**Server push** (Websocket vs SSE)

**Reference 1 (**[**https://stackoverflow.com/questions/5195452/websockets-vs-server-sent-events-eventsource**](https://stackoverflow.com/questions/5195452/websockets-vs-server-sent-events-eventsource)**):**

Websockets and SSE (Server Sent Events) are both capable of pushing data to browsers, however they are not competing technologies.

Websockets connections can both send data to the browser and receive data from the browser. A good example of an application that could use websockets is a chat application.

SSE connections can only push data to the browser. Online stock quotes, or twitters updating timeline or feed are good examples of an application that could benefit from SSE.

In practice since everything that can be done with SSE can also be done with Websockets, Websockets is getting a lot more attention and love, and many more browsers support Websockets than SSE.

However, it can be overkill for some types of application, and the backend could be easier to implement with a protocol such as SSE.

Furthermore SSE can be polyfilled into older browsers that do not support it natively using just JavaScript. Some implementations of SSE polyfills can be found on the [Modernizr github page](https://github.com/Modernizr/Modernizr/wiki/HTML5-Cross-Browser-Polyfills).

**Gotchas:**

* SSE suffers from a limitation to the maximum number of open connections, which can be specially painful when opening various tabs as the limit is *per browser* and set to a very low number (6). The issue has been marked as "Won't fix" in [Chrome](https://bugs.chromium.org/p/chromium/issues/detail?id=275955) and [Firefox](https://bugzilla.mozilla.org/show_bug.cgi?id=906896)
* Only WS can transmit both binary data and UTF-8, SSE is limited to UTF-8. (Thanks to Chado Nihi).

[HTML5Rocks](http://www.html5rocks.com/tutorials/eventsource/basics/) has some good information on SSE. From that page:

# **Server-Sent Events vs. WebSockets**

Why would you choose Server-Sent Events over WebSockets? Good question.

One reason SSEs have been kept in the shadow is because later APIs like WebSockets provide a richer protocol to perform bi-directional, full-duplex communication. Having a two-way channel is more attractive for things like games, messaging apps, and for cases where you need near real-time updates in both directions. However, in some scenarios data doesn't need to be sent from the client. You simply need updates from some server action. A few examples would be friends' status updates, stock tickers, news feeds, or other automated data push mechanisms (e.g. updating a client-side Web SQL Database or IndexedDB object store). If you'll need to send data to a server, XMLHttpRequest is always a friend.

SSEs are sent over traditional HTTP. That means they do not require a special protocol or server implementation to get working. WebSockets on the other hand, require full-duplex connections and new Web Socket servers to handle the protocol. In addition, Server-Sent Events have a variety of features that WebSockets lack by design such as automatic reconnection, event IDs, and the ability to send arbitrary events.

**Reference 2:**

[**https://codeburst.io/polling-vs-sse-vs-websocket-how-to-choose-the-right-one-1859e4e13bd9**](https://codeburst.io/polling-vs-sse-vs-websocket-how-to-choose-the-right-one-1859e4e13bd9)