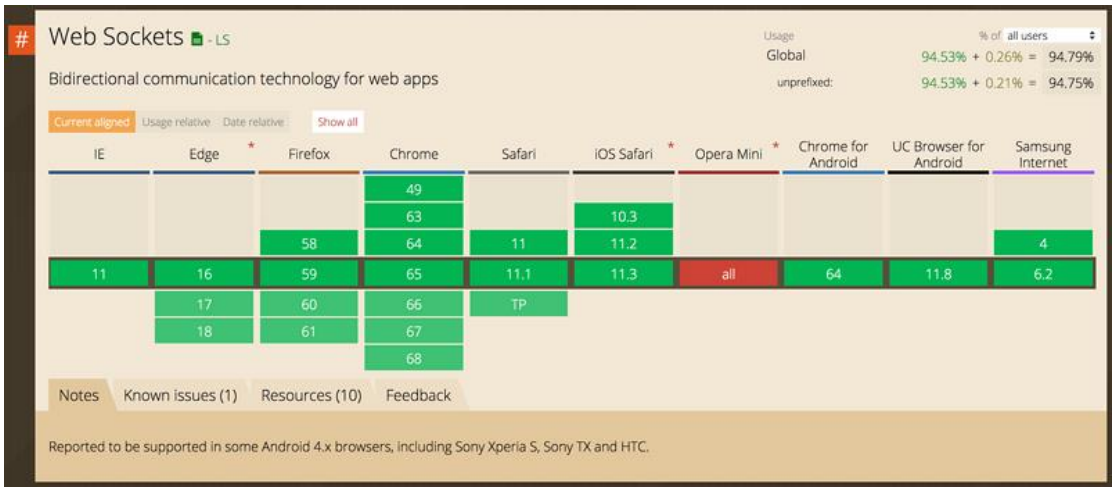
1 什么是 SockJS /Socket.IO

有一些浏览器中缺少对 WebSocket 的支持，而 SockJS 是一个浏览器的 JavaScript 库，它提供了一个类似于网络的对象，SockJS 提供了一个连贯的，跨浏览器的 JavaScriptAPI，它在浏览器和 Web 服务器之间创建了一个低延迟、全双工、跨域通信通道。SockJS 的一大好处在于提供了浏览器兼容性。即优先使用原生 WebSocket，如果浏览器不支持 WebSocket，会自动降为轮询的方式。如果你使用 Java 做服务端，同时又恰好使用 Spring Framework 作为框架，那么推荐使用 SockJS。



如果你使用 Node.js 做服务端，那么毫无疑问你该选择 Socket.IO，它本省就是从 Node.js 开始的，当然服务端也提供了 engine.io-server-java 实现。甚至你可以使用 netty-socketio。

2 Stomp 协议

2.1 简单什么是 Stomp？

STOMP 中文为“面向消息的简单文本协议”，STOMP 提供了能够协作的报文格式，以至于 STOMP 客户端可以与任何 STOMP 消息代理（Brokers）进行通信，从而为多语言，多平台和 Brokers 集群提供简单且普遍的消息协作。STOMP 协议可以建立在 WebSocket 之上，也可以建立在其他应用层协议之上。通过 Websocket 建立 STOMP 连接，也就是说在 Websocket 连接的基础上再建立 STOMP 连接。

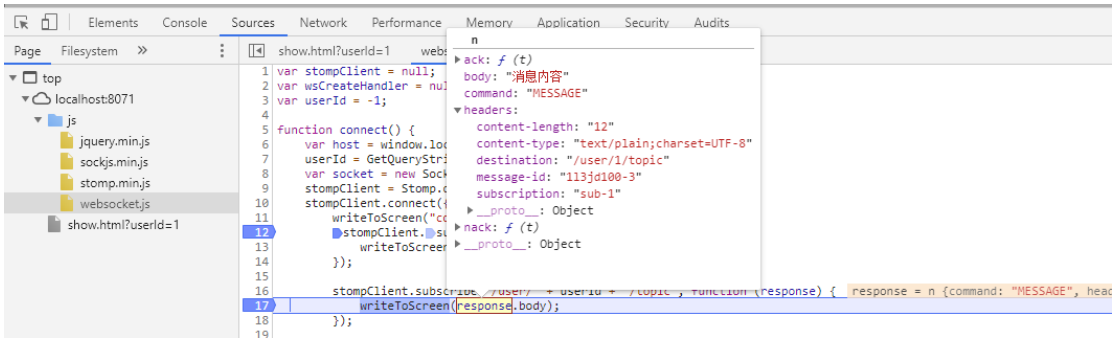
- 业界已经有很多优秀的 STOMP 的服务器/客户端的开源实现

STOMP 服务器：ActiveMQ、RabbitMQ、StompServer、…
STOMP 客户端库：stomp.js(javascript)、stomp.py(python)、Gozirra(java)、…

- Stomp 的特点是客户端的实现很容易，服务端相当于消息队列的 broker 或者是 server，一般不需要我们去实现，所以重点关注一下客户端如何使用

CONNECT
SEND
SUBSCRIBE
UNSUBSCRIBE
BEGIN
COMMIT
ABORT
ACK
NACK
DISCONNECT

- STOMP Frames



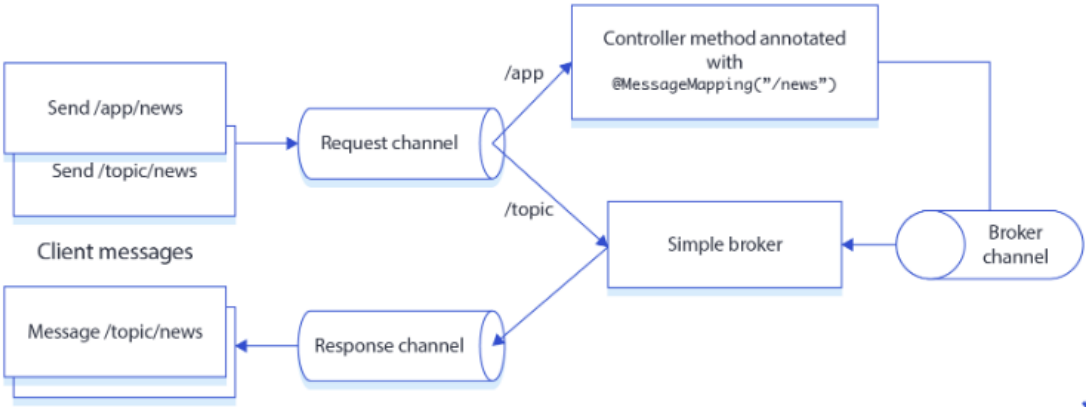
有 header 头、body 消息体

2.2 为什么使用 Stomp？

WebSocket 协议是一种相当低级的协议。它定义了如何将字节流转换为帧。帧可以包含文本或二进制消息。由于消息本身不提供有关如何路由或处理它的任何其他信息，因此很难在不编写其他代码的情况下实现更复杂的应用程序。幸运的是，WebSocket 规范允许在更高的应用程序级别上使用子协议。

2.3 如何使用？

当然这里的使用是基于 websocket 消息推送


Send /app/news
Send /topic/news
Client messages
Request channel
/app
/topic
Controller method annotated with @MessageMapping("/news")
Simple broker
Broker channel
Message /topic/news
Response channel