### 1 SessionStorage和localStorage还有cookie

**共同点**：都是保存在浏览器端、且同源的  
**不同点**：  
1.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  
sessionStorage和localStorage不会自动把数据发送给服务器，仅在本地保存。  
2.存储大小限制也不同，cookie数据不能超过4K，sessionStorage和localStorage可以达到5M  
3.sessionStorage：仅在当前浏览器窗口关闭之前有效；  
localStorage：始终有效，窗口或浏览器关闭也一直保存，本地存储，因此用作持久数据；  
cookie：只在设置的cookie过期时间之前有效，即使窗口关闭或浏览器关闭  
4.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：在所有同源窗口中都是共享的；也就是说只要浏览器不关闭，数据仍然存在  
cookie: 也是在所有同源窗口中都是共享的.也就是说只要浏览器不关闭，数据仍然存在

**除了cookie， webStorage还有哪些本地存储的方式**

IndexedDB是一种使用浏览器存储大量数据的方法，它创造的数据可以被查询，并且可以离线使用。

### 2 从输入URL到页面加载发生了什么？

**1.读取缓存**：搜索自身的 DNS 缓存。(如果 DNS 缓存中找到IP 地址就跳过了接下来查找 IP 地址步骤，直接访问该 IP 地址。)  
**2.DNS 解析:将域名解析成 IP 地址**，就是寻找哪台机器上有你需要资源的过程  
**3.TCP 连接**：TCP 三次握手，简易描述三次握手：  
 客户端：服务端你在么？  
 服务端：客户端我在，你要连接我么？  
 客户端：是的服务端，我要连接。  
 连接打通，可以开始请求来  
**4.发送 HTTP 请求**：发送HTTP请求的过程就是构建HTTP请求报文并通过TCP协议中发送到服务器指定端口(HTTP协议80/8080, HTTPS协议443)。  
**5.服务器处理请求并返回 HTTP 报文  
6.浏览器解析渲染页面  
7.断开连接：TCP 四次挥手**  
关于第六步浏览器解析渲染页面又可以聊聊如果返回的是html页面  
根据 HTML 解析出 DOM 树  
根据 CSS 解析生成 CSS 规则树  
结合 DOM 树和 CSS 规则树，生成渲染树  
根据渲染树计算每一个节点的信息  
根据计算好的信息绘制页面

### 3 同步和异步的区别?

****同步****：浏览器访问服务器请求，用户看得到页面刷新，重新发请求,等请求完，页面刷新，新内容出现，用户看到新内容,进行下一步操作。  
****异步****：浏览器访问服务器请求，用户正常操作，浏览器后端进行请求。等请求完，页面不刷新，新内容也会出现，用户看到新内容

### 4 什么是Ajax，以及优势和缺点

Ajax就是异步传输+js+xml，

**Ajax应用程序的优势在于：** 1. 通过异步模式，提升了用户体验  
 2. 优化了浏览器和服务器之间的传输，减少不必要的数据往返，减少了带宽占用  
 3. Ajax引擎在客户端运行，承担了一部分本来由服务器承担的工作，从而减少了大用户量下的服务器负载。  
**AJAX最大的特点：** Ajax可以实现局部刷新  
 就是能在不更新整个页面的前提下维护数据。这使得Web应用程序更为迅捷地回应用户动作，并避免了在网络上发送那些没有改变过的信息。

**AJAX缺点**：

1、ajax不支持浏览器back按钮。  
 2、安全问题 AJAX暴露了与服务器交互的细节。  
 3、对搜索引擎的支持比较弱。  
 4、破坏了程序的异常机制。  
 5、不容易调试。

### 5 AJAX应用和传统Web应用有什么不同

在传统的Javascript编程中，如果想得到服务器端数据库或文件上的信息，或者发送客户端信息到服务器，需要建立一个HTML form然后GET或者POST数据到服务器端。用户需要点击”Submit”按钮来发送或者接受数据信息，然后等待服务器响应请求，页面重新加载。  
因为服务器每次都会返回一个新的页面， 所以传统的web应用有可能很慢而且用户交互不友好。  
 使用AJAX技术， 就可以使Javascript通过XMLHttpRequest对象直接与服务器进行交互。通过HTTP Request， 一个web页面可以发送一个请求到web服务器并且接受web服务器返回的信息(不用重新加载页面)，展示给用户的还是通一个页面，用户感觉页面刷新，也看不到到Javascript后台进行的发送请求和接受响应。

### 6 AJAX使用了哪些技术？

HTML / XHTML和CSS - 这些技术用于显示内容和样式。

DOM - 用于动态显示和与数据交互。

XML - 用于与服务器之间传送数据

XMLHttpRequest - 用于客户端和服务器之间的异步通信。

JavaScript - 主要用于客户端验证。

### 7 AJAX中请求的准备状态有哪些？

在AJAX中有5个请求的就绪状态。

0 表示xhr被创建，但尚未使用xhr.open()方法

1 表示open() 方法被调用，建立了连接

2 表示send() 方法被调用，并取得了响应的状态值和响应头。

3 表示响应体正在下载中

4 表示下载体已经完成，可以直接使用responseText

### 8 AJAX代码

1. 创建ajax对象

var xhr = new XMLHttpRequest();

1. 打开请求，设置请求方式

*//请求方法自定，第三个参数通常设为true，异步请求*

xhr.open('GET', url, true);

1. 发送请求

*//可选，设置请求头，根据需要定，post请求的话要写* *//xhr.setRequestHeader("Content-type","application/x-www-form-urlencoded");*

xhr.send(要发送的数据);

1. 接收响应

//服务器响应状态(readyState)改变时都会被执行

xhr.onreadystatechange = function(){

//0 （未初始化）对象已经创建，但还没有调用open()方法

//1 （启动）已经调用open() 方法，但尚未调用send()方法发送请求

//2 （发送）send()方法已调用，请求已经发送完成，但尚未接收到响应

//3 （接收）已经接收到部分响应数据

//4 （完成）已经接收到了全部数据，而且已经可以在客户端使用了

//服务器响应状态(readyState)和响应的HTTP状态(status)同时满足才算成功

if (xhr.readyState==4 && xhr.status==200){

//通过xhr.responseText，获得服务器返回的内容

console.log(xhr.responseText)

}

}

**ajax和fetch的区别:**

**Ajax**利用的是XMLHttpRequest对象来请求数据的

**fetch是**window的一个方法 主要特点是  
1、第一个参数是URL  
2、第二个参数可选参数 可以控制不同的init对象  
3、使用了js 中的promise对象

主要区别：

1、fetch()返回的promise将不会拒绝http的错误状态，即使响应是一个HTTP 404或者500  
2、在默认情况下 fetch不会接受或者发送cookies

使用fetch遇到的问题：

所有的IE浏览器都不会支持 fetch()方法  
服务器端返回 状态码 400 500的时候 不会reject

### 9 HTTP状态码

**2XX 成功**

200：请求成功  
 202：服务器已接受请求，但尚未处理  
 204：服务器成功处理了请求，但未返回内容

**3XX 重定向**

301：(永久重定向)、 302： (临时重定向) 、

实施301后，新网址完全继承旧网址，旧网址的排名等完全清零；实施302后，对旧网址没有影响，但新网址不会有排名  
 303：http1.1协议，禁止被缓存  
 304：(协商缓存成功（资源未修改）的返回值)

**4XX 客户端错误**

400：客户端请求的语法错误，服务器无法理解  
 403：服务器理解请求客户端的请求，但是拒绝执行此请求  
 404：服务器无法根据客户端的请求找到资源（网页）

**5XX 服务器错误**

500 internal sever error，表示服务器端在执行请求时发生了错误

501 Not Implemented，表示服务器不支持当前请求所需要的某个功能

503 service unavailable，表明服务器暂时处于超负载或正在停机维护，无法处理请求

### 10 什么是同源

所谓同源是指，域名，协议，端口相同。

非同源受到的限制：

* 1 cookie不能读取 （如我在自己的站点无法读取博客园用户的cookie）
* 2 dom无法获得
* 3 ajax请求不能发送

### 11 实现跨域的几种方式

* 因为浏览器出于安全考虑，有同源策略。也就是说，如果协议、域名或者端口有一个不同就是跨域，Ajax 请求会失败。为来防止CSRF攻击

**1 JSONP：**  
 JSONP 的原理很简单，就是利用 <script> 标签没有跨域限制的漏洞。  
通过 <script> 标签指向一个需要访问的地址并提供一个回调函数来接收数据当需要通讯时。  
<script src="http://domain/api?param1=a&param2=b&callback=jsonp"></script>  
<script>  
 function jsonp(data) {  
 console.log(data)  
}  
</script>  
JSONP 使用简单且兼容性不错，但是只限于 get 请求。

**缺点：**

 2.1它只支持GET请求而不支持POST等其它类型的HTTP请求：

常见的实现代码其实就是document.createElement('script') 生成一个 script 标签，然后插 body 里而已。在这里根本没有设置请求格式的余地，所以只能get  
2.2它只支持跨域HTTP请求这种情况，不能解决不同域的两个页面之间如何进行JavaScript调用的问题。  
2.3 jsonp在调用失败的时候不会返回各种HTTP状态码。  
2.4缺点是安全性。万一假如提供jsonp的服务存在页面注入漏洞，即它返回的javascript的内容被人控制的。那么结果是什么？所有调用这个 jsonp的网站都会存在漏洞。于是无法把危险控制在一个域名下…所以在使用jsonp的时候必须要保证使用的jsonp服务必须是安全可信的。

**2 CORS 同域安全策略：**

通过跨域资源共享（CORS）发送AJAX请求，普通跨域请求只需在服务端设置Access-Control-Allow-Origin即可。而带有cookie的跨域请求，还需在服务端设置Access-Control-Allow-Credentials，且前端设置xhr对象的withCredentials属性为true。

**3 document.domain**  
该方式只能用于二级域名相同的情况下，比如 a.test.com 和 b.test.com 适用于该方式。  
只需要给页面添加 document.domain = 'test.com' 表示二级域名都相同就可以实现跨域 **4 Proxy代理**：由于同源策略只是浏览器的限制，服务器端并没有这个限制，所以只要A域客户端将请求发送一个代理服务器，然后由代理服务器去请求B域服务器就行了，比如前后端分离的工程，本地调试的时候我们启用nodejs代理服务、线上部署通过nginx代理转发等，都属于这个跨域模式。同样的，这个本质上也只是绕过浏览器的跨域限制而已。

### 12 HTTP和HTTPS区别

1、https协议需要到CA  （Certificate Authority，证书颁发机构）申请证书，一般免费证书较少，因而需要一定费用。(原来网易官网是http，而网易邮箱是https。)

2、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。

3、http和https使用的是完全不同的连接方式，用的端口也不一样，前者是80，后者是443。

4、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。(无状态的意思是其数据包的发送、传输和接收都是相互独立的。无连接的意思是指通信双方都不长久的维持对方的任何信息。)

### 13 Cookie、Session、Token、JWT

**什么是Cookie：**

HTTP 是无状态的协议（对于事务处理没有记忆能力，每次客户端和服务端会话完成时，服务端不会保存任何会话信息）：每个请求都是完全独立的，服务端无法确认当前访问者的身份信息，无法分辨上一次的请求发送者和这一次的发送者是不是同一个人。所以服务器与浏览器为了进行会话跟踪（知道是谁在访问我），就必须主动的去维护一个状态，这个状态用于告知服务端前后两个请求是否来自同一浏览器。而这个状态需要通过 cookie 或者 session 去实现。

cookie 存储在客户端： cookie 是服务器发送到用户浏览器并保存在本地的一小块数据，它会在浏览器下次向同一服务器再发起请求时被携带并发送到服务器上。

cookie 是不可跨域的： 每个 cookie 都会绑定单一的域名，无法在别的域名下获取使用，一级域名和二级域名之间是允许共享使用的（靠的是 domain）。

**什么是session:**

session 是另一种记录服务器和客户端会话状态的机制

session 是基于 cookie 实现的，session 存储在服务器端，sessionId 会被存储到客户端的cookie 中

用户第一次请求服务器的时候，服务器根据用户提交的相关信息，创建对应的 Session

请求返回时将此 Session 的唯一标识信息 SessionID 返回给浏览器

浏览器接收到服务器返回的 SessionID 信息后，会将此信息存入到 Cookie 中，同时 Cookie 记录此 SessionID 属于哪个域名

当用户第二次访问服务器的时候，请求会自动判断此域名下是否存在 Cookie 信息，如果存在自动将 Cookie 信息也发送给服务端，服务端会从 Cookie 中获取 SessionID，再根据 SessionID 查找对应的 Session 信息，如果没有找到说明用户没有登录或者登录失效，如果找到 Session 证明用户已经登录可执行后面操作。

**Cookie 和 Session 的区别**

**安全性**： Session 比 Cookie 安全，Session 是存储在服务器端的，Cookie 是存储在客户端的。

**存取值的类型不同**：Cookie 只支持存字符串数据，想要设置其他类型的数据，需要将其转换成字符串，Session 可以存任意数据类型。

**有效期不同：** Cookie 可设置为长时间保持，比如我们经常使用的默认登录功能，Session 一般失效时间较短，客户端关闭（默认情况下）或者 Session 超时都会失效。

**存储大小不同：** 单个 Cookie 保存的数据不能超过 4K，Session 可存储数据远高于 Cookie，但是当访问量过多，会占用过多的服务器资源。

**什么是Token：**

Token是服务端生成的一串字符串，以作客户端进行请求的一个令牌，当第一次登录后，服务器生成一个Token便将此Token返回给客户端，以后客户端只需带上这个Token前来请求数据即可，无需再次带上用户名和密码。token其实说的更通俗点可以叫暗号，在一些数据传输之前，要先进行暗号的核对，不同的暗号被授权不同的数据操作。

Token好处：

每一次请求都需要携带 token，需要把 token 放到 HTTP 的 Header 里

基于 token 的用户认证是一种服务端无状态的认证方式，服务端不用存放 token 数据。用解析 token 的计算时间换取 session 的存储空间，从而减轻服务器的压力，减少频繁的查询数据库

token 完全由应用管理，所以它可以避开同源策略

**什么是JWT：**

JSON Web Token（简称 JWT）是目前最流行的跨域认证解决方案。是一种认证授权机制。JWT 的原理是，服务器认证以后，生成一个 JSON 对象，发回给用户，就像下面这样。以后，用户与服务端通信的时候，都要发回这个 JSON 对象。服务器完全只靠这个对象认定用户身份。为了防止用户篡改数据，服务器在生成这个对象的时候，会加上签名。服务器就不保存任何 session 数据了，也就是说，服务器变成无状态了，从而比较容易实现扩展。

**Token和JWT区别：**

相同：

都是访问资源的令牌

都可以记录用户的信息

都是使服务端无状态化

都是只有验证成功后，客户端才能访问服务端上受保护的资源

区别：

Token：服务端验证客户端发送过来的 Token 时，还需要查询数据库获取用户信息，然后验证 Token 是否有效。

WT： 将 Token 和 Payload 加密后存储于客户端，服务端只需要使用密钥解密进行 校验（校验也是 JWT 自己实现的）即可，不需要查询或者减少查询数据库，因为 JWT 自 包含了用户信息和加密的数据。

### 14 TCP三次握手四次挥手

三次握手：客户端向服务端发送一个SYN报文，服务端收到报文确认后，发送一个SYN-ACK报文，客户端收到服务器的报文后，返回一个ACK报文，连接建立。

四次挥手：客户端向服务端发送一个FIN报文，服务端收到报文后，立即发送一个ACK报文，随即通知本地服务启动清理工作，当清理完成时，发送一个FIN报文给客户端，客户端收到后返回一个ACK报文，连接关闭。

由于服务端接受到客户端关闭连接请求时，可能一些数据还没有发送完成，因此不能立刻关闭连接，因此多了一次挥手。

15 Https的主要特点

https = http + 完整性保护（报文摘要） + 认证 + 加密

HTTPS只是HTTP通信接口部分用SSL或TLS等协议代替而已。当通信时，HTTP先与SSL通信，再由SSL和TCP进行通信。

https使用混合加密机制，首先使用非对称加密换取对称加密密钥，再使用对称加密传输数据信息

https使用数字证书认证机构（CA）和其相关机构颁发的公开密钥证书，可以使客户端验证服务器公开密钥的真实性。

https在发送数据时，会附加MAC报文摘要，从而防止报文在传输过程中遭遇篡改

### 16 xss和csrf是什么？各有什么防御手段？

xss为跨站脚本攻击，攻击者向web网页中恶意植入代码，用户访问该页面时，就会受到攻击。将一些隐私数据像 cookie、session 发送给攻击者，将受害者重定向到一个由攻击者控制的网站，在受害者的机器上进行一些恶意操作

#### 防御手段：

#### 1 **HttpOnly**

浏览器禁止页面的Javascript访问带有HttpOnly属性的cookie。（实质解决的是：XSS后的cookie劫持攻击）如今已成为一种“标准”的做法

#### **2 输入检查（XSS Filter）**

原理：让一些基于特殊字符的攻击失效。（常见的Web漏洞如XSS、SQLInjection等，都要求攻击者构造一些特殊字符）输入检查一般是检查用户输入的数据中是否包含 <，> 等特殊字符，如果存在，则对特殊字符进行过滤或编码

输入检查的逻辑，必须放在服务器端代码中实现。目前Web开发的普遍做法，是同时哎客户端Javascript中和服务端代码中实现相同的输入检查。客户端的输入检查可以阻挡大部分误操作的正常用户，节约服务器资源。

#### **3 输出检查**

在变量输出到HTML页面时，使用编码或转义的方式来防御XSS攻击

csrf为跨站请求伪造，指攻击者通过设置好的陷阱，强制对已完成认证的用户进行非预期的个人信息或设定信息等某些状态更新。本质CSRF攻击是攻击者利用用户身份操作用户账户的一种攻击方式，CSRF攻击是攻击者借助受害者的 Cookie 骗取服务器的信任，可以在受害者毫不知情的情况下以受害者名义伪造请求发送给受攻击服务器，从而在并未授权的情况下执行在权限保护之下的操作。

防御手段：验证码：CSRF 攻击往往是在用户不知情的情况下构造了网络请求。而验证码会强制用户必须与应用进行交互，才能完成最终请求验证，HTTP Refer（缺点：老版的浏览器可可以篡改Refer），toke验证（在http请求中加入随机产生的toke，服务端进行合法性验证，缺点对每个请求都加入token比较麻烦，且token可能会被refer属性暴露），自定义HTTP属性头（通过XMLHttpRequest这个类，给http请求加上属性头，缺点为只限于ajax）

### 17 Restful API是什么，如何设计RESTful API？

RESTful API是指符合REST设计风格的Web API，为了使的接口安全、易用、可维护以及可扩张，一般设计RESTful API需要考虑以下几个方面：

1. 通信用HTTPS安全协议
2. 在URL中加入版本号
3. URL中的路径不能有动词，只能用名词
4. 用HTTP方法对资源进行增删改查的操作
5. 用HTTP状态吗传达执行结果和失败原因
6. 为集合提供过滤、排序、分页功能
7. 用查询字符串或HTTP首部Accpet进行内容协商，指定返回结果的数据格式
8. 及时更新文档，每个接口都有对应的说明

### 18 HTTPS的握手过程

1 浏览器将自己支持的一套加密规则发送给服务器。

2 服务器从中选出一组加密算法与HASH算法，并将自己的身份信息以证书的形式发回给浏览器。证书里面包含了网站地址，加密公钥，以及证书的颁发机构等信息。

3 浏览器获得网站证书之后浏览器要做以下工作：

验证证书的合法

如果证书受信任，或者是用户接受了不受信的证书，浏览器会生成一串随机数的密码，并用证书中提供的公钥加密。

使用约定好的HASH算法计算握手消息，并使用生成的随机数对消息进行加密，最后将之前生成的所有信息发送给服务器

4 网站接收浏览器发来的数据之后要做以下的操作：

使用自己的私钥将信息解密取出密码，使用密码解密浏览器发来的握手消息，并验证HASH是否与浏览器发来的一致。

使用密码加密一段握手消息，发送给浏览器。

5 浏览器解密并计算握手消息的HASH，如果与服务端发来的HASH一致，此时握手过程结束，之后所有的通信数据将由之前浏览器生成的随机密码并利用对称加密算法进行加密。

### 19 HTTP：

HTTP（HyperText Transfer Protocol），超文本传输协议，是一个基于TCP实现的应用层协议，HTTP由请求和响应构成，是一个标准的客户端服务器模型（B/S）。HTTP协议永远都是客户端发起请求，服务器回送响应，HTTP是一个无状态的协议。无状态是指客户机（Web浏览器）和服务器之间不需要建立持久的连接，这意味着当一个客户端向服务器端发出请求，然后服务器返回响应(response)，连接就被关闭了，在服务器端不保留连接的有关信息.HTTP遵循请求(Request)/应答(Response)模型。客户机（浏览器）向服务器发送请求，服务器处理请求并返回适当的应答。所有HTTP连接都被构造成一套请求和应答

（基本概念：  
HTTP，全称为 HyperText Transfer Protocol，即为超文本传输协议。是互联网应用最为广泛的一种网络协议  
所有的 www 文件都必须遵守这个标准。  
  
http特性：  
HTTP 是无连接无状态的  
HTTP 一般构建于 TCP/IP 协议之上，默认端口号是 80  
HTTP 可以分为两个部分，即请求和响应。

http请求：  
HTTP 定义了在与服务器交互的不同方式，最常用的方法有 4 种  
分别是 GET，POST，PUT， DELETE。URL 全称为资源描述符，可以这么认为：一个 URL 地址  
对应着一个网络上的资源，而 HTTP 中的 GET，POST，PUT，DELETE  
就对应着对这个资源的查询，修改，增添，删除4个操作。  
  
HTTP 请求由 3 个部分构成，分别是：状态行，请求头(Request Header)，请求正文。  
HTTP 响应由 3 个部分构成，分别是：状态行，响应头(Response Header)，响应正文。  
HTTP 响应中包含一个状态码，用来表示服务器对客户端响应的结果。）

20 列举几个HTTP请求头字段和响应头字段？

**请求头**

**Accept：**指定客户端能够接收的内容类型。

**Accept-Charset：**浏览器可以接受的字符编码集。

**Accept-Encoding：**指定浏览器可以支持的web服务器返回内容压缩编码类型。

**Accept-Language：**浏览器可接受的语言。

**Accept-Ranges：**可以请求网页实体的一个或者多个子范围字段。

**AuthorizationHTTP：**授权的授权证书。

**Cache-Control：**指定请求和响应遵循的缓存机制。

**Connection：**表示是否需要持久连接。（HTTP 1.1默认进行持久连接）

**CookieHTTP：**请求发送时，会把保存在该请求域名下的所有cookie值一起发送给web服务器。

**Content-Length：**请求的内容长度。

**Content-Type：**请求的与实体对应的MIME信息。

**Date：**请求发送的日期和时间。

**Expect：**请求的特定的服务器行为。

**From：**发出请求的用户的Email。

**Host：**指定请求的服务器的域名和端口号。

**If-Match：**只有请求内容与实体相匹配才有效。

**If-Modified-Since：**如果请求的部分在指定时间之后被修改则请求成功，未被修改则返回304代码。

**If-None-Match：**如果内容未改变返回304代码，参数为服务器先前发送的Etag，与服务器回应的Etag比较判断是否改变。

**If-Range：**如果实体未改变，服务器发送客户端丢失的部分，否则发送整个实体。

**If-Unmodified-Since：**只在实体在指定时间之后未被修改才请求成功。

**Max-Forwards：**限制信息通过代理和网关传送的时间。

**Pragma：**用来包含实现特定的指令。

**Proxy-Authorization：**连接到代理的授权证书。

**Range：**只请求实体的一部分，指定范围。

**Referer：**先前网页的地址，当前请求网页紧随其后,即来路。

**TE：**客户端愿意接受的传输编码，并通知服务器接受接受尾加头信息。

**Upgrade：**向服务器指定某种传输协议以便服务器进行转换（如果支持。

**User-AgentUser-Agent：**的内容包含发出请求的用户信息。

**Via：**通知中间网关或代理服务器地址，通信协议。

**Warning：**关于消息实体的警告信息

**响应头：**

**Accept-Ranges：**表明服务器是否支持指定范围请求及哪种类型的分段请求。

**Age：**从原始服务器到代理缓存形成的估算时间（以秒计，非负）。

**Allow：**对某网络资源的有效的请求行为，不允许则返回405。

**Cache-Control：**告诉所有的缓存机制是否可以缓存及哪种类型。

**Content-Encodingweb：**服务器支持的返回内容压缩编码类型。。

**Content-Language：**响应体的语言。

**Content-Length：**响应体的长度。

**Content-Location：**请求资源可替代的备用的另一地址。

**Content-MD5：**返回资源的MD5校验值。

**Content-Range：**在整个返回体中本部分的字节位置。

**Content-Type：**返回内容的MIME类型。

**Date：**原始服务器消息发出的时间。

**ETag：**请求变量的实体标签的当前值。

**Expires：**响应过期的日期和时间。

**Last-Modified：**请求资源的最后修改时间。

**Location：**用来重定向接收方到非请求URL的位置来完成请求或标识新的资源。

**Pragma：**包括实现特定的指令，它可应用到响应链上的任何接收方。

**Proxy-Authenticate：**它指出认证方案和可应用到代理的该URL上的参数。

**refresh：**应用于重定向或一个新的资源被创造，在5秒之后重定向（由网景提出，被大部分浏览器支持）

**Retry-After：**如果实体暂时不可取，通知客户端在指定时间之后再次尝试。

**Serverweb：**服务器软件名称。

**Set-Cookie：**设置Http Cookie。

**Trailer：**指出头域在分块传输编码的尾部存在。

**Transfer-Encoding：**文件传输编码。

**Vary：**告诉下游代理是使用缓存响应还是从原始服务器请求。

**Via：**告知代理客户端响应是通过哪里发送的。

**Warning：**警告实体可能存在的问题。

**WWW-Authenticate：**表明客户端请求实体应该使用的授权方案。

### 21 HTTP1.0和HTTP2.0区别：

1 HTTP/2采用二进制格式而非文本格式

2 HTTP/2是完全多路复用的，而非有序并阻塞的——只需一个连接即可实现并行:多路复用代替了HTTP1.x的序列和阻塞机制，所有的相同域名请求都通过同一个TCP连接并发完成，同域名下所有通信都在单个连接上完成，消除了因多个TCP连接而带来的延时和内存消耗。单个连接上可以并行交错的请求和响应，之间互不干扰

在 HTTP 2.0 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。

3 使用报头压缩，HTTP/2降低了开销

4 HTTP/2让服务器可以将响应主动“推送”到客户端缓存中

### 22 Post和Get的区别？

1. GET提交的数据放在URL中，POST则不会。这点意味着GET相对不安全
2. GET回退浏览器无害，POST会再次提交请求（GET方法回退后浏览器在缓存中拿结果，POST每次都会创建新资源）
3. GET提交的数据大小有限制（是因为浏览器对URL的长度有限制，GET本身没有限制），POST没有
4. GET可以被保存为书签，POST不可以。这一点也能感受到。
5. GET能被缓存，POST不能
6. GET只允许ASCII字符，POST没有限制
7. GET会保存再浏览器历史记录中，POST不会。这点也能感受到。

### 23 HTTP连接优化

1. 并行连接（能够同一时候和多台server建立HTTP连接）：

并行连接能够在带宽资源充足的情况下同一时候建立多个HTTP连接，加快页面的载入速度。

缺点：

并行连接在带宽资源不足的情况下会是连接竞争资源

1. 持久连接：

重用已对目标server打开的空暇持久连接，就能够避免缓慢的连接建立阶段。同一时候，已经打开的连接还能够避免慢启动的拥塞适应阶段。以便更快的进行传输数据。

1. 管道化连接：

能够同意在持久连接上可选的使用请求管道。相当于流水线的功能。在对应到达之前，能够将多条请求放入队列

### 24 HTTPS用了哪些加密算法

HTTPS一般使用的加密与HASH算法如下：

* 非对称加密算法：RSA，DSA/DSS
* 对称加密算法：AES，RC4，3DES
* HASH算法：MD5，SHA1，SHA256

### 25 了解Service Worker嘛？

是运行在浏览器背后的独立线程，可用于实现缓存功能，传输协议必须是HTTPS。使用Service-Worker实现缓存功能一般分为三个步骤：首先注册，然后监听install事件缓存需要的文件，最后拦截请求事件，如果缓存中已经有请求的数据就直接使用。

### 26 HTTP1.0和HTTP1.1的一些区别

1 **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例如Entity tag，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。

2 **长连接**，HTTP 1.1支持长连接（PersistentConnection）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。

### 27 head跟其他的区别

GET可以说是最常见的了，它本质就是发送一个请求来取得服务器上的某一资源。资源通过一组HTTP头和呈现据（如HTML文本，或者图片或者视频等）返回给客户端。GET请求中，永远不会包含呈现数据。

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。

### 28 put和post的区别

本质上来讲， PUT和POST极为相似，都是向服务器发送数据，但它们之间有一个重要区别，PUT通常指定了资源的存放位置，而POST则没有，POST的数据存放位置由服务器自己决定。