# 作用

http的缺陷：只能有客户端发起。不能主动推送消息。（轮询十分消耗资源，http始终打开）

websocket提供一条持续打开的数据通道。

# 特点

1. 建立与TCP协议之上，服务端的实现比较容易
2. 与http协议有良好的兼容性，默认端口也是80和443，并且握手阶段采用http协议，握手不容易屏蔽，能通过各种http代理服务器。
3. 数据格式轻量，性能开销小，通信高效
4. 可以发送文本，也可以发送二进制数据。
5. 没有同源限制，客户端可以与任意服务器通信
6. 协议标识符是ws(加密则为wss)

# websocket握手

请求头：

Upgrade : Upgrade

通知服务器，如果可以的话，就升级到websoket协议。

origin

请求的域

sec-websocket-key:

base64 16字节随机编码，用于握手协议的秘钥。

响应头：

响应的状态码是101

connection： Upgrade

通知浏览器，需要改变协议

sec-websoket-accept:

服务器取sec-websocket-key，在添加‘258EAFA5-E914-47DA-95CA-C5AB0DC85B11’字符串，取SHA-1。浏览器对这个值进行验证，确保是目标服务器回应了websoket请求。

sec-websocket-location:

服务器位置

# API

构造函数 WebSocket

var ws = new WebSocket(url)

类属性

Websocket.

CONNECTING :正在连接 0

OPEN: 连接成功 1

CLOSING: 连接正在关闭 2

CLOSED: 连接已经关闭，或者打开失败 3

实例属性

ws.readyState: ws当前的状态 0-3

ws.onopen = fn 成功后的回调

指定多个回调函数

ws.addEventListener(‘open’, fn)

ws.onclose()

ws.onmessage = fn: 接收到服务器数据后的回调

var fn = function fn(event) {

if(event.data instanceof ArrayBuffer){

//二进制文件 blob或者 Arraybuffer

}

}

ws.onerror = fn 失败回调

指定多个回调函数

ws.addEventListener(‘message’, fn)

显示指定收到的二进制数据类型

ws.binaryType = “Blob” || “arraybuffer”

向服务器发送数据

ws.send()

多少字节的二进制数据没有发送出去

ws.bufferedAmout