HTML5 WebSocket Introduction

Gene

Agenda

1. Client/server communication
2. Intro to WebSocket
3. Demo

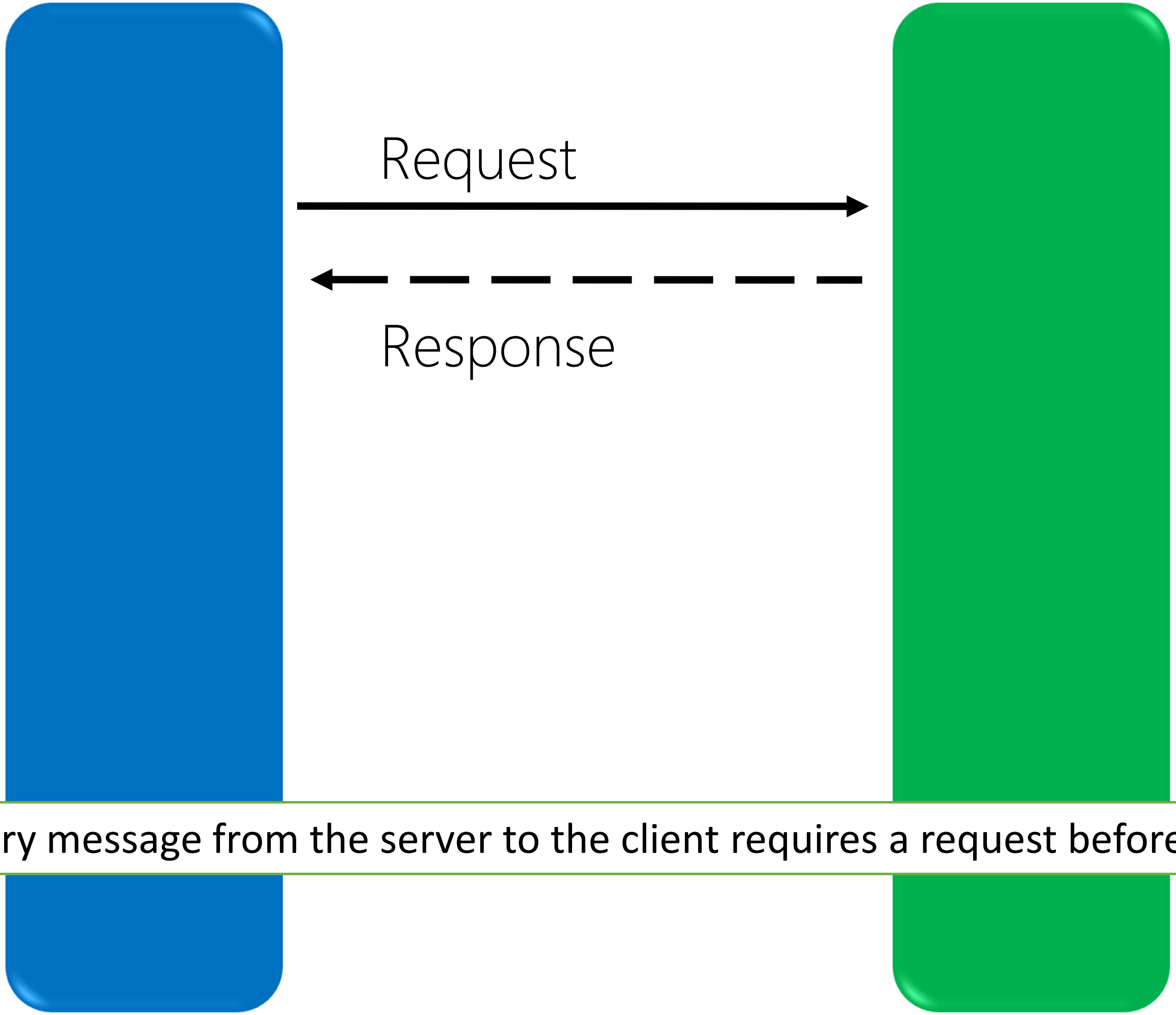
Client/Server Communication

Basic Web Applications

Basic Web Application

Browser

Server

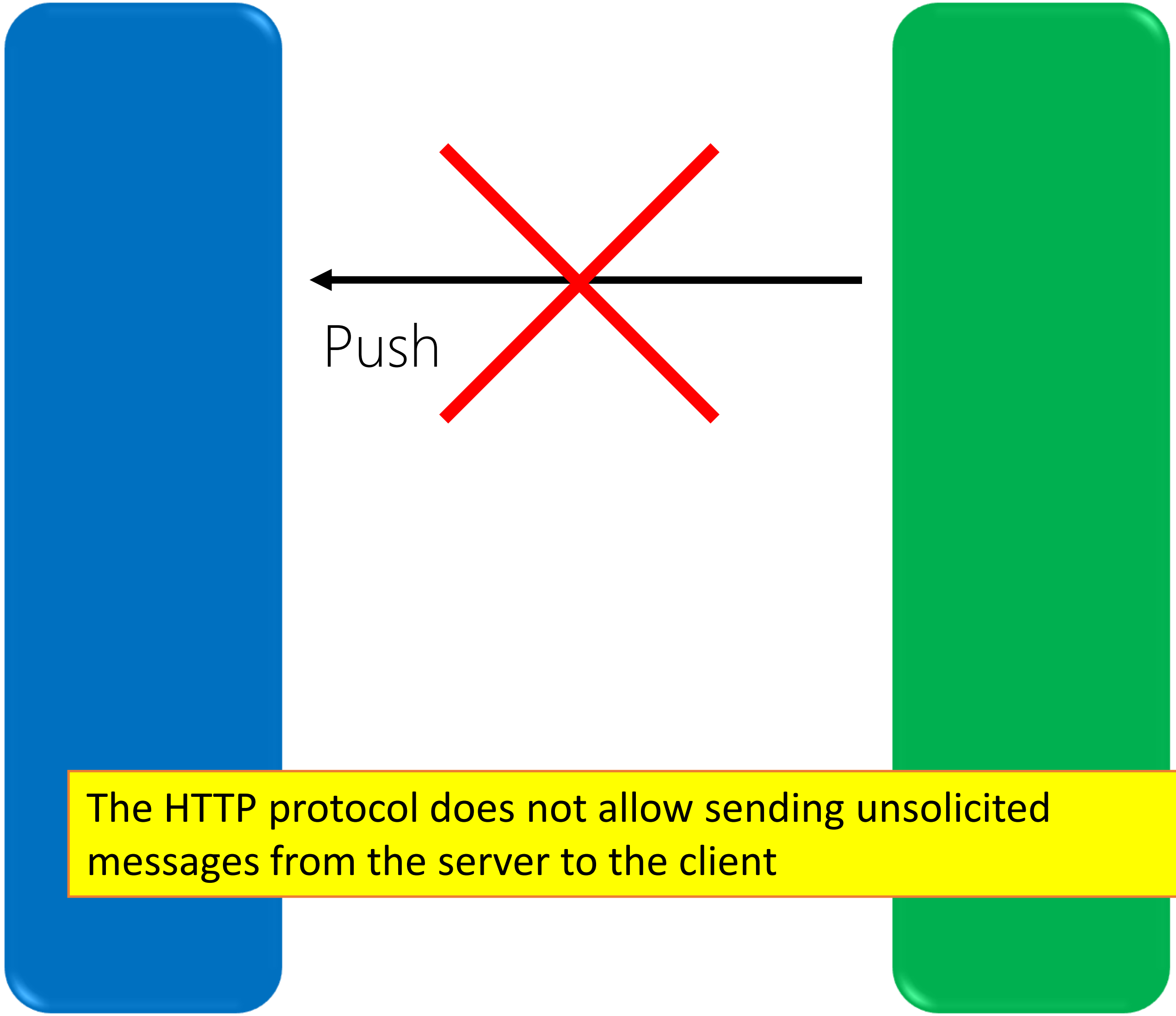


Every message from the server to the client requires a request beforehand

Basic Web Application

Browser

Server



The HTTP protocol does not allow sending unsolicited messages from the server to the client

Basic Web Application

Pros

- Simple
- Client gets what it wants(and when)
- Minimal interaction between the server and the client

Cons

- Server cannot initiate the communication, only the client
- new client request == new page load

How do we add more interaction between the client and the server?

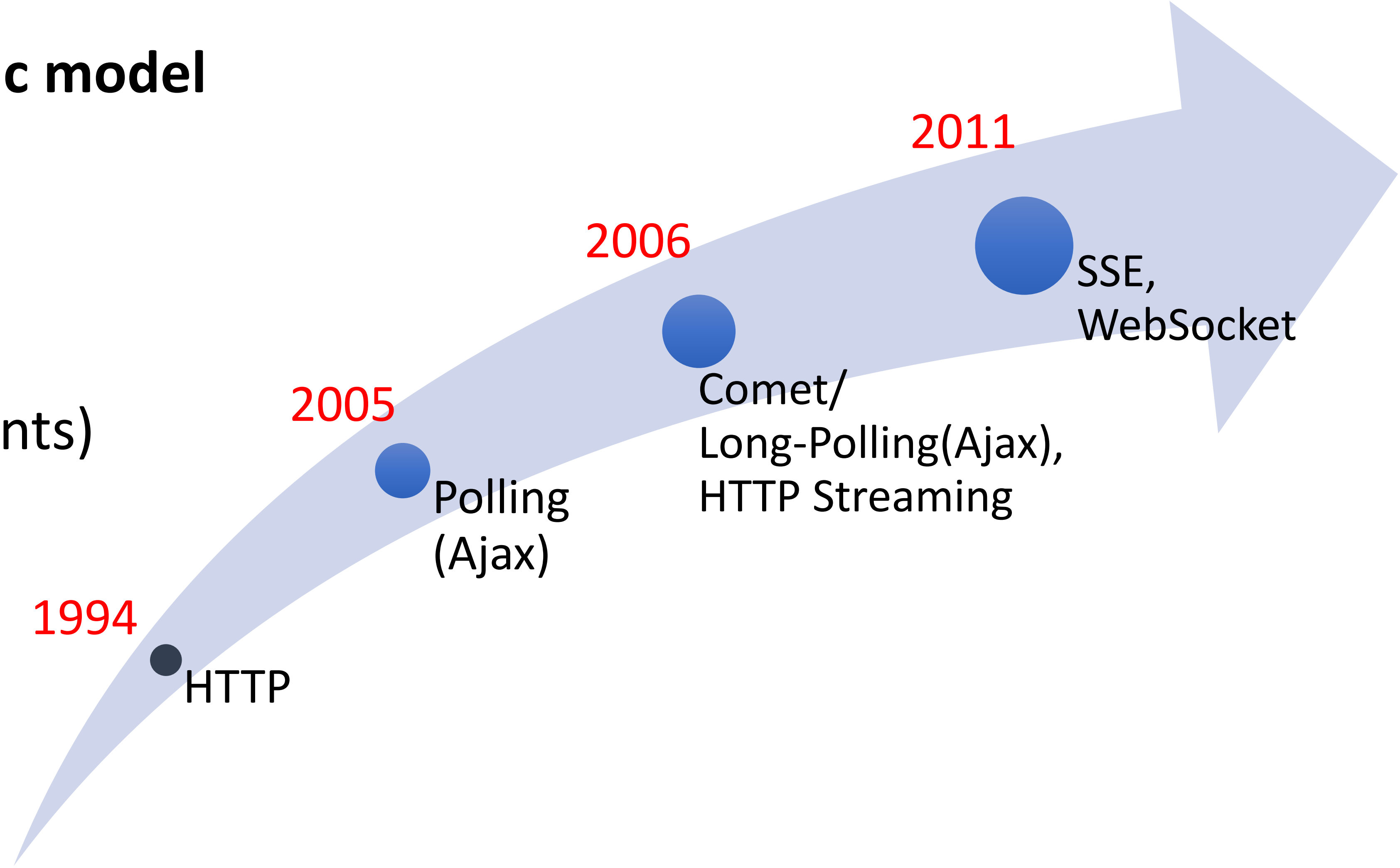
The Evolution of the Push

Variations on the basic model

- Polling (AJAX)
- Long-Polling (AJAX)

HTML5 new models

- SSE (Server-Sent Events)
- WebSocket



REF : http://www.evolutionoftheweb.com/?hl=zh-tw

WEB APPLICATION

=

HTML

圖片
按鈕
影片
表格
區塊
....
結構

＋

JavaScript

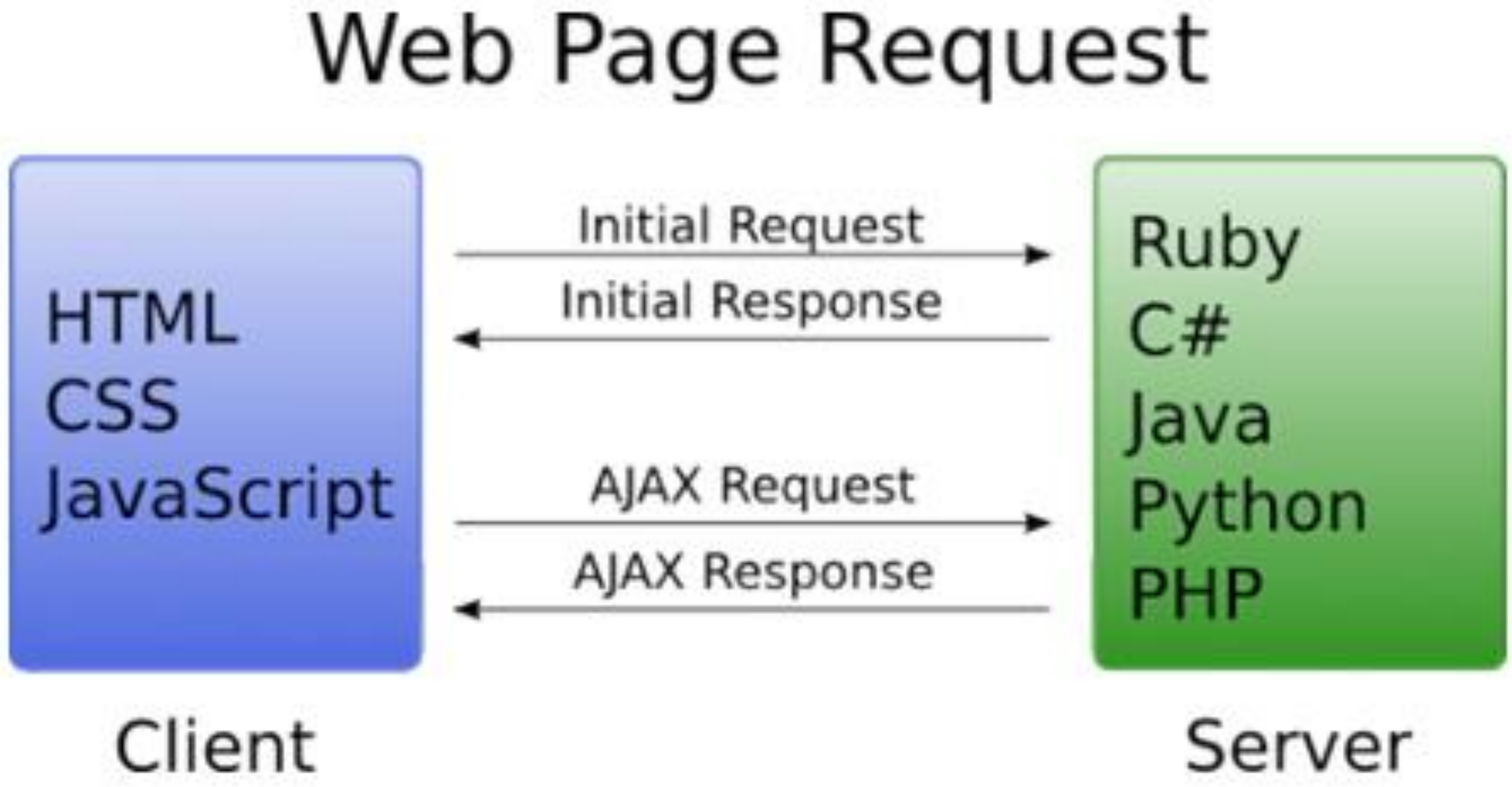
互動
元素更新
資料驗證
資料運算
特效
....
行為

＋

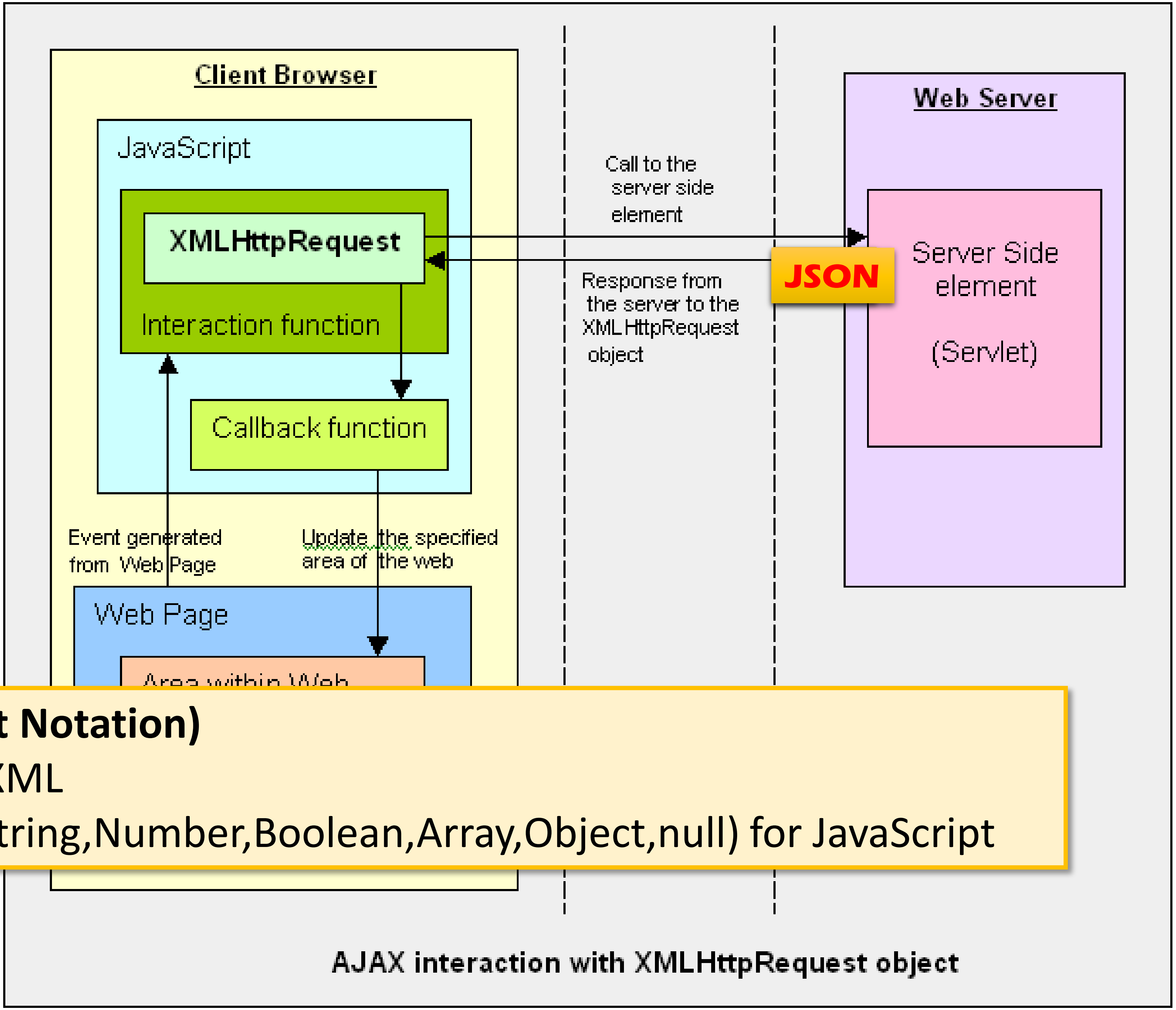
CSS

字型
顏色
寬高
位置
疊層
....
樣式

AJAX = Asynchronous JavaScript and XML



XmlHttpRequest()


Client Browser
JavaScript
XMLHttpRequest
Interaction function
Callback function
Event generated from Web Page
Update the specified area of the web
Web Page
Area within Web
Call to the server side element
Response from the server to the XMLHttpRequest object
JSON
Web Server
Server Side element
(Servlet)
AJAX interaction with XMLHttpRequest object

JSON (JavaScript Object Notation)

1. Smaller in size than XML
2. Explicit data types (String,Number,Boolean,Array,Object,null) for JavaScript

XML vs. JSON

XML

Attribute

Sub Element

```
<student name=“David" age="12">
  <address>
    <country>Taiwan</country>
    <city>Taipei</city>
    <district>...</district>
    ...
  </address>
</student>
```

All are string type

174 字

JSON

Number

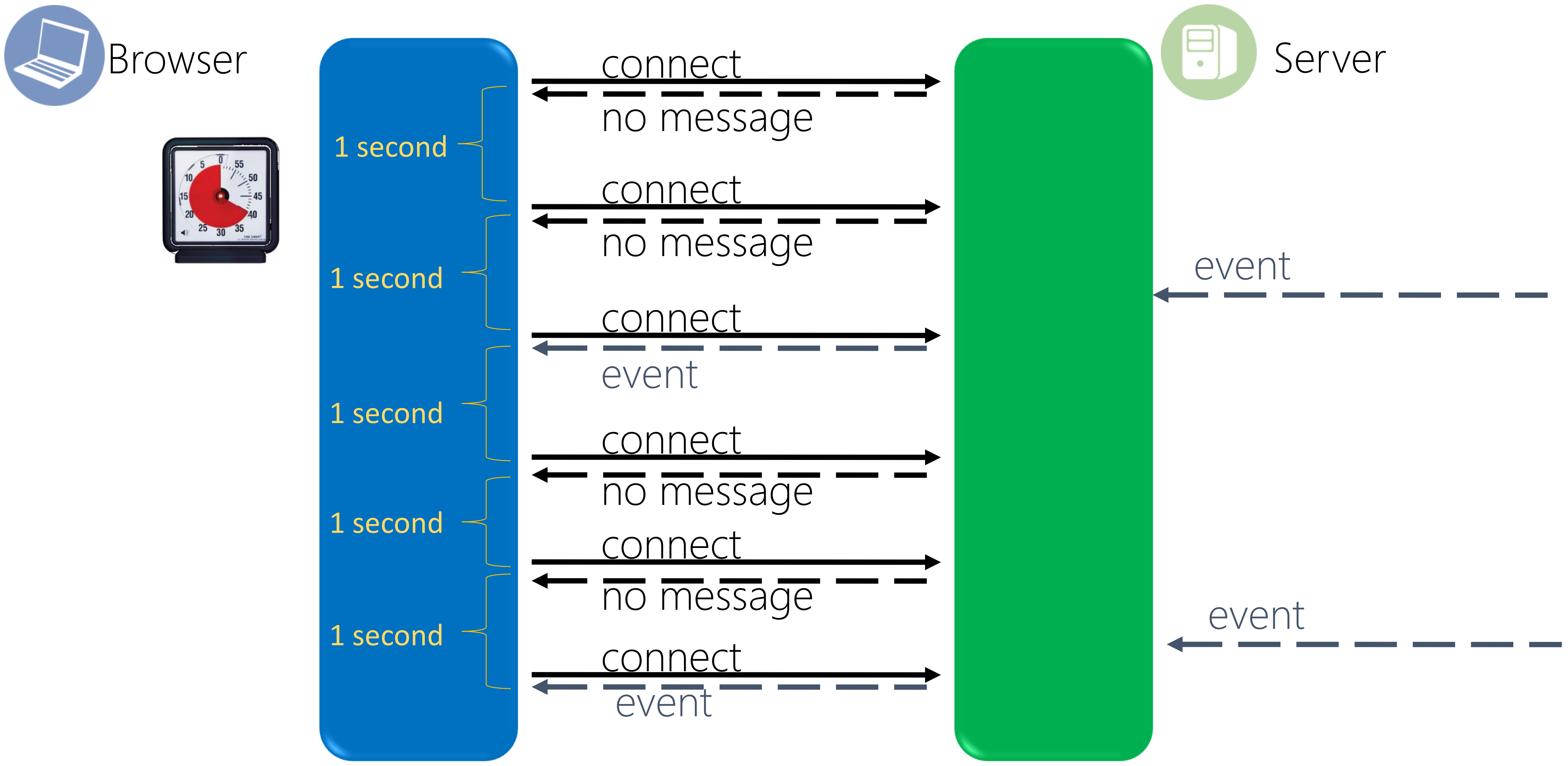
Object

```
{
  "name": “David",
  "age" : 12,
  "address" :
  {
    "country" : “Taiwan",
    "city" : “Taipei",
    "district" : "..",
    ...
  }
}
```

92 字

Polling (by Ajax)

Polling



Browser

Server

1 second

1 second

1 second

1 second

1 second

connect

no message

connect

no message

event

connect

event

connect

no message

connect

no message

event

connect

event

Polling

Pros

- Simple
- New client request != new page load (Ajax)

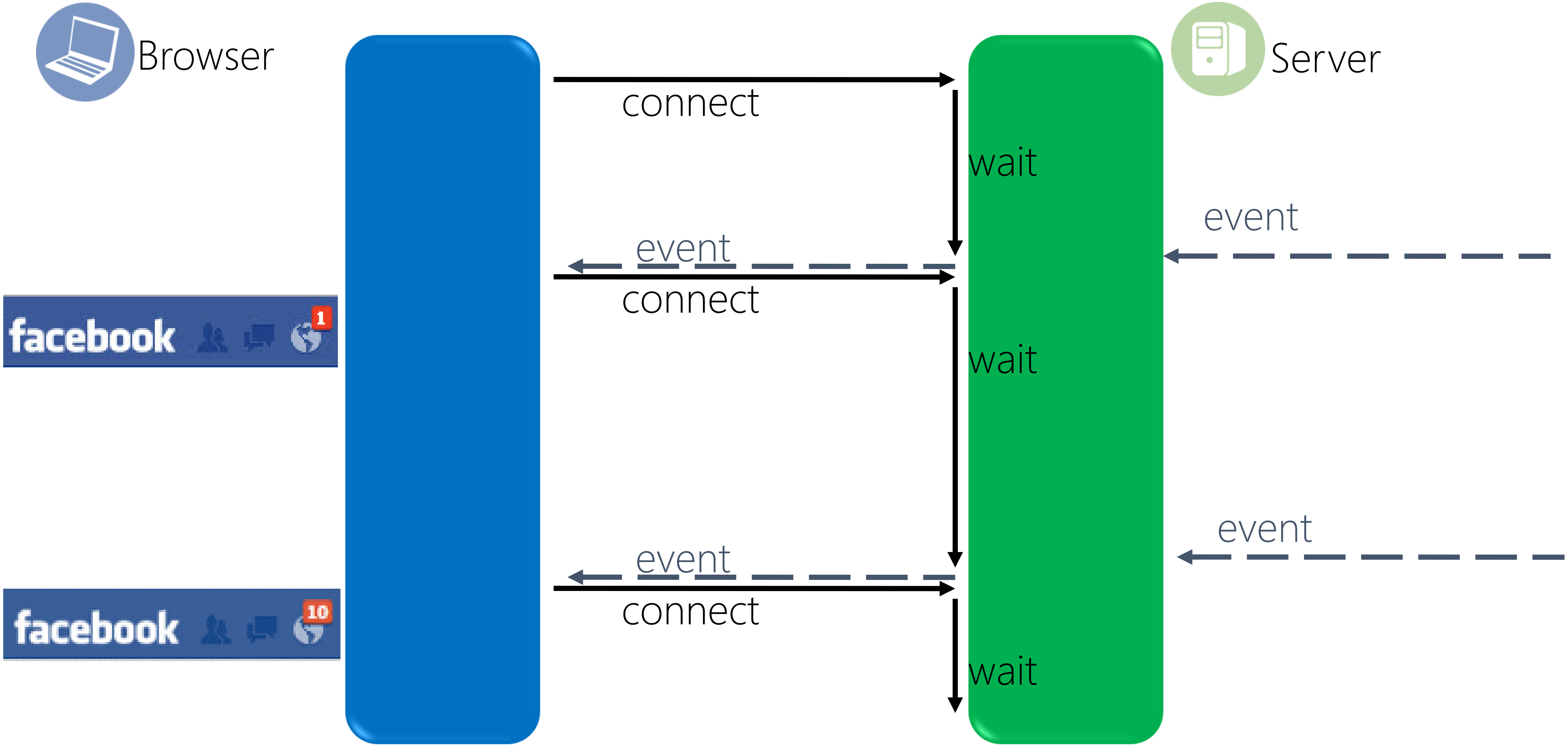
Cons

- Event latency depends on polling period (Some web apps need rapid/frequently updates)
- No real-time user experience
- Can increase polling rate , but wastes bandwidth, most requests return no message
- Frequent polling determine high server loads

Comet/Long Polling (by Ajax)

Comet/Long polling

facebook



Browser

Server

connect

wait

event

event

connect

wait

event

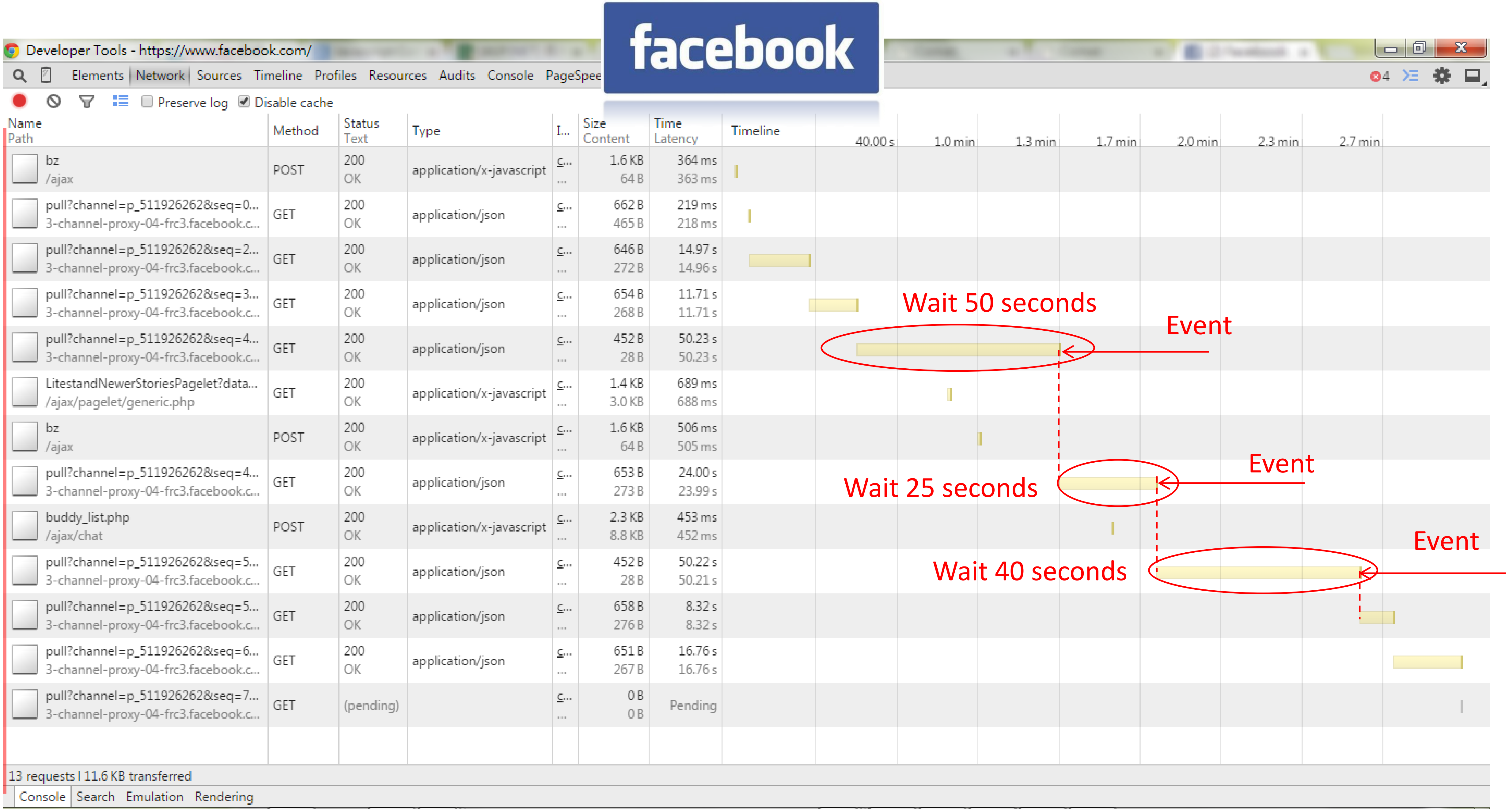
event

connect

wait

facebook 1

facebook 10



facebook
Developer Tools - https://www.facebook.com/
Elements Network Sources Timeline Profiles Resources Audits Console PageSpee
☐ Preserve log ☑ Disable cache
40.00 s 1.0 min 1.3 min 1.7 min 2.0 min 2.3 min 2.7 min
Console Search Emulation Rendering

Name Path	Method	Status Text	Type	I...	Size Content	Time Latency
bz /ajax	POST	200 OK	application/x-javascript	c... ...	1.6 KB 64 B	364 ms 363 ms
pull?channel=p_511926262&seq=0... 3-channel-proxy-04-frc3.facebook.c...	GET	200 OK	application/json	c... ...	662 B 465 B	219 ms 218 ms
pull?channel=p_511926262&seq=2... 3-channel-proxy-04-frc3.facebook.c...	GET	200 OK	application/json	c... ...	646 B 272 B	14.97 s 14.96 s
pull?channel=p_511926262&seq=3... 3-channel-proxy-04-frc3.facebook.c...	GET	200 OK	application/json	c... ...	654 B 268 B	11.71 s 11.71 s
pull?channel=p_511926262&seq=4... 3-channel-proxy-04-frc3.facebook.c...	GET	200 OK	application/json	c... ...	452 B 28 B	50.23 s 50.23 s
LitestandNewerStoriesPagelet?data... /ajax/pagelet/generic.php	GET	200 OK	application/x-javascript	c... ...	1.4 KB 3.0 KB	689 ms 688 ms
bz /ajax	POST	200 OK	application/x-javascript	c... ...	1.6 KB 64 B	506 ms 505 ms
pull?channel=p_511926262&seq=4... 3-channel-proxy-04-frc3.facebook.c...	GET	200 OK	application/json	c... ...	653 B 273 B	24.00 s 23.99 s
buddy_list.php /ajax/chat	POST	200 OK	application/x-javascript	c... ...	2.3 KB 8.8 KB	453 ms 452 ms
pull?channel=p_511926262&seq=5... 3-channel-proxy-04-frc3.facebook.c...	GET	200 OK	application/json	c... ...	452 B 28 B	50.22 s 50.21 s
pull?channel=p_511926262&seq=5... 3-channel-proxy-04-frc3.facebook.c...	GET	200 OK	application/json	c... ...	658 B 276 B	8.32 s 8.32 s
pull?channel=p_511926262&seq=6... 3-channel-proxy-04-frc3.facebook.c...	GET	200 OK	application/json	c... ...	651 B 267 B	16.76 s 16.76 s
pull?channel=p_511926262&seq=7... 3-channel-proxy-04-frc3.facebook.c...	GET	(pending)		c... ...	0 B 0 B	Pending

13 requests | 11.6 KB transferred

Wait 50 seconds

Event

Wait 25 seconds

Event

Wait 40 seconds

Event

Comet/Long polling

How

- Client does request; service maintains connection
- Fakes notifications by sending “event”/response when ready (Emulating a Push mechanism)
- Always a pending request

Pros

- Emulating a more real-time communication model

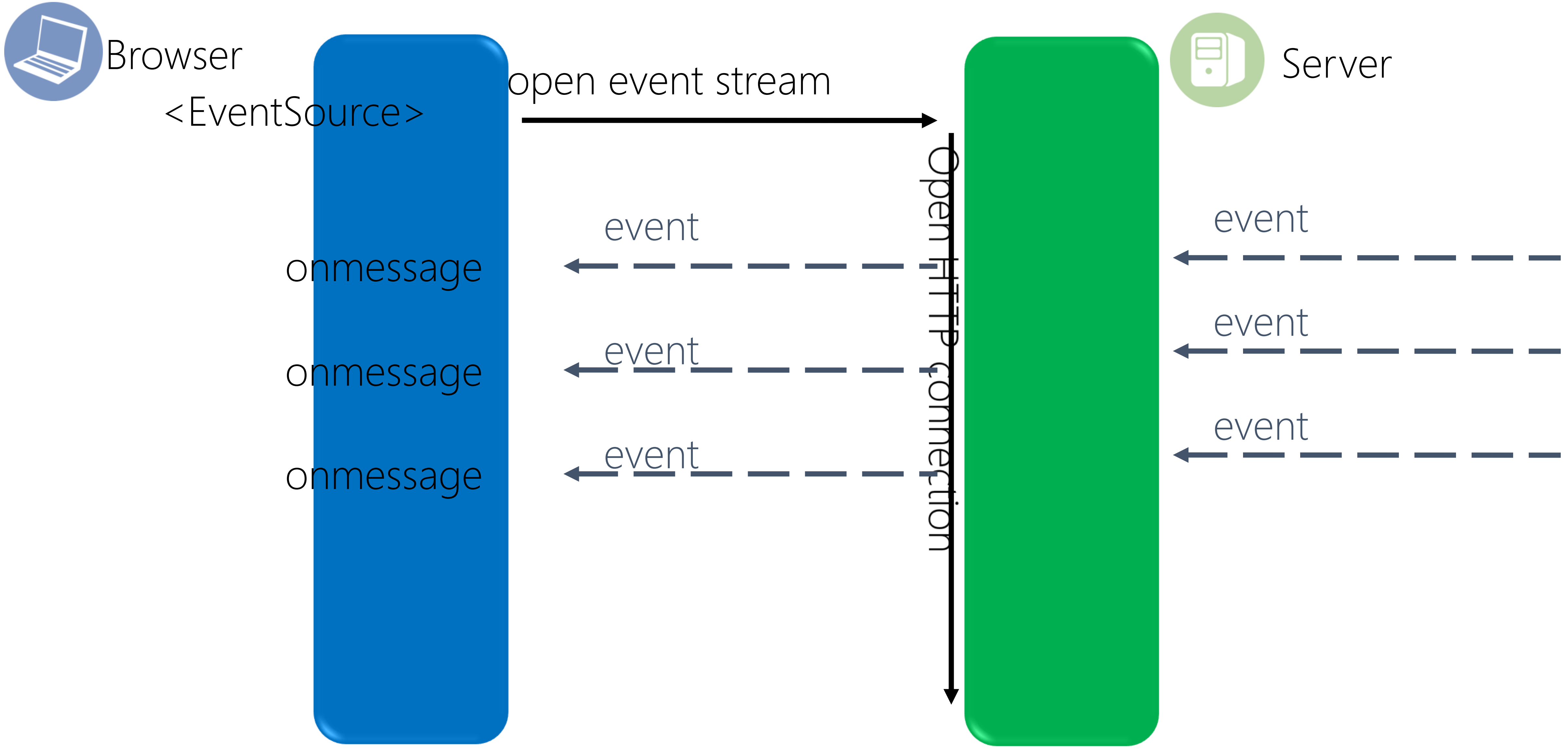
Cons

- HTTP overhead still a problem (not suited for low latency apps)

HTML
5

Server-Sent Events

Server-sent events



Server-sent events

How

- Simulates a server push channel **over HTTP**
- **Unidirectional**, from server to browser
- Standardizes some form of Comet/http streaming
- New html tag: <EventSource>
- New mime type: text/event-stream

Refs

- http://www.w3.org/TR/2009/WD-eventsource-20091029/
- http://www.html5rocks.com/en/tutorials/eventsource/basics/

Server-sent events (SSE)

Pros

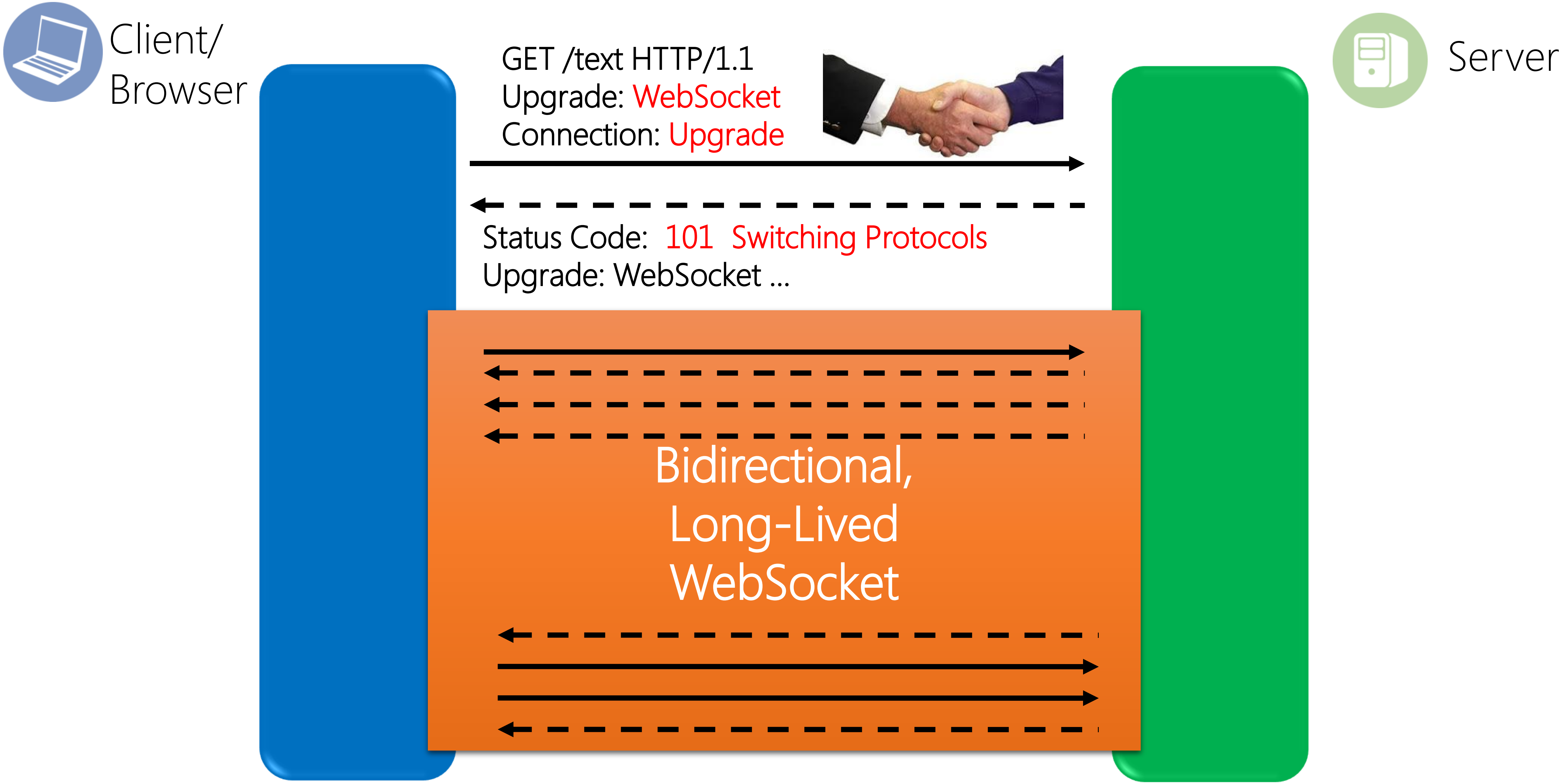
- A real-time communication model from server to client
- Auto reconnect after disconnected.

Cons

- Not bidirectional. Communication is only one way (Some web apps need two-way communication)
- Not more real-time
- Only text data

WebSocket

WebSocket



Client/
Browser

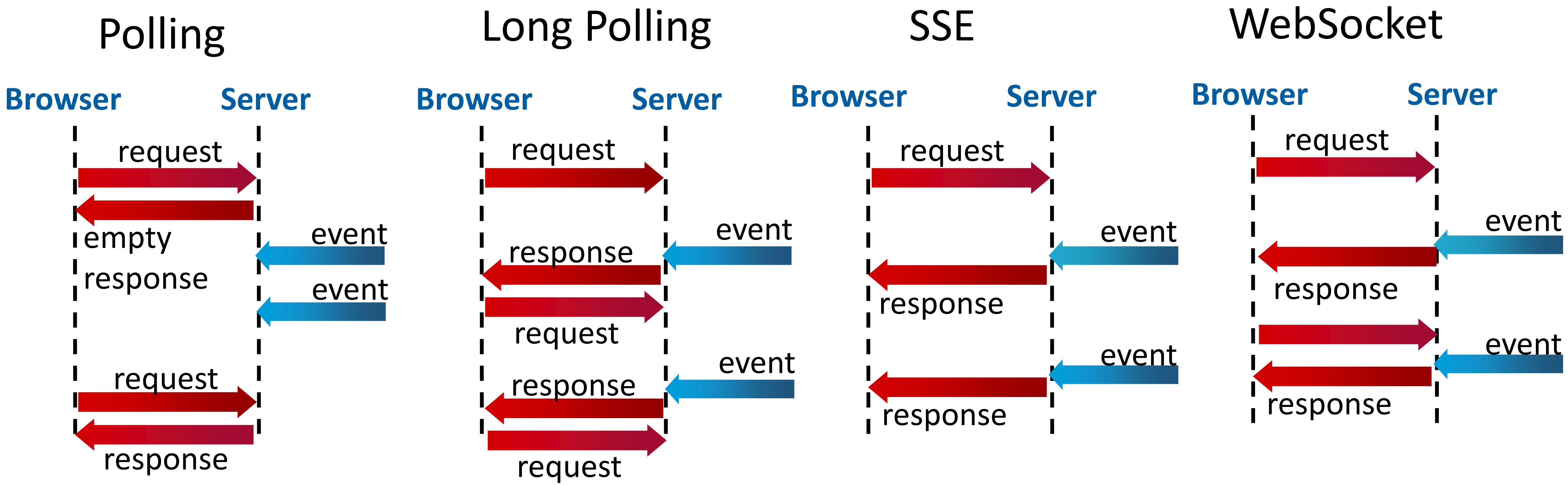
Server

GET /text HTTP/1.1
Upgrade: WebSocket
Connection: Upgrade

Status Code: 101 Switching Protocols
Upgrade: WebSocket ...

Bidirectional,
Long-Lived
WebSocket

Comparison



Polling

Browser Server

request

empty response

event

event

request

response

Long Polling

Browser Server

request

event

response

request

event

response

request

SSE

Browser Server

request

event

response

event

response

WebSocket

Browser Server

request

event

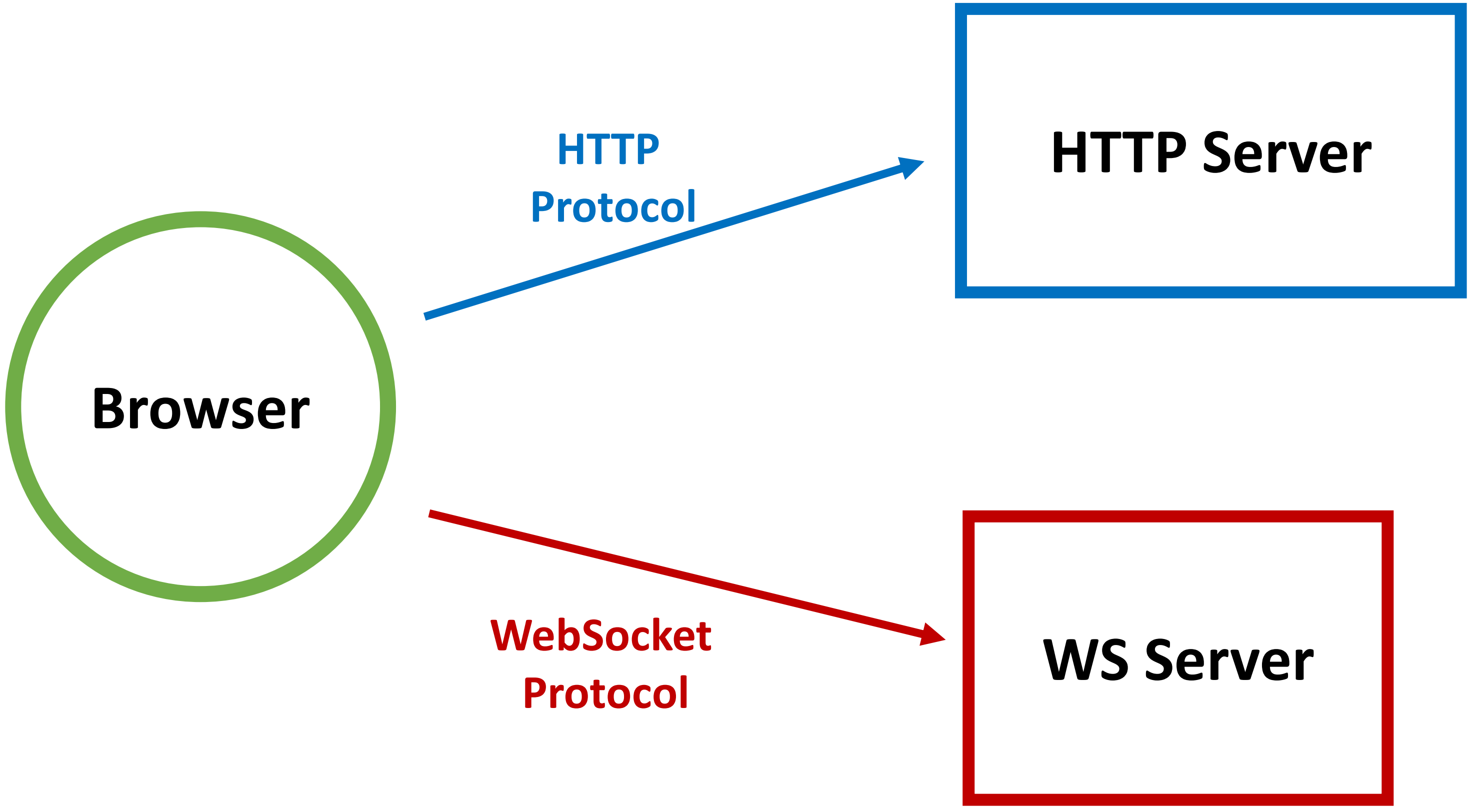
response

event

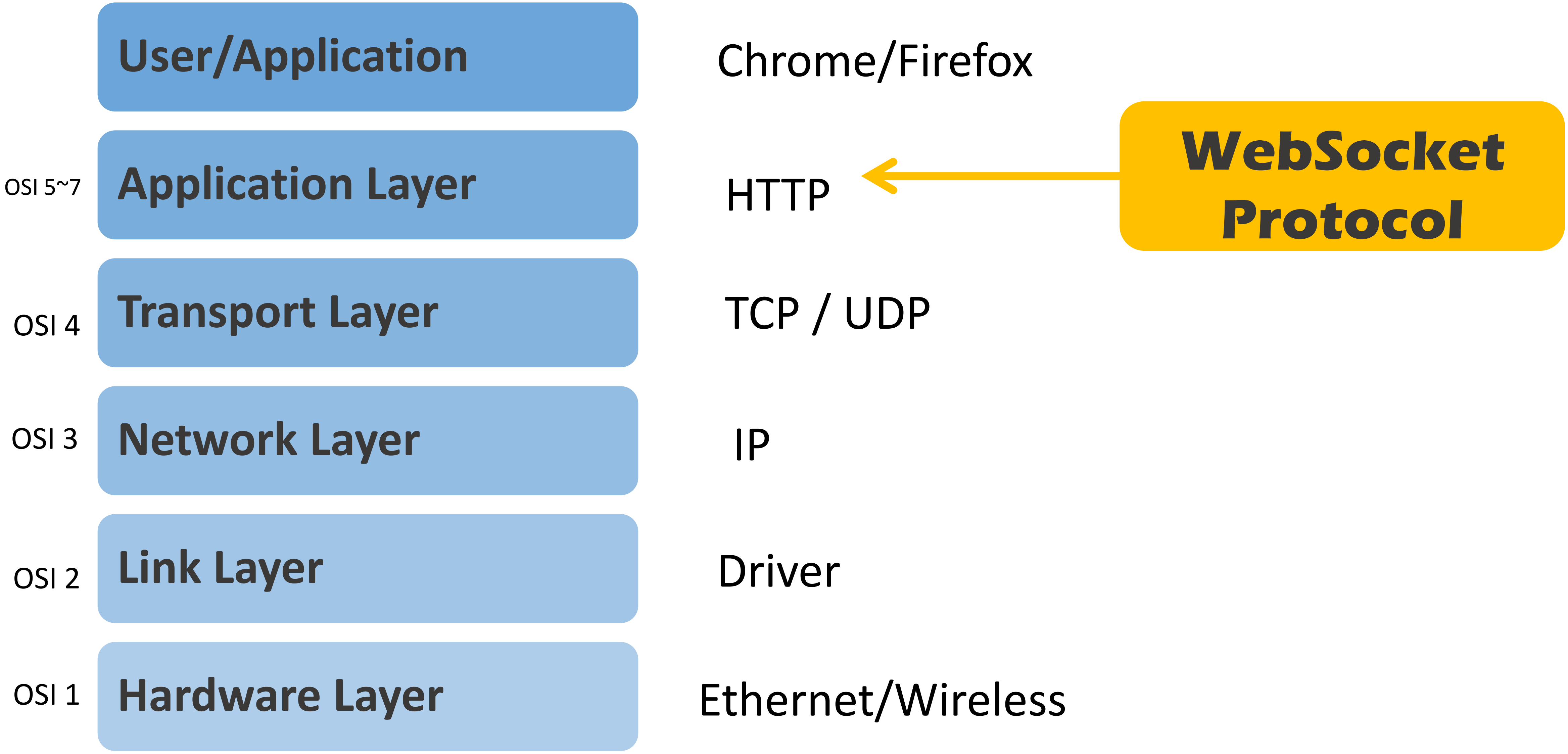
response

HTML5 WebSocket

- Server can send info to client anytime: update latency reduced
  ➔ Real-time full-duplex communication over TCP
- Single TCP socket
- Not HTTP, but uses HTTP to bootstrap
- Messages are either UTF-8 text or binary data
- Shares port with existing HTTP (80, 443)


HTTP Server
HTTP
Protocol
Browser
WebSocket
Protocol
WS Server

The TCP/IP Stack



	User/Application	Chrome/Firefox
OSI 5~7	Application Layer	HTTP
OSI 4	Transport Layer	TCP / UDP
OSI 3	Network Layer	IP
OSI 2	Link Layer	Driver
OSI 1	Hardware Layer	Ethernet/Wireless

WebSocket Protocol

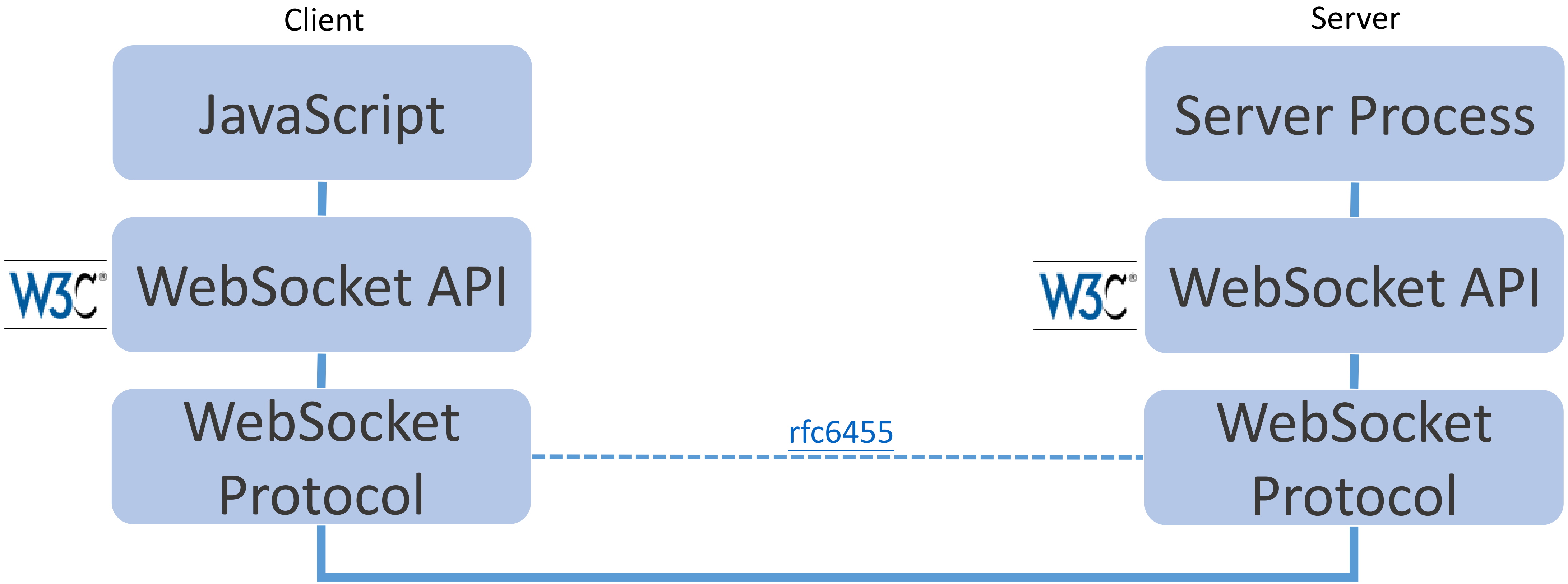
WebSocket

Pros

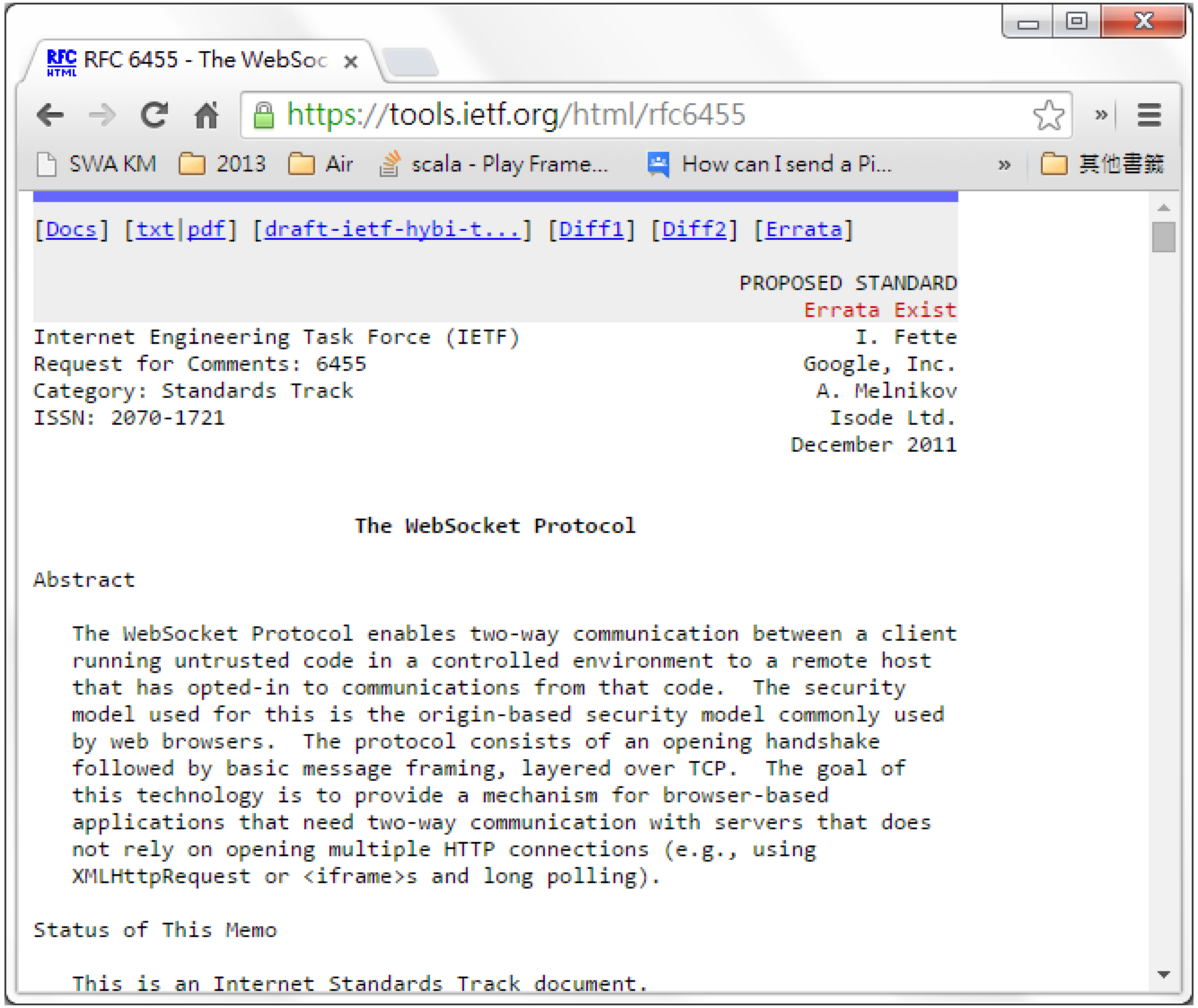
- A real-time communication model
- Bidirectional
- Reduction in unnecessary network traffic and latency (compared to the polling and long-polling)
- Tiny Header : Minimize is 2 bytes.
  Compares to HTTP Header (from ~200 bytes to over 2KB.)

WebSocket Specifications

- WebSocket protocol - https://tools.ietf.org/html/rfc6455
- WebSocket API - http://www.w3.org/TR/2011/WD-websockets-20110419/



The WebSocket Protocol



[Docs] [txt|pdf] [draft-ietf-hybi-t...] [Diff1] [Diff2] [Errata]

```
                                                       PROPOSED STANDARD
                                                           Errata Exist
Internet Engineering Task Force (IETF)                          I. Fette
Request for Comments: 6455                                  Google, Inc.
Category: Standards Track                                    A. Melnikov
ISSN: 2070-1721                                               Isode Ltd.
                                                           December 2011
```

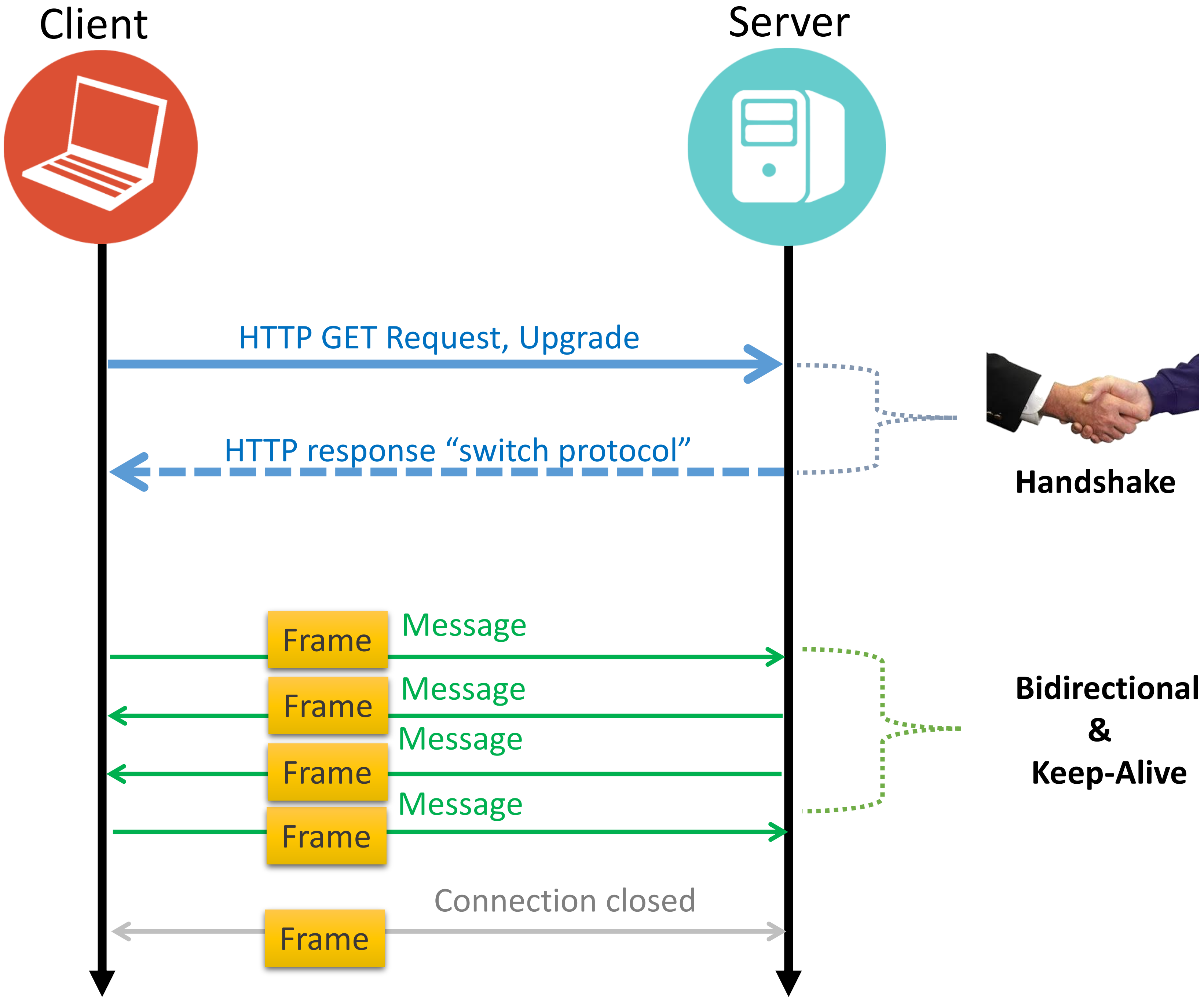
The WebSocket Protocol

Abstract

The WebSocket Protocol enables two-way communication between a client running untrusted code in a controlled environment to a remote host that has opted-in to communications from that code. The security model used for this is the origin-based security model commonly used by web browsers. The protocol consists of an opening handshake followed by basic message framing, layered over TCP. The goal of this technology is to provide a mechanism for browser-based applications that need two-way communication with servers that does not rely on opening multiple HTTP connections (e.g., using XMLHttpRequest or <iframe>s and long polling).

Status of This Memo

This is an Internet Standards Track document.



Client

Server

HTTP GET Request, Upgrade

HTTP response “switch protocol”

Handshake

Frame

Message

Frame

Message

Frame

Message

Frame

Message

Bidirectional

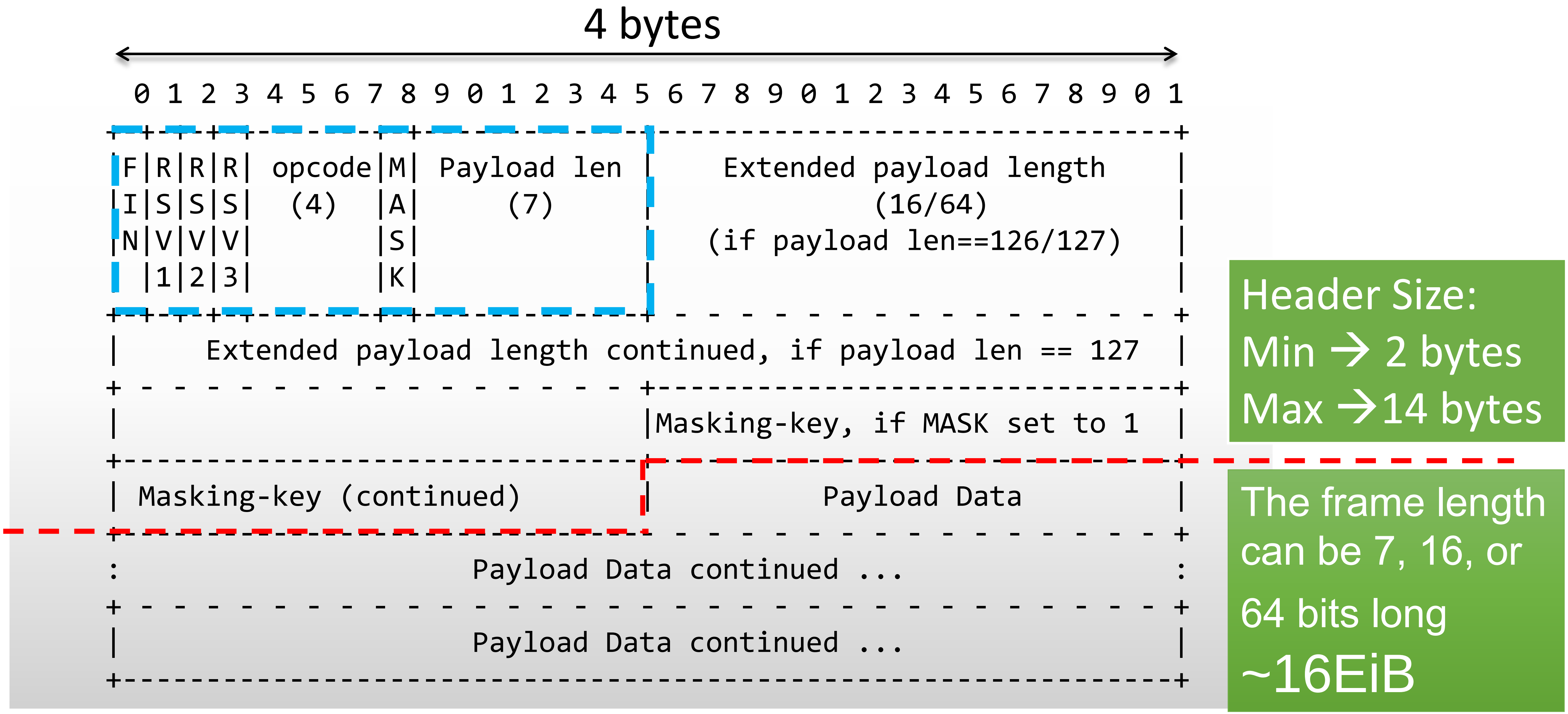
&

Keep-Alive

Frame

Connection closed

What's in a Frame?



4 bytes

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-------+-+-------------+-------------------------------+
|F|R|R|R| opcode|M| Payload len |    Extended payload length    |
|I|S|S|S|  (4)  |A|     (7)     |             (16/64)           |
|N|V|V|V|       |S|             |   (if payload len==126/127)   |
| |1|2|3|       |K|             |                               |
+-+-+-+-+-------+-+-------------+ - - - - - - - - - - - - - - - +
|     Extended payload length continued, if payload len == 127  |
+ - - - - - - - - - - - - - - - +-------------------------------+
|                               |Masking-key, if MASK set to 1  |
+-------------------------------+-------------------------------+
| Masking-key (continued)       |          Payload Data         |
+-------------------------------- - - - - - - - - - - - - - - - +
:                     Payload Data continued ...                :
+ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - +
|                     Payload Data continued ...                |
+---------------------------------------------------------------+
```

Header Size:
Min → 2 bytes
Max →14 bytes

The frame length can be 7, 16, or 64 bits long

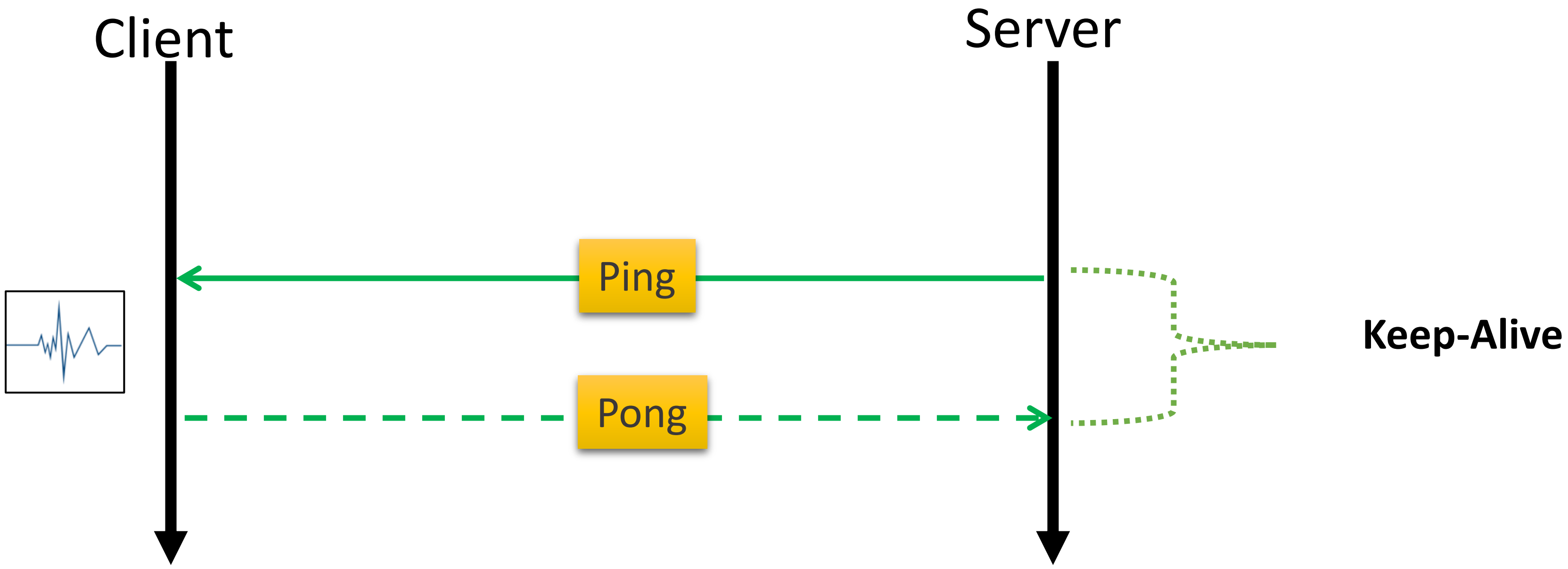
~16EiB

Opcode: 4 bits

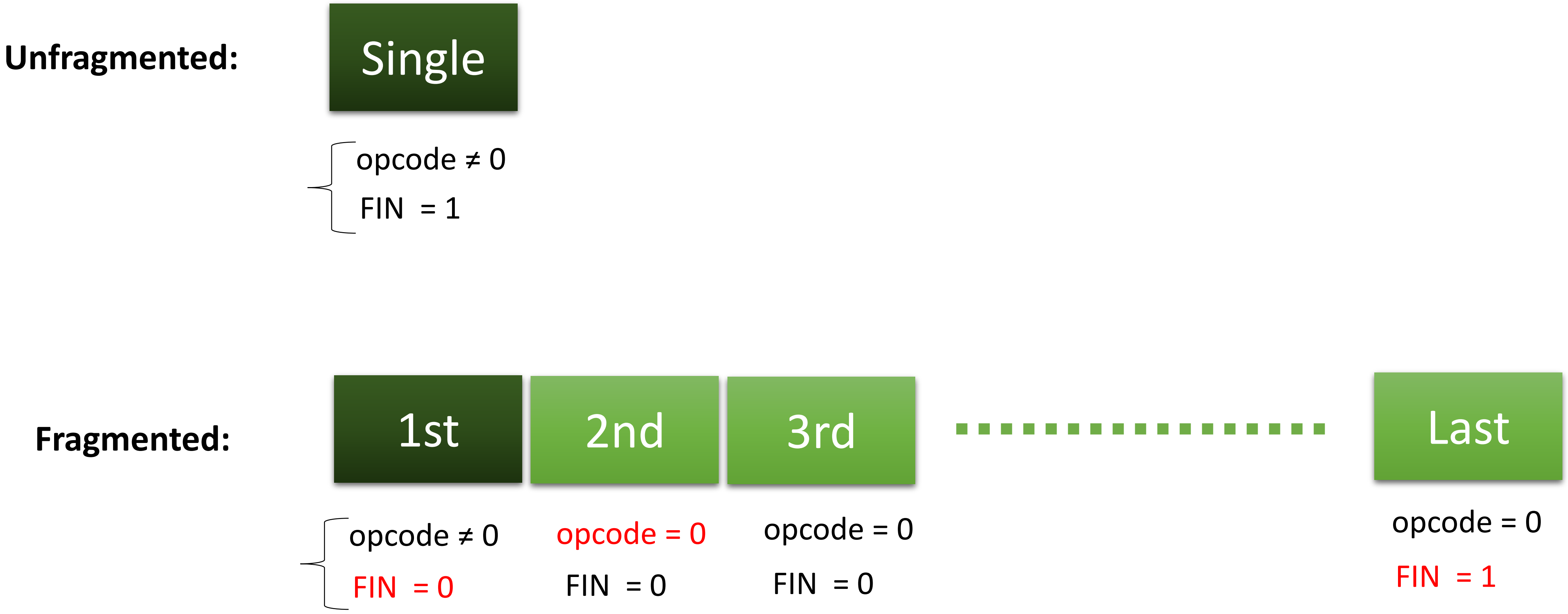
0	= denotes a continuation frame	Data Frames
1	= denotes a text frame (UTF-8)	
2	= denotes a binary frame	
8	= denotes a connection close	Control Frames
9	= denotes a ping : Confirmation of connection	
A	= denotes a pong : Reply for Ping	

Ping/Pong Frames

- A pong frame sent in response to a Ping frame. It must be implemented on client, so server can cleanup dead connections.
- Server may send a ping frame, on the other hand, server can push messages detect client is alive or not.



Data Frame



Unfragmented:

Single

opcode ≠ 0
FIN = 1

Fragmented:

1st

2nd

3rd

Last

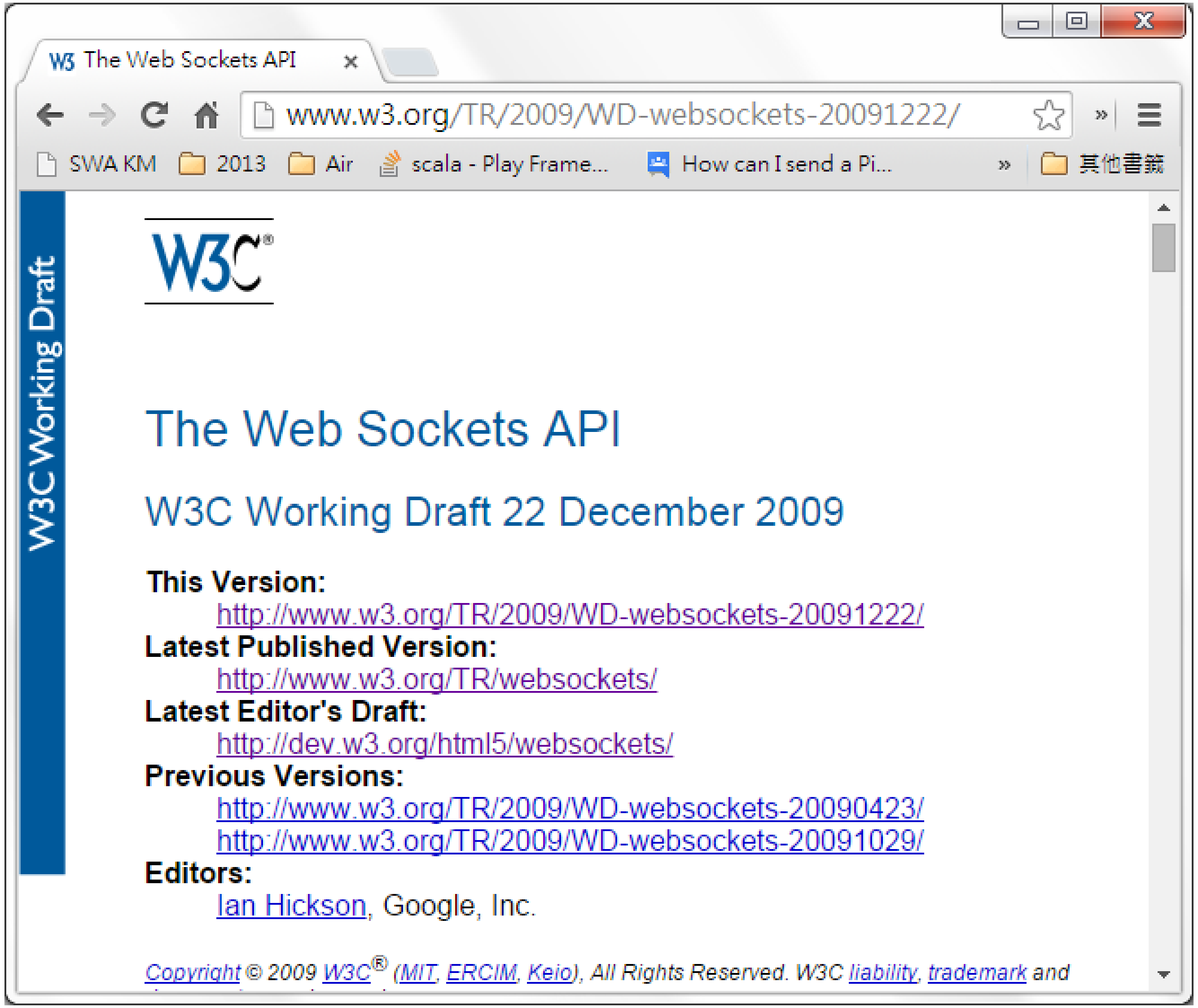
opcode ≠ 0
FIN = 0

opcode = 0
FIN = 0

opcode = 0
FIN = 0

opcode = 0
FIN = 1

W3C WebSockt API



The Web Sockets API

W3C Working Draft 22 December 2009

This Version:
http://www.w3.org/TR/2009/WD-websockets-20091222/

Latest Published Version:
http://www.w3.org/TR/websockets/

Latest Editor's Draft:
http://dev.w3.org/html5/websockets/

Previous Versions:
http://www.w3.org/TR/2009/WD-websockets-20090423/
http://www.w3.org/TR/2009/WD-websockets-20091029/

Editors:
Ian Hickson, Google, Inc.

Copyright © 2009 W3C® (MIT, ERCIM, Keio), All Rights Reserved. W3C liability, trademark and

WebSocket API

CONNECTION

- open
- close

SEND DATA

- send(message)

EVENT HANDLERS

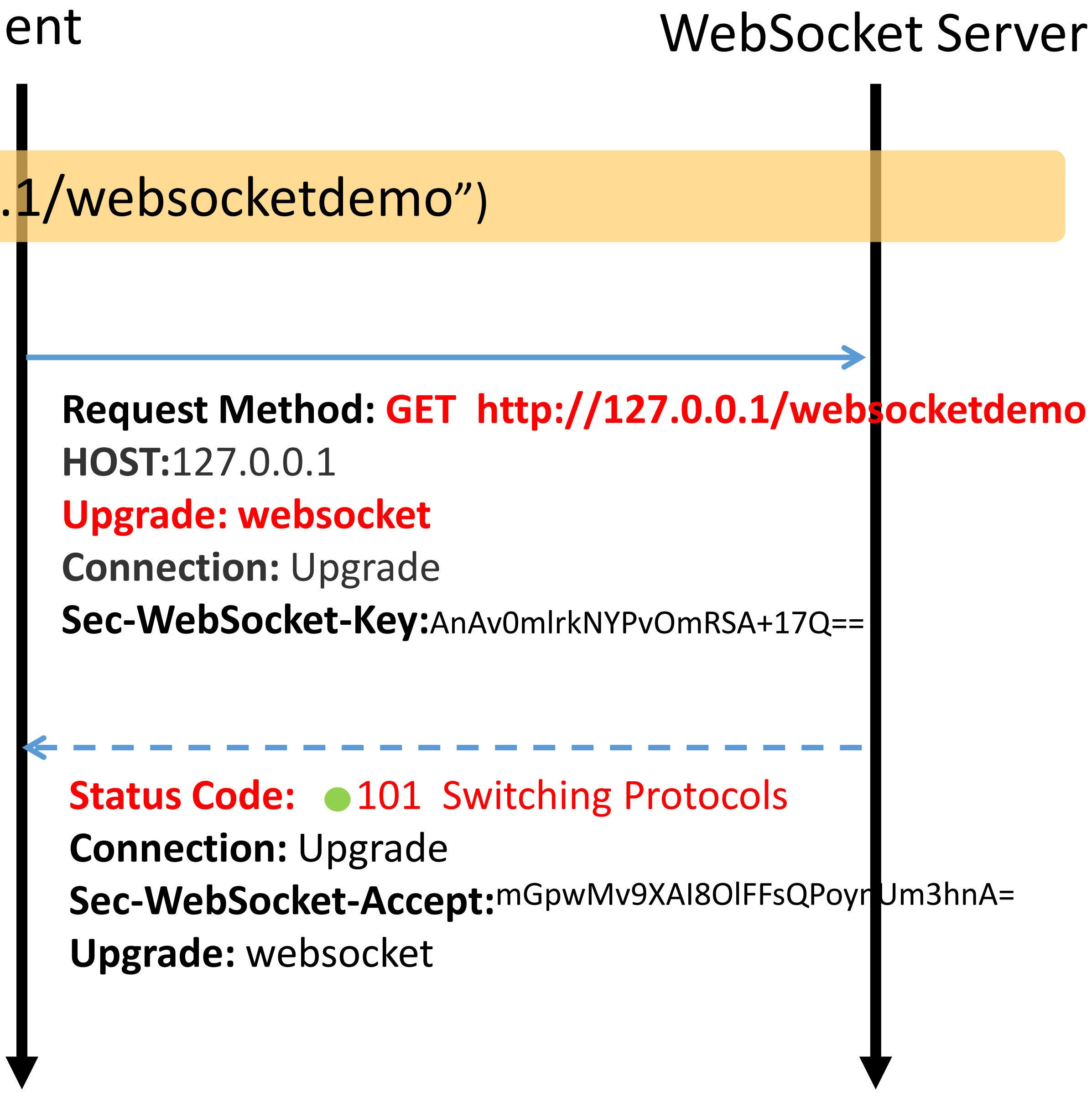
- onOpen
- onClosed
- onMessage(message)
- onError(error)

JavaScript

JavaScript

ws = new WebSocket("ws://127.0.0.1/websocketdemo")

1.A websocket connection is initiated via a standard HTTP GET request, where the client asks for an 'Upgrade'



Client

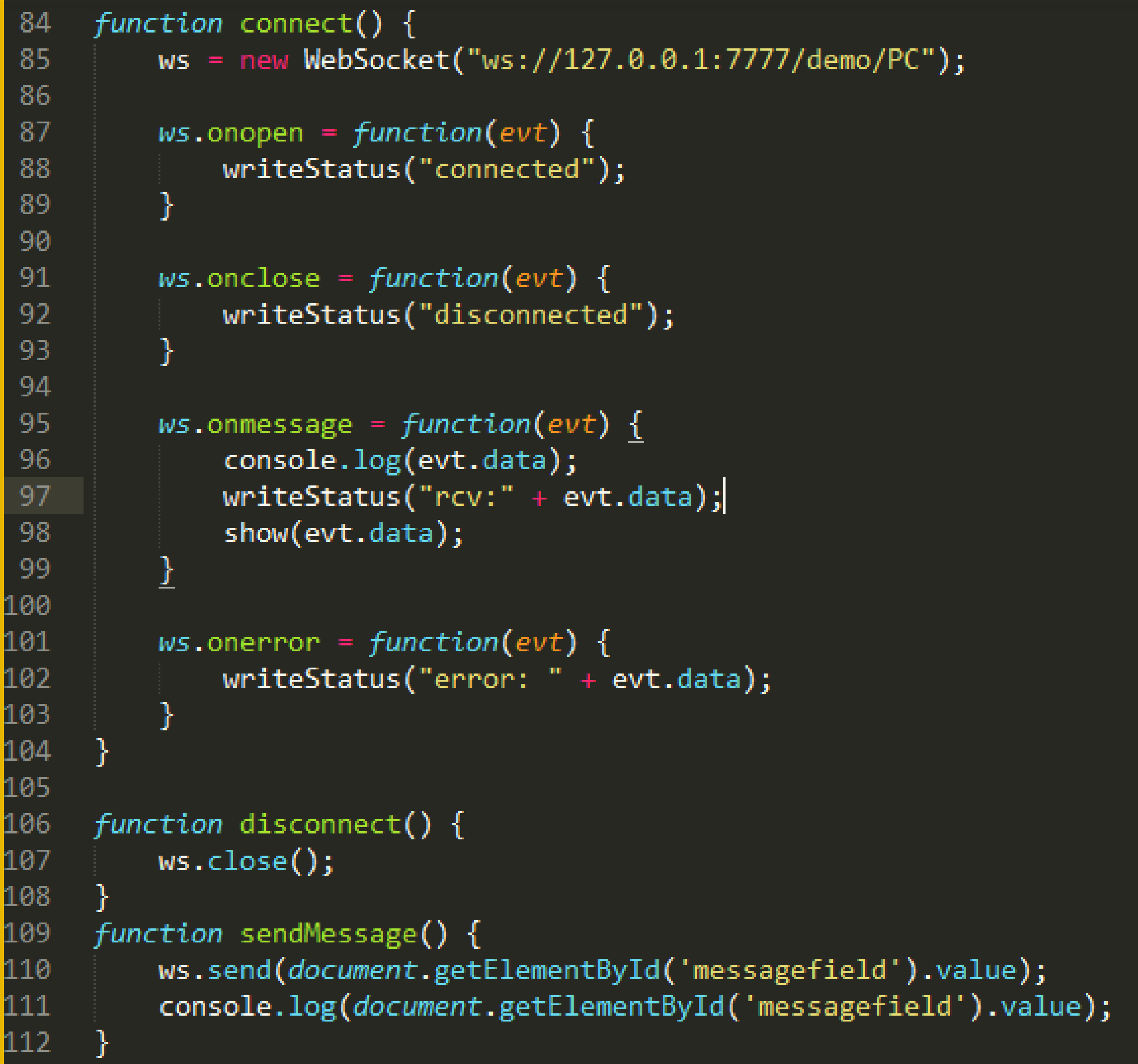
WebSocket Server

Request Method: GET http://127.0.0.1/websocketdemo HTTP/1.1
HOST:127.0.0.1
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: AnAv0mlrkNYPvOmRSA+17Q==

Status Code: ●101 Switching Protocols
Connection: Upgrade
Sec-WebSocket-Accept:mGpwMv9XAI8OlFFsQPoyrUm3hnA=
Upgrade: websocket

2.The response will be a 101 status, 'Switching protocol'

JavaScript Client Example



```javascript
function connect() {
    ws = new WebSocket("ws://127.0.0.1:7777/demo/PC");

    ws.onopen = function(evt) {
        writeStatus("connected");
    }

    ws.onclose = function(evt) {
        writeStatus("disconnected");
    }

    ws.onmessage = function(evt) {
        console.log(evt.data);
        writeStatus("rcv:" + evt.data);
        show(evt.data);
    }

    ws.onerror = function(evt) {
        writeStatus("error: " + evt.data);
    }
}

function disconnect() {
    ws.close();
}
function sendMessage() {
    ws.send(document.getElementById('messagefield').value);
    console.log(document.getElementById('messagefield').value);
}
```

WebSocket need maintenance and care:

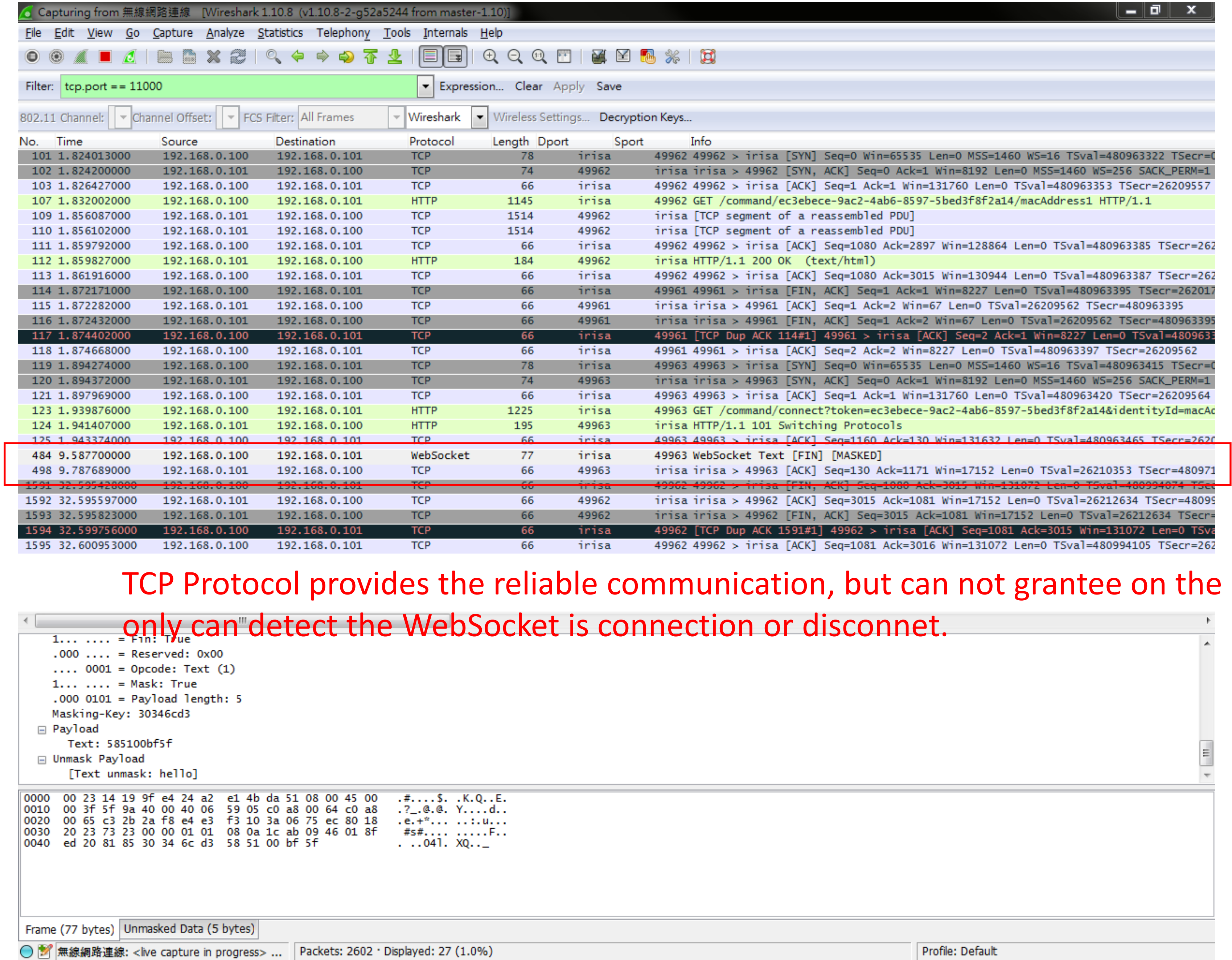
- **Re-open conn** if network hiccup or timeout
- **Back off** if server is down, don’t keep trying
- **Keep alive** if your connection times out
- **Buffer and re-send** msgs in the above cases

The reliability of WebSocket

No connection reliability

- No reconnect handling
- No guaranteed message delivery (same as Server-Sent Event)
- Reliability has to be in app (same as “keep alive message - Ping/Pong” – Server job)

Reliability of Websocket



Capturing from 無線網路連線 [Wireshark 1.10.8 (v1.10.8-2-g52a5244 from master-1.10)]

Filter: tcp.port == 11000

802.11 Channel: Channel Offset: FCS Filter: All Frames Wireshark

```
No.   Time          Source         Destination    Protocol   Length Dport  Sport  Info
 101  1.824013000   192.168.0.100  192.168.0.101  TCP          78   irisa  49962 49962 > irisa [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=16 TSval=480963322 TSecr=0
 102  1.824200000   192.168.0.101  192.168.0.100  TCP          74   49962  irisa irisa > 49962 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
 103  1.826427000   192.168.0.100  192.168.0.101  TCP          66   irisa  49962 49962 > irisa [ACK] Seq=1 Ack=1 Win=131760 Len=0 TSval=480963353 TSecr=26209557
 107  1.832002000   192.168.0.100  192.168.0.101  HTTP       1145   irisa  49962 GET /command/ec3ebece-9ac2-4ab6-8597-5bed3f8f2a14/macAddress1 HTTP/1.1
 109  1.856087000   192.168.0.101  192.168.0.100  TCP        1514   49962  irisa [TCP segment of a reassembled PDU]
 110  1.856102000   192.168.0.101  192.168.0.100  TCP        1514   49962  irisa [TCP segment of a reassembled PDU]
 111  1.859792000   192.168.0.100  192.168.0.101  TCP          66   irisa  49962 49962 > irisa [ACK] Seq=1080 Ack=2897 Win=128864 Len=0 TSval=480963385 TSecr=262
 112  1.859827000   192.168.0.101  192.168.0.100  HTTP        184   49962  irisa HTTP/1.1 200 OK  (text/html)
 113  1.861916000   192.168.0.100  192.168.0.101  TCP          66   irisa  49962 49962 > irisa [ACK] Seq=1080 Ack=3015 Win=130944 Len=0 TSval=480963387 TSecr=262
 114  1.872171000   192.168.0.100  192.168.0.101  TCP          66   irisa  49961 49961 > irisa [FIN, ACK] Seq=1 Ack=1 Win=8227 Len=0 TSval=480963395 TSecr=262017
 115  1.872282000   192.168.0.101  192.168.0.100  TCP          66   49961  irisa irisa > 49961 [ACK] Seq=1 Ack=2 Win=67 Len=0 TSval=26209562 TSecr=480963395
 116  1.872432000   192.168.0.101  192.168.0.100  TCP          66   49961  irisa irisa > 49961 [FIN, ACK] Seq=1 Ack=2 Win=67 Len=0 TSval=26209562 TSecr=480963395
 117  1.874402000   192.168.0.100  192.168.0.101  TCP          66   irisa  49961 [TCP Dup ACK 114#1] 49961 > irisa [ACK] Seq=2 Ack=1 Win=8227 Len=0 TSval=4809633
 118  1.874668000   192.168.0.100  192.168.0.101  TCP          66   irisa  49961 49961 > irisa [ACK] Seq=2 Ack=2 Win=8227 Len=0 TSval=480963397 TSecr=26209562
 119  1.894274000   192.168.0.100  192.168.0.101  TCP          78   irisa  49963 49963 > irisa [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=16 TSval=480963415 TSecr=0
 120  1.894372000   192.168.0.101  192.168.0.100  TCP          74   49963  irisa irisa > 49963 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
 121  1.897969000   192.168.0.100  192.168.0.101  TCP          66   irisa  49963 49963 > irisa [ACK] Seq=1 Ack=1 Win=131760 Len=0 TSval=480963420 TSecr=26209564
 123  1.939876000   192.168.0.100  192.168.0.101  HTTP       1225   irisa  49963 GET /command/connect?token=ec3ebece-9ac2-4ab6-8597-5bed3f8f2a14&identityId=macAc
 124  1.941407000   192.168.0.101  192.168.0.100  HTTP        195   49963  irisa HTTP/1.1 101 Switching Protocols
 125  1.943374000   192.168.0.100  192.168.0.101  TCP          66   irisa  49963 49963 > irisa [ACK] Seq=1160 Ack=130 Win=131632 Len=0 TSval=480963465 TSecr=262
 484  9.587700000   192.168.0.100  192.168.0.101  WebSocket    77   irisa  49963 WebSocket Text [FIN] [MASKED]
 498  9.787689000   192.168.0.101  192.168.0.100  TCP          66   49963  irisa irisa > 49963 [ACK] Seq=130 Ack=1171 Win=17152 Len=0 TSval=26210353 TSecr=480971
1591 32.595428000   192.168.0.100  192.168.0.101  TCP          66   irisa  49962 49962 > irisa [FIN, ACK] Seq=1080 Ack=3015 Win=131072 Len=0 TSval=480994074 TSec
1592 32.595597000   192.168.0.101  192.168.0.100  TCP          66   49962  irisa irisa > 49962 [ACK] Seq=3015 Ack=1081 Win=17152 Len=0 TSval=26212634 TSecr=48099
1593 32.595823000   192.168.0.101  192.168.0.100  TCP          66   49962  irisa irisa > 49962 [FIN, ACK] Seq=3015 Ack=1081 Win=17152 Len=0 TSval=26212634 TSecr=
1594 32.599756000   192.168.0.100  192.168.0.101  TCP          66   irisa  49962 [TCP Dup ACK 1591#1] 49962 > irisa [ACK] Seq=1081 Ack=3015 Win=131072 Len=0 TSva
1595 32.600953000   192.168.0.100  192.168.0.101  TCP          66   irisa  49962 49962 > irisa [ACK] Seq=1081 Ack=3016 Win=131072 Len=0 TSval=480994105 TSecr=262
```

TCP Protocol provides the reliable communication, but can not grantee on the WebSocket layer, only can detect the WebSocket is connection or disconnet.

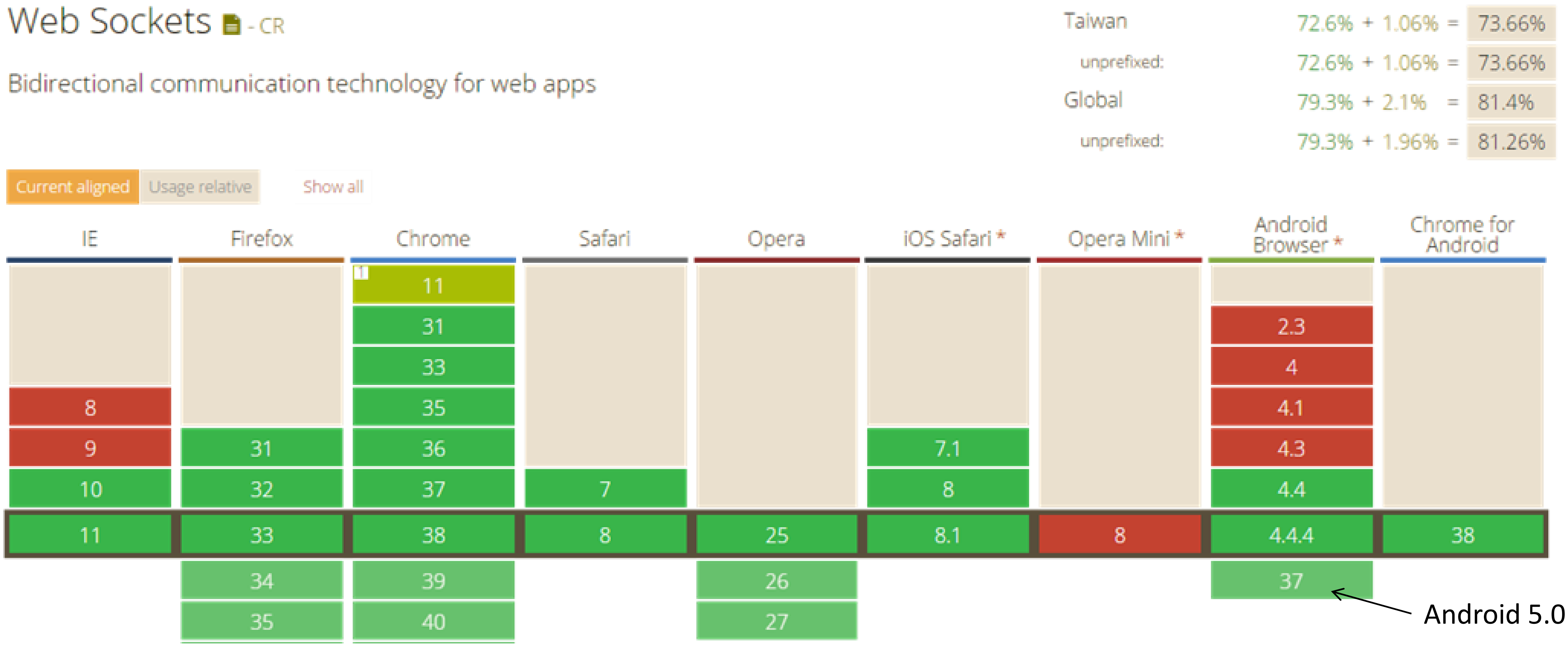
```
    1... .... = Fin: True
    .000 .... = Reserved: 0x00
    .... 0001 = Opcode: Text (1)
    1... .... = Mask: True
    .000 0101 = Payload length: 5
    Masking-Key: 30346cd3
  ⊟ Payload
      Text: 585100bf5f
  ⊟ Unmask Payload
      [Text unmask: hello]
```

```
0000  00 23 14 19 9f e4 24 a2  e1 4b da 51 08 00 45 00   .#....$. .K.Q..E.
0010  00 3f 5f 9a 40 00 40 06  59 05 c0 a8 00 64 c0 a8   .?_.@.@. Y....d..
0020  00 65 c3 2b 2a f8 e4 e3  f3 10 3a 06 75 ec 80 18   .e.+*... ..:.u...
0030  20 23 73 23 00 00 01 01  08 0a 1c ab 09 46 01 8f    #s#.... .....F..
0040  ed 20 81 85 30 34 6c d3  58 51 00 bf 5f            . ..04l. XQ.._
```

Frame (77 bytes) | Unmasked Data (5 bytes)

無線網路連線: <live capture in progress> ... Packets: 2602 · Displayed: 27 (1.0%) Profile: Default

WebSocket Browser support list



Web Sockets - CR

Bidirectional communication technology for web apps

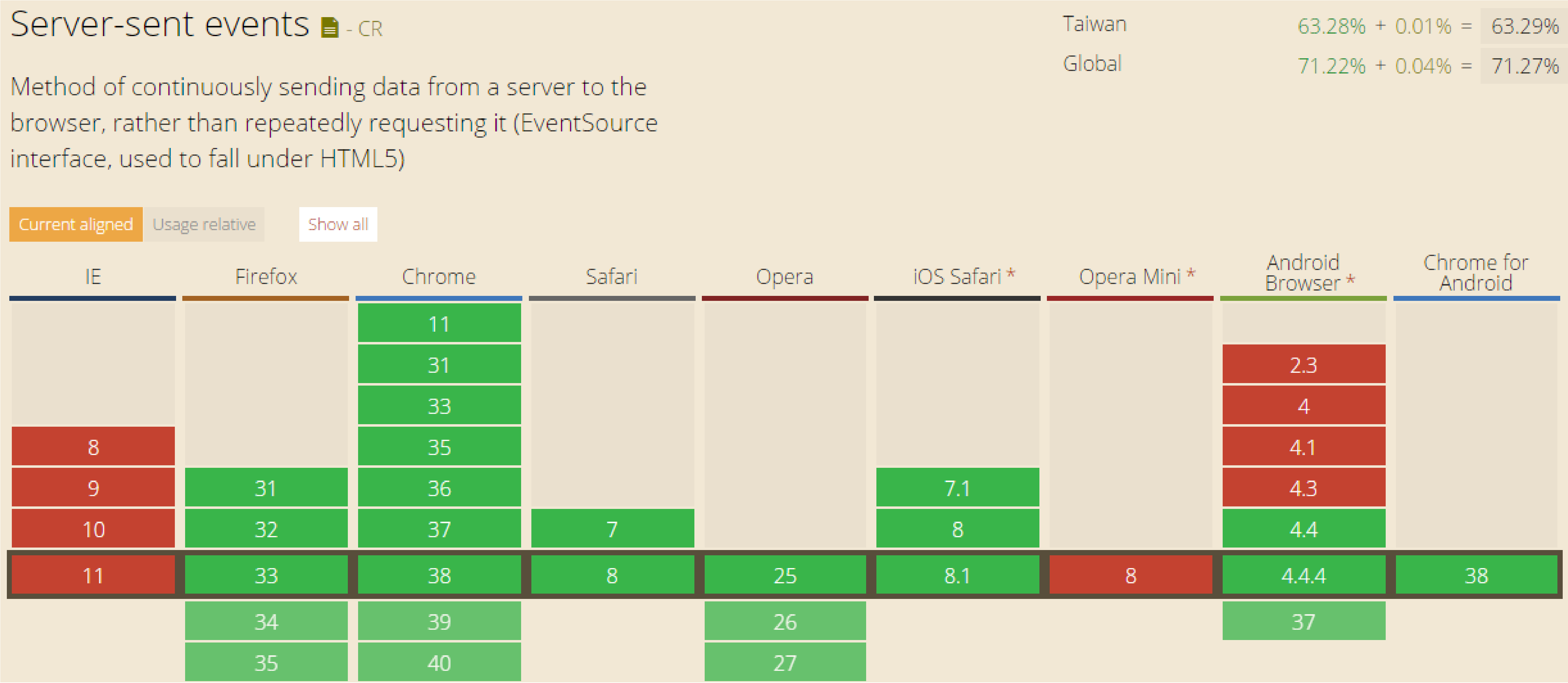
Taiwan	72.6%	+ 1.06%	=	73.66%
unprefixed:	72.6%	+ 1.06%	=	73.66%
Global	79.3%	+ 2.1%	=	81.4%
unprefixed:	79.3%	+ 1.96%	=	81.26%

Current aligned Usage relative Show all

IE	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
		11						
		31					2.3	
		33					4	
8		35					4.1	
9	31	36			7.1		4.3	
10	32	37	7		8		4.4	
11	33	38	8	25	8.1	8	4.4.4	38
	34	39		26			37	
	35	40		27				

Android 5.0

SSE Support list



Server-sent events - CR

Method of continuously sending data from a server to the browser, rather than repeatedly requesting it (EventSource interface, used to fall under HTML5)

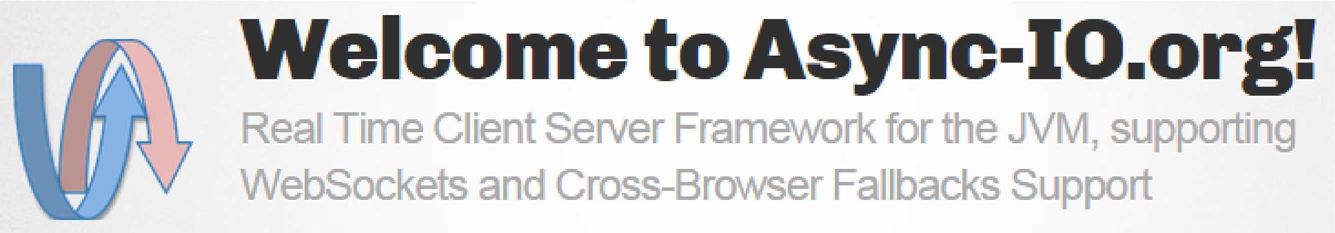
Taiwan	63.28%	+ 0.01%	=	63.29%
Global	71.22%	+ 0.04%	=	71.27%

IE	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
		🟩 11						
		🟩 31					🟥 2.3	
		🟩 33					🟥 4	
🟥 8		🟩 35					🟥 4.1	
🟥 9	🟩 31	🟩 36			🟩 7.1		🟥 4.3	
🟥 10	🟩 32	🟩 37	🟩 7		🟩 8		🟩 4.4	
🟥 11	🟩 33	🟩 38	🟩 8	🟩 25	🟩 8.1	🟥 8	🟩 4.4.4	🟩 38
	🟩 34	🟩 39		🟩 26			🟩 37	
	🟩 35	🟩 40		🟩 27				

WebSocket Fallbacks Support Solutions

- SockJS by Marek Majkowski
- socket.io by Guillermo Rauch
  - Abstracts API and adds features
- kaazing websocket gateway
  - Commercial product. pure polyfill.
- web-socket-js
  - Supports CORS fallback
- atmosphere jQuery plugin (Async-IO)
  - Fallback to comet long-polling
- Graceful WebSocket jQuery plugin by David Lindkvist
  - Fallback to comet long-polling
- Portal by Donghwan Kim
  - Server agnostic and supports Sharing connection, WebSocket, Server-Sent Events, Streaming and Long polling.
- DataChannel polyfill by Jesús Leganés Combarro "Piranna"
  - Add support for WebRTC DataChannels using a WebSockets proxy server as backend

socket.io



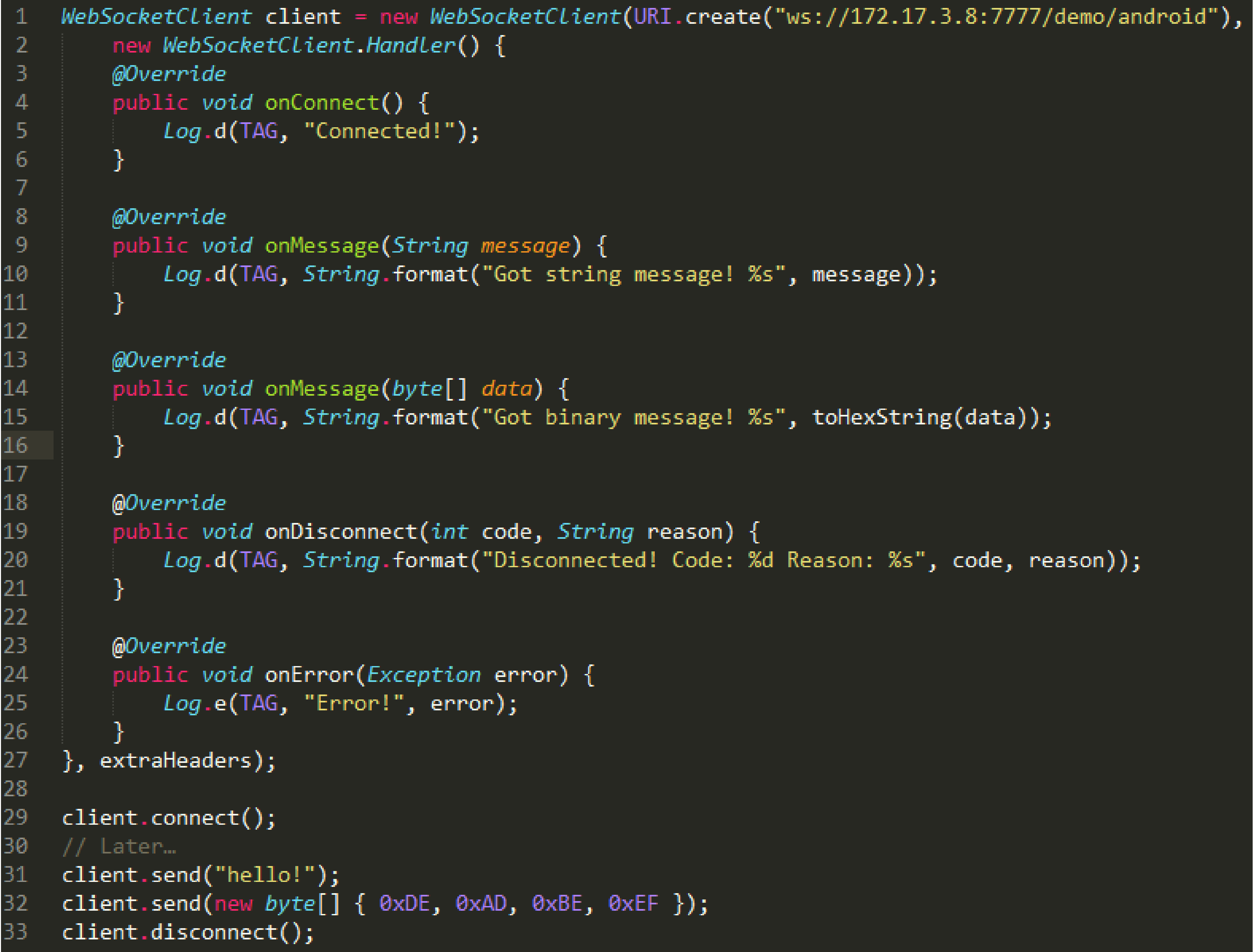
Welcome to Async-IO.org!

Real Time Client Server Framework for the JVM, supporting WebSockets and Cross-Browser Fallbacks Support

Android Client Solutions

1. Java-WebSocket : https://github.com/TooTallNate/Java-WebSocket
2. AndroidAsync : https://github.com/koush/AndroidAsync
3. codebutler/android-websockets : https://github.com/codebutler/android-websockets
4. moko365/android-browser-websocket : https://github.com/moko365/android-browser-websocket (Enable WebView to support WebSocket client connection.)
5. Autobahn|Android : http://autobahn.ws/
6. jWebSocket the open source sol : http://jwebsocket.org/documentation/installation-guide/android-client

Android Client Example



```java
WebSocketClient client = new WebSocketClient(URI.create("ws://172.17.3.8:7777/demo/android"),
    new WebSocketClient.Handler() {
    @Override
    public void onConnect() {
        Log.d(TAG, "Connected!");
    }

    @Override
    public void onMessage(String message) {
        Log.d(TAG, String.format("Got string message! %s", message));
    }

    @Override
    public void onMessage(byte[] data) {
        Log.d(TAG, String.format("Got binary message! %s", toHexString(data));
    }

    @Override
    public void onDisconnect(int code, String reason) {
        Log.d(TAG, String.format("Disconnected! Code: %d Reason: %s", code, reason));
    }

    @Override
    public void onError(Exception error) {
        Log.e(TAG, "Error!", error);
    }
}, extraHeaders);

client.connect();
// Later…
client.send("hello!");
client.send(new byte[] { 0xDE, 0xAD, 0xBE, 0xEF });
client.disconnect();
```

Where Can You Use Full-Duplex WebSocket?

Interactive

- Game apps
- Instant Messaging
- Collaborative editing

Real-Time Control

- Monitoring apps
- Controlling apps

Dynamic Data Update

- Social networking apps (Activity feeds)
- Financial apps

Demos

Interactive

- Game apps - http://browserquest.mozilla.org/
- Instant Messaging - http://shunjikonishi.github.io/room-sandbox/sample/chat.html
- Collaborative editing - http://shunjikonishi.github.io/room-sandbox/sample/canvas.html

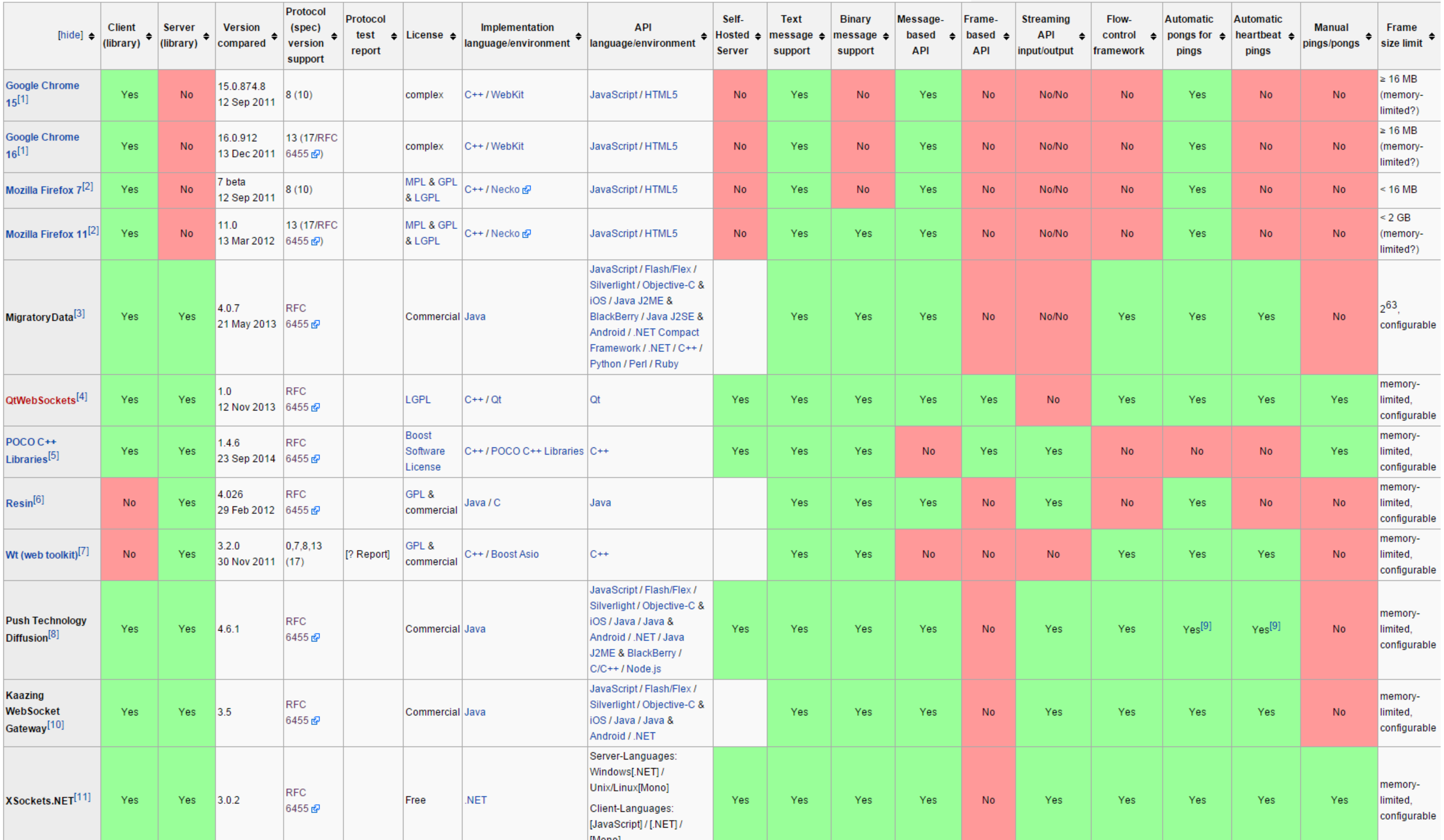
Real-Time Control

- Monitoring apps - http://live.embeda.com.tw:8080/probe.html
- Controlling apps - http://172.17.3.7:7777/demo/PC

Dynamic Data Update

- Social networking apps (Activity feeds)
- Financial apps - http://demo.kaazing.com/livefeed/

Comparison of WebSocket implementations



[hide]	Client (library)	Server (library)	Version compared	Protocol (spec) version support	Protocol test report	License	Implementation language/environment	API language/environment	Self-Hosted Server	Text message support	Binary message support	Message-based API	Frame-based API	Streaming API input/output	Flow-control framework	Automatic pongs for pings	Automatic heartbeat pings	Manual pings/pongs	Frame size limit
Google Chrome 15[1]	Yes	No	15.0.874.8 12 Sep 2011	8 (10)		complex	C++ / WebKit	JavaScript / HTML5	No	Yes	No	Yes	No	No/No	No	Yes	No	No	≥ 16 MB (memory-limited?)
Google Chrome 16[1]	Yes	No	16.0.912 13 Dec 2011	13 (17/RFC 6455)		complex	C++ / WebKit	JavaScript / HTML5	No	Yes	No	Yes	No	No/No	No	Yes	No	No	≥ 16 MB (memory-limited?)
Mozilla Firefox 7[2]	Yes	No	7 beta 12 Sep 2011	8 (10)		MPL & GPL & LGPL	C++ / Necko	JavaScript / HTML5	No	Yes	No	Yes	No	No/No	No	Yes	No	No	< 16 MB
Mozilla Firefox 11[2]	Yes	No	11.0 13 Mar 2012	13 (17/RFC 6455)		MPL & GPL & LGPL	C++ / Necko	JavaScript / HTML5	No	Yes	Yes	Yes	No	No/No	No	Yes	No	No	< 2 GB (memory-limited?)
MigratoryData[3]	Yes	Yes	4.0.7 21 May 2013	RFC 6455		Commercial	Java	JavaScript / Flash/Flex / Silverlight / Objective-C & iOS / Java J2ME & BlackBerry / Java J2SE & Android / .NET Compact Framework / .NET / C++ / Python / Perl / Ruby		Yes	Yes	Yes	No	No/No	Yes	Yes	Yes	No	2^{63}, configurable
QtWebSockets[4]	Yes	Yes	1.0 12 Nov 2013	RFC 6455		LGPL	C++ / Qt	Qt	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	memory-limited, configurable
POCO C++ Libraries[5]	Yes	Yes	1.4.6 23 Sep 2014	RFC 6455		Boost Software License	C++ / POCO C++ Libraries	C++	Yes	Yes	Yes	No	Yes	Yes	No	No	No	Yes	memory-limited, configurable
Resin[6]	No	Yes	4.026 29 Feb 2012	RFC 6455		GPL & commercial	Java / C	Java		Yes	Yes	Yes	No	Yes	No	Yes	No	No	memory-limited, configurable
Wt (web toolkit)[7]	No	Yes	3.2.0 30 Nov 2011	0,7,8,13 (17)	[? Report]	GPL & commercial	C++ / Boost Asio	C++		Yes	Yes	No	No	No	Yes	Yes	Yes	No	memory-limited, configurable
Push Technology Diffusion[8]	Yes	Yes	4.6.1	RFC 6455		Commercial	Java	JavaScript / Flash/Flex / Silverlight / Objective-C & iOS / Java / Java & Android / .NET / Java J2ME & BlackBerry / C/C++ / Node.js	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes[9]	Yes[9]	No	memory-limited, configurable
Kaazing WebSocket Gateway[10]	Yes	Yes	3.5	RFC 6455		Commercial	Java	JavaScript / Flash/Flex / Silverlight / Objective-C & iOS / Java & Android / .NET		Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	No	memory-limited, configurable
XSockets.NET[11]	Yes	Yes	3.0.2	RFC 6455		Free	.NET	Server-Languages: Windows[.NET] / Unix/Linux[Mono] Client-Languages: [JavaScript] / [.NET] / [Mono]	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	memory-limited, configurable

A few useful links

- https://tools.ietf.org/html/rfc6455 (official doc)
- http://www.html5rocks.com/en/tutorials/websockets/basics/ (basic tutorial)
- http://www.websocket.org (the echo server folks)
- http://www.slideshare.net/peterlubbers/websocketsthe-new-network-stack-by-peter-lubbers-and-frank-greco
- You can test websocket that can work on your browser via websocketstest.com
- http://codepen.io/matt-west/pen/tHlBb - Online Codepen