参考网址其一 <http://interview.wzcu.com/%E7%BD%91%E7%BB%9C%E4%B8%8E%E5%8D%8F%E8%AE%AE/%E5%8D%8F%E8%AE%AE.html>

http协议和https协议的区别主要是：传输信息安全性不同、连接方式不同、端口不同、证书申请方式不同

一、传输信息安全性不同

http协议：是超文本传输协议，信息是明文传输。如果攻击者截取了Web浏览器和网站服务器之间的传输报文，就可以直接读懂其中的信息。

https协议：是具有安全性的ssl加密传输协议，为浏览器和服务器之间的通信加密，确保数据传输的安全。

二、连接方式不同

http协议：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。

三、端口不同

http协议：使用的端口是80。

https协议：使用的端口是443．

四、证书申请方式不同

http协议：免费申请。

https协议：需要到ca申请证书，一般免费证书很少，需要交费

Csrf攻击

       攻击者盗用了你的身份，以你的名义发送恶意请求，对服务器来说这个请求是完全合法的，但是却完成了攻击者所期望的一个操作，比如以你的名义发送邮件、发消息，盗取你的账号，添加系统管理员，甚至于购买商品、虚拟货币转账等。 如下：其中Web A为存在CSRF漏洞的网站，Web B为攻击者构建的恶意网站，User C为Web A网站的合法用户。

****CSRF攻击攻击原理及过程如下：****

       1. 用户C打开浏览器，访问受信任网站A，输入用户名和密码请求登录网站A；

       2.在用户信息通过验证后，网站A产生Cookie信息并返回给浏览器，此时用户登录网站A成功，可以正常发送请求到网站A；

       3. 用户未退出网站A之前，在同一浏览器中，打开一个TAB页访问网站B；

       4. 网站B接收到用户请求后，返回一些攻击性代码，并发出一个请求要求访问第三方站点A；

       5. 浏览器在接收到这些攻击性代码后，根据网站B的请求，在用户不知情的情况下携带Cookie信息，向网站A发出请求。网站A并不知道该请求其实是由B发起的，所以会根据用户C的Cookie信息以C的权限处理该请求，导致来自网站B的恶意代码被执行。

http 1.0和http2.0的区别

ttp1.1在传输消息的时候，1.0通常都要传输几次消息，一次无法给浏览器进行渲染，但是每一次传输的时候客户端都需要重新请求建立链接, 常说的3次握手4次挥手 http1.1通过引入长连接和流水线技术处理了这个问题，就是让tcp连接一直开着 直到客户端通知关闭了在关闭，这样再次传输数据的时候就不用了重新建立连接了，但是他还是需要按照顺序请求和响应的，也就是如果第一个请求没有得到响应就会造成后面的请求堵塞，引发队头堵塞问题，虽然可以建立多个tcp连接 ，但是会浪费很多的资源。比如需要传输：hello world，只能从h到d一个一个的传输，不能并行传输，因为接收端并不知道这些字符的顺序，所以并行传输在HTTP1.1是不能实现的

HTTP/2引入二进制数据帧和流的概念，其中帧对数据进行顺序标识，如下图所示，这样浏览器收到数据之后，就可以按照序列对数据进行合并，而不会出现合并后数据错乱的情况。同样是因为有了序列，服务器就可以并行的传输数据，这就是流所做的事情。

滑动窗口大小的设计

滑动窗口越大，效率就越高，但是存在一个问题，滑动窗口越大，那么就要有一个相对的缓冲区，缓冲区也就是一块内存。内存占用过大是不科学的；

而影响窗口大小的还有接收方的接受速率，如果接收方处理数据慢，而发送数据却发送得很快，就会导致丢包。此时就要限制发送发的窗口大小。因此就有了 流量控制机制 ！

而两台主机之间传输数据会有很多中间节点，也就是路由器，路由器的处理数据能力并不知道，为了防止丢包，因此就有了 拥塞控制机制。

滑动窗口 的大小取决于 拥塞窗口 和 流量窗口 中的最小值。

.TCP、UDP的区别

TCP(transfer control protocol)是一种面向连接的、可靠的、基于字节流的传输层通信协议。

特点：

1.面向连接

2.点到点的通信

3.高可靠性：三次握手

4.占用系统资源多、效率低

生活案例：打电话

UDP(User DatagramProtocol)是一种无连接的传输层协议，提供面向事务的简单不可靠信息传送服务。

特点：

1.非面向连接，传输不可靠，可能丢失

2.发送不管对方是否准备好，接收方收到也不确认

3.可以广播发送

4.开销小

生活案例：发短信

2.TCP建立连接三次握手，断开连接四次握手具体过程

1.建立连接三次握手：

第一次握手是客户端连接到服务器端；

第二次握手是服务器端接收到请求后，向客服端发送一个信息；

第三次是客服端向服务器端发送的，是对第二次握手信息的确认。

之后客户端和服务器端就开始通讯。

2.断开连接四次握手

断开连接的一端发送close请求是第一次握手；

另外一端接收到断开连接的请求之后需要对close进行确认，发送一次信息，这是第二次握手；

发送了确认消息之后还要再发close消息，用来关闭连接，这是第三次握手；

最初发送close请求的一端接收到消息之后，进入time\_wait状态，第四次握手是最初发送close请求的一端接收到消息之后，对消息的确认。

3.套接字Socket

Socket是网络传输层供给应用层的编程接口，使用Socket编程可以开发客户机和服务器应用程序，可以在本地网络上进行通信，也可以通过Internet在全球范围内通信

4.IP地址和端口

IP地址用来标识一台计算机，端口用来区分计算机上的应用程序。

IP地址好比每个人的地址（门牌号），端口好比是房间号，必须同时指定IP地址和端口号才能够正确的发送数据。

————————————————

当一个请求url的****协议、域名、端口****三者之间任意一个与当前页面url不同即为跨域

解决方法

【6】websocket

Websocket 是 HTML5 的一个持久化的协议，它实现了浏览器与服务器的全双工通信，同时也是跨域的一种解决方案。WebSocket 和 HTTP 都是应用层协议，都基于 TCP 协议。但是 WebSocket 是一种双向通信协议，在建立连接之后，WebSocket 的 服务器与 客户端都能主动向对方发送或接收数据。同时，WebSocket 在建立连接时需要借助 HTTP 协议，连接建立好了之后 client 与 server 之间的双向通信就与 HTTP 无关了。

【7】Nginx反向代理

Nginx 实现原理类似于 Node 中间件代理，需要你搭建一个中转 nginx 服务器，用于转发请求。

使用 nginx 反向代理实现跨域，是最简单的跨域方式。只需要修改 nginx 的配置即可解决跨域问题，支持所有浏览器，支持 session，不需要修改任何代码，并且不会影响服务器性能。

我们只需要配置nginx，在一个服务器上配置多个前缀来转发http/https请求到多个真实的服务器即可。这样，这个服务器上所有url都是相同的域 名、协议和端口。因此，对于浏览器来说，这些url都是同源的，没有跨域限制。而实际上，这些url实际上由物理服务器提供服务。这些服务器内的 javascript可以跨域调用所有这些服务器上的url。

————————————————

、http get跟head

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。

１５、http 401,403

400 bad request，请求报文存在语法错误

401 unauthorized，表示发送的请求需要有通过 HTTP 认证的认证信息

403 forbidden，表示对请求资源的访问被服务器拒绝

404 not found，表示在服务器上没有找到请求的资源

、http能不能一次连接多次请求，不等后端返回

http本质上市使用socket连接，因此发送请求，接写入tcp缓冲，是可以多次进行的，这也是http是无状态的原因

### **TCP握手为什么是三次，不能是两次？不能是四次？**

不能是两次主要有三个原因：

防止已过期的连接请求报文突然又传送到服务器，因而产生错误和资源浪费。

在双方两次握手即可建立连接的情况下，假设客户端发送 A 报文段请求建立连接，由于网络原因造成 A 暂时无法到达服务器，服务器接收不到请求报文段就不会返回确认报文段。

客户端在长时间得不到应答的情况下重新发送请求报文段 B，这次 B 顺利到达服务器，服务器随即返回确认报文并进入 ESTABLISHED 状态，客户端在收到 确认报文后也进入 ESTABLISHED 状态，双方建立连接并传输数据，之后正常断开连接。

此时姗姗来迟的 A 报文段才到达服务器，服务器随即返回确认报文并进入 ESTABLISHED 状态，但是已经进入 CLOSED 状态的客户端无法再接受确认报文段，更无法进入 ESTABLISHED 状态，这将导致服务器长时间单方面等待，造成资源浪费。

三次握手才能让双方均确认自己和对方的发送和接收能力都正常。

第一次握手：客户端只是发送处请求报文段，什么都无法确认，而服务器可以确认自己的接收能力和对方的发送能力正常；

第二次握手：客户端可以确认自己发送能力和接收能力正常，对方发送能力和接收能力正常；

第三次握手：服务器可以确认自己发送能力和接收能力正常，对方发送能力和接收能力正常；

可见三次握手才能让双方都确认自己和对方的发送和接收能力全部正常，这样就可以愉快地进行通信了。

告知对方自己的初始序号值，并确认收到对方的初始序号值。

TCP 实现了可靠的数据传输，原因之一就是 TCP 报文段中维护了序号字段和确认序号字段，通过这两个字段双方都可以知道在自己发出的数据中，哪些是已经被对方确认接收的。这两个字段的值会在初始序号值得基础递增，如果是两次握手，只有发起方的初始序号可以得到确认，而另一方的初始序号则得不到确认。

不需要四次是因为三次握手已经可以确认双方的发送接收能力正常，双方都知道彼此已经准备好，而且也可以完成对双方初始序号值得确认，也就无需再第四次握手了。

* 第一次握手：服务端确认“自己收、客户端发”报文功能正常。
* 第二次握手：客户端确认“自己发、自己收、服务端收、客户端发”报文功能正常，客户端认为连接已建立。
* 第三次握手：服务端确认“自己发、客户端收”报文功能正常，此时双方均建立连接，可以正常通信。

### **TIME-WAIT 状态为什么需要等待 2MSL**

2MSL，2 Maximum Segment Lifetime，即两个最大段生命周期

* 1个 MSL 保证四次挥手中主动关闭方最后的 ACK 报文能最终到达对端
* 1个 MSL 保证对端没有收到 ACK 那么进行重传的 FIN 报文能够到达

### **http，tcp，ip分别处于OSI哪一层？**

add: 2019-12-28

* 传输层协议：TCP、UDP、SCTP
* 网络层协议：IP、ARP、RARP、ICMP、IGMP、OSPF
* 应用层协议：http，FTP、SMTP、RIP、DNS

### **. OSI有哪几层，会画出来，知道主要几层的各自作用?**

应用层(数据): 确定进程之间通信的性质以满足用户需要以及提供网络与用户应用

表示层(数据): 主要解决拥护信息的语法表示问题，如加密解密

会话层(数据): 提供包括访问验证和会话管理在内的建立和维护应用之间通信的机制，如服务器验证用户登录便是由会话层完成的

运输层(段): 实现网络不同主机上用户进程之间的数据通信，可靠与不可靠的传输，传输层的错误检测，流量控制等

网络层(包): 提供逻辑地址（IP）、选路，数据从源端到目的端的传输

数据链路层(帧): 将上层数据封装成帧，用MAC地址访问媒介，错误检测与修正

物理层（比特流）：设备之间比特流的传输，物理接口，电气特性等

### **TCP/IP有哪几层，会画出来，知道所有层数的作用，会列举各层主要的协议名称?**

1. 应用层（TELNET、FTP、SMTP）
2. 运输层（TCP、UDP）
3. 网际层（IP、ICMP）
4. 网络接口层（PPP）

### **为什么要3次握手，4次挥手**

握手：防止已过期的连接请求报文突然又传送到服务器，因而产生错误

挥手：确保数据能够完成传输，但关闭连接时，当收到对方的FIN报文通知时，它仅仅表示对方没有数据发送给你了；但未必你所有的数据都全部发送给对方了，所以你可以未必会马上会关闭SOCKET,也即你可能还需要发送一些数据给对方之后，再发送FIN报文给对方来表示你同意现在可以关闭连接了，所以它这里的ACK报文和FIN报文多数情况下都是分开发送的

### **Cookies 和 Session的区别**

* cookie 是一种发送到客户浏览器的文本串句柄，并保存在客户机硬盘上，可以用来在某个WEB站点会话间持久的保持数据
* session其实指的就是访问者从到达某个特定主页到离开为止的那段时间。 Session其实是利用Cookie进行信息处理的，当用户首先进行了请求后，服务端就在用户浏览器上创建了一个Cookie，当这个Session结束时，其实就是意味着这个Cookie就过期了
* cookie数据保存在客户端，session数据保存在服务器端

### **单工、半双工、全双工的区别**

#### **单工**

单工是指信道的全部带宽都用于发送方给接收方发送数据，也就是说信息只能往一个方向传送。发送方只能发送数据，不需要有接收数据的能力；接收方只能接收数据，不需要有发送数据的能力。比如无线电广播就属于单工通信，广播电台只负责发送数据，听众的收音机只负责接收数据。

#### **半双工**

半双工就是通信的双方都可以发送数据和接收数据，但不能同时发送和接收数据。比如A向B发送数据时，信道的全部带宽都用于A向B发送数据，此时B不能向A发送数据；当A向B发完数据后，B可以向A发送数据，此时信道的全部带宽都用于B向A发送数据，此时A不能向B发送数据。比如无线对讲机就属于半双工通信。

#### **全双工**

全双工和半双工一样通信的双方都可以发送数据和接收数据，不同的地方在于双方收发数据可以同时进行。比如现代的电话通信就属于全双工通信

### **什么是 SYN洪泛攻击？如何防范?**

SYN洪泛攻击属于 DOS 攻击的一种，它利用 TCP 协议缺陷，通过发送大量的半连接请求，耗费 CPU 和内存资源。

****原理****：

* 在三次握手过程中，服务器发送 [SYN/ACK] 包（第二个包）之后、收到客户端的 [ACK] 包（第三个包）之前的 TCP 连接称为半连接（half-open connect），此时服务器处于 SYN\_RECV（等待客户端响应）状态。如果接收到客户端的 [ACK]，则 TCP 连接成功，如果未接受到，则会****不断重发请求****直至成功。
* SYN 攻击的攻击者在短时间内****伪造大量不存在的 IP 地址****，向服务器不断地发送 [SYN] 包，服务器回复 [SYN/ACK] 包，并等待客户的确认。由于源地址是不存在的，服务器需要不断的重发直至超时。
* 这些伪造的 [SYN] 包将长时间占用未连接队列，影响了正常的 SYN，导致目标系统运行缓慢、网络堵塞甚至系统瘫痪。

检测：当在服务器上看到大量的半连接状态时，特别是源 IP 地址是随机的，基本上可以断定这是一次 SYN 攻击。

****防范****：

* 通过防火墙、路由器等过滤网关防护。
* 通过加固 TCP/IP 协议栈防范，如增加最大半连接数，缩短超时时间。
* SYN cookies技术。SYN Cookies 是对 TCP 服务器端的三次握手做一些修改，专门用来防范 SYN 洪泛攻击的一种手段。

http常见状态码

常见状态码：

* 200：服务器已成功处理了请求。 通常，这表示服务器提供了请求的网页。
* 301 ： (永久移动) 请求的网页已永久移动到新位置。 服务器返回此响应(对 GET 或 HEAD 请求的响应)时，会自动将请求者转到新位置。
* 302：(临时移动) 服务器目前从不同位置的网页响应请求，但请求者应继续使用原有位置来进行以后的请求。
* 400 ：客户端请求有语法错误，不能被服务器所理解。
* 403 ：服务器收到请求，但是拒绝提供服务。
* 404 ：(未找到) 服务器找不到请求的网页。
* 500： (服务器内部错误) 服务器遇到错误，无法完成请求。

### **HTTP1.0和HTTP1.1的区别**

****长连接****：HTTP 1.1支持长连接（Persistent Connection）和请求的流水线（Pipelining）处理，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，在HTTP1.1中默认开启Connection： keep-alive，一定程度上弥补了HTTP1.0每次请求都要创建连接的缺点。

****缓存处理****：在HTTP1.0中主要使用header里的If-Modified-Since,Expires来做为缓存判断的标准，HTTP1.1则引入了更多的缓存控制策略，可供选择的缓存头来控制缓存策略。

****带宽优化及网络连接的使用****：HTTP1.0中，存在一些浪费带宽的现象，例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，并且不支持断点续传功能，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（Partial Content），这样就方便了开发者自由的选择以便于充分利用带宽和连接。

****错误通知的管理****：在HTTP1.1中新增了24个错误状态响应码，如409（Conflict）表示请求的资源与资源的当前状态发生冲突；410（Gone）表示服务器上的某个资源被永久性的删除。

****Host头处理****：在HTTP1.0中认为每台服务器都绑定一个唯一的IP地址，因此，请求消息中的URL并没有传递主机名（hostname）。但随着虚拟主机技术的发展，在一台物理服务器上可以存在多个虚拟主机（Multi-homed Web Servers），并且它们共享一个IP地址。HTTP1.1的请求消息和响应消息都应支持Host头域，且请求消息中如果没有Host头域会报告一个错误（400 Bad Request）。

### **HTTP1.1和 HTTP2.0的区别**

HTTP2.0相比HTTP1.1支持的特性：

****新的二进制格式****：HTTP1.1的解析是基于文本。基于文本协议的格式解析存在天然缺陷，文本的表现形式有多样性，要做到健壮性考虑的场景必然很多，二进制则不同，只认0和1的组合。基于这种考虑HTTP2.0的协议解析决定采用二进制格式，实现方便且健壮。

****多路复用****，即连接共享，即每一个request都是用作连接共享机制的。一个request对应一个id，这样一个连接上可以有多个request，每个连接的request可以随机的混杂在一起，接收方可以根据request的 id将request再归属到各自不同的服务端请求里面。

****头部压缩****，HTTP1.1的头部（header）带有大量信息，而且每次都要重复发送；HTTP2.0使用encoder来减少需要传输的header大小，通讯双方各自cache一份header fields表，既避免了重复header的传输，又减小了需要传输的大小。

****服务端推送****：服务器除了对最初请求的响应外，服务器还可以额外的向客户端推送资源，而无需客户端明确的请求。

### **1.1.37. 在浏览器中输入www.baidu.com后执行的全部过程?**

* 域名解析（域名 [www.baidu.com](http://www.baidu.com/" \t "http://interview.wzcu.com/%E7%BD%91%E7%BB%9C%E4%B8%8E%E5%8D%8F%E8%AE%AE/_blank)变为 ip 地址）。
* ****浏览器搜索自己的DNS缓存****（维护一张域名与IP的对应表）；若没有，则搜索****操作系统的DNS缓存****（维护一张域名与IP的对应表）；若没有，则搜索操作系统的****hosts文件****（维护一张域名与IP的对应表）。
* 若都没有，则找 tcp/ip 参数中设置的首选 dns 服务器，即****本地 dns 服务器****（递归查询），****本地域名服务器查询自己的dns缓存****，如果没有，则进行迭代查询。将本地dns服务器将IP返回给操作系统，同时缓存IP。
* 发起 tcp 的三次握手，建立 tcp 连接。浏览器会以一个随机端口（1024-65535）向服务端的 web 程序 ****80**** 端口发起 tcp 的连接。
* 建立 tcp 连接后发起 http 请求。
* 服务器响应 http 请求，客户端得到 html 代码。服务器 web 应用程序收到 http 请求后，就开始处理请求，处理之后就返回给浏览器 html 文件。
* 浏览器解析 html 代码，并请求 html 中的资源。
* 浏览器对页面进行渲染，并呈现给用户
* 断开连接