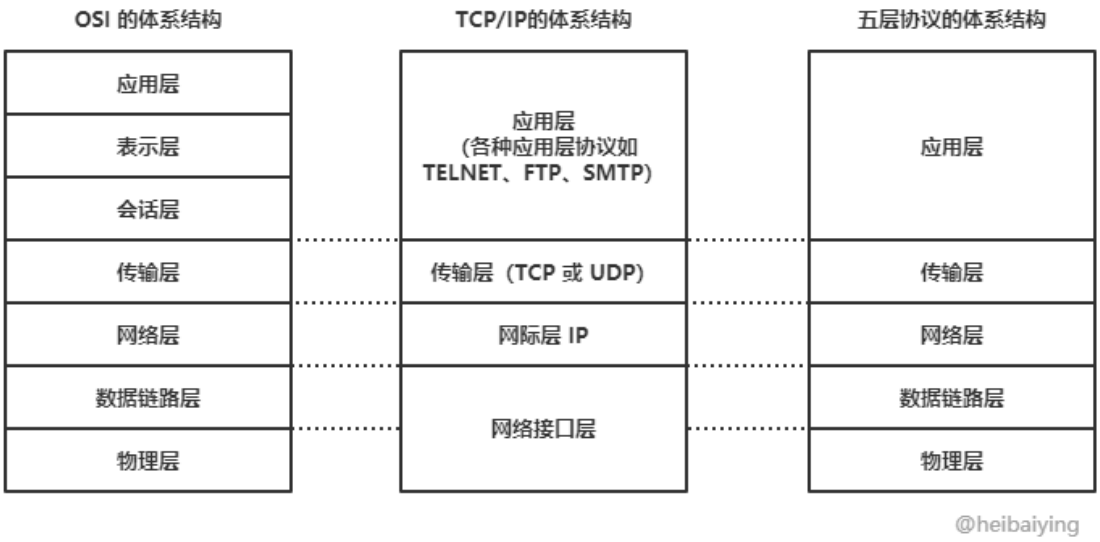
1. OSI七层模型

`ISO`为了更好的使网络应用更为普及，推出了`OSI`参考模型。



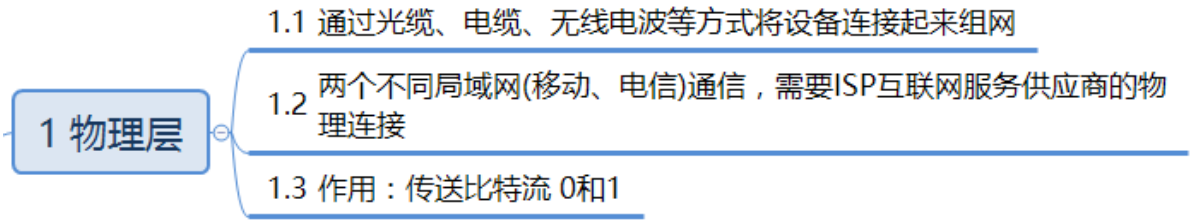
2. TCP/IP五层协议

`TCP/IP`五层协议和`OSI`的七层协议对应关系如下：



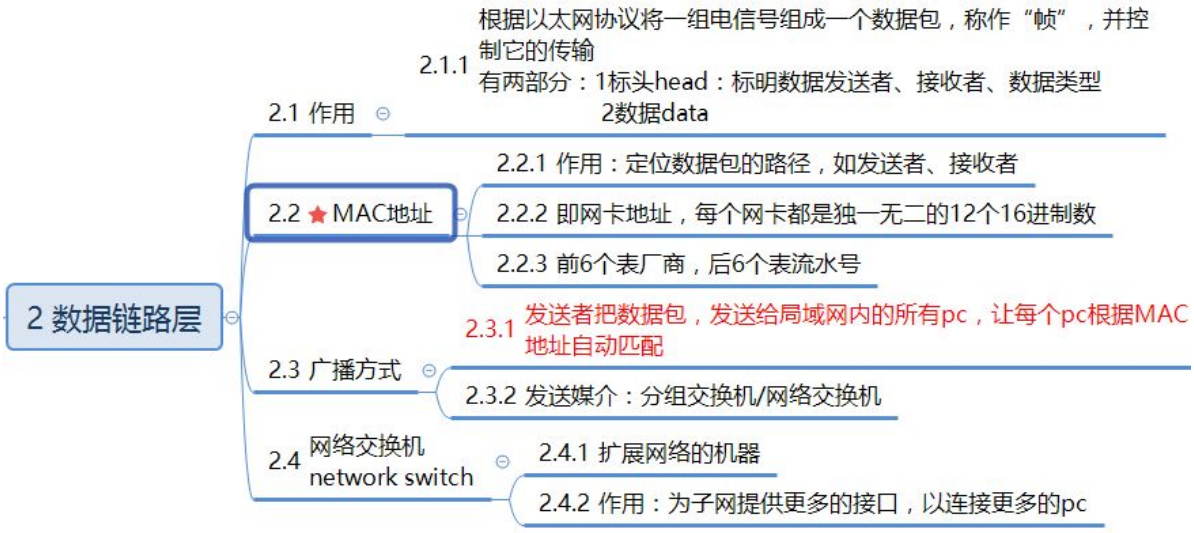
物理层

确保数据可以在各种物理媒介上进行传输，为数据的传输提供可靠的环境。



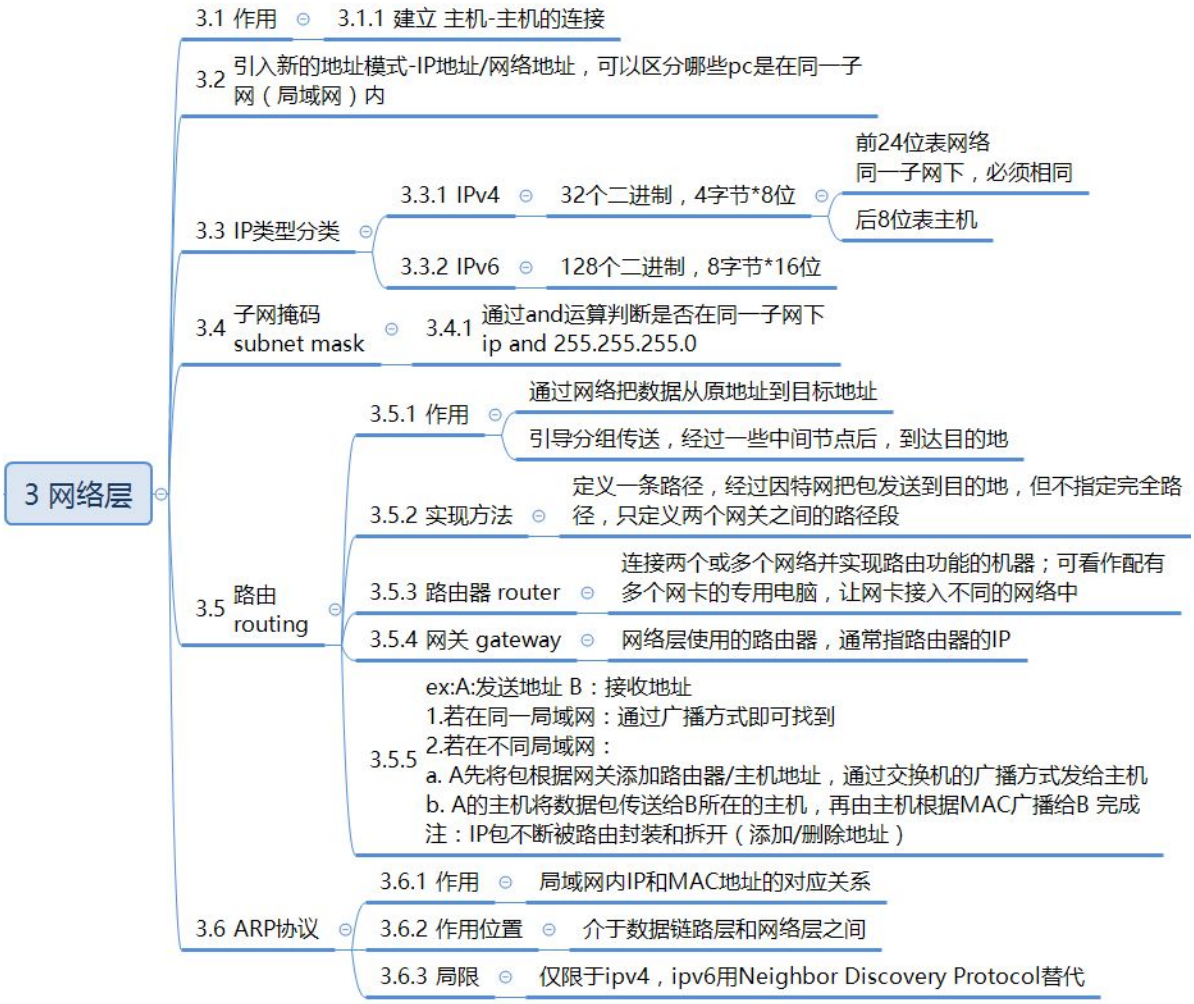
数据链路层

负责将网络层交下来的IP数据报封装成帧，并在链路的两个相邻节点间传送帧，每一帧都包含数据和必要的控制信息。



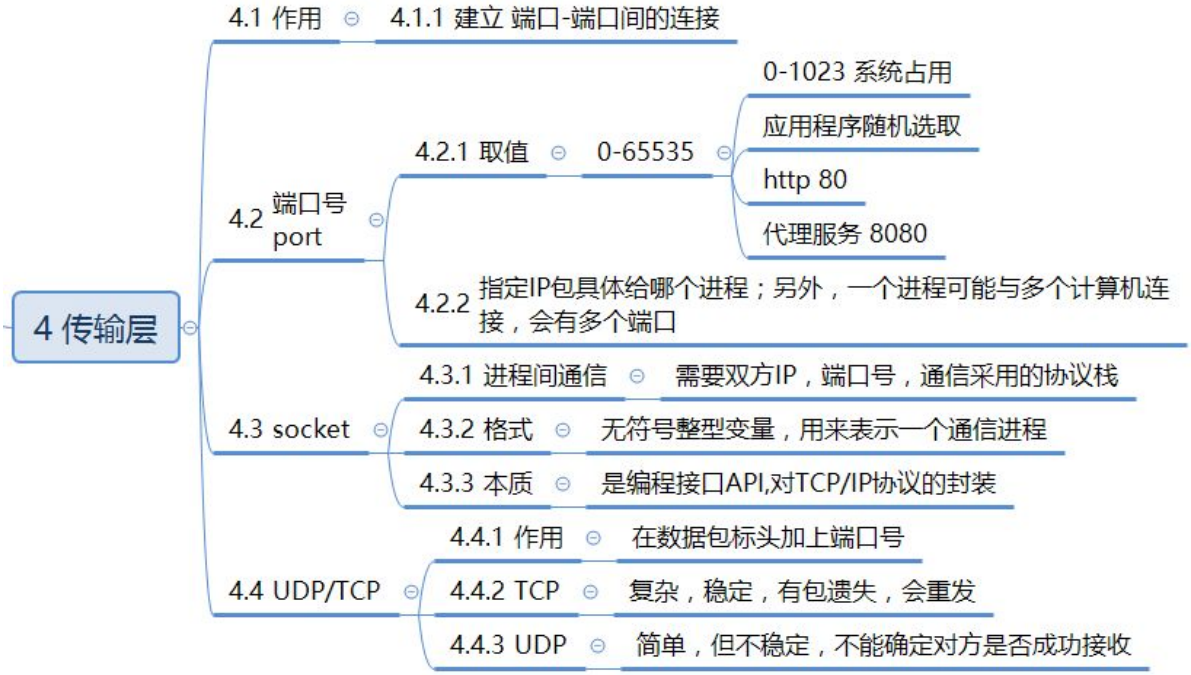
网络层

IP寻址和路由选择



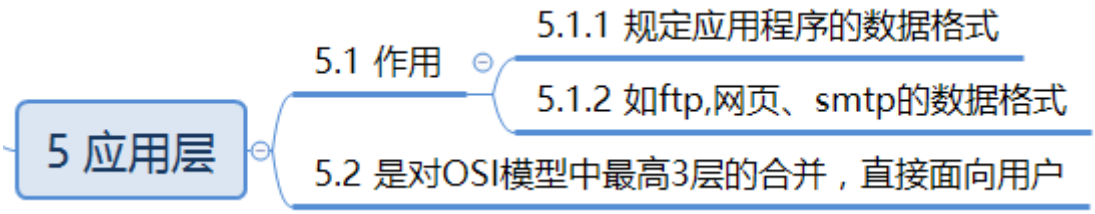
传输层

为两台主机中的进程提供通信服务。建立、维护、管理端到端的连接。



应用层

直接为应用进程提供服务。



五、WebSocket

1. 对WebSocket的理解

WebSocked是HTML5提供的一种浏览器与服务器进行**全双工通讯**的网络技术，属于应用层协议。它基于TCP传输协议，并复用HTTP的握手通道。浏览器和服务器只需要完成一次握手，两者之间就直接可以创建持久性的连接，并进行双向数据传输。

WebSocket的出现就解决了半双工通信的弊端。它最大的特点是：**服务器可以向客户端主动推动消息，客户端也可以主动向服务器推送消息。**

WebSocket原理：客户端向WebSocket服务器通知（notify）一个带有所有接收者ID（recipients IDs）的事件（event），服务器接收后立即通知所有活跃的（active）客户端，只有ID在接收者ID序列中的客户端才会处理这个事件。

WebSocket特点如下：

- 支持双向通信，实时性更强
- 可以发送文本，也可以发送二进制数据
- 建立在TCP协议之上，服务端的实现比较容易
- 数据格式比较轻量，性能开销小，通信高效
- 没有同源限制，客户端可以与任意服务器通信
- 协议标识符是ws（如果加密，则为wss），服务器网址就是URL
- 与http协议有良好的兼容性。默认端口也是80和443，并且握手阶段采用http协议，因此握手时不容易屏蔽，能通过各种http代理服务器。

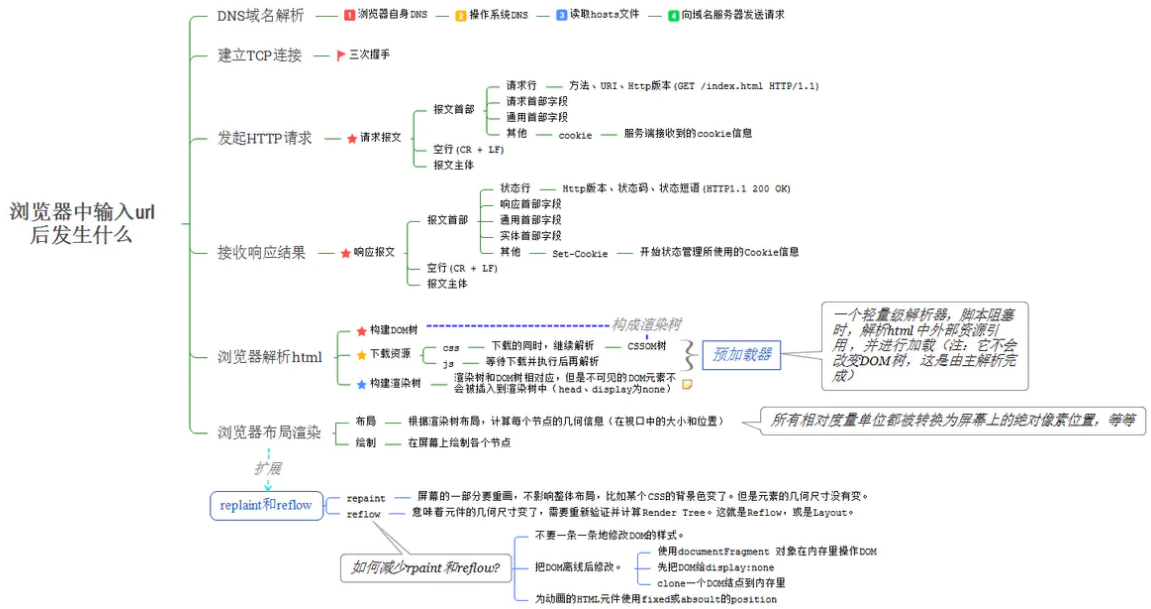
WebSocket的使用方法：

在客户端中：

```
//在index.html中直接写WebSocket，设置服务端的端口为9999
let ws = new WebSocket('ws://localhost:9999');
//在客户端与服务端建立连接后触发
ws.onopen = function(){
    console.log("Connection open.");
    ws.send('hello');
};
//在服务端给客户端发来消息的时候触发
ws.onmessage = function(res) {
    console.log(res);  //打印的是MessageEvent对象
    console.log(res.data);  //打印的是收到的消息
};
//在客户端与服务端建立关闭后触发
ws.onclose = function(evt){
    console.log("Connection closed");
};
```

六、场景题

1. 在地址栏输入一个URL之后会发生什么?



输入URL后，首先**需要找到这个URL域名服务器ip**，为了寻找这个ip，**浏览器首先会寻找缓存**，查看缓存中是否有记录，缓存的查找记录为：浏览器缓存-》系统缓存-》路由器缓存，缓存中没有则**查找系统的hosts文件**是否有记录，如果没有则**查询DNS服务器**，得到服务器的ip地址后，浏览器根据这个ip以及相应的端口号，**构造一个http请求**，这个请求报文会包括这次请求的信息，主要是请求方法，请求说明和请求附带的数据，**并将这个http请求封装在一个tcp包中**，这个tcp包会经过传输层，网络层，数据链路层，物理层到达服务器，**服务器解析这个请求来做出响应**，返回相应的HTML给浏览器，因为HTML是一个树型结构，**浏览器根据这个HTML来构建DOM树**，在dom数的构建过程中如果遇到JS脚本和外部JS连接，则会停止构建DOM树来执行和下载相应的代码，这会造成阻塞，这就是为什么推荐JS代码应该放在HTML代码后面，之后根据外部样式，内部样式，**内联样式构建一个css对象模型树CSSOM树，构建完成后和DOM树合并为渲染树**，这里主要做的是排除非视觉节点，比如script，meta标签和排除display为none的节点，之后**进行布局**，布局主要是确定各个元素的位置和尺寸，之后是**渲染页面**，因为html文件中会含有图片、视频、音频等资源，在解析DOM的过程中，遇到这些都会进行并行下载，浏览器对每个域的并行下载数量有一定的限制，一般是4-6个。

当然在这些所有的请求中我们还需要关注的就是缓存，缓存一般通过Cache-Control、Last-Modify、Expires等首部字段控制。Cache-Control和Expires的区别在于Cache-Control使用相对时间，Expires使用的是基于服务器端的绝对时间，因为存在时差问题，一般采用Cache-Control，在请求这些有设置了缓存的数据时，会先查看是否过期，如果没有过期则直接使用本地缓存，过期则请求并在服务器校验文件是否修改，如果上一次响应设置了ETag值会在这次请求的时候作为If-None-Match的值交给服务器校验，如果一致，继续校验Last-Modified，没有设置ETag则直接验证Last-Modified，在决定是否返回304。

【直观版】：

1. 浏览器解析URL获取协议，域名，端口，路径
2. 查看浏览器是否有资源的缓存
    1. 有。判断是否过期
        1. 没过期。直接读取缓存
        2. 过期。
            1. Etag和If-None-Match

        2. Last-Modify和If-Modified-Since
        3. 文件修改了则把新资源发给浏览器（状态码200），没修改则告诉浏览器读取缓存（状态码304）
    2. 没有则进行下一步
3. 首先进行DNS解析
    1. 寻找浏览器是否存在缓存，若没有
    2. 寻找操作系统是否存在缓存，若没有
    3. 寻找hosts文件中是否有域名和ip的对应关系，若没有
    4. 查找路由器中是否有缓存
    5. 寻找DNS服务器是否没缓存，若没有
    6. 向根域名服务器发送请求
4. 生成HTTP请求
5. 建立TCP连接，三次握手
    1. 客户端发送一个SYN=1,Seq=X的TCP包
    2. 服务端发回一个SYN=1,ACK=X+1，Seq=Y的TCP包
    3. 客户端发送ACK=Y+1，Seq=Y + 1的TCP包
6. 如果是HTTP请求

    对HTTP报文进行报文分割并标记序号和端口号
7. 如果是HTTPS请求
    1. 将HTTP报文交给TLS处理，TLS和服务端进行TLS握手，交换版本信息，加密算法，压缩算法，随机数（浏览器一个，客户端一个）。
    2. 服务端发送证书，浏览器用CA的公钥对其进行验证。
    3. 浏览器用服务端的公钥加密生成的预备主密码发送给服务端，两个随机数和预备主密码生成主密码
    4. 使用主密码生成对称加密的密钥对，消息认证码的密钥对，对称加密的CBC分组（分组模式）需要的初始化向量密钥对。
    5. 握手之后进行加密，对HTTP报文分组，分组后压缩，压缩后的数据和MAC一起加密。
    6. 对称加密保障私密性，消息认证码保障完整性，数字证书保障认证，防止中间人攻击。
8. 对TCP报文打包，加入源IP地址和目标IP地址。
9. 根据目标IP地址和路由表，查询下一跳路由。使用ARP查询下一跳路由的MAC地址。
10. 对IP报文打包并附上MAC地址。
11. 发送数据，服务端接收到请求并返回响应。
12. 浏览器接收到HTTP响应，关闭TCP连接或保持复用，四次挥手。
13. （如果返回了HTML）根据响应头的字符集进行解码
14. 如果响应头没有字符集，则浏览器会默认用一套解码规则，当解析html解析到meta标签中的编码规则时，则替换成新的解码方式重新解码。
15. 资源预解析，会将一些请求资源提前加入请求队列中
16. 解析HTML为DOM树
    1. 标记化（tokenizing）: 将HTML解析成标记
    2. 构建树（tree construction）: 根据标记生成DOM树
17. 解析CSS为CSSOM
18. 根据DOM树和CSSOM生成DOM渲染树

    从DOM的根节点遍历所有可见节点，对其应用对应的CSSOM规则。不可见节点包括（script, meta标签， 被css隐藏的节点）
19. 布局：浏览器获取每个渲染对象的位置和尺寸

20. 绘制：将计算好的像素绘制到屏幕
21. 渲染层/合成层合并

【简单总结就是】：

https://www.jianshu.com/p/c1dfc6caa520

1. DNS解析
2. 建立TCP连接
3. 发起HTTP请求
4. 接受响应结果
5. 浏览器解析HTML
6. 浏览器布局渲染