WebSocket解读，资料获取WX：18040505058

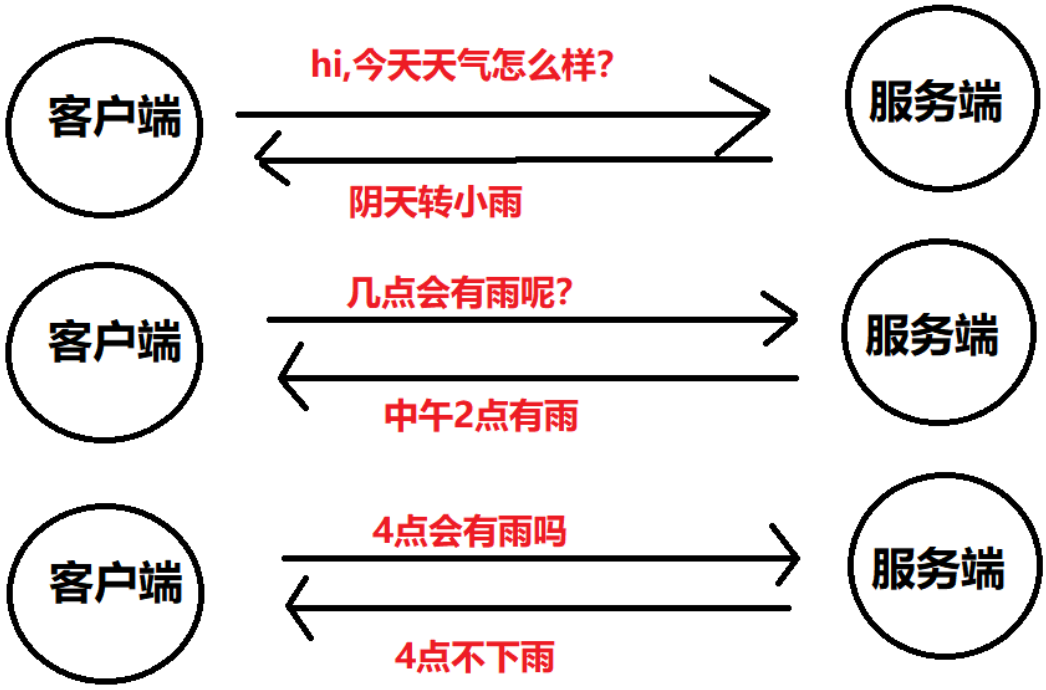
WebSocket解读

1、WebSocket是什么?

WebSocket 是一种网络通信协议
WebSocket 是 HTML5 开始提供的一种在单个 TCP 连接上进行全双工通讯的协议。

2、为什么需要 WebSocket ?

初次接触 WebSocket 的人，都会问同样的问题：我们已经有了 HTTP 协议，为什么还需要另一个协议？它能带来什么好处?
答案很简单，因为 HTTP 协议有一个缺陷：通信只能由客户端发起。
举例来说，我们想了解今天的天气，只能是客户端向服务器发出请求，服务器返回查询结果。HTTP 协议做不到服务器主动向客户端推送信息。



传统的HTTP协议是无状态的，每次请求（request）都要由客户端（如 浏览器）主动发起，服务端进行处理后返回response结果，而服务端很难主动向客户端发送数据；这种客户端是主动方，服务端是被动方的传统Web模式 对于信息变化不频繁的Web应用来说造成的麻烦较小，而对于涉及实时信息的Web应用却带来了很大的不便，如带有即时通信、实时数据、订阅推送等功能的应 用。

HTTP工作原理

HTTP协议定义Web客户端如何从Web服务器请求Web页面，以及服务器如何把Web页面传送给客户端。HTTP协议采用了请求/响应模型。客户端向服务器发送一个请求报文，请求报文包含请求的方法、URL、协议版本、请求头部和请求数据。服务器以一个状态行作为响应，响应的内容包括协议的版本、成功或者错误代码、服务器信息、响应头部和响应数据。

以下是 HTTP 请求/响应的步骤：

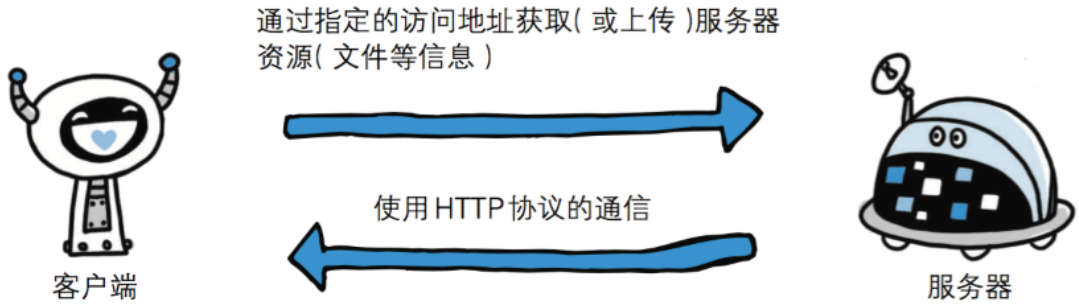
1. 客户端连接到Web服务器
   一个HTTP客户端，通常是浏览器，与Web服务器的HTTP端口（默认为80）建立一个TCP套接字连

接。例如，http://www.abc.com

2. 发送HTTP请求
   通过TCP套接字，客户端向Web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4部分组成。
3. 服务器接受请求并返回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
   若connection 模式为close，则服务器主动关闭TCP连接，客户端被动关闭连接，释放TCP连接;若connection 模式为keepalive，则该连接会保持一段时间，在该时间内可以继续接收请求;
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。

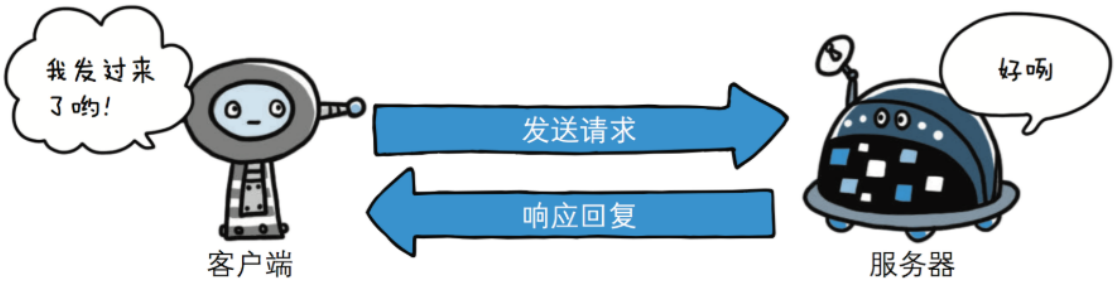
例如：在浏览器地址栏键入URL，按下回车之后会经历以下流程：

1. 浏览器向 DNS 服务器请求解析该 URL 中的域名所对应的 IP 地址;
2. 解析出 IP 地址后，根据该 IP 地址和默认端口 80，和服务器建立TCP连接;
3. 浏览器发出读取文件(URL 中域名后面部分对应的文件)的HTTP 请求，该请求报文作为 TCP 三次握手的第三个报文的数据发送给服务器;
4. 服务器对浏览器请求作出响应，并把对应的 html 文本发送给浏览器;
5. 释放 TCP连接;
6. 浏览器将该 html 文本并显示内容;



基于 请求-响应 的模式

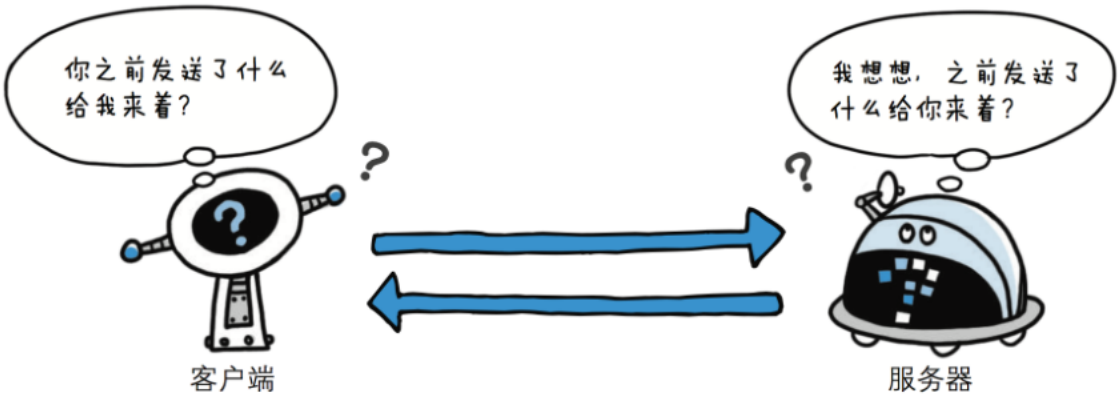
HTTP协议规定,请求从客户端发出,最后服务器端响应该请求并 返回。换句话说,肯定是先从客户端开始建立通信的,服务器端在没有 接收到请求之前不会发送响应



图：请求必定由客户端发出，而服务器端回复响应

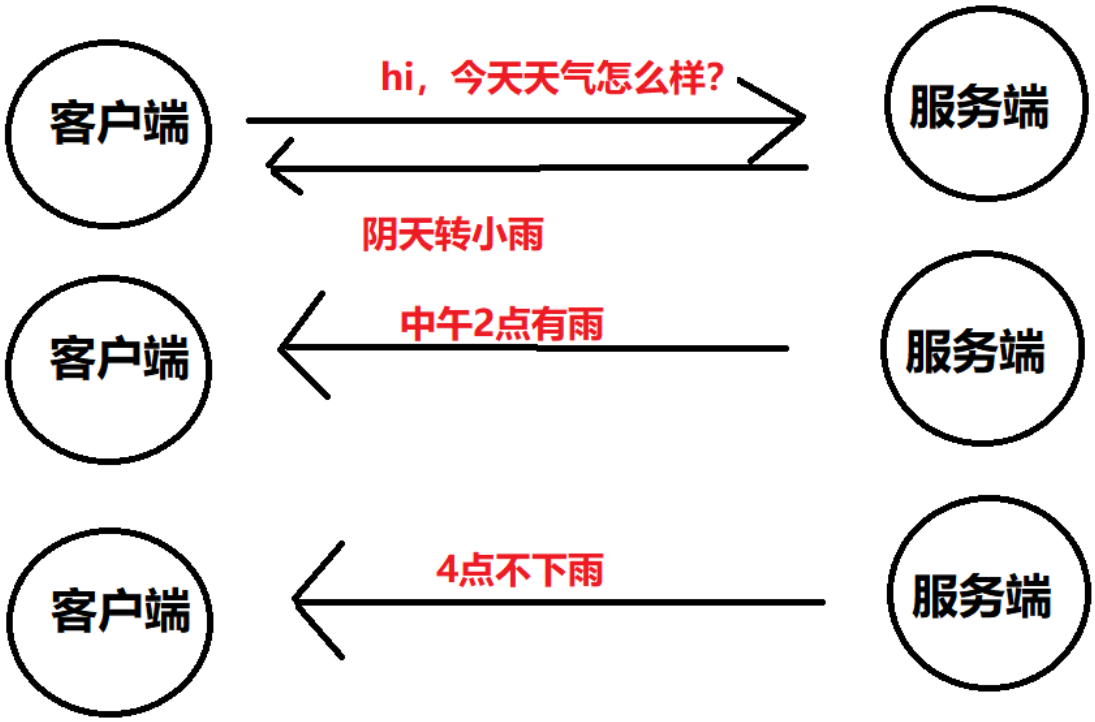
无状态保存

HTTP是一种不保存状态,即无状态(stateless)协议。HTTP协议 自身不对请求和响应之间的通信状态进行保存。也就是说在HTTP这个 级别,协议对于发送过的请求或响应都不做持久化处理。



图：HTTP 协议自身不具备保存之前发送过的请求或响应的功能

WebSocket协议



3、WebSocket的优势

相对于传统HTTP每次请求-应答都需要客户端与服务端建立连接的模式，WebSocket是类似Socket的TCP长连接通讯模式。一旦WebSocket连接建立后，后续数据都以帧序列的形式传输。在客户端断开WebSocket连接或Server端中断连接前，不需要客户端和服务端重新发起连接请求。在海量并发及客户端与服务器交互负载流量大的情况下，极大的节省了网络带宽资源的消耗，有明显的性能优势，且客户端发送和接受消息是在同一个持久连接上发起，实时性优势明显。

4、实现方式

WebSocket需要像TCP一样，先建立连接，需要客户端和服务端进行握手连接，连接成功后才能相互通信。

连接成功建立的回调方法

```
ws.onopen = function () {
    alert("WebSocket连接成功")
}
```

连接发生错误的回调方法

```
ws.onerror = function () {
    alert("WebSocket连接发生错误")
}
```

接收到消息的回调方法

```
ws.onmessage = function (data) {
    alert(data)
}
```

连接关闭的回调方法

```
ws.onclose = function () {
    alert("WebSocket连接关闭")
}
```

5、接口文档

服务端访问地址：ws://127.0.0.1:8090

接收消息

资源URL：ws://localhost:8090		
协议：ws		
返回参数：{"type":"类型","nickname":"昵称","message":"消息内容"}		
notification	通知	
message	普通消息	
nick_update	更新昵称	