


- [Mobile](#)
- [HTML5](#)
- [JavaScript](#)
- [APM](#)
- [Java](#)
- [Cloud](#)
- [Continuous Delivery](#)
- [.NET](#)

[All topics](#)

You are here: [InfoQ Homepage](#) [News](#) WebSockets versus REST?

WebSockets versus REST?

by [Mark Little](#) on Feb 26, 2012 | [24 Discuss](#)

- Share 
- |
-
-
-
-
-
-
-
-
-
-
- ["Read later"](#)
- ["My Reading List"](#)

For the past few years [WebSockets](#) has been gaining in popularity and availability. At the end of last year it moved one step closer to a standard by becoming a [W3C Candidate Recommendation](#). Oracle and others have also recently submitted a request to start a standardisation effort around WebSockets ([JSR 356](#)) in the next version of Java Enterprise Edition. All of the major browsers, such as Chrome, Firefox, Safari and IE, support one of the WebSockets revisions and will ultimately adopt whatever the standard eventually becomes. In a [relatively short period](#) of time, [WebSockets has almost become](#) an [integral part of the Web](#). However, there remains a segment of developers who are uncertain as to how or whether WebSockets fits into the architectural style of the Web: REST. Some, such as Nathan Evans, go so far as [to suggest](#) that WebSockets could overshadow REST:

After reading up about the standard in detail and absorbing the various online discussion around it, it is becoming increasingly clear to me that this standard is going to steal a large chunk of mind share from RESTful web services. What I mean is that there will come a stage in product development where somebody will have to ask the question:

"Right guys, shall we use WebSockets or REST for this project?"

I expect that WebSockets will, within a year or two, begin stunting the growth of RESTful web

services – at least as we know them today.

In his article Nathan believes that based on his experiences of using REST, it is not the "silver bullet" that some other people portray it as, although he admits that WebSockets is also not perfect. He then outlines a number of reasons why WebSockets is a threat to REST, including "sub-par" REST frameworks reliance on text-based protocols. Some of the important benefits that WebSockets offers over REST include bi-directional interactions:

The true bi-directional capability offered by WebSockets is a first for any HTTP-borne protocol. It is something that neither SOAP nor REST have. And which Comet/push/long-polling can only emulate, inefficiently. The bi-directional capability is inherently so good that you could tunnel a real-time TCP protocol such as Remote Desktop or VNC over a WebSocket, if you wanted.

Nathan believes that the benefits of WebSockets outweigh those of REST (HTTP) and that developers will migrate towards WebSockets in preference to it.

REST will probably remain the default choice for projects that need highly visible and cross-platform interoperable web services. Projects without those requirements will probably opt for WebSockets instead and either run JSON over it, or use a bespoke wire protocol. [...] Even though they are competing, the good thing is that REST and WebSockets can actually co-exist with one another. In fact, because they are both built upon HTTP fundamentals they will actually complement each other.

However, Nathan isn't the only one to raise the question of "WebSockets or/versus REST". For instance, [Shay Bannon](#) wondered in 2010 whether it is even possible to use the principles of REST with WebSockets:

First and foremost, how do you represent a URI? Second, how do you represent the HTTP methods (GET, PUT, POST, ...)? And last, how do you represent HTTP uri parameters and headers? It seems like maybe a solution for this is to build some sort of schema into the content that goes into that text string. Something like a JSON string that has a "uri" field, and "params" and so on. But thats annoying, since with HTTP, you can create very simple gateways that simply use the headers or parameters without needing to parse the body...

He wonders why WebSockets does not have notion of URI or header, and as a result whether there is a need for a REST-over-WebSockets specification before people reinvent REST and end up doing so "not in a uniform manner"? At the time he received a mixed response to this article. For instance one respondent, who mentioned he worked for a WebSockets company, so perhaps not too objective, stated:

REST and WebSocket communication seems to be two different types of distributed computing plumbing. REST is the old-school, sit on top of HTTP, synchronous style of web rpc. WebSocket is the newer, sit along side HTTP, usually asynchronous style of web communication. Imho, in the long term, WebSocket will dramatically reduce the need for REST for WAN computing. With WebSocket, all the protocols we've known and loved for the past few decades can now be extended over the web without the clumsy and performance-sucking mapping to HTTP.

And another:

My take is REST involves the conventional request/response paradigm. In contrast sockets cater for the comet/long polling scenario where the line of communication stays open for several

communication cycles. Also, the initial handshake to WS still occurs from HTTP, so in reality they are not mutually exclusive. I also thought the whole point of the WS protocol was to get rid of the cruft in the HTTP Headers as it becomes redundant and just adds to the payload size.

However, while agreeing that REST and WebSockets can and should co-exist, several commenters disagreed with the notion of a REST-over-WebSockets specification at all, with one adding:

if you consider REST in the Fielding sense, with a web of addressable objects (or resources), then that doesn't really work in a duplex comms format. You don't expect the resources to initiate the conversation. WebSockets will transform the web (if they take off), but not as a protocol for REST-style communications.

And another giving a more detailed point of view:

WebSockets are like a conversation between two people with ADD. It's full duplex, so both sides can talk at the same time, and both sides have to keep their listening caps on **while** they're talking. REST is stateless and synchronous, dealing only with request->response. You would have to expand the concept of REST to get the benefit of unprompted server->client communication. I could see there being a library that implements REST in WebSockets, but it would only be useful for applications that already have a RESTful API and want to get the benefit of reduced overhead that a single connection would afford without refactoring their code.

Then there are some in the REST community, [such as Jim Webber](#), who believe that WebSockets [are not right for the Web](#):



With WebSockets almost at the point of becoming a standard, as well as being supported and used in browsers, [mobile](#) and [in the cloud](#), it is interesting to wonder how much of an impact they will have on developers who are currently using REST and HTTP, or do they address a different developer segment? Worse still, are we at risk as some believe, of "breaking the Web"?

- Personas
- [Architecture & Design](#)
- [Development](#)
- Topics
- [Enterprise Architecture](#)