

# Websockets-01 Introduction to Websockets

---

## 1 What is a websocket?

### 1.1 What type of thing is it?

It is a **connection protocol** between a server and a client. Another example of a connection protocol is the HTTP protocol.

### 1.2 What does the protocol look like?

On the server side, one has to **register** the URL to be used by clients to connect to it. The server can handle events such as "**open** – which happens when a client *connects*", "**message** – when client message is received", "**close** – when the connection is closed". The server can also **send** messages to the client.

On the client side, one has to **connect** to the server by using the URL. The client can handle events such as "**open** – which happens when the connection is established", "**message** – when server message is received", "**close** – when the connection is closed". The client can also **send** messages to the server.

### 1.3 What are some special properties of this protocol?

Firstly, this protocol is compatible with HTTP. It uses the same port as HTTP (i.e. 80 for unsecure connections and 443 for secure connections). The client can make both normal HTTP and WS requests on the same **port** to the server! (Do remember that a single client can open many connections with the same server). Each new connection is treated differently based on if it is an HTTP request or a WS request.

Secondly, UNLIKE HTTP, it maintains a **full-duplex** and **persistent** connection. HTTP connections are one-way. That means client sends a request and server responds.

Half-duplex means communication can only be in one direction at a time. This is very similar to how walkie-talkies function. In contrast, Websockets are full-duplex, which means they allow both sending and receiving to occur at the **same time**. This is similar to a phone call in which you can talk while the other person is also talking.



Figure 1 Half-Duplex vs Full-Duplex

HTTP connections are short-lived. Once the server responds, the connection is closed. WEBSOCKET connections remain open until the client or server decides to close the connection.

## 2 Why do we want to use websockets?

**Reason-1:** If you use HTTP, servers have no way to let clients know when some event happens.

For example, maybe some other user just logged in. Maybe someone sent a chat message. Maybe the server is low on resources and is going to be shut down for maintenance. Clients would need to POLL the server to get the latest information. This is not ideal. Using websockets, the server could just let the client know immediately.

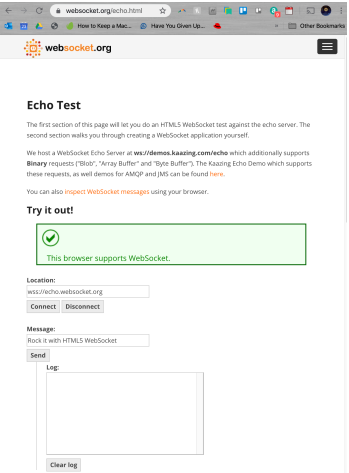
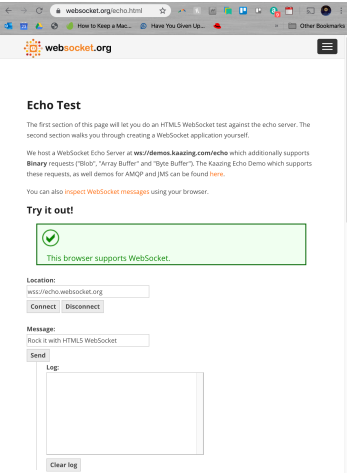
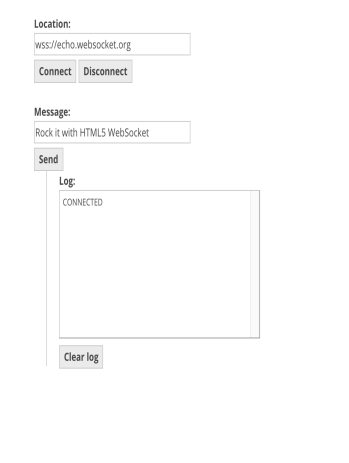
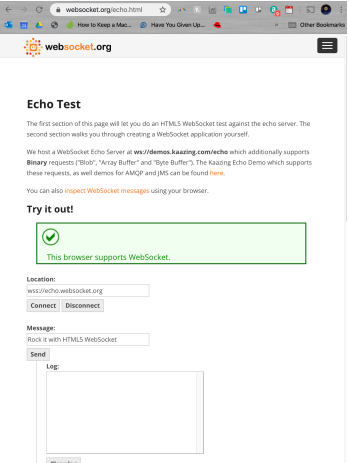
Similarly, when a multi-player game is being implemented, all the player clients would need to be notified of game events quickly.

**Reason-2:** Most organizations use firewalls but keep allow web connections. Since websockets use webconnections (i.e. port 80 etc), applications using websockets would still work.

## 3 Show me an example of a websocket.

Websocket.org has a simple ECHO server and client implementation. Multiple clients can connect to the websocket server (which is at <ws://echo.websocket.org>). Note that instead of using http:// we need to use ws:// to connect to use websocket protocol.

Here is a short demo of that website. You can do the same.

<p>Open <b>TWO</b> chrome tabs at <a href="http://www.websocket.org/echo.html">www.websocket.org/echo.html</a></p> <p>These will serve as <b>TWO CLIENTS</b> to the websocket server.</p>		
<p>On the left hand one, click on <b>CONNECT</b>. This will establish a connection with the websocket server.</p> <p>The server will respond with <b>CONNECTED</b>.</p>		

Type "Message from client#1" in the message text box and then

CLICK on Send button.

You will see both the SENT and RECEIVED message. The blue message was sent by the server.

**Note how the connection is persistent. Until you click disconnect – you will remain connected to the server.**

Location:  
ws://echo.websocket.org

Connect Disconnect

Message:  
MESSAGE FROM CLIENT #1

Send

Log:

CONNECTED

SENT: MESSAGE FROM CLIENT #1

RECEIVED: MESSAGE FROM CLIENT #1

Clear log

websocket.org

### Echo Test

The first section of this page will let you do an HTML5 WebSocket test against the echo server. The second section walks you through creating a WebSocket application yourself.

We host a WebSocket Echo Server at [www.demos.bazinga.com/echo](http://www.demos.bazinga.com/echo) which additionally supports Binary requests ("Blob", "Array Buffer" and "Typed Array"). The Bazinga Echo Demo which supports these requests, as well as demos for AMQP and JMS can be found [here](#).

You can also [inspect WebSocket messages](#) using your browser.

Try it out!

✓ This browser supports WebSocket.

Location:  
ws://echo.websocket.org

Connect Disconnect

Message:  
Rock it with HTML5 WebSocket

Send

Log:

Clear log

Now, on the right hand side client, click CONNECT.

You will see "Connected

Location:  
ws://echo.websocket.org

Connect Disconnect

Message:  
MESSAGE FROM CLIENT #1

Send

Log:

CONNECTED

SENT: MESSAGE FROM CLIENT #1

RECEIVED: MESSAGE FROM CLIENT #1

Clear log

Location:  
ws://echo.websocket.org

Connect Disconnect

Message:  
Rock it with HTML5 WebSocket

Send

Log:  
CONNECTED

Clear log

Instru...rinn...

Type "Message from client#2" in the message text box for the right hand side client and then

CLICK on Send button.

You will see both the SENT and RECEIVED message. The blue message was sent by the server.

Location:  
ws://echo.websocket.org

Connect Disconnect

Message:  
MESSAGE FROM CLIENT #1

Send

Log:

CONNECTED

SENT: MESSAGE FROM CLIENT #1

RECEIVED: MESSAGE FROM CLIENT #1

Clear log

Location:  
ws://echo.websocket.org

Connect Disconnect

Message:  
MESSAGE FROM CLIENT#2

Send

Log:  
CONNECTED

SENT: MESSAGE FROM CLIENT#2

RECEIVED: MESSAGE FROM CLIENT#2

Clear log

This was a really simple example. It shows a simple ECHO server in use. You can use it to TEST your own websocket clients.

## 4 Summary

- WebSocket is a protocol that is compatible with HTTP.
- Websockets allow full-duplex, persistent communications.
- On server side, URL needs to be registered. After that, server can respond to open, message, close events. Server can send messages to clients.
- On client side, client needs to connect to server. After that, client can respond to open, message, close events. Client can send messages to server.
- [websocket.org/echo.html](http://websocket.org/echo.html) is a websocket client.
- <ws://echo.websocket.org> is a websocket server (that you can use to test your client).