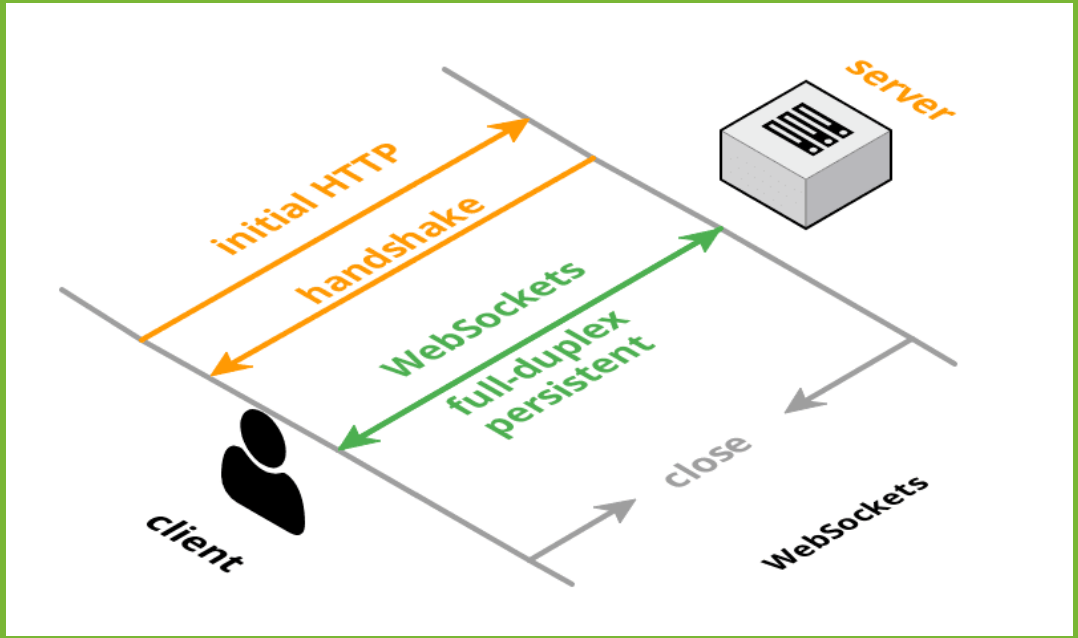
Web Sockets

Peer-to-peer connections for Real-time Web



Feiner/Ulm/Schwab

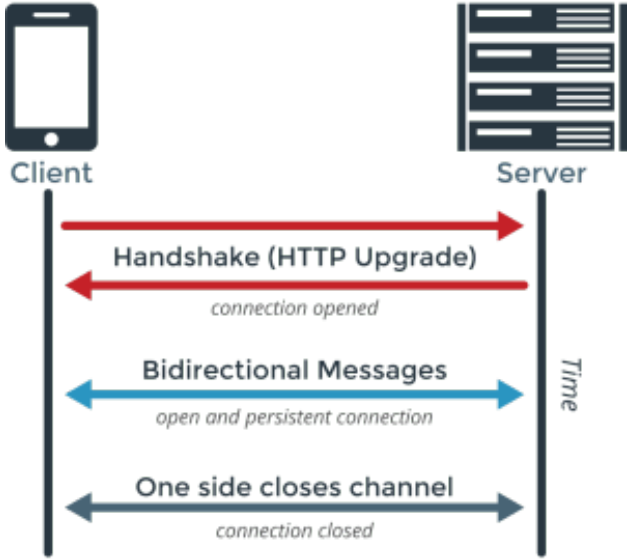
https://www.ably.io/concepts/websockets

WebSocket for "Real-time Web"

- "Classic" HTTP is uni-directional with Request-Response

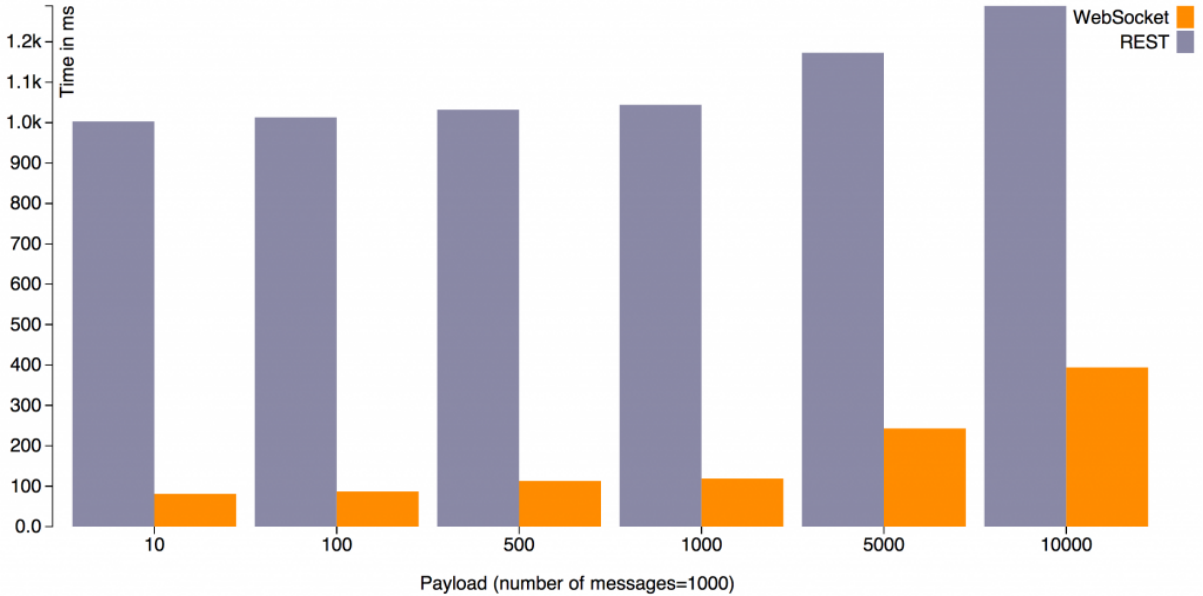
- WebSocket enables a **bi-directional communication** between WebApp and Server (compare WebRTC data channel)
- Mobile App initially connects to Server. **Reuse** of **existing connection, protocol upgrade.**

- Server is able to "**push**" information to Client



http://saurabh29july.blog/understanding-websockets-using-java-embedded-jetty-server-a-simple-html-javascript-client/

Payload comparison REST vs. WebSocket



http://blog.arungupta.me/rest-vs-websocket-comparison-benchmarks/

WebSocket readyState attribute

Identifies current connection state

(0) **CONNECTING**

Client establishes currently a connection to Server

(1) **OPEN**

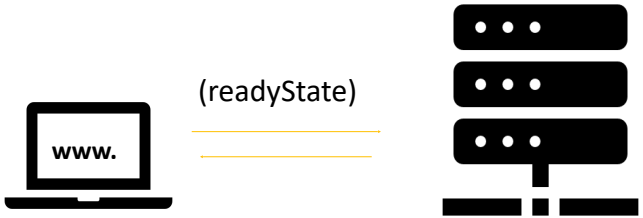
The Web Socket communication is open

(2) **CLOSING**

Current connection is currently closing

(3) **CLOSED**

The connection is closed.



WebSocket Events and Event Handlers

Each events will be bind with an Event Handler

open / onopen

Connection is established

message / onmessage

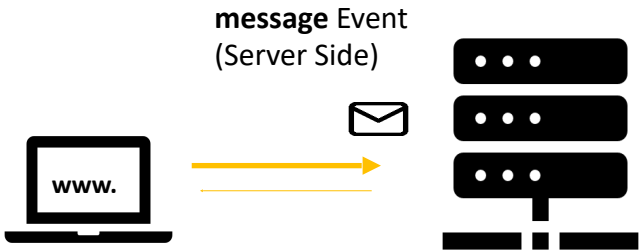
Client receives data from server

error / onerror

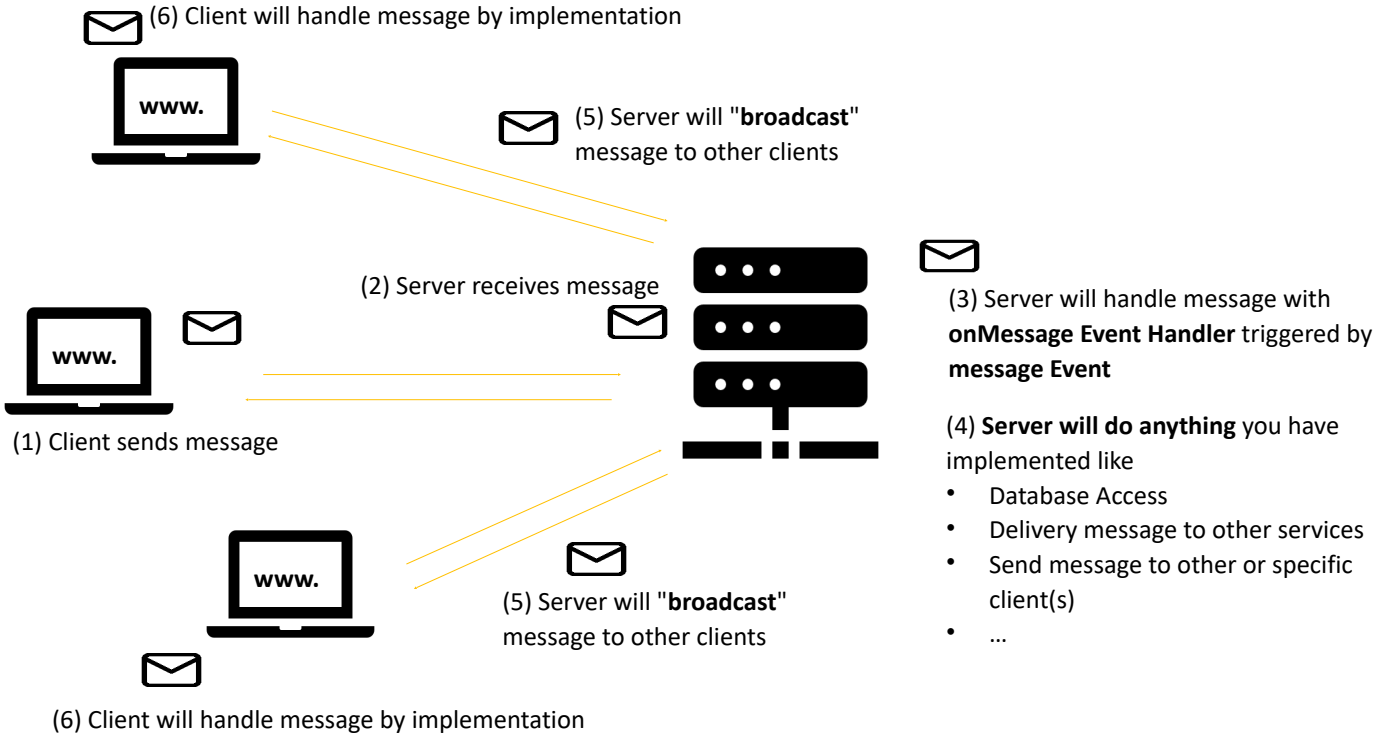
Occurs on any error in communication

close / onclose

Connection is closed

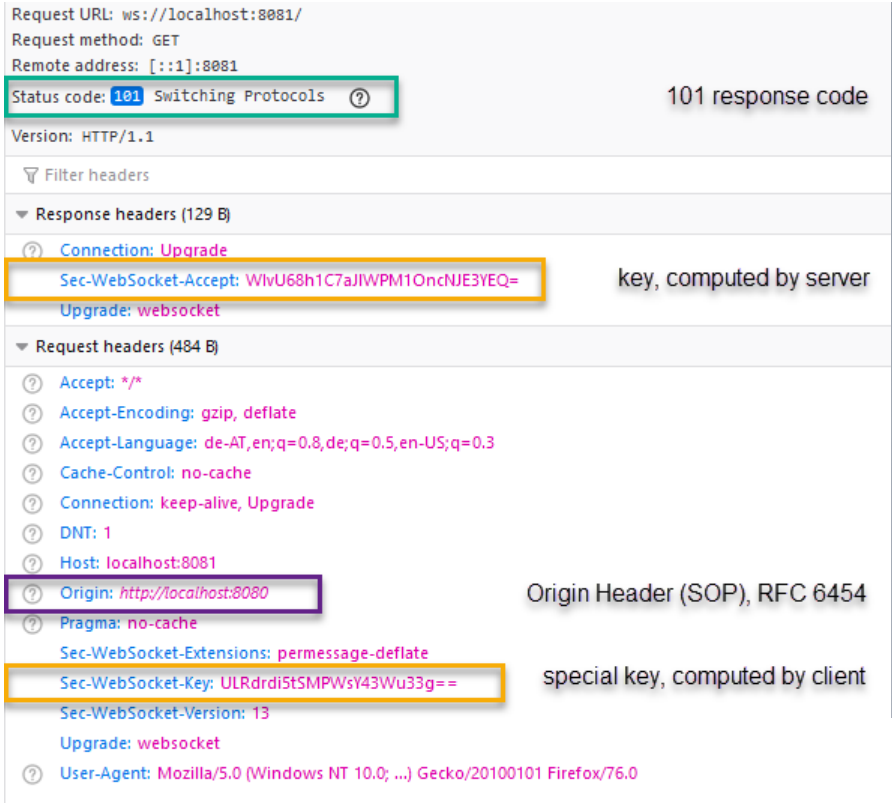


possible Infrastructure and Design



Handshake

- HTTP based
- Client computes **Key** and sends upgrade request to server
- Server calculates **Accept** key and sends back 101 response



https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Sec-WebSocket-Accept

https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/101

https://tools.ietf.org/html/rfc6454

Message

- Event-Listener
  - onmessage
- Event-Object*
  - message
- Data**
  - message.data
- Typically serialised JSON
  - String
  - ArrayBuffer
  - Blob

```
const connection = new WebSocket('wss://example.com:1234');

connection.onmessage = (message) => {
  const msg = JSON.parse(message.data);
  console.log(msg);
};
```

Browser side example

* Depends on WebSocket implementation!
** Depending on the implementation, the event object can already be the pure data stream

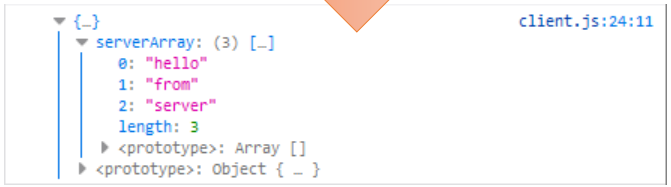
https://developer.mozilla.org/en-US/docs/Web/API/WebSocket/send

Message

```
connection.onmessage = (message) => {
  console.log(message);
};
```

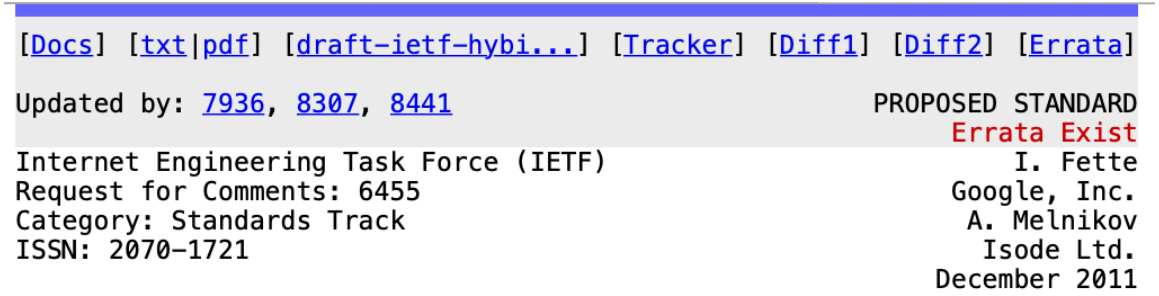
https://www.w3.org/TR/websockets/

```
connection.onmessage = (message) => {
  const msg = JSON.parse(message.data);
  console.log(msg);
};
```



https://github.com/Rigbin/ws-example

Selected references



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

Updated by: 7936, 8307, 8441 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).

https://tools.ietf.org/html/rfc6455