###### http常见消息头？消息头可以分为哪几类？

请求头：

Accept-Charset

Accept：MIME

Accept-Language

Accept-Encoding

Cookie

Host

Origin

Referer：当前页面的来源页面的地址

实体头：

Content-Length

Content-Encoding

Content-Language

Content-Type ：实体头部用于指示资源的MIME类型

响应头：

Allow:枚举资源所支持的http方法集合

Etag

Expires:指定过期日期

Server

Vary：它的值是请求头的名称，可以是一个或者多个，告诉缓存服务器，当客户端第二次请求同一个资源时，请求头和Vary中列出的请求头的值也一样时，才可以读缓存，否则要从原始服务器中请求资源。常常用在内容协商机制。常用的值：Accept-Language、User-Agent、Cookie。

Set-Cookie

通用头：

Cache-Control

Connection

Date

Via: 由代理服务器添加，列出请求或者响应经过哪里代理服务器，用来追踪消息转发情况，防止循环请求

Keep-Alive: max=1000,timeout=秒

timeout：指定了一个空闲连接需要保持打开状态的最小时长（以秒为单位）。需要注意的是，如果没有在传输层设置 keep-alive TCP message 的话，大于 TCP 层面的超时设置会被忽略。

max：在连接关闭之前，在此连接可以发送的请求的最大值。在非管道连接中，除了 0 以外，这个值是被忽略的，因为需要在紧跟着的响应中发送新一次的请求。HTTP 管道连接则可以用它来限制管道的使用。

http1.x有什么特点？优点/缺点都有哪些？

特点：

1 基于响应-请求模式

2 无状态：对于发送过的请求或响应都不做持久化处理

3 通过headers使得http可扩展，只要客户端和服务端达成语义一致，非常容易拓展。

缺点：

1.使用明文传输，不安全

2.头部携带大量且重复的信息

3.队头阻塞带来高延迟问题（队头阻塞：客户端一次性发送多个请求，如果前一个请求遇到了阻塞，那么后面的请求即使处理完了也要等前面的请求先处理才会返回到客户端，响应的顺序和请求顺序一致）

4.只能由客户端发起请求

###### https的加密解密过程

S1.向CA机构申请证书

S2.从CA机构获取证书及密钥，把密钥配置到服务器

C3.客户端发起请求

S4.服务器把证书发给客户端

C5.客户端验证证书，验证通过或者接受不受信任到证书，从证书中获取公钥

C6.客户端生成随机数，使用证书到公钥加密，发送给服务端

S7.服务端使用私钥解密，获取客户端的随机数

S8.服务器告诉客户端，已收消息，后面可以开始使用他们两都知道都密钥进行对称加密算法进行通讯

###### http2和http1.x相比有什么改进的地方

1.二进制分帧技术，把报文分成头部Header帧和数据Data帧

2.对头部进行压缩，使用“HPACK”算法，在客户端和服务端建立字典，使用索引号表示重复对字符串

3.使用多路复用技术，解决队头阻塞问题

4.使用server push，使服务端具备主动发起请求对能力

###### http的请求方法有哪些？

http1.0：

GET/

POST/

HEAD

http1.1:

TRACE/

OPTION/

DELETE/

PUT/

CONNECTION

###### 常见http状态码

200:OK

201:Created 通常作为POST请求的响应码

204:请求已经响应成功，但是客户端不需要离开当前页，通常使用在PUT请求

206：断点续传成功

301:重定向

304:从缓存读取

400：Bad Request 表示语法无效，服务器无法理解该请求，一般是参数不对

401:没有权限

404:没有找到资源

412 (Precondition Failed，先决条件失败)

415:不支持的媒体类型

500：服务器错误

###### http的内容协商机制

定义：一个url请求常常代表若干个不同的资源，而服务器根据请求头的信息，选择合适的资源且返回，这个机制叫做HTTP的内容协商机制。

协商头包括：Accent/Accept-Charset/Accept-Language/Accept-Encoding

有三种方式可以决定服务器返回哪个资源给客户端：

1.客户端驱动

客户端发起请求，服务器发送可选项列表，客户端作出选择后在发送第二次请求。

优点：比较容易实现；

缺点：增加了时延，至少要发送两次请求，第一次请求获取资源列表，第二次获取选择的副本；如何实现？

2.服务器驱动

服务器检查客户端的请求首部集并决定提供哪个版本的页面。

优点：比客户端驱动的协商要快。HTTP提供了q机制，允许服务器近似匹配，还提供了vary首部供服务器告知下游的设备（如代理服务器）如何对请求估值；

缺点：首部集不匹配，服务器要做猜测；

3.透明协商

某个中间设备（通常是缓存代理）代表客户端进行协商。使用Vary实现

优点：免除了web服务器的协商开销，比客户端驱动的协商要快；

缺点：HTTP并没有提供相应的规范；

###### 断点续传的原理

断点续传一般在文件下载的情况使用，通过使用HTTP的Accept-Range、Range、If-Range、Range-Content等头部实现。步骤如下：

1.客户端请求一个资源

2.服务器返回资源信息，头部包含：Content-Length和Accept-Range告诉客户端，资源的长度和接收断点续传。

3.客户端如果下载中段了，需要接着下载，那么发请求，头部包含：If-Range和Range，If-Range可以是Last-Modify的值或者是Etag的值，Range是bytes的起始范围

4.服务端验证如果资源没有过期，即If-Range满足条件，那么从请求头Range指定的范围返回资源，状态码为206；如果If-Range不满足，那么从头开始返回资源，状态码为200

###### Content-Security-Policy响应头有什么作用

简称CSP，用来防止跨站脚本攻击（XSS-Cross Site Scripting）.开发者配置网站的哪些外部资源可以加载和执行，实质就是给网站配置白名单。也可以通过meta标签实现。

注意：

（1）script-src和object-src是必设的，除非设置了default-src。

因为攻击者只要能注入脚本，其他限制都可以规避。而object-src必设是因为 Flash 里面可以执行外部脚本。

（2）script-src不能使用unsafe-inline关键字（除非伴随一个nonce值），也不能允许设置data:URL。

下面是两个恶意攻击的例子。

<img src="x" onerror="evil()">

<script src="data:text/javascript,evil()"></script>

（3）必须特别注意 JSONP 的回调函数。

<script

src="/path/jsonp?callback=alert(document.domain)//">

</script>

Content-Security-Policy: default-src 'self'

限制所有外部资源

<meta http-equiv="Content-Security-Policy" content="script-src 'self'; object-src 'none'; style-src cdn.example.org third-party.org; child-src https:">

脚本：只信任当前域名

<object>标签：不信任任何URL，即不加载任何资源

样式表：只信任cdn.example.org和third-party.org

框架（frame）：必须使用HTTPS协议加载

其他资源：没有限制

###### websocker有什么特点，如何通讯的？

WebSocket是html5推出的协议。

1建立连接后，既可以由客户端发起请求，也可以由服务器发起请求，客户端和服务器是对等的。

2是一种持久化的协议，建立连接后就可以一直通讯了

3建立在 TCP 协议之上，服务器端的实现比较容易。

4与 HTTP 协议有着良好的兼容性。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。Upgrade: websocket

Connection: Upgrade

5数据格式比较轻量，性能开销小，通信高效。

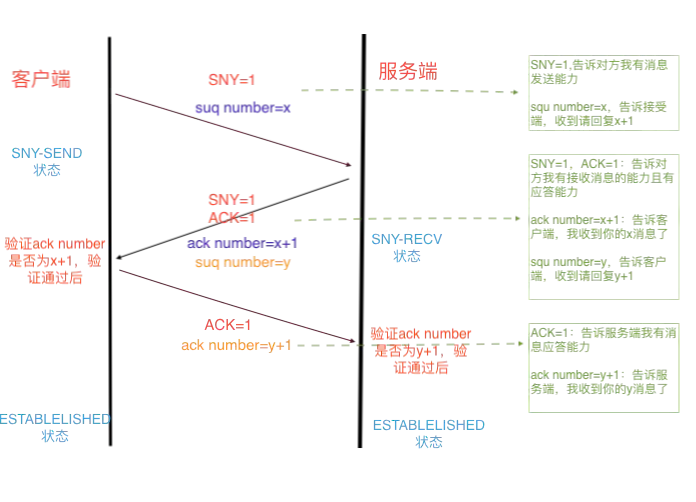
6可以发送文本，也可以发送二进制数据。

7没有同源限制，客户端可以与任意服务器通信。

8协议标识符是ws（如果加密，则为wss），服务器网址就是 URL。

###### http建立连接的三次握手？为什么要三次握手？

SNY/ACK标记为了确认对方具备收发能力；suq number防止已经失效的报文又突然发送到服务端



###### http建立连接的四次挥手，为什么要四次挥手

TCP协议是面向连接的，可靠的，基于字节流的运输层协议，是全双工通信模式

1.客户端发送FIN断开报文时，告诉客户端发送消息完毕，但是客户端还可以接受消息

2.服务端收到FIN报文，返回ACK报文，告诉客户端它已经知道客户端发送完了消息，但是服务端还可以发送消息（可能手头还有事情没处理完）

3.服务端发送FIN报文，表示服务端也不再发送消息了

4.客户端收到FIN报文，断开TCP连接

###### 什么是http的长连接，短连接，如何实现长连接

在http1.0，默认使用短连接，通用头Connect：close，就是客户端发起一个请求，服务器返回响应，然后连接就断开了。生命周期是连接-传输数据-断开连接

在http1.1，默认使用长连接，通用头Connect：keep-alive。生命周期：连接-传输数据-传输数据-传输数据-...-断开连接。可以配合通用头Keep-Alive使用

###### http如何自动升级为https

（在url地址栏输入http协议，返回的页面却是https）

1.地址栏输入http://baidu.com

2.返回307状态吗，响应头Location: https://www.baidu.com/

Non-Authoritative-Reason: HSTS

3.重定向到Location指定的url

###### 客户端如何判断请求所得到的响应数据已经接收完成

1.通过实体头Content-Length来判断

2.通过响应头Transfer-Encoding:chunk判断，它的格式是：

/r/n /r/n之后+响应内容(16进制的长度+内容)+0+/r/n /r/n，如果有0+/r/n /r/n说明内容传输完毕

3.如果既没有Content-Length也没有Transfer-Encoding，那么客户端等到服务器断开tcp连接才认为数据传送完了

4.Trailer用于验证消息到完整性

HTTP/1.1 200 OK

Content-Type: text/plain

Transfer-Encoding: chunked

Trailer: Expires

7\r\n

Mozilla\r\n

9\r\n

Developer\r\n

7\r\n

Network\r\n

0\r\n

Expires: Wed, 21 Oct 2015 07:28:00 GMT\r\n

\r\n

###### 哪些消息头是为代理服务器设置的？

Via:通用首部，由代理服务器添加，正向代理和反响代理都可使用，记录了协议名称/版本，代理名称或url/端口

X-Forward-Host：请求首部，它的值是客户端ip/代理ip/代理ip，最右端的IP地址表示最近通过的代理服务器，而最左端的IP地址表示最初发起请求的客户端的IP地址。

Via: 1.1 vegur

Via: HTTP/1.1 GWA

Via: 1.0 fred, 1.1 p.example.net

X-Forwarded-For: 203.0.113.195, 70.41.3.18, 150.172.238.178

###### 哪些消息头是为缓存服务器设置的？

Vary

###### 条件请求头有哪些？及他们的应用场景

If-Modified-Since：日期，用来判断缓存是否可用，

If-None-Match：Etag值，用来判断缓存是否可用If-Match：Etag值。

If-Match:

1.对于HEAD/GET方法，搭配Range使用，如果Etage值不匹配，那么返回416范围请求无法满足

2.对于PUT方法，用来防止上传的资源覆盖原始资源更新后的资源。

If-Range：Etag值/日期

If-Unmodified-Since：

1.与 non-safe 方法如 POST 搭配使用，可以用来优化并发控制，例如在某些wiki应用中的做法：假如在原始副本获取之后，服务器上所存储的文档已经被修改，那么对其作出的编辑会被拒绝提交。

2.与含有 If-Range 消息头的范围请求搭配使用，用来确保新的请求片段来自于未经修改的文档。

###### 跨站脚本攻击是什么？(反射性)

在被攻击站点网页注入恶意脚本，比如

1.盗取用户信息，发送到发起攻击者的网站

2.cookie劫持：HTTPOnly

###### 跨域问题有哪些解决方式？

1.jsonp

原理：创建一个<script>标签的src属性可以加载跨域资源，定义一个全局函数赋值给请求url的callback参数，src的请求结果是一个包含数据的函数调用。

优点：使用简单

缺点：只适合get请求

2.cors

原理：利用http

优点：支持所有类型的请求

缺点：老浏览器不支持，服务端也要支持cors

###### CORS是什么？

CORS（跨域资源共享），是W3C的一个标准，它允许浏览器向服务端发起xmlHttpRequest请求。使用它需要浏览器和服务端都支持。目前大部分浏览器都支持该功能，并且浏览器会自动完成，不需要开发者和用户做其他的操作。所以实现cors的关键是服务器。

服务端把跨域请求分为两类：简单请求和非简单请求。简单请求的条件：1，请求方法是HEAD/GET/POST，2）HTTP的头信息不超出以下几种字段：

Accept

Accept-Language

Content-Language

Last-Event-ID

Content-Type：只限于三个值application/x-www-form-urlencoded、multipart/form-data、text/plain

简单请求流程：

1.客户端发起跨域请求，浏览器自动加上Origin请求头。

2.服务端接收到请求报文，查看Origin的值，决定是否同意这次请求。

3.如果Origin指定的源，不在许可范围，那么返回一个正常的http响应。响应头不包含Access-Control-Allow-Origin头，被XmlHttpRequest对象的onerror回调捕获。

4.如何Origin在许可范围内，那么会多出几个头：

1⃣Access-Control-Allow-Origin: http://api.bob.com --- 必须，也可以是\*

2⃣️Access-Control-Allow-Credentials: true --- 可选，表示是否允许发送Cookie开发者必须在AJAX请求中打开withCredentials属性。

var xhr = new XMLHttpRequest();

xhr.withCredentials = true;

如果要发送Cookie，Access-Control-Allow-Origin就不能设为星号，必须指定明确的、与请求网页一致的域名

3⃣️Access-Control-Expose-Headers: FooBar --- 可选

非简单请求：

非简单请求的CORS请求，会在正式通信之前，增加一次HTTP查询请求，称为"预检"请求（preflight）。

1.浏览器发现非简单跨域请求时，会自动发起一个预检请求，请求方法是OPTIONS,自动加上三个属性：

Origin: http://api.bob.com

Access-Control-Request-Method: PUT

Access-Control-Request-Headers: X-Custom-Header

2.如果服务端不同意预检请求，返回一个正常的http响应，没有Access-Control-Allow-Origin字段。浏览器通过onerror回调捕获。

3.如果服务端同意预检请求，那么返回

Access-Control-Allow-Origin --- 必须

Access-Control-Allow-Methods: GET, POST, PUT -- 必须

Access-Control-Allow-Headers: X-Custom-Header -- 如果请求头有Access-Control-Request-Header，那么必须有

Access-Control-Allow-Credentials: true -- 可选

Access-Control-Max-Age: 1728000 -- 可选，预检请求的有效期

4.服务器通过了预检请求，那么后面的非简单请求就和简单请求一样了

Sec-Fetch-Dest: document

Sec-Fetch-Mode: navigate

Sec-Fetch-Site: none

Sec-Fetch-User: ?1

Upgrade-Insecure-Requests: 1

###### 六个简单响应首部

Cache-Control

Content-Language

Content-Type

Expires

Last-Modified

Pragma

###### Access-Control-Expose-Headers有什么用处？

CORS请求时，XMLHttpRequest对象的getResponseHeader()方法只能拿到6个基本字段：Cache-Control、Content-Language、Content-Type、Expires、Last-Modified、Pragma。如果想拿到其他字段，就必须在Access-Control-Expose-Headers里面指定。