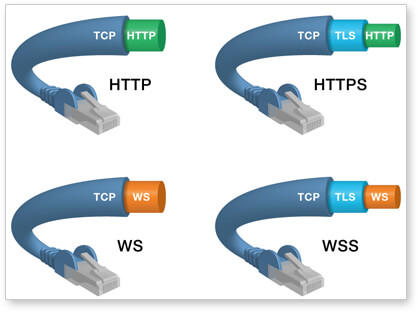
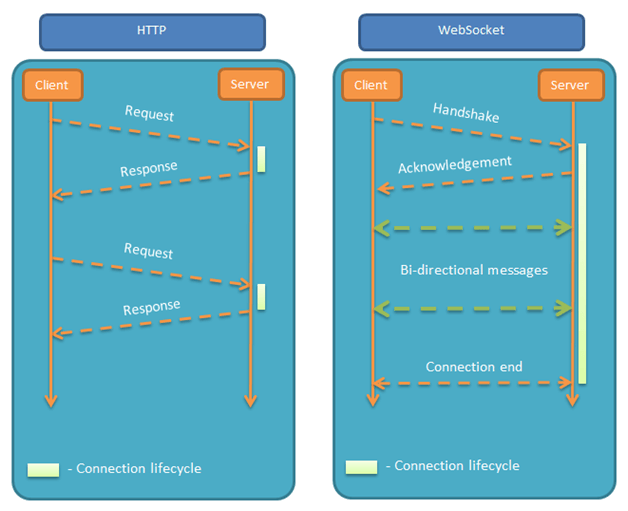
Problems with HTTP：

1. Client request> Server ，Server response> Client单向请求
2. Stateless, half dulpex



WebSocket 协议本质上是一个基于 TCP 的协议。为了建立一个 WebSocket 连接，客户端浏览器首先要向服务器发起一个 HTTP 请求，这个请求和通常的 HTTP 请求不同，包含了一些附加头信息，其中附加头信息"Upgrade: WebSocket"表明这是一个申请协议升级的 HTTP 请求，服务器端解析这些附加的头信息然后产生应答信息返回给客户端，客户端和服务器端的 WebSocket 连接就建立起来了，双方就可以通过这个连接通道自由的传递信息，并且这个连接会持续存在直到客户端或者服务器端的某一方主动的关闭连接。

Socket.IO

建立连接：Server监听Client请求

通讯：Client <---> Server

断开连接：Client发起Server响应

Socket.on保留事件: connect, disconnect, message

**Server**

var app = require('express')();

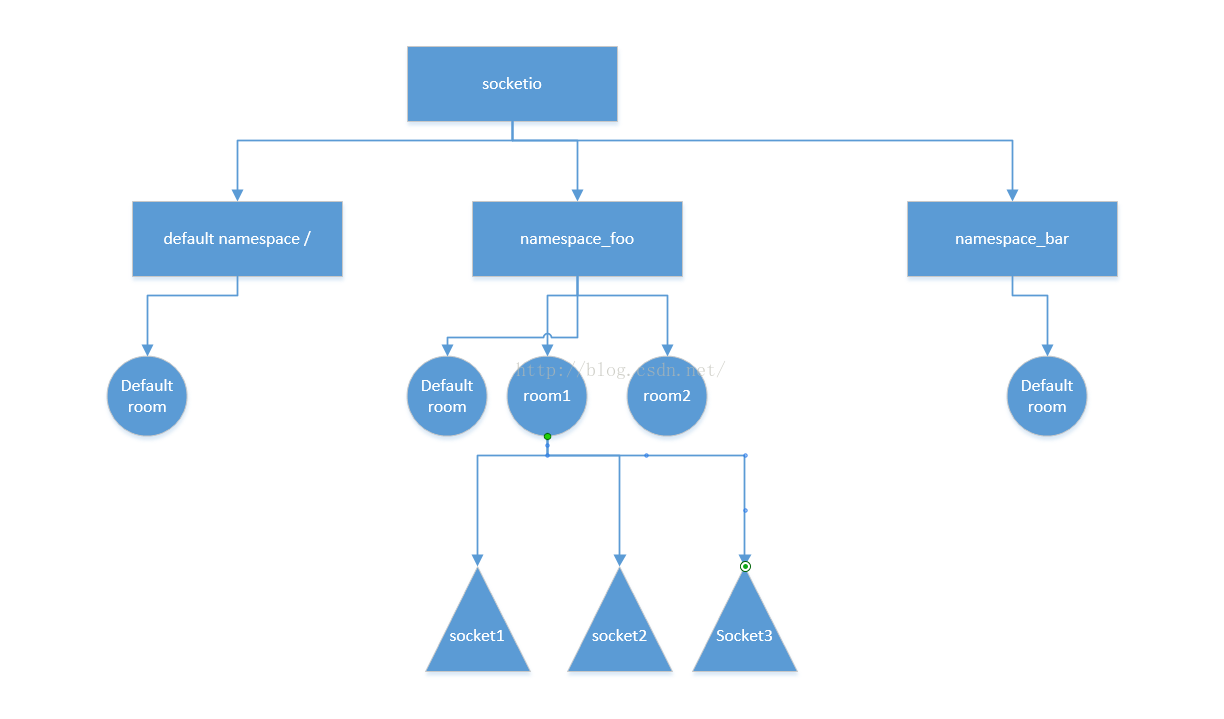
var my\_server = require('http').Server(app);

var server\_io = require('socket.io')(my\_server);

**Client**

var client\_socket = io.connect('/private');

Mechanism:

IO/Namespace/Room/Socket [emit---on]

Acknowledgements: Sometimes, you might want to get a callback when the client confirmed the message reception.连接确认ACK

The main idea behind Socket.IO is that you can send and receive any events you want, with any data you want. Any objects that can be encoded as JSON will do, and binary data is supported too.数据传输

**Multiplexing support 多路复用**

Namespaces, which will act as separate communication channels but will share the same underlying connection. Represents a pool of sockets connected under a given scope identified by a pathname.

**Room support 群**

Within each Namespace, you can define arbitrary channels, called Rooms, that sockets can join and leave. You can then broadcast to any given room, reaching every socket that has joined it.