WIKIPEDIA

WebSocket

WebSocket is a computer communications protocol, providing full-duplex communication channels over a single TCP connection. The WebSocket protocol was standardized by the IETF as RFC 6455 in 2011, and the WebSocket API in Web IDL is being standardized by the W3C.

WebSocket is a different protocol from HTTP. Both protocols are located at layer 7 in the OSI model and depend on TCP at layer 4. Although they are different, RFC 6455 states that WebSocket "is designed to work over HTTP ports 80 and 443 as well as to support HTTP proxies and intermediaries" thus making it compatible with the HTTP protocol. To achieve compatibility, the WebSocket handshake uses the HTTP Upgrade header[1] to change from the HTTP protocol to the WebSocket protocol.

The WebSocket protocol enables interaction between a web browser (or other client application) and a web server with lower overheads, facilitating real-time data transfer from and to the server. This is made possible by providing a standardized way for the server to send content to the client without being first requested by the client, and allowing messages to be passed back and forth while keeping the connection open. In this way, a two-way ongoing conversation can take place between the client and the server. The communications are done over TCP port number 80 (or 443 in the case of TLS-encrypted connections), which is of benefit for those environments which block non-web Internet connections using a firewall. Similar two-way browser-server communications have been achieved in non-standardized ways using stopgap technologies such as Comet.

Most browsers support the protocol, including Google Chrome, Microsoft Edge, Internet Explorer, Firefox, Safari and Opera.

Contents

Overview
History
Browser implementation
Protocol handshake
Proxy traversal
See also
Notes
References
External links

Overview

Unlike HTTP, WebSocket provides full-duplex communication.[2][3] Additionally, WebSocket enables streams of messages on top of TCP. TCP alone deals with streams of bytes with no inherent concept of a message. Before WebSocket, port 80 full-duplex communication was attainable using Comet channels; however, Comet implementation is nontrivial, and due to the TCP handshake and HTTP header overhead, it is inefficient for small messages. The WebSocket protocol aims to solve these problems without compromising security assumptions of the web.

The WebSocket protocol specification defines `ws` (WebSocket) and `wss` (WebSocket Secure) as two new uniform resource identifier (URI) schemes[4] that are used for unencrypted and encrypted connections, respectively. Apart from the scheme name and fragment (`#` is not supported), the rest of the URI components are defined to use URI generic syntax.[5]

Using browser developer tools, developers can inspect the WebSocket handshake as well as the WebSocket frames.[6]

History

WebSocket was first referenced as TCPConnection in the HTML5 specification, as a placeholder for a TCP-based socket API.[7] In June 2008, a series of discussions were led by Michael Carter that resulted in the first version of the protocol known as WebSocket.[8]

The name "WebSocket" was coined by Ian Hickson and Michael Carter shortly thereafter through collaboration on the #whatwg IRC chat room,[9] and subsequently authored for inclusion in the HTML5 specification by Ian Hickson, and announced on the cometdaily blog by Michael Carter.[10] In December 2009, Google Chrome 4 was the first browser to ship full support for the standard, with WebSocket enabled by default.[11] Development of the WebSocket protocol was subsequently moved from the W3C and WHATWG group to the IETF in February 2010, and authored for two revisions under Ian Hickson.[12]

After the protocol was shipped and enabled by default in multiple browsers, the RFC was finalized under Ian Fette in December 2011.[13]

Browser implementation

A secure version of the WebSocket protocol is implemented in Firefox 6,[14] Safari 6, Google Chrome 14,[15] Opera 12.10 and Internet Explorer 10.[16] A detailed protocol test suite report[17] lists the conformance of those browsers to specific protocol aspects.

An older, less secure version of the protocol was implemented in Opera 11 and Safari 5, as well as the mobile version of Safari in iOS 4.2.[18] The BlackBerry Browser in OS7 implements WebSockets.[19] Because of vulnerabilities, it was disabled in Firefox 4 and 5,[20] and Opera 11.[21]

Implementation status

Protocol, version	Draft date	Internet Explorer	Firefox[22] (PC)	Firefox (Android)	Chrome (PC, Mobile)	Safari (Mac, iOS)	Opera (PC, Mobile)	Android Browser
hixie-75 (https://tools.ietf.org/html/draft-hixie-thewebsocketprotocol-75)	February 4, 2010				4	5.0.0		
hixie-76 (https://tools.ietf.org/html/draft-hixie-thewebsocketprotocol-76) hybi-00 (https://tools.ietf.org/html/draft-ietf-hybi-thewebsocketprotocol-00)	May 6, 2010 May 23, 2010		4.0 (disabled)		6	5.0.1	11.00 (disabled)	
hybi-07 (https://tools.ietf.org/html/draft-ietf-hybi-thewebsocketprotocol-07), v7	April 22, 2011		6[23][a]					
hybi-10 (https://tools.ietf.org/html/draft-ietf-hybi-thewebsocketprotocol-10), v8	July 11, 2011		7[25][a]	7	14[26]			
RFC 6455 (https://tools.ietf.org/html/rfc6455), v13	December, 2011	10[27]	11	11	16[28]	6	12.10[29]	4.4

Protocol handshake

To establish a WebSocket connection, the client sends a WebSocket handshake request, for which the server returns a WebSocket handshake response, as shown in the example below.[30]

Client request (just like in HTTP, each line ends with `\r\n` and there must be an extra blank line at the end):

```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: x3JJHMbDL1EzLkh9GBhXDw==
Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 13
Origin: http://example.com
```

Server response:

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: HSmrc0sMlYUkAGmm5OPpG2HaGWk=
Sec-WebSocket-Protocol: chat
```

The handshake starts with an HTTP request/response, allowing servers to handle HTTP connections as well as WebSocket connections on the same port. Once the connection is established, communication switches to a bidirectional binary protocol which does not conform to the HTTP protocol.

In addition to `Upgrade` headers, the client sends a `Sec-WebSocket-Key` header containing base64-encoded random bytes, and the server replies with a hash of the key in the `Sec-WebSocket-Accept` header. This is intended to prevent a caching proxy from re-sending a previous WebSocket conversation,[31] and does not provide any authentication, privacy, or integrity. The hashing function appends the fixed string `258EAFA5-E914-47DA-95CA-C5AB0DC85B11` (a GUID) to the value from `Sec-WebSocket-Key` header (which is not decoded from base64), applies the SHA-1 hashing function, and encodes the result using base64.[32]

Once the connection is established, the client and server can send WebSocket data or text frames back and forth in full-duplex mode. The data is minimally framed, with a small header followed by payload.[33] WebSocket transmissions are described as "messages", where a single message can optionally be split across several data frames. This can allow for sending of messages where initial data is available but the complete length of the message is unknown (it sends one data frame after another until the end is reached and marked with the FIN bit). With extensions to the protocol, this can also be used for multiplexing several streams simultaneously (for instance to avoid monopolizing use of a socket for a single large payload).[34]

It is important (from a security perspective) to validate the "Origin" header during the connection establishment process on the server side (against the expected origins) to avoid Cross-Site WebSocket Hijacking attacks, which might be possible when the connection is authenticated with Cookies or HTTP authentication. It is better to use tokens or similar protection mechanisms to authenticate the WebSocket connection when sensitive (private) data is being transferred over the WebSocket.[35]

Proxy traversal

WebSocket protocol client implementations try to detect if the user agent is configured to use a proxy when connecting to destination host and port and, if it is, uses HTTP CONNECT method to set up a persistent tunnel.

While the WebSocket protocol itself is unaware of proxy servers and firewalls, it features an HTTP-compatible handshake thus allowing HTTP servers to share their default HTTP and HTTPS ports (80 and 443) with a WebSocket gateway or server. The WebSocket protocol defines a ws:// and wss:// prefix to indicate a WebSocket and a WebSocket Secure connection, respectively. Both schemes use an HTTP upgrade mechanism to upgrade to the WebSocket protocol. Some proxy servers are transparent and work fine with WebSocket; others will prevent WebSocket from working correctly, causing the connection to fail. In some cases, additional proxy server configuration may be required, and certain proxy servers may need to be upgraded to support WebSocket.

If unencrypted WebSocket traffic flows through an explicit or a transparent proxy server without WebSockets support, the connection will likely fail.[36]

If an encrypted WebSocket connection is used, then the use of Transport Layer Security (TLS) in the WebSocket Secure connection ensures that an HTTP CONNECT command is issued when the browser is configured to use an explicit proxy server. This sets up a tunnel, which provides low-level end-to-end TCP communication through the HTTP proxy, between the WebSocket Secure client and the WebSocket server. In the case of transparent proxy servers, the browser is

unaware of the proxy server, so no HTTP CONNECT is sent. However, since the wire traffic is encrypted, intermediate transparent proxy servers may simply allow the encrypted traffic through, so there is a much better chance that the WebSocket connection will succeed if WebSocket Secure is used. Using encryption is not free of resource cost, but often provides the highest success rate since it would be travelling through a secure tunnel.

A mid-2010 draft (version hixie-76) broke compatibility with reverse proxies and gateways by including eight bytes of key data after the headers, but not advertising that data in a `Content-Length: 8` header.[37] This data was not forwarded by all intermediates, which could lead to protocol failure. More recent drafts (e.g., hybi-09[38]) put the key data in a `Sec-WebSocket-Key` header, solving this problem.

See also

- BOSH
- Comparison of WebSocket implementations
- Network socket
- Push technology
- Server-sent events
- XMLHttpRequest
- HTTP/2
- Internet protocol suite
- WebRTC

Notes

a. Gecko-based browsers versions 6–10 implement the WebSocket object as "MozWebSocket",[24] requiring extra code to integrate with existing WebSocket-enabled code.

References

1. Ian Fette; Alexey Melnikov (December 2011). "Relationship to TCP and HTTP" (https://tools.ietf.org/html/rfc6455#section-1.7). *RFC 6455 The WebSocket Protocol* (https://tools.ietf.org/html/rfc6455). IETF. sec. 1.7. doi:10.17487/RFC6455 (https://doi.org/10.17487%2FRFC6455). RFC 6455.
2. "Glossary:WebSockets" (https://developer.mozilla.org/en-US/docs/Glossary/WebSockets). Mozilla Developer Network. 2015.
3. HTML5 WebSocket: A Quantum Leap in Scalability for the Web (http://www.websocket.org/quantum.html)
4. Graham Klyne, ed. (2011-11-14). "IANA Uniform Resource Identifer (URI) Schemes" (https://www.iana.org/assignments/uri-schemes.html). Internet Assigned Numbers Authority. Retrieved 2011-12-10.
5. Ian Fette; Alexey Melnikov (December 2011). "WebSocket URIs" (https://tools.ietf.org/html/rfc6455#section-3). *RFC 6455 The WebSocket Protocol* (https://tools.ietf.org/html/rfc6455). IETF. sec. 3. doi:10.17487/RFC6455 (https://doi.org/10.17487%2FRFC6455). RFC 6455.
6. Wang, Vanessa; Salim, Frank; Moskovits, Peter (February 2013). "APPENDIX A: WebSocket Frame Inspection with Google Chrome Developer Tools" (http://my.safaribooksonline.com/book/-/9781430247401/appendix-a-inspecting-websocket-traffic/sec1_xhtml). *The Definitive Guide to HTML5 WebSocket*. Apress. ISBN 978-1-4302-4740-1. Retrieved 7 April 2013.
7. "HTML 5" (https://www.w3.org/TR/2008/WD-html5-20080610/comms.html#tcp-connections). *www.w3.org*. Retrieved 2016-04-17.

8. "[whatwg] TCPConnection feedback from Michael Carter on 2008-06-18 (whatwg.org from June 2008)" (https://lists.w3.org/Archives/Public/public-whatwg-archive/2008Jun/0165.html). *lists.w3.org*. Retrieved 2016-04-17.
9. "IRC logs: freenode / #whatwg / 20080618" (http://krijnhoetmer.nl/irc-logs/whatwg/20080618#l-1145). *krijnhoetmer.nl*. Retrieved 2016-04-18.
10. "Comet Daily » Blog Archive » Independence Day: HTML5 WebSocket Liberates Comet From Hacks" (http://cometdaily.com/2008/07/04/html5-websocket/). Retrieved 2016-04-17.
11. "Web Sockets Now Available In Google Chrome" (https://blog.chromium.org/2009/12/web-sockets-now-available-in-google.html). *Chromium Blog*. Retrieved 2016-04-17.
12. <ian@hixie.ch>, Ian Hickson. "The WebSocket protocol" (https://tools.ietf.org/html/draft-hixie-thewebsocketprotocol-75). *tools.ietf.org*. Retrieved 2016-04-17.
13. <ian@hixie.ch>, Ian Hickson. "The WebSocket protocol" (https://tools.ietf.org/html/rfc6455). *tools.ietf.org*. Retrieved 2016-04-17.
14. Dirkjan Ochtman (May 27, 2011). "WebSocket enabled in Firefox 6" (https://developer.mozilla.org/en/WebSockets). *Mozilla.org*. Retrieved 2011-06-30.
15. "Chromium Web Platform Status" (https://www.chromium.org/developers/web-platform-status). Retrieved 2011-08-03.
16. "WebSockets (Windows)" (https://msdn.microsoft.com/en-us/library/ie/hh673567(v=vs.85).aspx). Microsoft. 2012-09-28. Retrieved 2012-11-07.
17. "WebSockets Protocol Test Report" (http://autobahn.ws/testsuite/reports/clients/index.html). Tavendo.de. 2011-10-27. Retrieved 2011-12-10.
18. Katie Marsal (November 23, 2010). "Apple adds accelerometer, WebSockets support to Safari in iOS 4.2" (http://www.appleinsider.com/articles/10/11/23/apple_adds_accelerometer_websockets_support_to_safari_in_ios_4_2.html). *AppleInsider.com*. Retrieved 2011-05-09.
19. "Web Sockets API" (https://web.archive.org/web/20110610191150/http://docs.blackberry.com/en/developers/deliverables/29271/Web_Sockets_support_1582781_11.jsp). BlackBerry. Archived from the original (http://docs.blackberry.com/en/developers/deliverables/29271/Web_Sockets_support_1582781_11.jsp) on June 10, 2011. Retrieved 8 July 2011.
20. Chris Heilmann (December 8, 2010). "WebSocket disabled in Firefox 4" (https://hacks.mozilla.org/2010/12/websockets-disabled-in-firefox-4/). *Hacks.Mozilla.org*. Retrieved 2011-05-09.
21. Aleksander Aas (December 10, 2010). "Regarding WebSocket" (https://web.archive.org/web/20101215010748/http://my.opera.com/chooseopera/blog/2010/12/10/regarding-websocket). *My Opera Blog*. Archived from the original (http://my.opera.com/chooseopera/blog/2010/12/10/regarding-websocket) on 2010-12-15. Retrieved 2011-05-09.
22. "WebSockets (support in Firefox)" (https://developer.mozilla.org/en/WebSockets). *developer.mozilla.org*. Mozilla Foundation. 2011-09-30. Retrieved 2011-12-10.
23. "Bug 640003 - WebSockets - upgrade to ietf-06" (https://bugzilla.mozilla.org/show_bug.cgi?id=640003). Mozilla Foundation. 2011-03-08. Retrieved 2011-12-10.
24. "WebSockets - MDN" (https://developer.mozilla.org/en/WebSockets). *developer.mozilla.org*. Mozilla Foundation. 2011-09-30. Retrieved 2011-12-10.
25. "Bug 640003 - WebSockets - upgrade to ietf-07(comment 91)" (https://bugzilla.mozilla.org/show_bug.cgi?id=640003#c91). Mozilla Foundation. 2011-07-22.
26. "Chromium bug 64470" (https://code.google.com/p/chromium/issues/detail?id=64470). *code.google.com*. 2010-11-25. Retrieved 2011-12-10.
27. "WebSockets in Windows Consumer Preview" (https://blogs.msdn.com/b/ie/archive/2012/03/19/websockets-in-windows-consumer-preview.aspx). *IE Engineering Team*. Microsoft. 2012-03-19. Retrieved 2012-07-23.
28. "WebKit Changeset 97247: WebSocket: Update WebSocket protocol to hybi-17" (https://trac.webkit.org/changeset/97249). *trac.webkit.org*. Retrieved 2011-12-10.

29. "A hot Opera 12.50 summer-time snapshot" (https://web.archive.org/web/20120805234006/http://my.opera.com/ODIN/blog/2012/08/03/a-hot-opera-12-50-summer-time-snapshot). Opera Developer News. 2012-08-03. Archived from the original (http://my.opera.com/ODIN/blog/2012/08/03/a-hot-opera-12-50-summer-time-snapshot) on 2012-08-05. Retrieved 2012-08-03.
30. Ian Fette; Alexey Melnikov (December 2011). "Protocol Overview" (https://tools.ietf.org/html/rfc6455#section-1.2). *RFC 6455 The WebSocket Protocol* (https://tools.ietf.org/html/rfc6455). IETF. sec. 1.2. doi:10.17487/RFC6455 (https://doi.org/10.17487%2FRFC6455). RFC 6455.
31. "Main Goal of WebSocket protocol" (https://trac.tools.ietf.org/wg/hybi/trac/wiki/FAQ). IETF. Retrieved 25 July 2015. "The computation [...] is meant to prevent a caching intermediary from providing a WS-client with a cached WS-server reply without actual interaction with the WS-server."
32. Ian Fette; Alexey Melnikov (December 2011). "Opening Handshake" (https://tools.ietf.org/html/rfc6455#section-1.3). *RFC 6455 The WebSocket Protocol* (https://tools.ietf.org/html/rfc6455). IETF. p. 8. sec. 1.3. doi:10.17487/RFC6455 (https://doi.org/10.17487%2FRFC6455). RFC 6455.
33. Ian Fette; Alexey Melnikov (December 2011). "Base Framing Protocol" (https://tools.ietf.org/html/rfc6455#section-5.2). *RFC 6455 The WebSocket Protocol* (https://tools.ietf.org/html/rfc6455). IETF. sec. 5.2. doi:10.17487/RFC6455 (https://doi.org/10.17487%2FRFC6455). RFC 6455.
34. John A. Tamplin; Takeshi Yoshino (2013). *A Multiplexing Extension for WebSockets* (https://tools.ietf.org/html/draft-ietf-hybi-websocket-multiplexing). IETF. I-D draft-ietf-hybi-websocket-multiplexing.
35. Christian Schneider (August 31, 2013). "Cross-Site WebSocket Hijacking (CSWSH)" (https://www.christian-schneider.net/CrossSiteWebSocketHijacking.html#main). *Web Application Security Blog*.
36. Peter Lubbers (March 16, 2010). "How Web Sockets Interact With Proxy Servers" (http://www.infoq.com/articles/Web-Sockets-Proxy-Servers). *Infoq.com*. C4Media Inc. Retrieved 2011-12-10.
37. Willy Tarreau (2010-07-06). "WebSocket -76 is incompatible with HTTP reverse proxies" (https://www.ietf.org/mail-archive/web/hybi/current/msg02149.html). *ietf.org* (email). Internet Engineering Task Force. Retrieved 2011-12-10.
38. Ian Fette (June 13, 2011). "Sec-WebSocket-Key" (https://tools.ietf.org/html/draft-ietf-hybi-thewebsocketprotocol-09#section-11.4#section-11.4). *The WebSocket protocol, draft hybi-09* (https://tools.ietf.org/html/draft-ietf-hybi-thewebsocketprotocol-09#section-11.4). sec. 11.4. Retrieved June 15, 2011.

External links

- IETF Hypertext-Bidirectional (HyBi) working group (https://datatracker.ietf.org/wg/hybi/charter/)
  - The WebSocket protocol (https://tools.ietf.org/html/rfc6455) - Proposed Standard published by the IETF HyBi Working Group
  - The WebSocket protocol (https://tools.ietf.org/html/draft-ietf-hybi-thewebsocketprotocol) - Internet-Draft published by the IETF HyBi Working Group
  - The WebSocket protocol (https://tools.ietf.org/html/draft-hixie-thewebsocketprotocol-76) - Original protocol proposal by Ian Hickson
- The WebSocket API (https://dev.w3.org/html5/websockets/) - W3C Working Draft specification of the API
- The WebSocket API (http://www.w3.org/TR/websockets/) - W3C Candidate Recommendation specification of the API
- WebSocket.org (https://www.websocket.org/) WebSocket demos, loopback tests, general information and community