### 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**服务器**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**，来确保数据的传输，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协议是由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协议的安全性更高。数据完整性**  
主要的区别如下：

Https协议需要**CA证书**，**费用较高**。

http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。**

使用不同的链接方式，端口也不同，一般而言，**http协议的端口为80**，**https的端口为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协议安全。

(3)https协议的工作原理

客户端在使用HTTPS方式与Web服务器通信时有以下几个步骤，如图所示。

客户使用https url访问服务器，则要求web 服务器**建立ssl链接。**

**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**。

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。

客户端浏览器通过双方协商一致的安全等级，**建立会话密钥**，然后通过网站的**公钥来加密会话密钥**，并传送给网站。

web服务器通过自己的**私钥解密出会话密钥**。

web服务器通过**会话密钥加密与客户端**之间的通信。

(4)https协议的优点

使用HTTPS协议**可认证用户和服务器**，确保数据发送到正确的客户机和服务器；

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，要比http协议安全，可防止数据在传输过程中**不被窃取、改变，确保数据的完整性**。

HTTPS是现行架构下最安全的解决方案，虽然不是绝对安全，但它大幅增加了中间人攻击的成本。

谷歌曾在2014年8月份调整搜索引擎算法，并称“比起同等HTTP网站，采用HTTPS加密的网站在搜索结果中的排名将会更高”。

(5)https协议的缺点

https握手阶段**比较费时**，会使页面加载时间延长50%，增加10%~20%**的耗电。**

https缓存不如http高效，会增加数据开销。

SSL证书也需要钱，**功能越强大的证书费用越高**。

SSL证书需要绑定IP，不能再同一个ip上绑定多个域名，ipv4资源支持不了这种消耗。

（6）自己的总结：http：信息明文传输、传输数据未加密。

### tcp三次握手，一句话概括（看后面的）

客户端和服务端都需要直到各自可收发，因此需要三次握手。

简化三次握手：

从图片可以得到三次握手可以简化为：C发起请求连接S确认，也发起连接C确认我们再看看每次握手的作用：第一次握手：S只可以确认 自己可以接受C发送的报文段第二次握手：C可以确认 S收到了自己发送的报文段，并且可以确认 自己可以接受S发送的报文段第三次握手：S可以确认 C收到了自己发送的报文段

### 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链接**。

（2）TCP提供**可靠的**服务。也就是说，通过TCP连接传送的数据，**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。 并且因为tcp可靠，面向连接，不会丢失数据因此**适合大数据量**的交换。

（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（因此会出现丢包，对实时的应用比如**IP电话和视频会议**等）。

（4）TCP只能是**1对1**的，UDP支持**1对1,1对多。**

（5）TCP的**首部较大为20字节**，而UDP**只有8字节**。

（6）TCP是面向连接的可靠性传输，而UDP是不可靠的。

### WebSocket的实现和应用

(1)什么是WebSocket?

WebSocket是HTML5中的协议，支持持久连续，http协议不支持持久性连接。Http1.0和HTTP1.1都不支持持久性的链接，HTTP1.1中的keep-alive，将多个http请求合并为1个

(2)WebSocket是什么样的协议，具体有什么优点？

HTTP的**生命周期通过Request**来界定，也就是Request一个Response，那么在Http1.0协议中，这次Http请求就结束了。在Http1.1中进行了改进，是的有一个connection：Keep-alive，也就是说，在一个Http连接中，可以发送多个Request，接收多个Response。但是必须记住，在Http中一个Request只能对应有一个Response，而且这个Response是被动的，不能主动发起。

WebSocket是基于Http协议的，或者说借用了Http协议来完成一部分握手，在握手阶段与Http是相同的。我们来看一个websocket握手协议的实现，基本是2个属性，upgrade，connection。

基本请求如下：

GET /chat HTTP/1.1

Host: server.example.com

Upgrade: websocket

Connection: Upgrade

### HTTP请求的方式，HEAD方式

head：**类似于get**请求，只不过返回的响应中没有具体的内容，用户获取报头，没有body

options：允许客户端查**看服务器的性能**，比如说服务器**支持的请求方式**等等。

### 一个图片url访问后直接下载怎样实现？

请求的返回头里面，用于浏览器解析的重要参数就是OSS的API文档里面的返回http头，决定用户下载行为的参数。

下载的情况下：

1. x-oss-object-type:

Normal

2. x-oss-request-id:

598D5ED34F29D01FE2925F41

3. x-oss-storage-class:

Standard

### 说一下web Quality（无障碍）

能够被残障人士使用的网站才能称得上一个易用的（易访问的）网站。

残障人士指的是那些带有残疾或者身体不健康的用户。

使用alt属性：

<img src="person.jpg"  alt="this is a person"/>

有时候浏览器会无法显示图像。具体的原因有：

用户**关闭了图像显示**

浏览器是**不支持图形显示的迷你浏览器**

浏览器是**语音浏览器**（供盲人和弱视人群使用）  
如果您使用了**alt 属性**，那么浏览器至少可以显示或读出有关图像的描述。

### 几个很实用的BOM属性对象方法?

什么是Bom? Bom是浏览器对象。有哪些常用的Bom属性呢？

(1)location对象（历史上访问过的地址）

location.href-- 返回或设置当前文档的URL  
location.search -- 返回URL中的查询字符串部分。例如 [http://www.dreamdu.com/dreamdu.php?id=5&name=dreamdu](http://www.dreamdu.com/dreamdu.php?id=5&name=dreamdu" \t "https://www.nowcoder.com/tutorial/96/_blank) 返回包括(?)后面的内容?id=5&name=dreamdu  
location.hash -- 返回URL#后面的内容，如果没有#，返回空（hash就是）  
location.host -- 返回URL中的域名部分，例如[www.dreamdu.com](http://www.dreamdu.com/" \t "https://www.nowcoder.com/tutorial/96/_blank)  
location.hostname -- 返回URL中的主域名部分，例如dreamdu.com（没有www）  
location.pathname -- 返回URL的域名后的部分。例如 [http://www.dreamdu.com/xhtml/](http://www.dreamdu.com/xhtml/" \t "https://www.nowcoder.com/tutorial/96/_blank) 返回/xhtml/（.com后面的）  
location.port -- 返回URL中的端口部分。例如 [http://www.dreamdu.com:8080/xhtml/](http://www.dreamdu.com:8080/xhtml/" \t "https://www.nowcoder.com/tutorial/96/_blank) 返回8080（端口号）  
location.protocol -- 返回URL中的协议部分。例如 [http://www.dreamdu.com:8080/xhtml/](http://www.dreamdu.com:8080/xhtml/" \t "https://www.nowcoder.com/tutorial/96/_blank) 返回(//)前面的内容http:  
location.assign -- 设置当前文档的URL  
location.replace() -- 设置当前文档的URL，并且在history对象的地址列表中移除这个URL location.replace(url);  
location.reload() -- 重载当前页面

(2)history对象

history.go() -- 前进或后退指定的页面数 history.go(num);  
history.back() -- 后退一页  
history.forward() -- 前进一页

(3)Navigator对象

navigator.userAgent -- 返回用户代理头的字符串表示(就是包括**浏览器版本信息等的字符串)**  
navigator.cookieEnabled -- 返回浏览器**是否支持(启用)cookie**

### 说一下HTML5 drag api

dragstart：事件主体是被拖放元素，在**开始拖放**被拖放元素时触发，。

darg：事件主体是被拖放元素，在**正在拖放**被拖放元素时触发。

dragenter：事件主体是目标元素，在被拖放元素**进入某元素**时触发。

dragover：事件主体是目标元素，在被拖放**在某元素内移动**时触发。

dragleave：事件主体是目标元素，在被拖放元素**移出目标元素**是触发。

drop：事件主体是目标元素，在目标元素**完全接受被拖放元素**时触发。

dragend：事件主体是被拖放元素，在**整个拖放操作结束**时触发

### 说一下http2.0

首先补充一下，http和https的区别，相比于http,https是基于ssl加密的http协议

简要概括：http2.0是基于1999年发布的http1.0之后的首次更新。

提升**访问速度**（可以对于，请求资源所需**时间更少**，访问**速度更快**，相比http1.0）

允许**多路复用**：多路复用允许同时通过单一的HTTP2.0连接**发送多重请求-响应信息**。改善了：在http1.1中，浏览器客户端在同一时间，针对同一域名下的请求有一定数量限制（连接数量），超过限制会被阻塞。

**二进制分帧**：HTTP 2.0会将所有的**传输信息分割为更小的信息或者帧**，并对他们进行**二进制编码**

**首部压缩**

**服务器端推送**

### 补充400和401、403状态码

(1)400状态码：请求无效

产生原因：

前端**提交数据的字段名称和字段类型**与**后台的实体没有保持一致**

前端提交到后台的数据应该是**json字符串类型**，但是前端没有将对象**JSON.stringify转化成字符串。**

解决方法：

对照字段的名称，保持一致性

将obj对象通过JSON.stringify实现序列化

(2)401状态码：当前请求需要**用户验证**

(3)403状态码：服务器已经**得到请求，但是拒绝执行**

### fetch发送2次请求的原因

Fetch类似于ajax，但是用promise。fetch发送post请求的时候，总是发送2次，第一次状态码是204，第二次才成功？

原因很简单，因为你用fetch的post请求的时候，导致fetch 第一次发送了一个Options请求，询问服务器**是否支持修改的请求头**，如果服务器支持，则在**第二次中发送真正的请求**。

### Cookie、sessionStorage、localStorage的区别

共同点：都是保存在浏览器端，并且是**同源**的

Cookie：cookie数据始终在同源**的http请求中携带**（即使不需要），即cookie在浏览器和服务器间来回传递。而sessionStorage和localStorage不会自动把数据发给服务器，仅在本地保存。cookie数据还有**路径（path）**的概念，可以限制cookie只属于某个路径下,**存储的大小**很小只有4K左右。 （key：可以在浏览器和服务器端来回传递，存储容量小，只有大约4K左右）

sessionStorage：仅在**当前浏览器窗口关闭前有效**，自然也就不可能持久保持，localStorage：始终有效，**窗口或浏览器关闭也一直保存**，因此用作持久数据；cookie只在设置的**cookie过期时间**之前一直有效，即使窗口或浏览器关闭。（key：本身就是一个回话过程，关闭浏览器后消失，session为一个回话，当页面不同即使是同一页面打开两次，也被视为同一次回话）

localStorage：localStorage 在所有同源窗口中都是共享的；cookie也是在所有同源窗口中都是共享的。（key：同源窗口都会共享，并且不会失效，不管窗口或者浏览器关闭与否都会始终生效）

补充说明一下cookie的作用：

保存用户登录状态。例如将**用户id存储于一个cookie内，**这样当用户下次访问该页面时就不需要重新登录了，现在很多论坛和社区都提供这样的功能。 cookie还可以设置过期时间，当超过时间期限后，cookie就会自动消失。因此，系统往往可以提示用户保持登录状态的时间：常见选项有一个月、三个 月、一年等。

跟踪用户行为。例如一个天气预报网站，能够根据用户选择的地区显示当地的天气情况。如果每次都需要选择所在地是烦琐的，当利用了cookie后就会显得很人性化了，系统能够记住上一次访问的地区，当下次再打开该页面时，它就会自动显示上次用户所在地区的天气情况。因为一切都是在后 台完成，所以这样的页面就像为某个用户所定制的一样，使用起来非常方便定制页面。如果网站提供了换肤或更换布局的功能，那么可以使用cookie来记录用户的选项，例如：背景色、分辨率等。当用户下次访问时，仍然可以保存上一次访问的界面风格。

### 说一下web worker

在HTML页面中，如果在**执行脚本**时，页面的状态是不可响应的，直到脚本执行完成后，页面才变成可相应。web worker是**运行在后台的js**，独立于其他脚本，不会影响页面你的性能。并且通过**postMessage将结果回传到主线程**。这样在进行复杂操作的时候，就不会阻塞主线程了。

如何创建web worker：

检测浏览器对于web worker的支持性

创建web worker文件（js，回传函数等）

创建web worker对象

### 对HTML语义化标签的理解

HTML5语义化标签是指正确的标签包含了正确的内容，结构良好，便于阅读，比如nav表示导航条，类似的还有article、header、footer等等标签。

### iframe是什么？有什么缺点？

定义：iframe元素会创建**包含另一个文档的内联框架**

提示：可以将提示文字放在<iframe></iframe>之间，来提示某些不支持iframe的浏览器

缺点：

会阻塞主页面的onload事件

搜索引擎无法解读这种页面，**不利于SEO**

**iframe和主页面共享连接池**，而浏览器对相同区域有限制所以会影响性能。

### Doctype作用?严格模式与混杂模式如何区分？它们有何意义?

Doctype声明于文档最前面，告诉浏览器以何种方式来渲染页面，这里有两种模式，**严格模式和混杂模式**。

严格模式的排版和JS 运作模式是 以该浏览器支持的**最高标准运行。**

**混杂模式，向后兼容**，模拟老式浏览器，防止浏览器无法兼容页面。

### Cookie如何防范XSS攻击

参考回答：

XSS（跨站脚本攻击）是指攻击者**在HTML中嵌入javascript脚本**，为了减轻这些攻击，需要在HTTP头部配上，set-cookie：

httponly-这个属性可以防止XSS,它会禁止javascript脚本来访问cookie。

secure - 这个属性告诉浏览器仅在请求为https的时候发送cookie。

结果应该是这样的：Set-Cookie=<cookie-value>.....

### Cookie和session的区别

HTTP是一个无状态协议，因此Cookie的最大的作用就是存储sessionId用来唯一标识用户

### 一句话概括RESTFUL

就是用URL定位资源，用HTTP描述操作

### 讲讲viewport和移动端布局

[响应式布局的常用解决方案对比(媒体查询、百分比、rem和vw/vh）](https://github.com/forthealllight/blog/issues/13" \t "https://www.nowcoder.com/tutorial/96/_blank)

### click在ios上有300ms延迟，原因及如何解决？

(1)粗暴型，**禁用缩放**

<meta name="viewport" content="width=device-width, user-scalable=no">

(2)利用FastClick，其原理是：

检测到touchend事件后，立刻出发模拟click事件，并且把浏览器300毫秒之后真正出发的事件给阻断掉

### addEventListener参数

addEventListener(event, function, useCapture)//事件、方法、冒泡

其中，event指定事件名；function指定要事件触发时执行的函数；useCapture指定事件是否在捕获或冒泡阶段执行。

### cookie sessionStorage localStorage区别

cookie数据始终在同源的http请求中携带(即使不需要)，即cookie在浏览器和服务器间来回传递。

cookie数据还有**路径（path）**的概念，可以限制。cookie只属于某个路径下

存储大小限制也不同，**cookie数据不能超过4K**，同时因为每次http请求都会携带cookie，所以cookie只适合保存很小的数据，如回话标识。

webStorage虽然也有存储大小的限制，但是比cookie大得多，**可以达到5M**或更大

数据的有效期不同sessionStorage：仅在当前**的浏览器窗口关闭**有效；localStorage：始终有效，**窗口或浏览器关闭也一直保**存，因此用作持久数据；cookie：只在设置的cookie过期时间之前一直有效，即使窗口和浏览器关闭

作用域不同sessionStorage：不在不同的浏览器窗口中共享，即使是同一个页面；localStorage：在所有同源窗口都是共享的；cookie：也是在所有同源窗口中共享的

### cookie session区别

1.    cookie数据存放在**客户的浏览器上**，session**数据放在服务器上**。

2.    cookie不**是很安全**，别人可以分析存放在本地的COOKIE并进行COOKIE欺骗  
考虑到安全应当**使用session。**

3.    session会在一定时间内保存在服务器上。当**访问增多**，会比较占用你服务器的性能  
考虑到减轻**服务器性能方面，应当使用COOKIE。**

4.    单个cookie保存的数**据不能超过4K**，很多浏览器都限制一个站点最**多保存20个cookie。**

### iframe通信，同源和不同源两种情况，多少种方法。同源我说了，根据父页面以及cookie，不同源我说了设置子域的方法。

### 介绍知道的http返回的状态码

100    Continue    继续。客户端应**继续其请求**

101    Switching Protocols   **切换协议**。服务器根据客户端的请求切换协议。只能切换到更高级的协议，例如，切换到HTTP的新版本协议

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的资源

202    Accepted    **已接受**。已经接受请求，但未处理完成

203    Non-Authoritative Information    非授权信息。请求成功。但返回的meta信息不在原始的服务器，而是一个副本

204    No Content    无内容。服务器成功处理，但未返回内容。在未更新网页的情况下，可确保浏览器继续显示当前文档

205    Reset Content    重置内容。服务器处理成功，用户终端（例如：浏览器）应重置文档视图。可通过此返回码清除浏览器的表单域

206    Partial Content    部分内容。服务器成功处理了部分GET请求

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于用户终端（例如：浏览器）选择

301    Moved Permanently   **永久移动**。请求的资源已被永久的移动到新URI，返回信息会包括新的URI，浏览器会自动定向到新URI。今后任何新的请求都应使用新的URI代替

302    Found   **临时移动**。与301类似。但资源只是临时被移动。客户端应继续使用原有URI

303    See Other    **查看其它地址。与301类似。使用GET和POST请求查看**

304    Not Modified    **未修改**。所请求的资源未修改，服务器返回此状态码时，不会返回任何资源。客户端通常会缓存访问过的资源，通过提供一个头信息指出客户端希望只返回在指定日期之后修改的资源

305    Use Proxy   **使用代理**。所请求的资源必须通过代理访问

306    Unused    已经被**废弃的HTTP状态码**

307    Temporary Redirect    **临时重定向。与302类似**。使用**GET请求重定向**

400    Bad Request    客户端请求的语法错误，服务器无法理解

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    服务器**无法根据客户端的请求找到资源**（网页）。通过此代码，网站设计人员可设置"您所请求的资源无法找到"的个性页面

405    Method Not Allowed    客户端请求中的方法被禁止

406    Not Acceptable    服务器无法根据客户端请求的内容特性完成请求

407    Proxy Authentication Required    请求要求代理的身份认证，与401类似，但请求者应当使用代理进行授权

408    Request Time-out    服务器等待客户端发送的请求时间过长，超时

409    Conflict    服务器完成客户端的PUT请求是可能返回此代码，服务器处理请求时发生了冲突

410    Gone    客户端请求的资源已经不存在。410不同于404，如果资源以前有现在被永久删除了可使用410代码，网站设计人员可通过301代码指定资源的新位置

411    Length Required    服务器无法处理客户端发送的不带Content-Length的请求信息

412    Precondition Failed    客户端请求信息的先决条件错误

413    Request Entity Too Large    由于请求的实体过大，服务器无法处理，因此拒绝请求。为防止客户端的连续请求，服务器可能会关闭连接。如果只是服务器暂时无法处理，则会包含一个Retry-After的响应信息

414    Request-URI Too Large    请求的URI过长（URI通常为网址），服务器无法处理

415    Unsupported Media Type    服务器无法处理请求附带的媒体格式

416    Requested range not satisfiable    客户端请求的范围无效

417    Expectation Failed    服务器无法满足Expect的请求头信息

500    Internal Server Erro**r    服务器内部错误，**无法完成请求

501    Not Implemented    **服务器不支持请求**的功能，无法完成请求

502    Bad Gateway    作为**网关或者代理工作**的服务器尝试执行请求时，从远程服务器接收到了一个无效的响应

503    Service Unavailable    由于**超载或系统维护**，服务器暂时的无法处理客户端的请求。延时的长度可包含在服务器的Retry-After头信息中

504    Gateway Time-out**充当网关或代理的服务器，未及时从远端服务器获取请求**

505    HTTP Version not supported    服务器不**支持请求的HTTP协议的版本**，无法完成处理

### http常用请求头

|  |  |
| --- | --- |
| 协议头 | 说明 |
| Accept | 可接受的响应内容类型（Content-Types）。 |
| Accept-Charset | 可接受的字符集 |
| Accept-Encoding | 可接受的响应内容的编码方式。 |
| Accept-Language | 可接受的响应内容语言列表。 |
| Accept-Datetime | 可接受的按照时间来表示的响应内容版本 |
| Authorization | 用于表示HTTP协议中需要认证资源的认证信息 |
| Cache-Control | 用来指定当前的请求/回复中的，是否使用缓存机制。 |
| Connection | 客户端（浏览器）想要优先使用的连接类型 |
| Cookie | 由之前服务器通过Set-Cookie（见下文）设置的一个HTTP协议Cookie |
| Content-Length | 以8进制表示的请求体的长度 |
| Content-MD5 | 请求体的内容的二进制 MD5 散列值（数字签名），以 Base64 编码的结果 |
| Content-Type | 请求体的MIME类型 （用于POST和PUT请求中） |
| Date | 发送该消息的日期和时间（以[RFC 7231](https://www.nowcoder.com/tutorial/96/24304825a0c04ea9a53cdb09cb664834" \l "section-7.1.1.1)中定义的"HTTP日期"格式来发送） |
| Expect | 表示客户端要求服务器做出特定的行为 |
| From | 发起此请求的用户的邮件地址 |
| Host | 表示服务器的域名以及服务器所监听的端口号。如果所请求的端口是对应的服务的标准端口（80），则端口号可以省略。 |
| If-Match | 仅当客户端提供的**实体与服务器**上对应的实体相匹配时，才进行对应的操作。主要用于像 PUT 这样的方法中，仅当从用户上次更新某个资源后，该资源未被修改的情况下，才更新该资源。 |
| **If-Modified-Since** | **允许在对应的资源未被修改的情况下返回304未修改** |
| **If-None-Match** | **允许在对应的内容未被修改的情况下返回304未修改（ 304 Not Modified ），参考 超文本传输协议 的实体标记** |
| If-Range | 如果该实体未被修改过，则向返回所缺少的那一个或多个部分。否则，返回整个新的实体 |
| If-Unmodified-Since | 仅当该实体自某个特定时间以来未被修改的情况下，才发送回应。 |
| Max-Forwards | 限制该消息可被代理及网关转发的次数。 |
| Origin | 发起一个针对[跨域资源共享](http://itbilu.com/javascript/js/VkiXuUcC.html)的请求（该请求要求服务器在响应中加入一个Access-Control-Allow-Origin的消息头，表示访问控制所允许的来源）。 |
| Pragma | 与具体的实现相关，这些字段可能在请求/回应链中的任何时候产生。 |
| Proxy-Authorization | 用于向代理进行认证的认证信息。 |
| Range | 表示请求某个实体的一部分，字节偏移以0开始。 |
| Referer | 表示浏览器所访问的前一个页面，可以认为是之前访问页面的链接将浏览器带到了当前页面。Referer其实是Referrer这个单词，但RFC制作标准时给拼错了，后来也就将错就错使用Referer了。 |
| TE | 浏览器预期接受的传输时的编码方式：可使用回应协议头Transfer-Encoding中的值（还可以使用"trailers"表示数据传输时的分块方式）用来表示浏览器希望在最后一个大小为0的块之后还接收到一些额外的字段。 |
| User-Agent | 浏览器的身份标识字符串 |
| Upgrade | 要求服务器升级到一个高版本协议。 |
| Via | 告诉服务器，这个请求是由哪些代理发出的。 |
| Warning | 一个一般性的警告，表示在实体内容体中可能存在错误。 |

### 强，协商缓存

缓存分为两种：强缓存和协商缓存，根据响应的header内容来决定。

|  |  |  |  |
| --- | --- | --- | --- |
|  | 获取资源形式 | 状态码 | 发送请求到服务器 |
| 强缓存 | 从缓存取 | 200（from cache） | 否，直接从缓存取 |
| 协商缓存 | 从缓存取 | 304（not modified） | 是，通过服务器来告知缓存是否可用 |

强缓存相关字段有expires，cache-control。如果cache-control与expires同时存在的话，cache-control的优先级高于expires。

协商缓存相关字段有Last-Modified/If-Modified-Since，Etag/If-None-Match

### HTTP状态码说说你知道的

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的资源

202    Accepted    **已接受**。已经接受请求，但未处理完成

203    Non-Authoritative Information    非授权信息。请求成功。但返回的meta信息不在原始的服务器，而是一个副本

204    No Content    无内容。服务器成功处理，但未返回内容。在未更新网页的情况下，可确保浏览器继续显示当前文档

205    Reset Content    重置内容。服务器处理成功，用户终端（例如：浏览器）应重置文档视图。可通过此返回码清除浏览器的表单域

206    Partial Content    部分内容。服务器成功处理了部分GET请求

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于用户终端（例如：浏览器）选择

301    Moved Permanently    永久移动。请求的资源已被永久的移动到新URI，返回信息会包括新的URI，浏览器会自动定向到新URI。今后任何新的请求都应使用新的URI代替

302    Found    临时移动。与301类似。但资源只是临时被移动。客户端应继续使用原有URI

303    See Other    查看其它地址。与301类似。使用GET和POST请求查看

304    Not Modified    未修改。所请求的资源未修改，服务器返回此状态码时，不会返回任何资源。客户端通常会缓存访问过的资源，通过提供一个头信息指出客户端希望只返回在指定日期之后修改的资源

305    Use Proxy    使用代理。所请求的资源必须通过代理访问

306    Unused    已经被废弃的HTTP状态码

307    Temporary Redirect    临时重定向。与302类似。使用GET请求重定向

400    Bad Request    客户端请求的语法错误，服务器无法理解

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    服务器无法根据客户端的请求找到资源（网页）。通过此代码，网站设计人员可设置"您所请求的资源无法找到"的个性页面

500    Internal Server Error    服务器内部错误，无法完成请求

501    Not Implemented    服务器不支持请求的功能，无法完成请求

502    Bad Gateway    作为网关或者代理工作的服务器尝试执行请求时，从远程服务器接收到了一个无效的响应

503    Service Unavailable    由于超载或系统维护，服务器暂时的无法处理客户端的请求。延时的长度可包含在服务器的Retry-After头信息中

504    Gateway Time-out    充当网关或代理的服务器，未及时从远端服务器获取请求

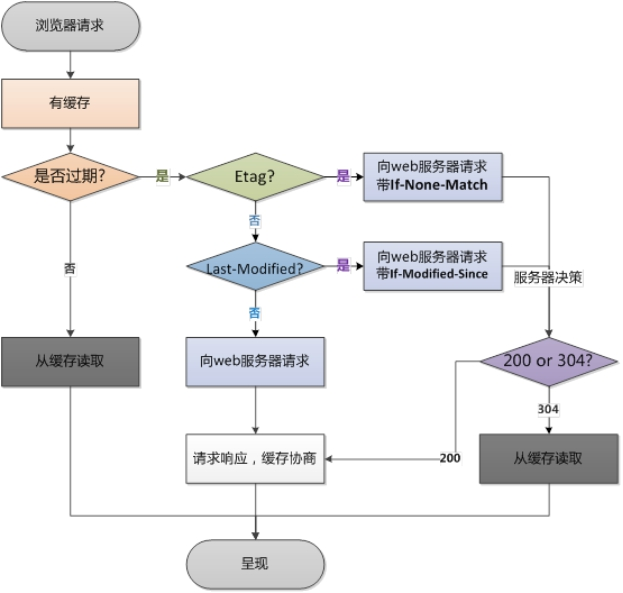
505    HTTP Version not supported    服务器不支持请求的HTTP协议的版本，无法完成处理

### 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。

### 强缓存、协商缓存什么时候用哪个

因为服务器上的资源不是一直固定不变的，大多数情况下它会更新，这个时候如果我们还访问本地缓存，那么对用户来说，那就相当于资源没有更新，用户看到的还是旧的资源；所以我们希望服务器上的资源更新了浏览器就请求新的资源，没有更新就使用本地的缓存，以最大程度的减少因网络请求而产生的资源浪费。



### 前端优化

降低请求量：合并资源，减少HTTP 请求数，minify / gzip 压缩，webP，lazyLoad。

加快请求速度：预解析DNS，减少域名数，并行加载，CDN 分发。

缓存：HTTP 协议缓存请求，离线缓存 manifest，离线数据缓存localStorage。

渲染：JS/CSS优化，加载顺序，服务端渲染，pipeline。

### GET和POST的区别（地方、长度、安全、编码方式、缓存、历史记录、数据包）

get参数通过**url传递**，post放在**request body**中。

get请求在**url中传递的参数是有长度限制的**，而post没有。

get比post更不安全，因为参数直接暴露在url中，所以不能用来传递敏感信息。

get请求只能进行url编码，而post支持多种编码方式

get请求会**浏览器主动cache**，而post支持多种编码方式。

get请求参数会被完整保留在浏览**历史记录**里，而post中的参数不会被保留。

GET和POST本质上就是**TCP链接**，并无差别。但是由于HTTP的规定和浏览器/服务器的限制，导致他们在应用过程中体现出一些不同。

GET产生**一个TCP数据包；POST产生两个TCP数据包。**

### 301和302的区别

301 Moved Permanently 被**请求的资源已永久移动到新位置**，并且将来任何对此资源的引用都应该使用本响应返回的若干个URI之一。如果可能，拥有链接编辑功能的客户端应当自动把请求的地址修改为从服务器反馈回来的地址。除非额外指定，否则这个响应也是可缓存的。

302 Found 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。

字面上的区别就是301是永久重定向，而302是临时重定向。

301比较常用的场景是使用域名跳转。302用来做临时跳转 比如未登陆的用户访问用户中心重定向到登录页面。

### HTTP支持的方法

GET, POST, HEAD, OPTIONS, PUT, DELETE, TRACE, CONNECT

### 如何画一个三角形

div {

**width**:0px;

**height**:0px;

**border-top**:10px solid red;

**border-right**:10px solid transparent;

**border-bottom**:10px solid transparent;

**border-left**:10px solid transparent;

}

### 状态码304和 200

状态码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**。即**返回的数据为全量的数据**，如果文件**不通过GZIP压缩**的话，**文件是多大，则要有多大传输量**。

状态码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状态码。即客户端和服务器端只需要传输很少的数据量来做文件的校验，如果文件没有修改过，则不需要返回全量的数据。

### 说一下浏览器缓存

缓存分为两种：强缓存和协商缓存，根据响应的header内容来决定。

强缓存相关字段有expires，cache-control。如果cache-control与expires同时存在的话，cache-control的优先级高于expires。

协商缓存相关字段有Last-Modified/If-Modified-Since，Etag/If-None-Match

### HTML5新增的元素

首先html5为了更好的实践web语义化，增加了header，footer，nav，aside,sectio，artical，n等语义化标签，在表单方面，为了增强表单，为input增加了color，emial，data ,range等类型，**在存储方面，提供了sessionStorage，localStorage,和离线存储**，通过这些存储方式方便数据在客户端的存储和获取，在多媒体方面规定了音频和视频元素audio和vedio，另外还有地理定位，canvas画布，拖放drag，多线程编程的web worker和websocket协议

### 在地址栏里输入一个URL,到这个页面呈现出来，中间会发生什么？

这是一个必考的面试问题，

输入url后，首先需要找到这个url域名的服务器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文件中是否有记录**，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包会依次经过传输层，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作出响应，返回相应的html给浏览器，因为html是一个树形结构，浏览器根据这个html来构建DOM树，在dom树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代码的后面，之后根据外部央视，内部央视，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，再决定是否返回304

### cookie和session的区别，localstorage和sessionstorage的区别

Cookie和session都可用来存储用户信息，cookie存放于客户端，session存放于服务器端，因为cookie存放于客户端有可能被窃取，所以cookie一般用来存放不敏感的信息，比如用户设置的网站主题，敏感的信息用session存储，比如用户的登陆信息，session可以存放于文件，数据库，内存中都可以，cookie可以服务器端响应的时候设置，也可以客户端通过JS设置cookie会在请求时在http首部发送给客户端，cookie一般在客户端有大小限制，一般为4K，

下面从几个方向区分一下cookie，localstorage，sessionstorage的区别

1、生命周期：

Cookie：可设置失效时间，否则默认为关闭浏览器后失效

Localstorage:除非被手动清除，否则永久保存

Sessionstorage：仅在当前网页会话下有效，关闭页面或浏览器后就会被清除

2、存放数据：

Cookie：4k左右

Localstorage和sessionstorage：可以保存5M的信息

3、http请求：

Cookie：每次都会携带在http头中，如果使用cookie保存过多数据会带来性能问题

其他两个：仅在客户端即浏览器中保存，**不参与和服务器的通信**

4、易用性：

Cookie：**需要程序员自己封装，原生的cookie接口不友好**

其他两个：**即可采用原生接口，亦可再次封装**

5、应用场景：

从安全性来说，因为每次http请求都回携带cookie信息，这样子浪费了带宽，所以cookie应该尽可能的少用，此外cookie还需要指定作用域，不可以跨域调用，限制很多，但是用户识别用户登陆来说，cookie还是比storage好用，其他情况下可以用storage，localstorage可以用来在页面传递参数，sessionstorage可以用来保存一些临时的数据，防止用户刷新页面后丢失了一些参数，

### 常见的HTTP的头部

可以将http首部分为通用首部，请求首部，响应首部，实体首部

通用首部表示一些通用信息，比如date表示报文创建时间，

请求首部就是请求报文中独有的，如cookie，和缓存相关的如if-Modified-Since

响应首部就是响应报文中独有的，如set-cookie，和重定向相关的location，

实体首部用来描述实体部分，如allow用来描述可执行的请求方法，content-type描述主题类型，content-Encoding描述主体的编码方式

### HTTP2.0 的特性

http2.0的特性如下：

1、**内容安全**，应为http2.0是基于https的，天然具有安全特性，通过http2.0的特性可以避免单纯使用https的性能下降

2、**二进制格式**，http1.X的解析是基于文本的，http2.0将所有的**传输信息**分割为更小的消息和帧，并对他们采用二进制格式编码，**基于二进制可以让协议有更多的扩展性**，比如引入了帧来传输数据和指令

3、**多路复用**，这个功能相当于是长连接的增强，每个request请求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，另外多路复用中也支持了流的优先级，允许客户端告诉服务器那些内容是更优先级的资源，可以优先传输，

### cache-control的值有哪些

cache-control是一个通用消息头字段被用于HTTP请求和响应中，通过指定指令来实现缓存机制，这个缓存指令是单向的，常见的取值有private、no-cache、max-age、must-revalidate等，默认为private。

### 浏览器在生成页面的时候，会生成那两颗树？

构造两棵树，DOM树和CSSOM规则树

当浏览器接收到服务器相应来的HTML文档后，会遍历文档节点，生成DOM树，

CSSOM规则树由浏览器解析CSS文件生成，

### csrf和xss的网络攻击及防范

CSRF：跨站请求伪造，可以理解为攻击者**盗用了用户的身份**，以用户的名义发送了恶意请求，比如用户登录了一个网站后，立刻在另一个页面访问量攻击者用来制造攻击的网站，这个网站要求访问刚刚登陆的网站，并发送了一个恶意请求，这时候CSRF就产生了，比如这个制造攻击的网站使用一张图片，但是这种图片的链接却是可以修改数据库的，这时候攻击者就可以以用户的名义操作这个数据库，防御方式的话：使用**验证码，**检查**https头部的refer**，使用**token**

XSS：跨站脚本攻击，是说攻击者通过注入恶意的脚本，在用户浏览网页的时候进行攻击，比如获取cookie，或者其他用户身份信息，可以分为存储型和反射型，存储型是攻击者输入一些数据并且存储到了数据库中，其他浏览者看到的时候进行攻击，反射型的话不存储在数据库中，往往表现为将攻击代码放在url地址的请求参数中，防御的话为cookie设置httpOnly属性，对用户的输入进行检查，进行特殊字符过滤

### 怎么看网站的性能如何

检测页面加载时间一般有两种方式，一种是**被动去测**：就是在被检测的页面**置入脚本或探针**，当用户访问网页时，**探针自动采集数据并传回数据库进行分析**，另一种**主动监测**的方式，即主动的搭建分布式受控环境，**模拟用户发起页面访问请求，主动采集性能数据并分析**，在检测的精准度上，专业的第三方工具效果更佳，比如说性能极客

### 介绍HTTP协议(特征)

HTTP是一个基于TCP/IP通信协议来传递数据（HTML 文件, 图片文件, 查询结果等）HTTP是一个属于应用层的面向对象的协议，由于其简捷、快速的方式，适用于分布式超媒体信息系统。它于1990年提出，经过几年的使用与发展，得到不断地完善和扩展。目前在WWW中使用的是HTTP/1.0的第六版，HTTP/1.1的规范化工作正在进行之中，而且HTTP-NG(Next Generation of HTTP)的建议已经提出。HTTP协议工作于客户端-服务端架构为上。浏览器作为HTTP客户端通过URL向HTTP服务端即WEB服务器发送所有请求。Web服务器根据接收到的请求后，向客户端发送响应信息。

### 输入URL到页面加载显示完成发生了什么?

DNS解析

TCP连接

发送HTTP请求

服务器处理请求并返回HTTP报文

浏览器解析渲染页面

连接结束

### 说一下对Cookie和Session的认知，Cookie有哪些限制？

1.cookie数据存放在客户**的浏览器**上，session数据放在**服务器上。**

2. cookie不**是很安全，别**人可以分析存放在本地的COOKIE并进行COOKIE欺骗  
考虑到安全应当**使用session。**

3. session会在**一定时间内**保存在服务器上。当访问增多，会比较占用你服务器的性能  
考虑到**减轻服务器性能方面**，应当使用COOKIE。

4. 单个cookie保存的数据**不能超过4K**，很多浏览器都限制一个站点最多保**存20个cookie**。

### 描述一下XSS和CRSF攻击？防御方法？

XSS, 即为（Cross Site Scripting）, 中文名为跨站脚本, 是发生在目标用户的浏览器层面上的，当渲染DOM树的过程成发生了不在预期内执行的JS代码时，就发生了XSS攻击。大多数XSS攻击的主要方式是嵌入一段远程或者第三方域上的JS代码。实际上是在目标网站的作用域下执行了这段js代码。

CSRF（Cross Site Request Forgery，跨站请求伪造），字面理解意思就是在别的站点伪造了一个请求。专业术语来说就是在**受害者访问一个网站时**，其 **Cookie 还没有过期的情况**下，攻击者**伪造一个链接地址发送受害者并欺骗让其点击**，从而形成 CSRF 攻击。

XSS防御的总体思路是：对输入(和URL参数)进行过滤，对输出进行编码。也就是对提交的**所有内容进行过滤**，对url中的参数进行过滤，过滤掉会导致脚本执行的相关内容；然后对动态输出到页面的内容进行html编码，使脚本无法在浏览器中执行。虽然对输入过滤可以被绕过，但是也还是会拦截很大一部分的XSS攻击。

防御CSRF 攻击主要有三种策略：**验证 HTTP Referer 字段；在请求地址中添加 token 并验证；在 HTTP 头中自定义属性并验证。**

### 知道304吗，什么时候用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。

### 具体有哪些请求头是跟缓存相关的

缓存分为两种：强缓存和协商缓存，根据响应的header内容来决定。

强缓存相关字段有expires，cache-control。如果cache-control与expires同时存在的话，cache-control的优先级高于expires。

协商缓存相关字段有Last-Modified/If-Modified-Since，Etag/If-None-Match

### cookie和session的区别

1.    cookie数据存放在客户的**浏览器**上，session数据放在**服务器上。**

2.    cookie不是**很安全，别**人可以分析存放在本地的COOKIE并进行COOKIE欺骗  
考虑到安全应当**使用session。**

3.    session会在一**定时间内保存在服务器上**。当访问增多，会比较占用你服务器的性能  
考虑到减**轻服务器性能方面，应当使用COOKIE**。

4.    单个cookie保存的数据不能**超过4K，很**多浏览器都限制一个站点最多保**存20个cookie。**

### cookie有哪些字段可以设置

name字段为一个cookie的名称。

value字段为一个cookie的值。

**domain字段为可以访问此cookie的域名。**

非顶级域名，如二级域名或者三级域名，设置的cookie的domain只能为顶级域名或者二级域名或者三级域名本身，不能设置其他二级域名的cookie，否则cookie无法生成。

顶级域名只能设置domain为顶级域名，不能设置为二级域名或者三级域名，否则cookie无法生成。

二级域名能读取设置了domain为顶级域名或者自身的cookie，不能读取其他二级域名domain的cookie。所以要想cookie在多个二级域名中共享，需要设置domain为顶级域名，这样就可以在所有二级域名里面或者到这个cookie的值了。  
顶级域名只能获取到domain设置为顶级域名的cookie，其他domain设置为二级域名的无法获取。

**path字段为可以访问此cookie的页面路径**。 比如domain是abc.com,path是/test，那么只有/test路径下的页面可以读取此cookie。

expires/Max-Age 字段为此cookie超时时间。若设置其值为一个时间，那么当到达此时间后，此cookie失效。不设置的话默认值是Session，意思是cookie会和session一起失效。当浏览器关闭(不是浏览器标签页，而是整个浏览器) 后，此cookie失效。

Size字段 此cookie大小。

http字段  cookie的httponly属性。若此属性为true，则只有在http请求头中会带有此cookie的信息，而不能通过document.cookie来访问此cookie。

secure 字段 设置是否只能通过https来传递此条cookie

### cookie有哪些编码方式？

encodeURI（）

### 既然你看过图解http，那你回答下200和304的区别

200    OK    请求成功。一般用于GET与POST请求

304    Not Modified    未修改。所请求的资源未修改，服务器返回此状态码时，不会返回任何资源。客户端通常会缓存访问过的资源，通过提供一个头信息指出客户端希望只返回在指定日期之后修改的资源

### 除了cookie，还有什么存储方式。说说cookie和localStorage的区别

还有localStorage，sessionStorage，indexdDB等

cookie和localStorage的区别：

cookie数据始终在同源的http请求中携带(即使不需要)，即cookie在浏览器和服务器间来回传递

cookie数据还**有路径（path）的概念，可以**限制。cookie只属于某个路径下

存储大小限制也不同，cookie数据**不能超过4K，**同时因为每次http请求都会携带cookie，所以cookie只适合保存很小的数据，如回话标识。

localStorage虽然也有**存储大小的限制，但是比cookie大得多，可以达到5M或更大**

localStorage始终有效，窗口或浏**览器关闭也一直保存，**因此用作持久数据；cookie只在设置的cookie过期时间之前一直有效，即使窗口和浏览器关闭。

### 浏览器输入网址到页面渲染全过程

DNS解析

TCP连接

发送HTTP请求

服务器处理请求并返回HTTP报文

浏览器解析渲染页面

连接结束

### HTML5和CSS3用的多吗？你了解它们的新属性吗？有在项目中用过吗？

html5：

1）标签增删

8个语义元素 header section footer aside nav main article figure

内容元素mark高亮 progress进度

新的表单控件calander date time email url search

新的input类型 color date datetime datetime-local email

移除过时标签big font frame frameset

2）canvas绘图，支持内联SVG。支持MathML

3）多媒体audio video source embed track

4）本地离线存储，把需要离线存储在本地的文件列在一个manifest配置文件

5）web存储。localStorage、SessionStorage

css3：

CSS3边框如border-radius，box-shadow等；CSS3背景如background-size，background-origin等；CSS3 2D，3D转换如transform等；CSS3动画如animation等。 参考[https://www.cnblogs.com/xkweb/p/5862612.html](https://www.cnblogs.com/xkweb/p/5862612.html" \t "https://www.nowcoder.com/tutorial/96/_blank)

### HTTP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的资源

202    Accepted    已接受。已经接受请求，但未处理完成

203    Non-Authoritative Information    非授权信息。请求成功。但返回的meta信息不在原始的服务器，而是一个副本

204    No Content    无内容。服务器成功处理，但未返回内容。在未更新网页的情况下，可确保浏览器继续显示当前文档

205    Reset Content    重置内容。服务器处理成功，用户终端（例如：浏览器）应重置文档视图。可通过此返回码清除浏览器的表单域

206    Partial Content    部分内容。服务器成功处理了部分GET请求

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于用户终端（例如：浏览器）选择

301    Moved Permanently    永久移动。请求的资源已被永久的移动到新URI，返回信息会包括新的URI，浏览器会自动定向到新URI。今后任何新的请求都应使用新的URI代替

302    Found    临时移动。与301类似。但资源只是临时被移动。客户端应继续使用原有URI

303    See Other    查看其它地址。与301类似。使用GET和POST请求查看

304    Not Modified    未修改。所请求的资源未修改，服务器返回此状态码时，不会返回任何资源。客户端通常会缓存访问过的资源，通过提供一个头信息指出客户端希望只返回在指定日期之后修改的资源

305    Use Proxy    使用代理。所请求的资源必须通过代理访问

306    Unused    已经被废弃的HTTP状态码

307    Temporary Redirect    临时重定向。与302类似。使用GET请求重定向

400    Bad Request    客户端请求的语法错误，服务器无法理解

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    服务器无法根据客户端的请求找到资源（网页）。通过此代码，网站设计人员可设置"您所请求的资源无法找到"的个性页面

500    Internal Server Error    服务器内部错误，无法完成请求

501    Not Implemented    服务器不支持请求的功能，无法完成请求

502    Bad Gateway    作为网关或者代理工作的服务器尝试执行请求时，从远程服务器接收到了一个无效的响应

503    Service Unavailable    由于超载或系统维护，服务器暂时的无法处理客户端的请求。延时的长度可包含在服务器的Retry-After头信息中

504    Gateway Time-out    充当网关或代理的服务器，未及时从远端服务器获取请求

505    HTTP Version not supported    服务器不支持请求的HTTP协议的版本，无法完成处理

### http常见的请求方法

get、post，这两个用的是最多的，还有很多比如patch、delete、put、options等等

### get和post的区别

GET - 从指定的资源请求数据。

POST - 向指定的资源提交要被处理的数据。

GET：不同的浏览器和服务器不同，一般限制在2~8K之间，更加常见的是1k以内。

GET和POST的底层也是TCP/IP，GET/POST都是TCP链接。

GET产生一个TCP数据包；POST产生两个TCP数据包。

**对于GET方式的请求，浏览器会把http header和data一并发送出去，服务器响应200（返回数据）；**

**而对于POST，浏览器先发送header，服务器响应100 continue，浏览器再发送data，服务器响应200 ok（返回数据）。**

### 说说302，301，304的状态码

301    Moved Permanently    永久移动。请求的资源已被永久的移动到新URI，返回信息会包括新的URI，浏览器会自动定向到新URI。今后任何新的请求都应使用新的URI代替

302    Found    临时移动。与301类似。但资源只是临时被移动。客户端应继续使用原有URI

304    Not Modified    未修改。所请求的资源未修改，服务器返回此状态码时，不会返回任何资源。客户端通常会缓存访问过的资源，通过提供一个头信息指出客户端希望只返回在指定日期之后修改的资源

### web性能优化

降低请求量：合并资源，减少HTTP 请求数，minify / gzip 压缩，webP，lazyLoad。

加快请求速度：预解析DNS，减少域名数，并行加载，CDN 分发。

缓存：HTTP 协议缓存请求，离线缓存 manifest，离线数据缓存localStorage。

渲染：JS/CSS优化，加载顺序，服务端渲染，pipeline。

### 浏览器缓存机制

缓存分为两种：强缓存和协商缓存，根据响应的header内容来决定。

强缓存相关字段有expires，cache-control。如果cache-control与expires同时存在的话，cache-control的优先级高于expires。

协商缓存相关字段有Last-Modified/If-Modified-Since，Etag/If-None-Match

### post和get区别

GET - 从指定的资源请求数据。

POST - 向指定的资源提交要被处理的数据。

GET：不同的浏览器和服务器不同，一般限制在2~8K之间，更加常见的是1k以内。

GET和POST的底层也是TCP/IP，GET/POST都是TCP链接。

GET产生一个TCP数据包；POST产生两个TCP数据包。

对于GET方式的请求，浏览器会把http header和data一并发送出去，服务器响应200（返回数据）；

而对于POST，浏览器先发送header，服务器响应100 continue，浏览器再发送data，服务器响应200 ok（返回数据）。

[上一节](https://www.nowcoder.com/tutorial/96/24304825a0c04ea9a53cdb09cb664834)