1. 请说一下get和post的区别

- 从用途来说，GET和POST是http请求方式。GET向特定的资源发出请求。POST向指定资源提交数据进行处理请求（例如提交表单或者上传文件）。
- 对于数据存放方式而言，要给GET加上request body，给POST带上url参数，是完全可以的。不同服务器的处理方式也是不同的，有些服务器会读出数据，有些服务器直接忽略，所以，虽然GET可以带request body，不能保证一定能被接收到。所以采用了url的方式。安全不足以作为他们的主要区别，只是因为数据存放位置不同而已。
- 对于数据量来说。数据量太大对浏览器和服务器都是很大负担。业界不成文的规定是，（大多数）浏览器通常都会限制url长度在2K个字节，而（大多数）服务器最多处理64K大小的url。超过的部分，恕不处理。所以就有了我们现在知道的数据长度受限的问题。
- 最本质的一点，GET和POST最大的区别主要是GET请求是幂等性的。副作用指对服务器上的资源做改变，搜索是无副作用的，数组的增删改是副作用的。副作用指对服务器上的资源做改变，搜索是无副作用的，数组的增删改是副作用的。

2. 说一说你知道的HTTP状态码

- 2XX 成功
  - 200 OK，表示从客户端发来的请求在服务器端被正确处理
  - 204 No content，表示请求成功，但响应报文不含实体的主体部分
  - 205 Reset Content，表示请求成功，但响应报文不含实体的主体部分，但是与 204 响应不同在于要求请求方重置内容
  - 206 Partial Content，进行范围请求
- 3XX 重定向
  - 301 moved permanently，永久性重定向，表示资源已被分配了新的 URL
  - 302 found，临时性重定向，表示资源临时被分配了新的 URL
  - 303 see other，表示资源存在着另一个 URL，应使用 GET 方法丁香获取资源
  - 304 not modified，表示服务器允许访问资源，但因发生请求未满足条件的情况
  - 307 temporary redirect，临时重定向，和302含义相同
- 4XX 客户端错误
  - 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
- 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

3. 你是否了解过websocket?

webSocket和http一样，同属于应用层协议。它最重要的用途是实现了客户端与服务端之间的全双工通信，当服务端数据变化时，可以第一时间通知到客户端。

除此之外，它与http协议不同的地方还有：

- http只能由客户端发起，而webSocket是双向的。
- webSocket传输的数据包相对于http而言很小，很适合移动端使用
- 没有同源限制，可以跨域共享资源

4. 请你手写实现一下ajax

实现流程：

1. 创建XMLHttpRequest对象，也就是创建一个异步调用对象。
2. 创建一个新的HTTP请求，并指定该HTTP请求的方法、URL及验证信息。
3. 设置响应HTTP请求状态变化的函数。
4. 发送HTTP请求 。
5. 获取异步调用返回的数据。
6. 使用JavaScript和DOM实现局部刷新。

代码实现：

```
    function ajax (options) {
        let url = options.url
        const method = options.method.toLocaleLowerCase() || 'get'
        const async = options.async != false // default is true
        const data = options.data
        const xhr = new XMLHttpRequest()

        if (options.timeout && options.timeout > 0) {
            xhr.timeout = options.timeout
        }

        return new Promise ( (resolve, reject) => {
            xhr.ontimeout = () => reject && reject('请求超时')
            xhr.onreadystatechange = () => {
                if (xhr.readyState == 4) {
                    if (xhr.status >= 200 && xhr.status < 300 || xhr.status
== 304) {
                        resolve && resolve(xhr.responseText)
                    } else {
                        reject && reject()
                    }
                }
            }
            xhr.onerror = err => reject && reject(err)

            let paramArr = []
            let encodeData
            if (data instanceof Object) {
                for (let key in data) {
                    // 参数拼接需要通过 encodeURIComponent 进行编码
                    paramArr.push( encodeURIComponent(key) + '=' +
encodeURIComponent(data[key]) )
                }
```

```
                encodeData = paramArr.join('&')
            }

            if (method === 'get') {
                // 检测 url 中是否已存在 ? 及其位置
                const index = url.indexOf('?')
                if (index === -1) url += '?'
                else if (index !== url.length -1) url += '&'
                // 拼接 url
                url += encodeData
            }

            xhr.open(method, url, async)
            if (method === 'get') xhr.send(null)
            else {
                // post 方式需要设置请求头
                xhr.setRequestHeader('Content-Type','application/x-www-
form-urlencoded;charset=UTF-8')
                xhr.send(encodeData)
            }
        } )
    }
```

5. http 2.0对于http 1.x有哪些优点?

- 多路复用：多路复用允许同时通过单一的 HTTP/2 连接发起多重的请求-响应消息。由于http 1.x的时代中，浏览器向同一域名下发送的http请求数量是受限的，当超出数量限制时，请求会被阻塞，大大降低了用户体验。而HTTP/2 的多路复用允许同时通过单一的 HTTP/2 连接发起多重的请求-响应消息。
- 二进制分帧：HTTP/2在应用层和传输层之间追加了一个二进制分帧层，最终使得多个数据流共用一个连接，更加高效的使用tcp连接。从而使得服务器的连接压力减轻，降低了内存的消耗，增大了网络的吞吐量。
- 首部压缩：HTTP/2引入了HPACK算法对头部进行压缩，大大减小了数据发送的字节数。

6. HTTP的缺点以及HTTPS

缺点：

- 通信使用明文不加密，内容可能被窃听。
- 不验证通信方身份，可能遭到伪装。
- 无法验证报文完整性，可能被篡改。

HTTPS就是HTTP加上SSL加密处理（一般是SSL安全通信线路）+认证+完整性保护

客户端浏览器在使用HTTPS方式与Web服务器通信时有以下几个步骤：

1. 客户使用https的URL访问Web服务器，要求与Web服务器建立SSL连接。
2. Web服务器收到客户端请求后，会生成一对公钥和私钥，并把公钥放在证书中发给客户端浏览器。
3. 客户端浏览器根据双方同意的SSL连接的安全等级，建立会话密钥，然后用公钥将会话密钥加密，并传送给服务器。

4. Web服务器用自己的私钥解密出会话密钥。
5. Web服务器利用会话密钥加密与客户端之间的通信。

7. 请说一下从输入 URL 到页面加载完成的过程

1. 首先做 DNS 查询，如果这一步做了智能 DNS 解析的话，会提供访问速度最快的 IP 地址回来。
2. 接下来是 TCP 握手，应用层会下发数据给传输层，这里 TCP 协议会指明两端的端口号，然后下发给网络层。网络层中的 IP 协议会确定 IP 地址，并且指示了数据传输中如何跳转路由器。然后包会再被封装到数据链路层的数据帧结构中，最后就是物理层面的传输了。
3. TCP 握手结束后会进行 TLS 握手，然后就开始正式的传输数据。
4. 数据在进入服务端之前，可能还会先经过负责负载均衡的服务器，它的作用就是将请求合理的分发到多台服务器上，这时假设服务端会响应一个 HTML 文件。
5. 首先浏览器会判断状态码是什么，如果是 200 那就继续解析，如果 400 或 500 的话就会报错，如果 300 的话会进行重定向，这里会有个重定向计数器，避免过多次的重定向，超过次数也会报错。
6. 浏览器开始解析文件，如果是 gzip 格式的话会先解压一下，然后通过文件的编码格式知道该如何去解码文件。
7. 文件解码成功后会正式开始渲染流程，先会根据 HTML 构建 DOM 树，有 CSS 的话会去构建 CSSOM 树。如果遇到 script 标签的话，会判断是否存在 async 或者 defer ，前者会并行进行下载并执行 JS，后者会先下载文件，然后等待 HTML 解析完成后顺序执行，如果以上都没有，就会阻塞住渲染流程直到 JS 执行完毕。遇到文件下载的会去下载文件，这里如果使用 HTTP 2.0 协议的话会极大的提高多图的下载效率。
8. 初始的 HTML 被完全加载和解析后会触发 DOMContentLoaded 事件。
9. CSSOM 树和 DOM 树构建完成后会开始生成 Render 树，这一步就是确定页面元素的布局、样式等等诸多方面的东西。
10. 在生成 Render 树的过程中，浏览器就开始调用 GPU 绘制，合成图层，将内容显示在屏幕上了。

8. 描述一下三次握手的过程，三次握手的作用?

TCP是一种面向连接的、可靠的、基于字节流的运输层（Transport layer）通信协议。是专门为了在不可靠的互联网络上提供一个可靠的端到端字节流而设计的。每一次TCP连接都需要三个阶段：连接建立、数据传送和连接释放。“三次握手”就发生在连接建立阶段。

1. 第一次握手：客户端发送一个TCP的SYN标志位置1的包，指明客户打算连接的服务器的端口，以及初始序号X, 保存在包头的序列号(Sequence Number)字段里。
2. 第二次握手： 服务器发回确认包(ACK)应答。即SYN标志位和ACK标志位均为1同时，将确认序号(Acknowledgement Number)设置为客户的ISN加1以.即X+1。
3. 第三次握手：客户端再次发送确认包(ACK) SYN标志位为0,ACK标志位为1.并且把服务器发来ACK的序号字段+1,放在确定字段中发送给对方.并且在数据段放写ISN的+1。

三次握手的目的:

- 使收发端的数据发送和接收同步
- 协调可以收发的数据量
- 建立虚连接

9. 请描述一下四次挥手的过程

所谓四次挥手（Four-Way Wavehand）即终止TCP连接，就是指断开一个TCP连接时，需要客户端和服务端总共发送4个包以确认连接的断开。由客户端或服务端任一方执行close来触发。

1. 第一次挥手：Client发送一个FIN，用来关闭Client到Server的数据传送，Client进入FIN_WAIT_1状态。
2. 第二次挥手：Server收到FIN后，发送一个ACK给Client，确认序号为收到序号+1（与SYN相同，一个FIN占用一个序号），Server进入CLOSE_WAIT状态。
3. 第三次挥手：Server发送一个FIN，用来关闭Server到Client的数据传送，Server进入LAST_ACK状态。
4. 第四次挥手：Client收到FIN后，Client进入TIME_WAIT状态，接着发送一个ACK给Server，确认序号为收到序号+1，Server进入CLOSED状态，完成四次挥手。

10. TCP UDP的区别？什么时候使用TCP，什么时候用UDP?

他们之间有四方面区别：

1. TCP是基于连接的，可靠性高;UDP基于无连接，可靠性较低。
2. 由于TCP是连接的通信，需要有三次握手、重新确认等连接过程，会有时延，实时性差；同时过程复杂，也使其易于被攻击；而UDP无连接，无建立连接的过程，因而实时性较强，也稍安全。
3. 在传输相同大小的数据时，TCP首部开销20字节;UDP的首部开销小，只有8个字节，TCP报头比UDP复杂，故实际包含的用户数据较少。TCP无丢包，而UDP有丢包，故TCP的开销大，UDP开销较小。
4. 每一条TCP连接只能是点到点的;UDP支持一对一，一对多，多对一和多对多的交互通信。

应用方面：

1. 由于TCP的实时性差，故对实时性要求高和高速传输的场合需用UDP。
2. TCP适用于传输大量数据，对可靠性要求高的环境；而在可靠性要求较低，追求效率时可用UDP。

11.cookie有什么作用？它的缺点是什么?

cookie可以解决http的无状态的问题，与服务器进行交互，作为http规范存在。它具有极高的简便性、可扩展性和可用性，也可以通过加密和SSL技术来提高其安全性。因此推荐使用cookie作为标识而不是身份验证的工具。

缺点有：

1. 大小和数目受限制。浏览器对一个域cookie的条目数有上限要求，且每个cookie的大小不得超过4kb。
2. 存在安全性问题，易被人拦截。
3. 需要指定域，不可以跨域。
4. 浪费带宽，因为我每次请求一个新的页面，cookie都会被自动发送过去。
5. 有的移动端浏览器不支持cookie或浏览器禁用cookie。
6. 有些状态不可能保存在客户端。例如，为了防止重复提交表单，我们需要在服务器端保存一个计数器。如果我们把这个计数器保存在客户端，那么它起不到任何作用。

12. Cookie的工作原理?

- 用户首次访问Web站点时，Web服务器对用户一无所知。
- Web服务器通过Set-Cookie首部将cookie存放到浏览器中的cookie数据库中。cookie中包含了N个键值对，例如Cookie: id=“1234”。cookie中可以包含任意信息，但它们通常都只包含一个服务器为了进行跟踪而产生的独特的识别码。
- 将来用户再次访问同一站点时，浏览器会从cookie数据库中挑中那个服务器设置的cookie，并在cookie请求首部中（Cookie: id=“1234”）将其传回给服务器。

- 服务器可以通过id="1234"这个键值对来查找服务器为其访问积累的信息（购物历史、地址信息等）。