Building Real-Time Web Applications using WebSockets - Part 1

**Contents:**

1. Introduction
2. What is WebSockets?
3. Comparison with AJAX (or XHR)
4. How it works?
5. WebSocket API and Socket.IO
6. Browser Compatibility
7. More reads

**Introduction:**

The web was built around the idea that a client’s job is to request data from a server, and a server’s job is to fulfill those requests. This method of making requests by the client to the server remains unchallenged for a long time and is known as polling.

For building real-time web applications, we can think of multiple strategies, including periodic polling using AJAX. But still all HTTP communications have to be steered by the client for getting the data refreshed. What we needed was some technology which can enable server pushing data to its all connected clients in the very moment it knows some new data is available and that too without any data request made by the client.

One of the most popular technology in this field has been long-polling, which creates an illusion of server initiated transaction by keeping an HTTP connection initiated by client open until the server has some data to push down to the clients.

The problem with all these solution is that they carry the overhead of HTTP. Every time we make an HTTP request a bunch of headers and cookie data are transferred to the server. What we really need is a way of creating a persistent, low latency connection that can support transactions initiated by either the client or server.

**What is WebSockets?**

WebSockets is an advance technology that provides a persistent, low-latency and bi-directional connection between a client and a server that both parties can use to send data any time.

Communication happens over single TCP socket using ws(unsecure) and wss(secure) protocol and can be used by any client or server application.

**Comparison with AJAX (or XHR):**

1. Ajax technologies require a request made by the client whereas WebSocket clients and Servers can push each other messages any time.

2. XHR is limited by the domain, whereas WebSocket API allows cross-domain messaging with no fuss.

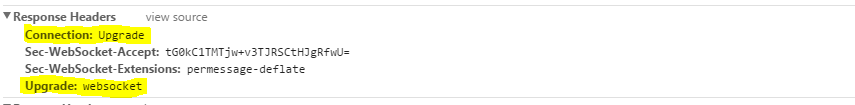
**How it works?**

The client establishes a WebSocket connection through a process known as the WebSocket handshake.

1. The client sending a regular HTTP request to the server. An Upgrade header is included in this request that informs the server that the client wishes to establish a WebSocket connection.



2. If the server supports the WebSocket protocol, it agrees to the upgrade and communicates this through an Upgrade header in the response.



3. Now the handshake is complete, the initial HTTP connection is replaced by a WebSocket connection that uses the same underlying TCP/IP connection. At this point, both parties server and client can start sending data anytime.

**WebSocket API and Socket.IO:**

Socket.IO is a WebSocket API that provide WebSocket event handling, fallback transports and a server side solution, all within one API.

Socket.IO use feature detection to decide if the connection will be established with WebSocket, AJAX long polling, Flash, etc., making creating real-time apps that work everywhere.

It also provide an API for Node.js which looks very much similar to the client side API.

We will discuss more on developing apps using Socket.IO in second part of this article series.

**Browser Compatibility:**

WebSockets are supported in almost all the modern web browsers. The only exceptions being the Opera mini and older version of Android browser. For up-to date information on the list of browsers support, Please check out: [Can I use WebSockets](http://caniuse.com/#feat=websockets).

**More Reads:**

* <http://chimera.labs.oreilly.com/books/1230000000545/ch17.html>
* <http://www.websocket.org/>
* <https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API>