+5 6 3 help

HTML5 WebSocket vs Long Polling vs AJAX vs WebRTC vs Server-Sent Events [closed]

So I'm building a small chat application for friends and, as I'm building it, I've been going back and forth about how to get information in a timely manner that is not manual or as rudimentary as forcing a page refresh.

That said, I've put in simple Ajax but alas, it's with constant hits to the server via a short timer. Someone suggested to me (on another thread of mine here on SO) about long/short polling etc, and as I researched, I ran across HTML5 websockets...

It SEEMS easy to implement and although not full spec yet (I think) (plus I think only some browsers support it) I'm wondering, in a HTML5 WebSockets vs AJAX long/short polling, what are the advantages/disadvantages to them?

Since I am learning, I don't want to learn both if one is better... They kind of do the same thing (I'm assuming) but I'm wondering are there certain scenarios where one would use one technique over the other? Or is HTML5 WS supposed to take over in general?

javascript ajax html5 websocket network-protocols

edited Jun 26 '15 at 19:11
Nakilon
18.1k 8 59 85

asked Apr 5 '12 at 12:39
somdow
2,169 6 21 42

closed as primarily opinion-based by gnat, rink.attendant.6, DreadPirateShawn, Peter Pei Guo, HaveNoDisplayName Jun 22 '15 at 17:37

Many good questions generate some degree of opinion based on expert experience, but answers to this question will tend to be almost entirely based on opinions, rather than facts, references, or specific expertise.

If this question can be reworded to fit the rules in the help center, please edit the question.

4 Would be nice if the moderators adjusted this question to be "appropriately worded" so it could be re-opened. There is a great question and answer here. – Ash Blue Jan 27 at 1:18

3 Answers

WebSockets - is definitely the future. Long polling is dirty workaround of preventing creating connections for each request like AJAX does -- but long polling was created when WebSockets didn't exist. Now due to WebSockets, Long Polling is going away. And WebRTC allows peer-to-peer communication.

I recommend learning WebSockets.

Comparison:

of different communication techniques in web

- **AJAX** - `request → response` . Creates connection to server, sends request headers with optional data, gets response from server, closes connection. *Supported in all major browsers.*
- **Long poll** - `request → wait → response` . Creates connection to server like AJAX does, but keep-alive connection open for some time (not long though), during connection open client can receive data from server. Client have to reconnect periodically after connection is closed due to timeouts or data eof. On server side it is still treated like HTTP request same as AJAX, except the answer on request will happen now or some time in the future defined by application logic. *Supported in all major browsers.*
- **WebSockets** - `client ↔ server` . Create TCP connection to server, and keep it as long as needed. Server or client can easily close it. Client goes through HTTP compatible handshake process, if it succeeds, then server and client can exchange data both directions at any time. It is very efficient if application requires frequent data exchange in both ways. WebSockets do have data framing that includes masking for each message sent from client to server so data is simply encrypted. *support chart (very good)*
- **WebRTC** - `peer ↔ peer` . Transport to establish communication between clients and is transport-agnostic so uses UDP, TCP or even more abstract layers. By design it allows to transport data in reliable as well as unreliable ways. This is generally used for high volume

data transfer such as video/audio streaming where reliability - is secondary and few frames or reduction in quality progression can be sacrificed in favour of response time and at least delivering something. Both sides (peers) can push data to each other independently. While it can be used totally independent from any centralised servers it still require some way of exchanging endPoints data, where in most cases developers still use centralised servers to "link" peers. This is required only to exchange essential data for connection establishing - after connection is established server on aside is not required. *support chart (medium)*

- **Server-Sent Events** - `client ← server` . Client establishes persistent and long-term connection to server. Only server can send data to client. If client wants to send data to server it would require to use other technology/protocol to do so. This protocol is HTTP compatible and simple to implement in most server-side platforms. This is preferable protocol to be used instead of Long Polling. *support chart (good, except IE)*

Advantages:

Main advantage of **WebSockets** for server, is that it is not HTTP request (after handshake), but proper message based communication protocol. That **allows you to achieve huge performance and architecture advantages**. For example in node.js you can share the same memory for different socket connections, so that way they can access shared variables. So you don't need to use database as exchange point in the middle (like with AJAX or Long Polling and for example PHP). You can store data in RAM, or even republish between sockets straight away.

Security considerations

People often are concerned regarding security of WebSockets. Reality is that it makes little difference or even puts WebSockets as better option. First of all with AJAX there is a higher chance of MITM as each request is new TCP connection and traversing through internet infrastructure. With WebSockets, once it's connected it is far more challenging to intercept in between, with additionally enforced frame masking when data is streamed from client to server as well as additional compression, that requires more effort to probe data. **All modern protocols support both: HTTP and HTTPS (encrypted).**

P.S.

Remember that WebSockets generally have a very different approach of logic for networking, more like real-time games had all this time, and not like http.

edited Nov 20 '15 at 11:56

answered Apr 5 '12 at 13:15

moka

11.9k 2 26 50

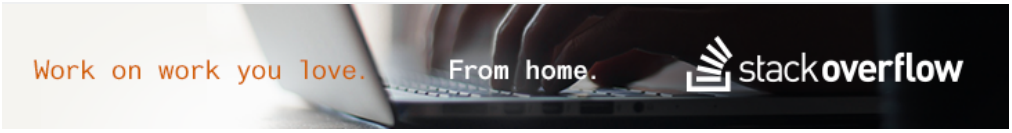
1 Tell me if I'm wrong but even though WebSocket provides a lot of improvement it does not completely overlap the functionality of long polling. For instance you cannot just swap comet for websocket to poll a REST api. – Reno Aug 7 '13 at 9:23

7 It's not about compatibility it self. Most important that it is about to fully rethink the way communication is happening. As RESTful APIs work with Request>Response pattern, bi-directional communication here would be pointless. So trying to use WebSockets to query RESTful API - is a bit weird attempt, and can't see any benefit of it at all. If you need data from RESTful API but in real-time manner, then you create WebSockets api to push data that will work with bidirectional communication like WebSockets. You are trying to compare things in angle that they are not comparable :) – moka Aug 7 '13 at 15:00

2 Hi @pithhelmet it all depends on server-side software (language/tech) it self. WebSocket is layer over TCP, and there are many ways of doing TCP stream implementations. Modern web servers use event-based architecture, and are very efficient with thread pools. Which tech you are using? Node.js uses events behind the scenes for IO, and event with single thread in execution context, so it is amazingly efficient. Using thread for each connection - is very inefficient in terms of RAM (1mb+ per thread) as well as CPU, as those threads will just idle or worse - infinite loop of checking for data. – moka May 13 '14 at 14:49

2 Long polling isn't a dirty workaround, and it's different from webSocket. These two are meant to used in different scenario. – bagz_man Aug 6 '14 at 7:10

3 @moka: Cloudflare's *free-tier* will absorb a sustained 400+Gbps attack. Can your wallet absorb the AWS bill? Also AWS and Cloudflare have opposite views when it comes to dealing with complaints against your origin. It's just something to keep in mind as long as we're discussing the tradeoffs. :) – danneu Jan 2 '15 at 21:21



One contending technology you've omitted is Server-Sent Events / Event Source. What are Long-Polling, Websockets, Server-Sent Events (SSE) and Comet? has a good discussion of all of these. Keep in mind that some of these are easier than others to integrate with on the

server side.

edited Aug 27 '13 at 20:27

answered Aug 27 '13 at 17:14

bmm6o
3,277 2 14 41

Out of all of these, whick would you suggest to look into? – somdow Dec 29 '13 at 12:52

I've had success with long-polling, the only trick (for it and other technologies) is not tying up a server thread. If you don't use asynchronous server code it won't scale. – bmm6o Dec 29 '13 at 13:59

1 @somdow Maksims-Mihejevs answered your question nicely in the first two paragraphs of his answer. Use websockets. – Jeff Sheffield Jan 28 '14 at 4:56

For chat applications or any other application that is in constant conversation with the server, `WebSockets` are the best option. However, you can only use `WebSockets` with a server that supports them, so that may limit your ability to use them if you cannot install the required libraries. In which case, you would need to use `Long Polling` to obtain similar functionality.

edited Jul 11 '13 at 13:31

answered Apr 5 '12 at 12:44

Brant Olsen
3,916 5 24 44

5 WebSockets are supported by every server... You just need to install node.js or something similar. – noob Apr 5 '12 at 12:46

6 Tweaked a bit to explain that yes any server will support WebSockets. However, if you are using hosting service, you may not be able to use them. – Brant Olsen Apr 5 '12 at 12:49

not at heroku they are not supported yet – Muhammad Umer Jul 21 '13 at 23:09

I realize this thread is a bit old but... WebSockets may not be the best answer for all bi-directional communication. I recently noticed that the documentation for Spring 4's web socket support suggests that WebSockets are better suited for moving large amounts of data or low latency. If those are not applicable or are not a priority then they I believe they suggest using long polling. I don't know the full justification for this view, I just figured the Spring folks know what they are talking about in general. – Stoney Aug 29 '14 at 12:49

@Stoney apart from the fact that you would need to setup websocket on the server (handlers, etc.) There is simply no reason to use Long polling over websocket. Websocket is much faster (low latency) and allows the server to "talk" to the client without the client asking it to. Nowadays I use signalr (one of the best implementations of websocket ever made in my opinion - it runs on the client and server and allows the client to call methods on the server and the server on the client as if there is no difference) on every website I make - dynamic content loading, bottomless pages, etc. – Random User Aug 11 '15 at 22:31

protected by Tushar Gupta Aug 14 '15 at 17:07

Thank you for your interest in this question. Because it has attracted low-quality or spam answers that had to be removed, posting an answer now requires 10 reputation on this site.

Would you like to answer one of these unanswered questions instead?