


02

OPEN ORIENTED

凹凸实验室

谈谈 WebSocket 那些事

WebSocket 介绍

Web Sockets  - LS

Global93.91% + 0.36% = 94.26%

unprefixed:93.91% + 0.31% = 94.22%

Bidirectional communication technology for web apps

Current alignedUsage relativeDate relativeShow all

IE	Edge*	Firefox	Chrome	Safari	iOS Safari*	Opera Mini*	Chrome for Android	UC Browser for Android	Samsung Internet
			49		9.3				
			61		10.2				
	15		62	10.1	10.3				4
11	16	57	63	11	11.2	all	62	11.4	6.2
	17	58	64	TP					
		59	65						
		60	66						

Notes

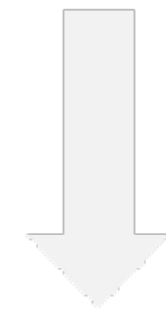
Known issues (1)

Resources (10)

Feedback

Reported to be supported in some Android 4.x browsers, including Sony Xperia S, Sony TX and HTC.

实时性



Long Pull (长轮询)

客户端发起请求之后，服务端没有消息就一直不返回.直到有消息，服务端返回response，客户端再次建立链接，周而复始。

Ajax 轮询

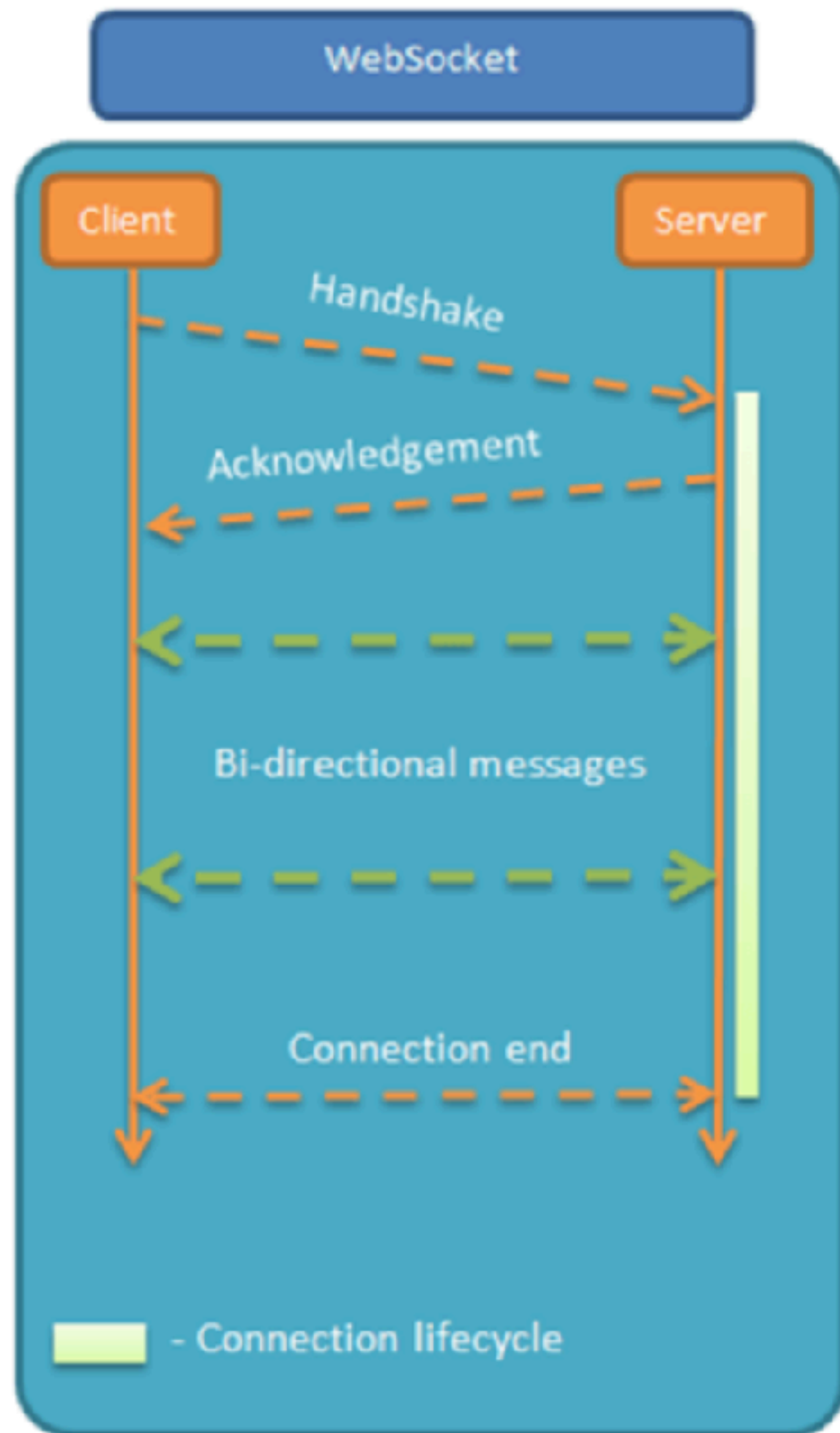
每隔一会发起一个请求，轮询服务端是否有新的信息

WebSocket

双向通信

WebSocket 原理

02



1. 客户端向服务端发起连接请求

```
Connection:Upgrade      #通知服务器协议升级
Upgrade:websocket       #协议升级为websocket协议
Sec-WebSocket-Key:K8==  #传输给服务器的key
Sec-WebSocket-Version:13 #websocket协议版本13
```

2. 服务端返回握手应答

```
Connection:Upgrade #协议升级成功
Sec-WebSocket-Accept:Gnog= #服务端处理之后的key
Sec-WebSocket-Version:13 #协议版本
Upgrade:websocket   #协议升级为websocket
```


WebSocket 原理

```
var ws = new WebSocket('ws://localhost:8080'); //协议使用ws://开头，安全的使用 wss://开头/

ws.onopen = function () {
  ws.send('连接成功!');
}

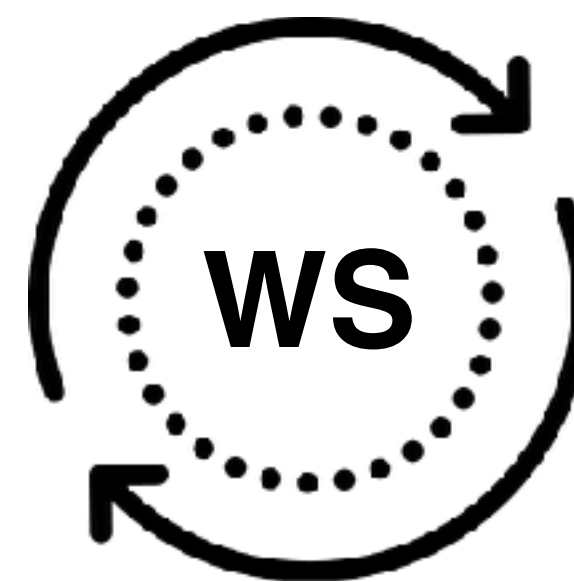
ws.onclose = function () {
  ws.send('关闭连接!'); // 发送数据到服务端 [传送的数据可能是文本，可能是二进制 (blob) ]
}

ws.onmessage = function () {
  console.log('我接收到消息了'); // 发送数据到服务端
}
```

实战 WebSocket



socket.io



WS

1. 服务端

```
var app = require('koa')();
var server = require('http').createServer(app.callback());
var io = require('socket.io')(server);
io.on('connection', function(){
  // listen to the event
  socket.on('reply', function(){ /* */ });
  // emit an event to the socket
  socket.emit('request', /* */);
});
server.listen(3000);
```


2. 客户端

```
<script src="http://localhost:3000/socket.io/socket.io.js"></script>
var socket = new io('ws://127.0.0.1:3000/')
socket.on('changeUserAuth', function (res) {
  fn('用户1接收到信息了')
})
socket.emit('changeUserStatus', {data}, function (res) {
  console.log(res)
})
```

有多个活动的情况，如何通知？

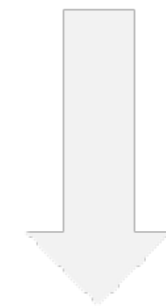
广播



```
socket.broadcast.emit('an event', { data });
```

所有连接的用户都会收到通知

Rooms



```
socket.join(actId); //加入 actId 房间  
io.to(actId).emit('msg', {data}); //发送消息给这个房间的所有人
```

坑：服务端使用 socket.io ，小程序无法连接

socket.io 底层是 engine.io

engine.io： 使用了 WebSocket 和 XMLHttpRequest（或JSONP） 封装了一套自己的 Socket 协议（暂时叫 EIO Socket）， 在低版本浏览器里面使用长轮询替代 Web

► WebSocket connection to 'ws://localhost:3000/' failed: Connection closed before receiving a handshake response

小程序采用的是标准的websocket 协议

1. 服务端

```
var app = require('koa')();
var server = require('http').createServer(app.callback());
const wss = new WebSocket.Server({ server });
wss.on('connection', function(ws){
  // listen to the event
  ws.on('message', function(){ /* */ });
  // emit an event to the socket
  ws.send('request', /* */);
});
server.listen(3000);
```

2. 客户端

```
var ws = new WebSocket('ws://localhost:3000/')
ws.onopen= function open() {
  ws.send('我连接上了');
  ws.onmessage = function (event) {
    console.log(event);
  }
};
```

Q & A

THANKS
FOR YOUR WATCHING

