

OFFER ENDS IN **01 HRS 49 MINS 45 SECS**



Get All Our Books And Courses For 50% Off ([https://www.sitepoint.com/premium/L/Join?Ref\\_source=Sitepoint&Ref\\_medium=Noticebar&Ref\\_campaign=2-Years-For-1](https://www.sitepoint.com/premium/L/Join?Ref_source=Sitepoint&Ref_medium=Noticebar&Ref_campaign=2-Years-For-1))

**Programming**(<https://www.sitepoint.com/programming/>) - April 17, 2012 - By [Brian Raymor](https://www.sitepoint.com/author/braymor/) (<https://www.sitepoint.com/author/braymor/>)



## WebSockets: Stable and Ready for Developers

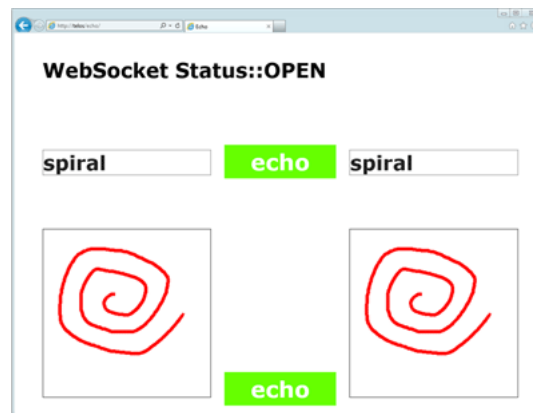
WebSockets are stable and ready for developers to start creating innovative applications and services. This tutorial provides a simple introduction to the W3C [WebSocket API](http://www.w3.org/TR/websockets/) (<http://www.w3.org/TR/websockets/>) and its underlying [WebSocket protocol](http://www.rfc-editor.org/rfc/rfc6455.txt) (<http://www.rfc-editor.org/rfc/rfc6455.txt>). The updated [Flipbook demo](https://websockets.interop.msftlabs.com/flipbook/) (<https://websockets.interop.msftlabs.com/flipbook/>) uses the latest version of the API and protocol.

Working groups have made significant progress and the [WebSocket API](http://www.w3.org/TR/websockets/) (<http://www.w3.org/TR/websockets/>) is a W3C Candidate Recommendation. [Internet Explorer 10](http://www.ietestdrive.com/) (<http://www.ietestdrive.com/>) implements this version of the spec. You can learn about the evolution of the spec [here](http://blogs.msdn.com/b/ie/archive/2011/09/15/site-ready-websockets.aspx) (<http://blogs.msdn.com/b/ie/archive/2011/09/15/site-ready-websockets.aspx>).

WebSockets enable Web applications to deliver real-time notifications and updates in the browser. Developers have faced problems in working around the limitations in the browser's original HTTP request-response model, which was not designed for real-time scenarios. WebSockets enable browsers to open a bidirectional, full-duplex communication channel with services. Each side can then use this channel to immediately send data to the other. Now, sites from social networking and games to financial sites can deliver better real-time scenarios, ideally using the same markup across different browsers.

## Introduction to the WebSocket API Using an Echo Example

The code snippets below use a simple echo server created with ASP.NET's [System.Web.WebSockets](http://msdn.microsoft.com/en-us/library/hh160729(v=vs.110).aspx) ([http://msdn.microsoft.com/en-us/library/hh160729\(v=vs.110\).aspx](http://msdn.microsoft.com/en-us/library/hh160729(v=vs.110).aspx)) namespace to echo back text and binary messages that are sent from the application. The application allows the user to type in text to be sent and echoed back as a text message, or draw a picture that can be sent and echoed back as a binary message.



For a more complex example that allows you to experiment with latency and performance differences between WebSockets and HTTP polling, see the [Flipbook demo](https://websockets.interop.msftlabs.com/flipbook/) (<https://websockets.interop.msftlabs.com/flipbook/>).

## Details of Connecting to a WebSocket Server

This simple explanation is based on a direct connection between the application and the server. If a proxy is configured, then IE10 starts the process by sending a HTTP CONNECT request to the proxy.

(<https://www.sitepoint.com/>)  
When a WebSocket object is created, a handshake is exchanged between the client and the server to establish the WebSocket connection.

OFFER ENDS IN 01 HRS 49 MINS 45 SECS

Client ← TCP → Server

Get All Our Books And Courses For 50% Off ([https://www.sitepoint.com/premium/l/join?ref\\_source=sitepoint&ref\\_medium=noticebar&ref\\_campaign=2-years-for-1](https://www.sitepoint.com/premium/l/join?ref_source=sitepoint&ref_medium=noticebar&ref_campaign=2-years-for-1))

IE10 starts the process by sending a HTTP request to the server.

```
GET /echo HTTP/1.1
Host: example.microsoft.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Origin: http://microsoft.com
Sec-WebSocket-Version: 13
```

Let’s look at each part of this request. The connection process starts with a standard HTTP GET request which allows the request to traverse firewalls, proxies, and other intermediaries:

```
GET /echo HTTP/1.1
Host: example.microsoft.com
```

The HTTP Upgrade header requests that the server switch the application-layer protocol from HTTP to the WebSocket protocol.

```
Upgrade: websocket
Connection: Upgrade
```

The server transforms the value in the Sec-WebSocket-Key header in its response to demonstrate that it understands the WebSocket protocol:

```
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
```

The Origin header is set by IE10 to allow the server to enforce origin-based security. (<http://tools.ietf.org/html/rfc6454>)

```
Origin: http://microsoft.com
```

The Sec-WebSocket-Version header identifies the requested protocol version. Version 13 is the final version in the IETF proposed standard:

```
Sec-WebSocket-Version: 13
```

If the server accepts the request to upgrade the application-layer protocol, it returns a HTTP 101 Switching Protocols response:



```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
```

To demonstrate that it understands the WebSocket Protocol, the server performs a standardized transformation on the Sec-WebSocket-Key from the client request and returns the results in the Sec-WebSocket-Accept header:

```
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+x0o=
```

IE10 then compares Sec-WebSocket-Key with Sec-WebSocket-Accept to validate that the server is a WebSocket server and not a HTTP server with delusions of grandeur.

The client handshake established a HTTP-on-TCP connection between IE10 and server. After the server returns its 101 response, the application-layer protocol switches from HTTP to WebSockets which uses the previously established TCP connection.

**HTTP is completely out of the picture at this point.** Using the lightweight WebSocket wire protocol, messages can now be sent or received by either endpoint at any time.



# Programming Connecting to a WebSocket Server

**Get All Our Books And Courses For 50% Off ([https://www.sitepoint.com/premium/l/join?ref\\_source=Sitepoint&ref\\_medium=Noticebar&ref\\_campaign=2-years-for-1](https://www.sitepoint.com/premium/l/join?ref_source=Sitepoint&ref_medium=Noticebar&ref_campaign=2-years-for-1))**  
The WebSocket protocol defines two new URI schemes which are similar to the HTTP schemes.

"ws:" "://" host [ ":" port ] path [ "?" query ] is modeled on the "http:" scheme. Its default port is 80. It is used for unsecure (unencrypted) connections.

"wss:" "://" host [ ":" port ] path [ "?" query ] is modeled on the "https:" scheme. Its default port is 443. It is used for secure connections tunneled over Transport Layer Security (<http://tools.ietf.org/html/rfc2818>).

When proxies or network intermediaries are present, there is a higher probability that secure connections will be successful, as intermediaries are less inclined to attempt to transform secure traffic.

The following code snippet establishes a WebSocket connection:

```
var host = "ws://example.microsoft.com";
var socket = new WebSocket(host);
```

## ReadyState – Ready ... Set ... Go ...

The WebSocket.readyState attribute represents the state of the connection: CONNECTING, OPEN, CLOSING, or CLOSED. When the WebSocket is first created, the readyState is set to CONNECTING. When the connection is established, the readyState is set to OPEN. If the connection fails to be established, then the readyState is set to CLOSED.

## Registering for Open Events

To receive notifications when the connection has been created, the application must register for open events.

```
socket.onopen = function (openEvent) {
  document.getElementById("serverStatus").innerHTML = 'Web Socket State::' + 'OPEN';
};
```

## Details Behind Sending and Receiving Messages

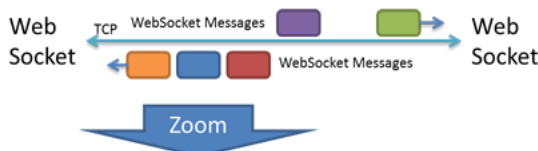
After a successful handshake, the application and the Websocket server may exchange WebSocket messages. A message is composed as a sequence of one or more message fragments or data "frames."

Each frame includes information such as:

Frame length

Type of message (binary or text) in the first frame in the message

A flag (FIN) indicating whether this is the last frame in the message



### WebSocket Frames

I'm the first binary  
fragment frame and  
I'm THIS big

I'm the next  
fragment frame and  
I'm THIS big

I'm the final  
fragment frame and  
I'm THIS big

(<https://www.sitepoint.com/>)  
IE10 reassembles the frames into a complete message before passing it to the script.

## Programming Sending and Receiving Messages

OFFER ENDS IN 01 HRS 49 MINS 45 SECS

The `send` API allows applications to send messages to a Websocket server as UTF-8 text,

Get All Our Books And Courses For 50% Off ([https://www.sitepoint.com/premium/L/Join?Ref\\_source=Sitepoint&Ref\\_medium=Noticebar&Ref\\_campaign=2-](https://www.sitepoint.com/premium/L/Join?Ref_source=Sitepoint&Ref_medium=Noticebar&Ref_campaign=2-)  
(<http://blogs.msdn.com/b/ie/archive/2011/12/01/working-with-binary-data-using-typed-arrays.aspx>)ArrayBuffers, or Blobs.

For example, this snippet retrieves the text entered by the user and sends it to the server as a UTF-8 text message to be echoed back. It verifies that the Websocket is in an OPEN readyState:

```
function sendTextMessage() {
    if (socket.readyState != WebSocket.OPEN)
        return;
    var e = document.getElementById("textmessage");
    socket.send(e.value);
}
```

This snippet retrieves the image drawn by the user in a canvas and sends it to the server as a binary message:

```
function sendBinaryMessage() {
    if (socket.readyState != WebSocket.OPEN)
        return;
    var sourceCanvas = document.getElementById('source');
    // msToBlob returns a blob object from a canvas image or drawing
    socket.send(sourceCanvas.msToBlob());
    // ...
}
```

## Registering for Message Events

To receive messages, the application must register for message events. The event handler receives a `MessageEvent` which contains the data in `MessageEvent.data`. Data can be received as text or binary messages.

When a binary message is received, the `WebSocket.binaryType` attribute controls whether the message data is returned as a `Blob` or an `ArrayBuffer` datatype. The attribute can be set to either "blob" or "arraybuffer." The examples below use the default value which is "blob."

This snippet receives the echoed image or text from the websocket server. If the data is a `Blob`, then an image was returned and is drawn in the destination canvas;

otherwise, a UTF-8 text message was returned and is displayed in a text field.

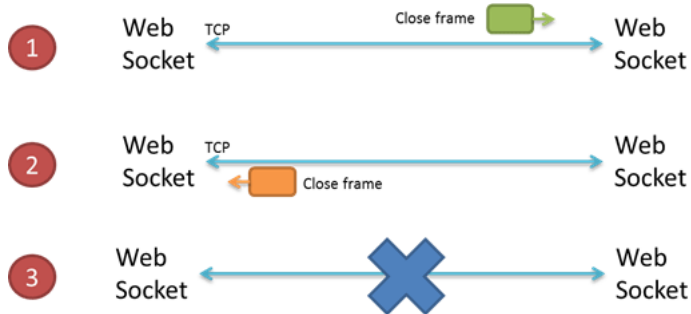
```
socket.onmessage = function (messageEvent) {
    if (messageEvent.data instanceof Blob) {
        var destinationCanvas = document.getElementById('destination');
        var destinationContext = destinationCanvas.getContext('2d');
        var image = new Image();
        image.onload = function () {
            destinationContext.clearRect(0, 0, destinationCanvas.width, destinationCanvas.height);
            destinationContext.drawImage(image, 0, 0);
        }
        image.src = URL.createObjectURL(messageEvent.data);
    } else {
        document.getElementById("textresponse").value = messageEvent.data;
    }
};
```

## Details of Closing a WebSocket Connection

Similar to the opening handshake, there is a closing handshake. Either endpoint (the application or the server) can initiate this handshake.

A special kind of frame – a close frame – is sent to the other endpoint. The close frame may contain an optional status code and reason for the close. The protocol defines a set of appropriate values for the status code. The sender of the close frame must not send further application data after the close frame.

When the other endpoint receives the close frame, it responds with its own close frame in response. It may send pending messages prior to responding with the close frame.



## Programming Closing a WebSocket and Registering for Close Events

The application initiates the close handshake on an open connection with the close API:

```
socket.close(1000, "normal close");
```

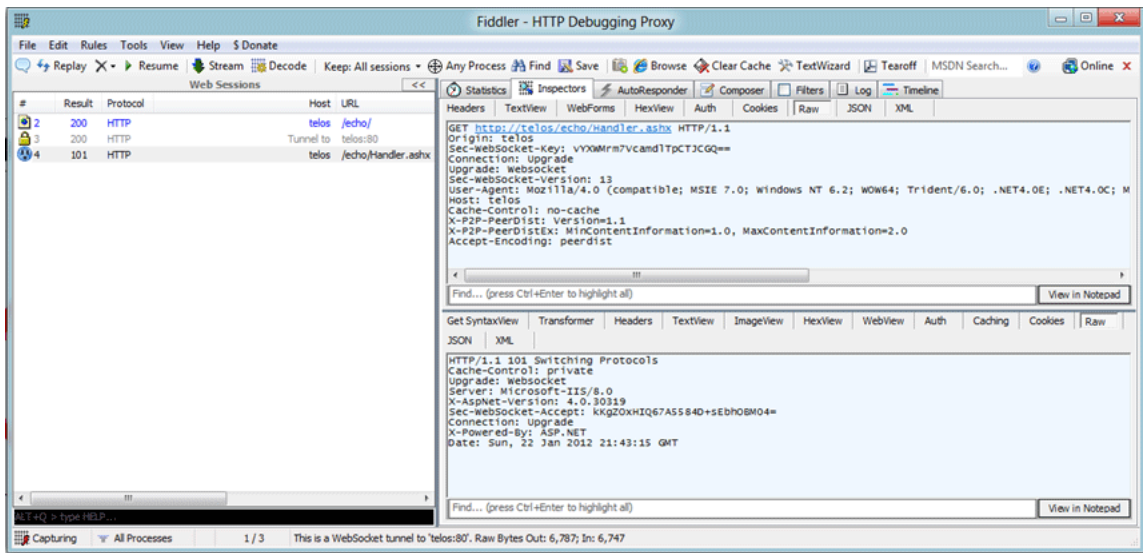
To receive notifications when the connection has been closed, the application must register for close events.

```
socket.onclose = function (closeEvent) {  
    document.getElementById("serverStatus").innerHTML = 'Web Socket State::' + 'CLOSED';  
};
```

The close API accepts two optional parameters: a status code as defined by the protocol and a description. The status code must be either 1000 or in the range 3000 to 4999. When close is executed, the readyState attribute is set to CLOSING. After IE10 receives the close response from the server, the readyState attribute is set to CLOSED and a close event is fired.

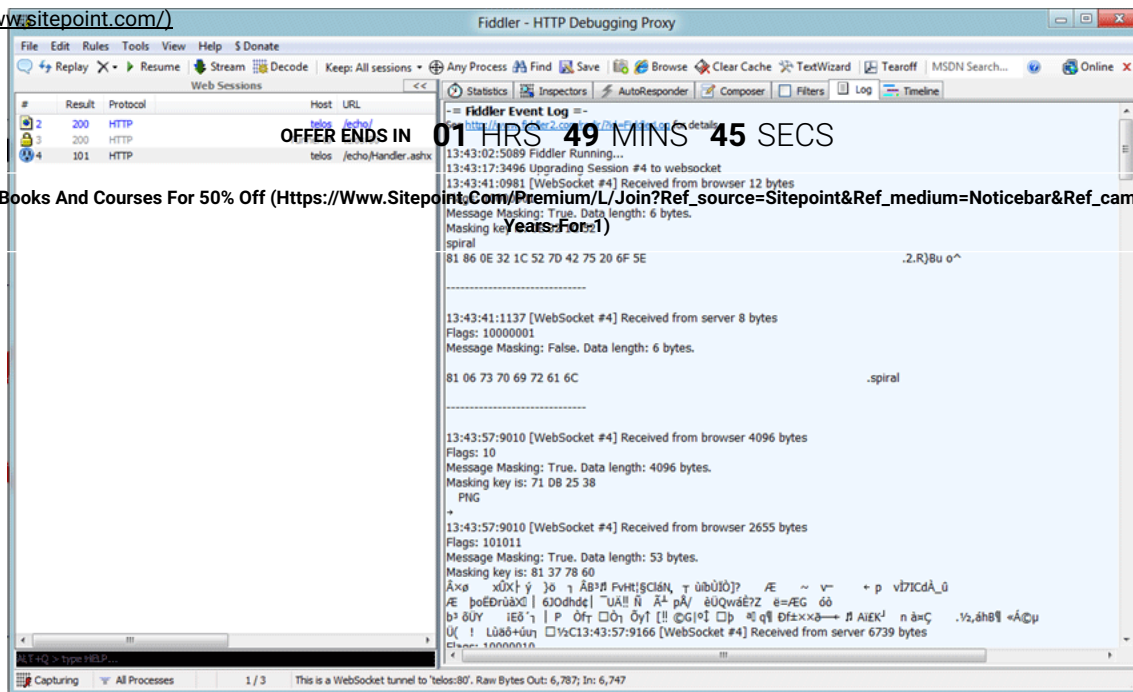
## Using Fiddler to See WebSocket Traffic

Fiddler (<http://www.fiddler2.com/fiddler2/>) is a popular HTTP debugging proxy. There is some support in the latest versions for the WebSocket protocol. You can inspect the headers exchanged in the WebSocket handshake:



All the WebSocket messages are also logged. In the screenshot below, you can see that "spiral" was sent to the server as a UTF-8 text message and echoed back:

(<https://www.sitepoint.com/>)



Get All Our Books And Courses For 50% Off ([https://www.sitepoint.com/premium/L/Join?Ref\\_source=Sitepoint&Ref\\_medium=Noticebar&Ref\\_campaign=2-Years-For-1](https://www.sitepoint.com/premium/L/Join?Ref_source=Sitepoint&Ref_medium=Noticebar&Ref_campaign=2-Years-For-1))

## Conclusion

If you want to learn more about WebSockets, you may watch these sessions from the Microsoft //Build/ conference from September 2011:

[Building real-time Web apps with HTML5 WebSockets \(http://channel9.msdn.com/events/BUILD/BUILD2011/PLAT-373C\)](http://channel9.msdn.com/events/BUILD/BUILD2011/PLAT-373C)

[Building real-time Web apps with WebSockets using IIS, ASP.NET and WCF](http://channel9.msdn.com/events/BUILD/BUILD2011/SAC-807T)

[\(http://channel9.msdn.com/events/BUILD/BUILD2011/SAC-807T\)](http://channel9.msdn.com/events/BUILD/BUILD2011/SAC-807T)

[Building Windows runtime sockets apps \(http://channel9.msdn.com/Events/BUILD/BUILD2011/PLAT-580T\)](http://channel9.msdn.com/Events/BUILD/BUILD2011/PLAT-580T)

If you're curious about using Microsoft technologies to create a WebSocket service, these posts are good introductions:

[Getting started with WebSockets in the Windows 8 developer preview \(http://www.paulbatum.com/2011/09/getting-started-with-websockets-in.html\)](http://www.paulbatum.com/2011/09/getting-started-with-websockets-in.html)

[Getting to know System.Net.WebSockets: A simple ASP.NET echo server \(http://www.paulbatum.com/2011/10/getting-to-know-systemnetwebsockets.html\)](http://www.paulbatum.com/2011/10/getting-to-know-systemnetwebsockets.html)

Start developing with WebSockets today!

[Electrical Socket \(http://www.shutterstock.com/cat.mhtml?](http://www.shutterstock.com/cat.mhtml?lang=en&search_source=search_form&version=llv1&anyorall=all&safesearch=1&searchterm=sockets&search_group=#id=34520161&src=45c66e58381057476152b535894b1502-1-25)

[lang=en&search\\_source=search\\_form&version=llv1&anyorall=all&safesearch=1&searchterm=sockets&search\\_group=#id=34520161&src=45c66e58381057476152b535894b1502-1-25](http://www.shutterstock.com/cat.mhtml?lang=en&search_source=search_form&version=llv1&anyorall=all&safesearch=1&searchterm=sockets&search_group=#id=34520161&src=45c66e58381057476152b535894b1502-1-25) image via Shutterstock



Meet the author

**[Brian Raymor \(https://www.sitepoint.com/author/braymor/\)](https://www.sitepoint.com/author/braymor/)**

Brian Raymor is a Senior Program Manager in the Windows Networking group at Microsoft.

**Login or Create Account to Comment**

(<https://www.sitepoint.com/>)







[Login \(/Premium/Sign-In?Ref\\_source=Sitepoint&Ref\\_medium=Comments&Redirect\\_path=%2Fwebsockets-Stable-And-Ready-For-Developers%2F%23commentsSection\)](#)

[Create Account \(/Premium/Sign-Up?Ref\\_source=Sitepoint&Ref\\_medium=Comments&Redirect\\_path=%2Fwebsockets-Stable-And-Ready-For-Developers%2F%23commentsSection\)](#)

Get All Our Books And Courses For 50% Off ([https://www.sitepoint.com/premium/l/join?Ref\\_source=Sitepoint&Ref\\_medium=Noticebar&Ref\\_campaign=2-Years-For-1](https://www.sitepoint.com/premium/l/join?Ref_source=Sitepoint&Ref_medium=Noticebar&Ref_campaign=2-Years-For-1))

## Popular In the Community



<b>HOW TO BUILD A REACT APP THAT WORKS WITH...</b>  <b>Hugo Hyz</b> 7 Dec Awesome content! I'm looking to transfer...	<b>24 PRODUCTIVITY TOOLS TO HELP YOU WITH YO...</b> <b>John Camacho</b> 18 Dec How is a wordpress theme a productivity...	<b>BUILDING A TRELLO LAYOUT WITH CSS GRID...</b>  <b>Martin H. Ande...</b> 6 Sep Thanks for a well written article. The...	<b>PUTTING THE APP IN PROGRESSIVE WEB APP...</b>  <b>Dev Agrawal</b> 9 Sep Excellent guide... definitely needed thi...	<b>USING PREACT AS A REACT ALTERNATIVE —...</b>  <b>Phil Mander</b> 12 Dec Thanks for the article Ahmed, I was lookin...	<b>FINDING A NICHE AND MAKING MONEY IN THE...</b>  <b>Simon Ho</b> 13 Dec Thank you for your inspiring article. I got...	<b>23 DEVELOPMENT TOO FOR BOOSTING WEBSIT</b>  <b>User_PCS</b> 14 Dec Very informative and veryhelpful list of...
--	--	--	--	--	--	---

## Conversation (9)



Sort by **Oldest** ▾



Add a comment...



**Phil Leggetter**




Great to see WebSockets covered here in a very comprehensive way.

Realtime bi-directional functionality has been achievable for quite some time via the Comet paradigm and using HTTP Long-Polling or HTTP Streaming for server -> client connection, with the addition of second HTTP short-lived connections for client -> server communication. It's great that we've finally now got a full duplex bi-directional method of doing this.

My personal opinion about how WebSockets are going to be used is that they won't be directly used too often by developers. How often do web developers directly use `document.getElementById`? Even if they write their own code that uses this function they'll frequently wrap that function to do a bit more. Many of us auto-include jQuery in our apps, and use `$('#someId').doStuff().doMoreStuff().wow()`.

My point is that as developers we'll end up using a library that 'under the hood' uses WebSockets and, whilst it is important to know how they work, I don't think it'll be something we worry about all that often.

[See more](#)

Reply · Share ·  





**Patrick**



Ready for developers? Um, not really. Support is limited to IE10, FF11+ and Chrome 16+. Nothing for Safari or Opera. I'm guessing that no mobile browser supports them. So yeah, if you have an up-to-date browser and are developing something for personal use, or if you want to showcase the technology, you can use websockets today. For any application developed for consumption by the general public, we'll be waiting at least a few years.

Great technology, but like so many new web innovations, its main purpose right now is just to tease developers and show them how much better things could be.

Reply · Share ·  



**Red Bucket** → Patrick

<http://caniuse.com/websockets>

Reply · Share ·  




**Orange Magnet** → Patrick

As Stephen's link shows support for WebSockets is actually:

- \* Firefox
- \* Chrome
- \* Safari
- \* Safari mobile
- \* Firefox mobile

(<https://www.sitepoint.com/>) become the default browser for Android)

[See more](#)


Reply · Share ·  

OFFER ENDS IN **01 HRS 49 MINS 45 SECS**



Frank

**Get All Our Books And Courses For 50% Off ([https://www.sitepoint.com/premium/L/Join?Ref\\_source=Sitepoint&Ref\\_medium=Noticebar&Ref\\_campaign=2-Years-For-1](https://www.sitepoint.com/premium/L/Join?Ref_source=Sitepoint&Ref_medium=Noticebar&Ref_campaign=2-Years-For-1))**  
I'm curious if it is possible to use sockets to stream audio and/or video? Or do we need to wait for the elusive "device" element?

Reply · Share ·  



Coco

Thank you for the great article. I have to say, though, I've only heard about WebSockets very briefly in the past so I am still trying to absorb the whole concept. A few questions if I may:

1) You say that both text and binary content can be sent and recieved with the WebSocket. How do you handle security on the server side? What would stop someone from writing their own client to connect to your WebSocket server and uploading some sort of rogue executable, for example? Can you incorporate traditional methods of web based password protection? Is there a way to validate (on the server side) what is being sent down the WebSocket connection? Are these concerns even relevant?

2) Does the WebSocket connection pass through a web server (like Apache or IIS) or is the WebSocket server a standalone beastie on its own port? Or does it pass through a web server only for the first part where it does the HTTP GET request?

Thanks again.

Reply · Share ·  



**Green Bottle** → Coco

1) You can do this in a few ways. At Pusher the URL that the WebSocket connects to has an application key which initially identifies that the connection is at least valid.

Once the connection is established you can add your own authentication as you require by checking the messages that are sent from client to server. Part of that could be an initial username/password message. Messages could all be signed for authenticity.

The executable example isn't very likely unless you take the data sent over the WebSocket connection, create a file out of it and then run it (or similar). So, you have the same control over the WebSocket connection as you do an HTTP connection.

2) IIS doesn't natively support WebSockets yet. We'll be waiting for Windows Server 8 for that. Apache has a module for WebSocket support. You can also get standalone WebSocket servers.

[See more](#)

Reply · Share ·  



Litost

Nice article! I am still weighing arguments about websockets, though. Should I go like "So now, we're back to where we were 15 years ago - yet with less powerful graphical controls", or like "Deployed as served, runs almost everywhere"...? Kind of reminds me of Java applets. Anyway, I am not sure that we'll have lots of relevant use cases for sockets in HTML, but I guess it's good to see some innovation. Cheers.

Reply · Share ·  




**Olive Cone** → Litost

Java applets were the first way that realtime server push could be achieved. Then we moved to native solutions such as a hidden IFRAME then solutions that used the XMLHttpRequest object. WebSockets are a native browser solution that solves the 'hacked' HTTP-based solutions.

The use cases are the same that the previous solutions were used for in addition to new innovative uses:

- \* Realtime data - simple updates such as share prices or sports data
- \* Notifications - Usually used along with Create, Update and Delete server/database events. e.g. a new news item is available or a new comment being added to a blog post.
- \* Chat - interaction between multiple users. I'm sure we've all seen chat before. But WebSockets are a perfect solution to this feature.
- \* Collaborative applications - e.g. Google docs where multiple users are editing the same document.

[See more](#)

Reply · Share ·  



OFFER ENDS IN 01 HRS 49 MINS 45 SECS

Stuff We Do Books And Courses For 50% Off ([https://www.sitepoint.com/Premium4444?Ref\\_source=Sitepoint&Ref\\_medium=Noticebar&Ref\\_campaign=2-Years-For-1](https://www.sitepoint.com/Premium4444?Ref_source=Sitepoint&Ref_medium=Noticebar&Ref_campaign=2-Years-For-1))

- [Premium \(/premium/\)](#)
- [Our Story \(/about-us/\)](#)
- [Contact Us \(/contact-us/\)](#)
- [Terms of Use \(/legals/\)](#)
- [Versioning \(/versioning/\)](#)
- [Press Room \(/press/\)](#)
- [FAQ \(<https://sitepoint.zendesk.com/hc/en-us>\)](#)
- [Privacy Policy \(/legals/#privacy\)](#)
- [Themes \(/themes/\)](#)
- [Write for Us \(/write-for-us/\)](#)
- [Forums \(/community/\)](#)
- [Advertise \(/advertise/\)](#)
- [References \(/html-css/css/\)](#)

## Connect

- [!\[\]\(30a147af384f9f71632c2ff17bc706c8\_img.jpg\) \(https://www.facebook.com/sitepoint\)](https://www.facebook.com/sitepoint) [!\[\]\(9b33568d5c136f08ca688ce48be37574\_img.jpg\) \(http://twitter.com/sitepointdotcom\)](http://twitter.com/sitepointdotcom) [!\[\]\(8c93063dab026f10e159986b27c41c64\_img.jpg\) \(/versioning/\)](/versioning/) [!\[\]\(8a17676a8da87a4e59299223a765e613\_img.jpg\) \(https://www.sitepoint.com/feed/\)](https://www.sitepoint.com/feed/) [!\[\]\(f7fdc7cc047b770fc5fdd2c2137c07d9\_img.jpg\) \(https://plus.google.com/+sitepoint\)](https://plus.google.com/+sitepoint)

© 2000 – 2018 SitePoint Pty. Ltd.

Recommended Hosting Partner:  (<https://www.siteground.com/go/sitepoint-siteground-promo>)

