1. 简述 TCP 连接的过程（淘系）

> 参考答案:
>
> TCP 协议通过三次握手建立可靠的点对点连接，具体过程是:
>
> 首先服务器进入监听状态，然后即可处理连接
>
> 第一次握手：建立连接时，客户端发送 syn 包到服务器，并进入 SYN_SENT 状态，等待服务器确认。在发送的包中还会包含一个初始序列号 seq。此次握手的含义是客户端希望与服务器建立连接。
>
> 第二次握手：服务器收到 syn 包，然后回应给客户端一个 SYN+ACK 包，此时服务器进入 SYN_RCVD 状态。此次握手的含义是服务端回应客户端，表示已收到并同意客户端的连接请求。
>
> 第三次握手：客户端收到服务器的 SYN 包后，向服务器再次发送 ACK 包，并进入 ESTAB_LISHED 状态。
>
> 最后，服务端收到客户端的 ACK 包，于是也进入 ESTAB_LISHED 状态，至此，连接建立完成

2. 介绍下 HTTPS 中间人攻击

> 参考答案:
>
> 针对 HTTPS 攻击主要有 SSL 劫持攻击和 SSL 剥离攻击两种。
>
> SSL 劫持攻击是指攻击者劫持了客户端和服务器之间的连接，将服务器的合法证书替换为伪造的证书，从而获取客户端和服务器之间传递的信息。这种方式一般容易被用户发现，浏览器会明确的提示证书错误，但某些用户安全意识不强，可能会点击继续浏览，从而达到攻击目的。
>
> SSL 剥离攻击是指攻击者劫持了客户端和服务器之间的连接，攻击者保持自己和服务器之间的 HTTPS 连接，但发送给客户端普通的 HTTP 连接，由于 HTTP 连接是明文传输的，即可获取客户端传输的所有明文数据。

3. 介绍下 `http1.0` 、 `http1.1` 、 `http2.0` 协议的区别?

> 参考答案:
>
> 首先说 http1.0
>
> 它的特点是每次请求和响应完毕后都会销毁 TCP 连接，同时规定前一个响应完成后才能发送下一个请求。这样做有两个问题:
>
> 1. 无法复用连接

每次请求都要创建新的 TCP 连接，完成三次握手和四次挥手，网络利用率低

2. 队头阻塞
   如果前一个请求被某种原因阻塞了，会导致后续请求无法发送。

然后是 http1.1

http1.1 是 http1.0 的改进版，它做出了以下改进：

1. 长连接
   http1.1 允许在请求时增加请求头 `connection:keep-alive` ，这样便允许后续的客户端请求在一段时间内复用之前的 TCP 连接
2. 管道化
   基于长连接的基础，管道化可以不等第一个请求响应继续发送后面的请求，但响应的顺序还是按照请求的顺序返回。
3. 缓存处理
   新增响应头 cache-control，用于实现客户端缓存。
4. 断点传输
   在上传/下载资源时，如果资源过大，将其分割为多个部分，分别上传/下载，如果遇到网络故障，可以从已经上传/下载好的地方继续请求，不用从头开始，提高效率

最后是 http2.0

http2.0 进一步优化了传输效率，它主要有以下改进：

1. 二进制分帧
   将传输的消息分为更小的二进制帧，每帧有自己的标识序号，即便被随意打乱也能在另一端正确组装
2. 多路复用
   基于二进制分帧，在同一域名下所有访问都是从同一个 tcp 连接中走，并且不再有队头阻塞问题，也无须遵守响应顺序
3. 头部压缩
   http2.0 通过字典的形式，将头部中的常见信息替换为更少的字符，极大的减少了头部的数据量，从而实现更小的传输量
4. 服务器推
   http2.0 允许服务器直接推送消息给客户端，无须客户端明确的请求

4. 为什么 HTTP1.1 不能实现多路复用（腾讯）

参考答案：

HTTP/1.1 不是二进制传输，而是通过文本进行传输。由于没有流的概念，在使用并行传输（多路复用）传递数据时，接收端在接收到响应后，并不能区分多个响应分别对应的请求，所以无法将多个响应的结果重新进行组装，也就实现不了多路复用。

5. 简单讲解一下 http2 的多路复用（网易）

> 参考答案：
>
> 在 HTTP/2 中，有两个非常重要的概念，分别是帧（frame）和流（stream）。 帧代表着最小的数据单位，每个帧会标识出该帧属于哪个流，流也就是多个帧组成的数据流。 多路复用，就是在一个 TCP 连接中可以存在多条流。换句话说，也就是可以发送多个请求，对端可以通过帧中的标识知道属于哪个请求。通过这个技术，可以避免 HTTP 旧版本中的队头阻塞问题，极大的提高传输性能。

6. 谈谈你对 TCP 三次握手和四次挥手的理解

> TCP 协议通过三次握手建立可靠的点对点连接，具体过程是：
>
> 首先服务器进入监听状态，然后即可处理连接
>
> 第一次握手：建立连接时，客户端发送 syn 包到服务器，并进入 SYN_SENT 状态，等待服务器确认。在发送的包中还会包含一个初始序列号 seq。此次握手的含义是客户端希望与服务器建立连接。
>
> 第二次握手：服务器收到 syn 包，然后回应给客户端一个 SYN+ACK 包，此时服务器进入 SYN_RCVD 状态。此次握手的含义是服务端回应客户端，表示已收到并同意客户端的连接请求。
>
> 第三次握手：客户端收到服务器的 SYN 包后，向服务器再次发送 ACK 包，并进入 ESTAB_LISHED 状态。
>
> 最后，服务端收到客户端的 ACK 包，于是也进入 ESTAB_LISHED 状态，至此，连接建立完成
>
> 当需要关闭连接时，需要进行四次挥手才能关闭
>
> 1. Client 向 Server 发送 FIN 包，表示 Client 主动要关闭连接，然后进入 FIN_WAIT_1 状态，等待 Server 返回 ACK 包。此后 Client 不能再向 Server 发送数据，但能读取数据。
> 2. Server 收到 FIN 包后向 Client 发送 ACK 包，然后进入 CLOSE_WAIT 状态，此后 Server 不能再读取数据，但可以继续向 Client 发送数据。
> 3. Client 收到 Server 返回的 ACK 包后进入 FIN_WAIT_2 状态，等待 Server 发送 FIN 包。
> 4. Server 完成数据的发送后，将 FIN 包发送给 Client，然后进入 LAST_ACK 状态，等待 Client 返回 ACK 包，此后 Server 既不能读取数据，也不能发送数据。
> 5. Client 收到 FIN 包后向 Server 发送 ACK 包，然后进入 TIME_WAIT 状态，接着等待足够长的时间（2MSL）以确保 Server 接收到 ACK 包，最后回到 CLOSED 状态，释放网络资源。
> 6. Server 收到 Client 返回的 ACK 包后便回到 CLOSED 状态，释放网络资源。

7. 介绍 HTTPS 握手过程

> 参考答案：
>
> 1. 客户端请求服务器，并告诉服务器自身支持的加密算法以及密钥长度等信息
> 2. 服务器响应公钥和服务器证书
> 3. 客户端验证证书是否合法，然后生成一个会话密钥，并用服务器的公钥加密密钥，把加密的结果通过请求发送给服务器
> 4. 服务器使用私钥解密被加密的会话密钥并保存起来，然后使用会话密钥加密消息响应给客户端，表示自己已经准备就绪
> 5. 客户端使用会话密钥解密消息，知道了服务器已经准备就绪。
> 6. 后续客户端和服务器使用会话密钥加密信息传递消息

8. HTTPS 握手过程中，客户端如何验证证书的合法性

> 参考答案：
>
> 1. 校验证书的颁发机构是否受客户端信任。
> 2. 通过 CRL 或 OCSP 的方式校验证书是否被吊销。
> 3. 对比系统时间，校验证书是否在有效期内。
> 4. 通过校验对方是否存在证书的私钥，判断证书的网站域名是否与证书颁发的域名一致。

9. Http 状态码 301 和 302 的应用场景分别是什么

> 参考答案：
>
> 301 表示永久重定向，302 表示临时重定向。
>
> 如果浏览器收到的是 301，则会缓存重定向的地址，之后不会再重新请求服务器，直接使用缓存的地址请求，这样可以减少请求次数。
>
> 但如果浏览器收到的是 302，则不会缓存重定向地址，浏览器将来会继续以原有地址请求。
>
> 因此，301 适合地址永久转移的场景，比如域名变更；而 302 适合临时转移的场景，比如首页临时跳转到活动页

10. cookie 和 token 都存放在 header 中，为什么不会劫持 token?

> 参考答案：
>
> 由于浏览器会自动发送 cookie 到服务器，因此攻击者可以利用这种特点进行 csrf 攻击。
>
> 而通常 token 是不放到 cookie 中的，需要浏览器端使用 JS 自行保存到 localstorage 中，在请求时也需要手动的加入到请求头中，因此不容易引发 csrf 攻击。

11. 介绍下如何实现 token 加密

参考答案：

以最常见的 token 格式 jwt 为例

token 分为三段，分别是 header、payload、signature

其中，header 标识签名算法和令牌类型；payload 标识主体信息，包含令牌过期时间、发布时间、发行者、主体内容等；signature 是使用特定的算法对前面两部分进行加密，得到的加密结果。

token 有防篡改的特点，如果攻击者改动了前面两个部分，就会导致和第三部分对应不上，使得 token 失效。而攻击者不知道加密秘钥，因此又无法修改第三部分的值。

所以，在秘钥不被泄露的前提下，一个验证通过的 token 是值得被信任的。

12. 说下单点登录（新东方）

参考答案：

SSO 一般都需要一个独立的认证中心（passport），子系统的登录均得通过 passport，子系统本身将不参与登录操作，当一个系统成功登录以后，passport 将会颁发一个令牌给各个子系统，子系统可以拿着令牌会获取各自的受保护资源，为了减少频繁认证，各个子系统在被 passport 授权以后，会建立一个局部会话，在一定时间内可以无需再次向 passport 发起认证。

具体流程是：

1. 用户访问系统 1 的受保护资源，系统 1 发现用户未登录，跳转至 sso 认证中心，并将自己的地址作为参数
2. sso 认证中心发现用户未登录，将用户引导至登录页面
3. 用户输入用户名密码提交登录申请
4. sso 认证中心校验用户信息，创建用户与 sso 认证中心之间的会话，称为全局会话，同时创建授权令牌
5. sso 认证中心带着令牌跳转会最初的请求地址（系统 1）
6. 系统 1 拿到令牌，去 sso 认证中心校验令牌是否有效
7. sso 认证中心校验令牌，返回有效，注册系统 1
8. 系统 1 使用该令牌创建与用户的会话，称为局部会话，返回受保护资源
9. 用户访问系统 2 的受保护资源
10. 系统 2 发现用户未登录，跳转至 sso 认证中心，并将自己的地址作为参数
11. sso 认证中心发现用户已登录，跳转回系统 2 的地址，并附上令牌
12. 系统 2 拿到令牌，去 sso 认证中心校验令牌是否有效
13. sso 认证中心校验令牌，返回有效，注册系统 2
14. 系统 2 使用该令牌创建与用户的局部会话，返回受保护资源

13. http1.1 是如何复用 tcp 连接的？（网易）

> 参考答案：
>
> 客户端请求服务器时，通过请求行告诉服务器使用的协议是 http1.1，同时在请求头中附带 `connection:keep-alive`（为保持兼容），告诉服务器这是一个长连接，后续请求可以重复使用这一次的 TCP 连接。
>
> 这样做的好处是减少了三次握手和四次挥手的次数，一定程度上提升了网络利用率。但由于 http1.1 不支持多路复用，响应顺序必须按照请求顺序抵达客户端，不能真正实现并行传输，因此在 http2.0 出现之前，实际项目中往往把静态资源，比如图片，分发到不同域名下的资源服务器，以便实现真正的并行传输。

14. 文件上传如何做断点续传（网易）

> 参考答案：
>
> 客户端将文件的二进制内容进行分片，每片数据按顺序进行序号标识，上传每片数据时同时附带其序号。服务器接收到每片数据时，将其保存成一个临时文件，并记录每个文件的 hash 和序号。
>
> 若上传中止，将来再次上传时，可以向服务器索要已上传的分片序号，客户端仅需上传剩余分片即可。
>
> 当全部分片上传完成后，服务器按照分片的顺序组装成完整的文件，并删除分片文件。

15. 介绍 SSL 和 TLS（寺库）

> 参考答案：
>
> 它们都是用于保证传输安全的协议，介于传输层和应用层之间，TLS 是 SSL 的升级版。
>
> 它们的基本流程一致：
>
> 1. 客户端向服务器端索要公钥，并使用数字证书验证公钥。
> 2. 客户端使用公钥加密会话密钥，服务端用私钥解密会话密钥，于是得到一个双方都认可的会话密钥
> 3. 传输的数据使用会话密钥加密，然后再传输，接收消息方使用会话密钥解密得到原始数据

16. 说说网络的五层模型（寺库）

> 参考答案：
>
> 从上到下分别为：应用层、传输层、网络层、数据链路层、物理层。在发送消息时，消息从上到下进行打包，每一层会在上一层基础上加包，而接受消息时，从下到上进行解包，最终得到原始信息。

其中：

应用层主要面向互联网中的应用场景，比如网页、邮件、文件中心等等，它的代表协议有 http、smtp、pop3、ftp、DNS 等等

传输层主要面向传输过程，比如 TCP 协议是为了保证可靠的传输，而 UDP 协议则是一种无连接的广播，它们提供了不同的传输方式

网络层主要解决如何定位目标的问题，比如 IP、ICMP、ARP 等等

数据链路层的作用是将数据可靠的传输到目标，比如常见的以太网协议、P2P 协议

物理层是要规范网络两端使用的物理设备，比如蓝牙、wifi、光纤、网线接头等等

17. GET 和 POST 的区别（流利说）

参考答案：

从 http 协议的角度来说，GET 和 POST 它们都只是请求行中的第一个单词，除了语义不同，其实没有本质的区别。

之所以在实际开发中会产生各种区别，主要是因为浏览器的默认行为造成的。

受浏览器的影响，在实际开发中，GET 和 POST 有以下区别：

1. 浏览器在发送 GET 请求时，不会附带请求体
2. GET 请求的传递信息量有限，适合传递少量数据；POST 请求的传递信息量是没有限制的，适合传输大量数据。
3. GET 请求只能传递 ASCII 数据，遇到非 ASCII 数据需要进行编码；POST 请求没有限制
4. 大部分 GET 请求传递的数据都附带在 path 参数中，能够通过分享地址完整的重现页面，但同时也暴露了数据，若有敏感数据传递，不应该使用 GET 请求，至少不应该放到 path 中
5. 刷新页面时，若当前的页面是通过 POST 请求得到的，则浏览器会提示用户是否重新提交。若是 GET 请求得到的页面则没有提示。
6. GET 请求的地址可以被保存为浏览器书签，POST 不可以

18. http 劫持是什么？

参考答案：

是指攻击者在客户端和服务器之间同时建立了连接通道，通过某种方式，让客户端请求发送到自己的服务器，然后自己就拥有了控制响应内容的能力，从而给客户端展示错误的信息。

19. HTTP 劫持、DNS 劫持与 XSS

参考答案：

> http 劫持是指攻击者在客户端和服务器之间同时建立了连接通道，通过某种方式，让客户端请求发送到自己的服务器，然后自己就拥有了控制响应内容的能力，从而给客户端展示错误的信息，比如在页面中加入一些广告内容。
>
> DNS 劫持是指攻击者劫持了 DNS 服务器，获得了修改 DNS 解析记录的权限，从而导致客户端请求的域名被解析到了错误的 IP 地址，攻击者通过这种方式窃取用户资料或破坏原有正常服务。
>
> XSS 是指跨站脚本攻击。攻击者利用站点的漏洞，在表单提交时，在表单内容中加入一些恶意脚本，当其他正常用户浏览页面，而页面中刚好出现攻击者的恶意脚本时，脚本被执行，从而使得页面遭到破坏，或者用户信息被窃取。
>
> 要防范 XSS 攻击，需要在服务器端过滤脚本代码，将一些危险的元素和属性去掉或对元素进行HTML实体编码。

20. 介绍 xss csrf 攻击

> 参考答案：
>
> XSS：
>
> XSS 是指跨站脚本攻击。攻击者利用站点的漏洞，在表单提交时，在表单内容中加入一些恶意脚本，当其他正常用户浏览页面，而页面中刚好出现攻击者的恶意脚本时，脚本被执行，从而使得页面遭到破坏，或者用户信息被窃取。
>
> 要防范 XSS 攻击，需要在服务器端过滤脚本代码，将一些危险的元素和属性去掉或对元素进行HTML实体编码。
>
> CSRF：
>
> CSRF 是跨站请求伪造，是一种挟制用户在当前已登录的Web应用上执行非本意的操作的攻击方法
>
> 它首先引导用户访问一个危险网站，当用户访问网站后，网站会发送请求到被攻击的站点，这次请求会携带用户的cookie发送，因此就利用了用户的身份信息完成攻击
>
> 防御 CSRF 攻击有多种手段：
>
> 1. 不使用cookie
> 2. 为表单添加校验的 token 校验
> 3. cookie中使用sameSite字段
> 4. 服务器检查 referer 字段

21. https 验证身份也就是 TSL/SSL 身份验证的过程

参考答案：

1. 客户端请求服务器，并告诉服务器自身支持的加密算法以及密钥长度等信息
2. 服务器响应公钥和服务器证书
3. 客户端验证证书是否合法，然后生成一个会话密钥，并用服务器的公钥加密密钥，把加密的结果通过请求发送给服务器
4. 服务器使用私钥解密被加密的会话密钥并保存起来，然后使用会话密钥加密消息响应给客户端，表示自己已经准备就绪
5. 客户端使用会话密钥解密消息，知道了服务器已经准备就绪。
6. 后续客户端和服务器使用会话密钥加密信息传递消息

22. 为什么需要 CA 机构对证书签名

参考答案：

主要是为了解决证书的可信问题。如果没有权威机构对证书进行签名，客户端就无法知晓证书是否是伪造的，从而增加了中间人攻击的风险，https 就变得毫无意义。

23. 身份验证过程中会涉及到密钥，对称加密，非对称加密，摘要的概念，请解释一下

参考答案：

密钥

密钥是一种参数，它是在明文转换为密文或将密文转换为明文的算法中输入的参数。密钥分为对称密钥与非对称密钥，分别应用在对称加密和非对称加密上。

对称加密

对称加密又叫做私钥加密，即信息的发送方和接收方使用同一个密钥去加密和解密数据。对称加密的特点是算法公开、加密和解密速度快，适合于对大数据量进行加密，常见的对称加密算法有 DES、3DES、TDEA、Blowfish、RC5 和 IDEA。

非对称加密

非对称加密也叫做公钥加密。非对称加密与对称加密相比，其安全性更好。对称加密的通信双方使用相同的密钥，如果一方的密钥遭泄露，那么整个通信就会被破解。而非对称加密使用一对密钥，即公钥和私钥，且二者成对出现。私钥被自己保存，不能对外泄露。公钥指的是公共的密钥，任何人都可以获得该密钥。用公钥或私钥中的任何一个进行加密，用另一个进行解密。

摘要

> 摘要算法又称哈希/散列算法。它通过一个函数，把任意长度的数据转换为一个长度固定的数据串（通常用 16 进制的字符串表示）。算法不可逆。

24. webSocket 协议是什么，能简述一下吗?

> 参考答案:
>
> websocket 协议 HTML5 带来的新协议，相对于 http，它是一个持久连接的协议，它利用 http 协议完成握手，然后通过 TCP 连接通道发送消息，使用 websocket 协议可以实现服务器主动推送消息。
>
> 首先，客户端若要发起 websocket 连接，首先必须向服务器发送 http 请求以完成握手，请求行中的 path 需要使用 `ws:` 开头的地址，请求头中要分别加
> 入 `upgrade`、`connection`、`Sec-WebSocket-Key`、`Sec-WebSocket-Version` 标记
>
> 然后，服务器收到请求后，发现这是一个 websocket 协议的握手请求，于是响应行中包
> 含 `Switching Protocols` ，同时响应头中包
> 含 `upgrade`、`connection`、`Sec-WebSocket-Accept` 标记
>
> 当客户端收到响应后即可完成握手，随后使用建立的 TCP 连接直接发送和接收消息。

25. webSocket 与传统的 http 有什么优势

> 参考答案:
>
> 当页面中需要观察实时数据的变化（比如聊天、k 线图）时，过去我们往往使用两种方式完成
>
> 第一种是短轮询，即客户端每隔一段时间就向服务器发送消息，询问有没有新的数据
>
> 第二种是长轮询，发起一次请求询问服务器，服务器可以将该请求挂起，等到有新消息时再进行响应。响应后，客户端立即又发起一次请求，重复整个流程。
>
> 无论是哪一种方式，都暴露了 http 协议的弱点，即响应必须在请求之后发生，服务器是被动的，无法主动推送消息。而让客户端不断的发起请求又白白的占用了资源。
>
> websocket 的出现就是为了解决这个问题，它利用 http 协议完成握手之后，就可以与服务器建立持久的连接，服务器可以在任何需要的时候，主动推送消息给客户端，这样占用的资源最少，同时实时性也最高。

26. 如何劫持 https 的请求，提供思路

> 参考答案:
>
> https 有防篡改的特点，只要浏览器证书验证过程是正确的，很难在用户不察觉的情况下进行攻击。但若能够更改浏览器的证书验证过程，便有机会实现 https 中间人攻击。

所以，要劫持 https，首先要伪造一个证书，并且要想办法让用户信任这个证书，可以有多种方式，比如病毒、恶意软件、诱导等。一旦证书被信任后，就可以利用普通中间人攻击的方式，使用伪造的证书进行攻击。

27. 怎样解决跨域问题?

参考答案:

1. 使用 JSONP
   这是一种古老的解决跨域问题的思路。
   在需要跨域请求时，事先准备好一个处理服务器数据的函数，然后生成一个 `<script>` 元素， `src` 指向跨域站点，同时把准备好的函数名通过地址参数传递到服务器。
   跨域站点返回一段调用该函数的脚本，当客户端接收到脚本后就会运行事先准备的函数，从而实现跨域获取数据。
   JSONP 实现简单、兼容性好，但缺点也很明显，它只支持 get 请求，同时也有安全性问题，并且对服务器端代码侵入性比较强。
2. 使用 cors
   在请求时，客户端使用一些特殊的请求头向服务器申请跨域访问，并通过这些请求头告诉服务器自己的行为。服务器根据自身的规则决定是否允许跨域，如果允许，则通过响应头告诉客户端可以发送跨域请求。
   cors 协议已被各种主流浏览器支持，它安全性高，同时也不会侵入服务器代码，是目前最主流的跨域方式

除此之外，远古时期的跨域处理还包括 iframe、form 等，由于它们缺陷非常明显，故很少使用了。

28. 前端如何实现即时通讯?

参考答案:

1. 短轮询。即客户端每隔一段时间就向服务器发送消息，询问有没有新的数据
2. 长轮询，发起一次请求询问服务器，服务器可以将该请求挂起，等到有新消息时再进行响应。响应后，客户端立即又发起一次请求，重复整个流程。
3. websocket，握手完毕后会建立持久性的连接通道，随后服务器可以在任何时候推送新消息给客户端

29. HTTP 常用状态码 301 302 304 403

参考答案:

301 永久重定向，浏览器会把重定向后的地址缓存起来，将来用户再次访问原始地址时，直接引导用户访问新地址

302 临时重定向，浏览器会引导用户进入新地址，但不会缓存原始地址，下一次用户访问源地址时，浏览器仍然要请求原地址的服务器

304 资源未修改，服务器通过该状态码告诉客户端，请求的资源和过去一样，并没有任何变化，建议自行使用过去的缓存。通常，304 状态码的响应中，服务器不会附带任何的响应体。

403 不允许访问。服务器通过该状态码告诉客户端，这个资源目前不允许访问。这种状态码通常出现在权限不足的情况下。

30. 在浏览器地址栏输入地址，并按下回车键后，发生了哪些事情?

    1. 参考答案:
        1. 浏览器自动补全协议、端口
        2. 浏览器自动完成url编码
        3. 浏览器根据url地址查找本地缓存，根据缓存规则看是否命中缓存，若命中缓存则直接使用缓存，不再发出请求
        4. 通过DNS解析找到服务器的IP地址
        5. 浏览器向服务器发出建立TCP连接的申请，完成三次握手后，连接通道建立
        6. 若使用了HTTPS协议，则还会进行SSL握手，建立加密信道。使用SSL握手时，会确定是否使用HTTP2
        7. 浏览器决定要附带哪些cookie到请求头中
        8. 浏览器自动设置好请求头、协议版本、cookie，发出GET请求
        9. 服务器处理请求，进入后端处理流程。完成处理后，服务器响应一个HTTP报文给浏览器。
        10. 浏览器根据使用的协议版本，以及Connection字段的约定，决定是否要保留TCP连接。
        11. 浏览器根据响应状态码决定如何处理这一次响应
        12. 浏览器根据响应头中的Content-Type字段识别响应类型，如果是text/html，则对响应体的内容进行HTML解析，否则做其他处理
        13. 浏览器根据响应头的其他内容完成缓存、cookie的设置
        14. 浏览器开始从上到下解析HTML，若遇到外部资源链接，则进一步请求资源
        15. 解析过程中生成DOM树、CSSOM树，然后一边生成，一边把二者合并为渲染树（rendering tree），随后对渲染树中的每个节点计算位置和大小（reflow），最后把每个节点利用GPU绘制到屏幕（repaint）
        16. 在解析过程中还会触发一系列的事件，当DOM树完成后会触发DOMContentLoaded事件，当所有资源加载完毕后会触发load事件

31. HTTPS 握手

    参考答案:

    1. 客户端请求服务器，并告诉服务器自身支持的加密算法以及密钥长度等信息

2. 服务器响应公钥和服务器证书
3. 客户端验证证书是否合法，然后生成一个会话密钥，并用服务器的公钥加密密钥，把加密的结果通过请求发送给服务器
4. 服务器使用私钥解密被加密的会话密钥并保存起来，然后使用会话密钥加密消息响应给客户端，表示自己已经准备就绪
5. 客户端使用会话密钥解密消息，知道了服务器已经准备就绪。
6. 后续客户端和服务器使用会话密钥加密信息传递消息

32. 网页验证码是干嘛的，是为了解决什么安全问题?

参考答案:

验证码主要用于让服务器区分请求是人还是机器发送的。这样做是为了避免某些程序恶意的提交大量信息到服务器，进而导致服务器产生大量的垃圾数据。有时，验证码也可以防止机器暴力破解用户密码，它通过在短时间内不断提交登录信息，尝试各种密码组合来达到破解的目的。

33. http1.0、http2.0、http3.0 之间的区别

参考答案:

http1.0

每次请求和响应完毕后都会销毁 TCP 连接，同时规定前一个响应完成后才能发送下一个请求。这样做有两个问题:

1. 无法复用连接
   每次请求都要创建新的 TCP 连接，完成三次握手和四次挥手，网络利用率低
2. 队头阻塞
   如果前一个请求被某种原因阻塞了，会导致后续请求无法发送。

http2.0

http2.0 优化了传输效率，它主要有以下改进:

1. 二进制分帧
   将传输的消息分为更小的二进制帧，每帧有自己的标识序号，即便被随意打乱也能在另一端正确组装
2. 多路复用
   基于二进制分帧，在同一域名下所有访问都是从同一个 tcp 连接中走，并且不再有队头阻塞问题，也无须遵守响应顺序
3. 头部压缩
   http2.0 通过字典的形式，将头部中的常见信息替换为更少的字符，极大的减少了头部的数据量，从而实现更小的传输量

4. 服务器推

http2.0 允许服务器直接推送消息给客户端，无须客户端明确的请求

http3.0

http3.0 目前还在草案阶段，它完全抛弃了 TCP 协议，转而使用 UDP 协议，是为了进一步提升性能。

虽然 http2.0 进行了大量的优化，但它无法摆脱 TCP 协议本身的问题，比如建立连接时间长、对头阻塞问题等等。

为了保证传输的可靠性，http3.0 使用了 QUIC 协议。

34. cookie/sessionStorage/localStorage 的区别

参考答案：

cookie、sessionStorage、localStorage 都是保存本地数据的方式

其中，cookie 兼容性较好，所有浏览器均支持。浏览器针对 cookie 会有一些默认行为，比如当响应头中出现 `set-cookie` 字段时，浏览器会自动保存 cookie 的值；再比如，浏览器发送请求时，会附带匹配的 cookie 到请求头中。这些默认行为，使得 cookie 长期以来担任着维持登录状态的责任。与此同时，也正是因为浏览器的默认行为，给了恶意攻击者可乘之机，CSRF 攻击就是一个典型的利用 cookie 的攻击方式。虽然 cookie 不断的改进，但前端仍然需要另一种更加安全的保存数据的方式

HTML5 新增了 sessionStorage 和 localStorage，前者用于保存会话级别的数据，后者用于更持久的保存数据。浏览器针对它们没有任何默认行为，这样一来，就把保存数据、读取数据的工作交给了前端开发者，这就让恶意攻击者难以针对登录状态进行攻击。
cookie 的大小是有限制的，一般浏览器会限制同一个域下的 cookie 总量为 4M，而 sessionStorage 和 localStorage 则没有限制
cookie 会与 domain、path 关联，而 sessionStorage 和 localStorage 只与 domain 关联

35. post 请求什么时候用 form data 什么时候用 request payload

参考答案：

form data 适合传递简单的键值对信息，由于传递的信息比较扁平，难以传递深层次嵌套的数据

request payload 适合传递任意格式的数据，包括单个数字、布尔、深层次嵌套的对象、数组等，但 request payload 不适合传递文件数据

在前后端分离的项目中，对于非文件数据的传递，都推荐使用 request payload 的形式，以传递最明确的数据类型和数据结构，而对于文件上传，则推荐使用传统的 form data

36. http 常见请求方法有哪些?

> 参考答案:
>
> - GET，表示向服务器获取资源
> - POST，表示向服务器提交信息，通常用于产生新的数据，比如注册
> - PUT，表示希望修改服务器的数据，通常用于修改
> - DELETE，表示希望删除服务器的数据
> - OPTIONS，发生在跨域的预检请求中，表示客户端向服务器申请跨域提交
> - TRACE，回显服务器收到的请求，主要用于测试和诊断
> - CONNECT，用于建立连接管道，通常在代理场景中使用，网页中很少用到

37. 列举优化网络性能方法

> 参考答案:
>
> - 优化打包体积
>   利用一些工具压缩、混淆最终打包代码，减少包体积
> - 多目标打包
>   利用一些打包插件，针对不同的浏览器打包出不同的兼容性版本，这样一来，每个版本中的兼容性代码就会大大减少，从而减少包体积
> - 压缩
>   现代浏览器普遍支持压缩格式，因此服务端的各种文件可以压缩后再响应给客户端，只要解压时间小于优化的传输时间，压缩就是可行的
> - CDN
>   利用 CDN 可以大幅缩减静态资源的访问时间，特别是对于公共库的访问，可以使用知名的 CDN 资源，这样可以实现跨越站点的缓存
> - 缓存
>   对于除 HTML 外的所有静态资源均可以开启协商缓存，利用构建工具打包产生的文件 hash 值来置换缓存
> - http2
>   开启 http2 后，利用其多路复用、头部压缩等特点，充分利用带宽传递大量的文件数据
> - 雪碧图
>   对于不使用 HTTP2 的场景，可以将多个图片合并为雪碧图，以达到减少文件的目的
> - defer、async
>   通过 defer 和 async 属性，可以让页面尽早加载 js 文件
> - prefetch、preload
>   通过 prefetch 属性，可以让页面在空闲时预先下载其他页面可能要用到的资源
>   通过 preload 属性，可以让页面预先下载本页面可能要用到的资源
> - 多个静态资源域

> 对于不使用 HTTP2 的场景，将相对独立的静态资源分到多个域中保存，可以让浏览器同时开启多个 TCP 连接，并行下载

38. session 怎么消除

> 参考答案：
>
> 1. 过期时间
>   当客户端长时间没有传递 sessionid 过来时，服务器可以在过期时间之后自动清除 session
> 2. 客户端主动通知
>   可以使用 JS 监听客户端页面关闭或其他退出操作，然后通知服务器清除 session

39. 什么是 DNS 域名解析？

> 参考答案：
>
> DNS 域名解析是指把域名解析成 IP 地址的过程。
>
> 在具体的实现上，域名解析是由多个层级的服务器共同完成的。在查询域名时，客户端会先检查自身的 DNS 映射表，若找不到解析记录，则使用用户配置的 DNS 服务器，若目标 DNS 服务器中找不到记录，则继续往上一个层级寻找，直到到达根域名服务器，根域名服务器会根据域名的类型，将解析任务分发到对应的子域名服务器依次查找，直到找到解析记录为止。