Websocket面试

**那么HTTP请求响应的工作流程是什么呢？**

1、客户端连接到web服务器，与web服务器的HTTP端口（默认是80）建立一个TCP套接字连接  
2、发送HTTP请求 通过TCP套接字，客户端向服务器发送一个文本的请求报文，一个请求报文由请求行，请求头部，空行和请求数据四个部分构成  
3、服务端接受请求并返回HTTP响应 web服务器解析请求，定位请求资源。服务器将资源复本写到TCP套接字，由客户端读取。一个响应由状态行、响应头部、空行和响应数据4部分组成。  
4、释放连接TCP连接 Web服务器主动关闭TCP套接字，释放TCP连接；客户端被动关闭TCP套接字，释放TCP连接。  
5、客户端浏览器解析HTML内容 客户端浏览器首先解析状态行，查看表明请求是否成功的状态代码。然后解析每一个响应头，响应头告知以下为若干字节的HTML文档和文档的字符集。客户端浏览器读取响应数据HTML，根据HTML的语法对其进行格式化，并在浏览器窗口中显示。

# HTTPS

HTTP协议传输的数据都是没有加密的，也就是明文的，因此使用HTTP协议传输隐私信息非常的不安全。\*\*为了让这些隐私数据能够加密传输，便设计了SSL协议对HTTP传输的数据进行加密，从而诞生了HTTPS。

## 加密过程如下:

1、浏览器将自己支持的一套加密规则发送给网站。  
2、网站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、获得网站证书之后浏览器要做以下工作：  
(1) 验证证书的合法性（颁发证书的机构是否合法，证书中包含的网站地址是否与正在访问的地址一致等），如果证书受信任，则浏览器栏里面会显示一个小锁头，否则会给出证书不受信的提示。  
(2) 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。  
(3) 使用约定好的HASH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网站。  
4、网站接收浏览器发来的数据之后要做以下的操作：  
(1) 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。  
(2) 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。

# websocket

HTTP获取数据的时候，需要不断的问服务端是否有我要的数据啊，如果有数据就返回数据，没有就过一段时间再次询问服务端是否有我需要的数据。那websocket呢，它只建立一次连接，那么这个连接就不会断，服务端如果有数据的话，会自动返回数据给客户端，还有一个问题，在HTTP中我们提到，HTTP是无状态的，意思它健忘，上一次的请求和这次的请求都没什么联系，我们需要引用cookie才能解决。那么在websockt中，因为是一次长连接，那么这就不用一次次加入cookie，是不是方便很多了。 下面的代码就是websocket在前端代码中的应用

|  |  |
| --- | --- |
|  | <!DOCTYPE HTML> |
|  | <html> |
|  | <head> |
|  | <meta charset="utf-8" /> |
|  | <title>websocket</title> |
|  | <style type="text/css"> |
|  |  |
|  | </style> |
|  | </head> |
|  | <body> |
|  | <div id="connected"></div> |
|  |  |
|  | <script type="text/javascript"> |
|  |  |
|  | var onOpen = function() { |
|  | console.log("Socket opened."); |
|  | socket.send("Hi, Server!"); |
|  | }, |
|  | onClose = function() { |
|  | console.log("Socket closed."); |
|  | }, |
|  |  |
|  |  |
|  | onMessage = function(data) { |
|  | console.log("We get signal:"); |
|  | console.log(data); |
|  | }, |
|  |  |
|  |  |
|  | onError = function() { |
|  | console.log("We got an error."); |
|  | }, |
|  |  |
|  |  |
|  | socket = new WebSocket("ws://127.0.0.1:8080/"); |
|  |  |
|  | socket.onopen = onOpen; |
|  | socket.onclose = onClose; |
|  | socket.onerror = onError; |
|  | socket.onmessage = onMessage; |
|  | </script> |
|  | </body> |
|  | </html> |

【nodejs】

var WebSocketServer = require('ws').Server;

var wss = new WebSocketServer({ port: 8080 });

wss.on('connection', function connection(ws) {

ws.on('message', function incoming(message) {

console.log('received: %s', message);

});

ws.send('something');

});

【复杂使用】<https://github.com/niuyueyang/nodeWebsocket>