What are Long-Polling, Websockets, Server-Sent Events (SSE) and Comet?

Asked 8 years, 10 months ago Active 3 months ago Viewed 189k times

1089

970

I have tried reading some articles, but I am not very clear on the concepts yet.

Would someone like to take a shot at explaining to me what these technologies are:

1. Long Polling
2. Server-Sent Events
3. Websockets
4. Comet

One thing that I came across every time was, the server keeps a connection open and pushes data to the client. How is the connection kept open, and how does the client get the pushed data? (How does the client use the data, maybe some code might help?)

Now, which one of them should I use for a real-time app. I have been hearing a lot about websockets (with socket.io [a node.js library]) but why not PHP?

php websocket comet long-polling server-sent-events

Share Improve this question

Follow

edited Nov 12 '16 at 21:27

DOK

31.3k 7 58 91

asked Jun 18 '12 at 6:28

user1437328

14.1k 9 28 39

1 Realtime websocket or webrtc? There is a library for websocket in php, you do need to write extra code in order for it to work using ZMQ or just socket programming, nodeJs is built for this so its easily available. The reason websocket is not readily available in php is that you have to run an extra terminal and kept it running so that websocket server is readily available, you will have two servers bottom line. and the structure, php is not an event structure like javascript so there's that, websocket uses a event structure in order to catch and send messages. – PauAI Apr 11 '17 at 0:31

Additionally: Comet and ServerSent Events are PHP's workaround of achieving almost realtime(not really) without creating 2 servers. – PauAI Apr 11 '17 at 0:49

4 Answers

Active | Oldest | Votes

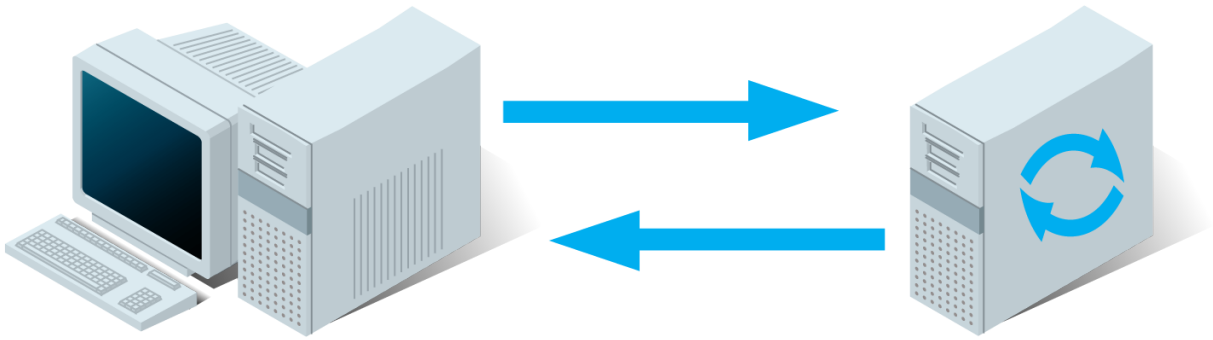
2149

In the examples below the client is the browser and the server is the webserver hosting the website.

Before you can understand these technologies, you have to understand *classic* HTTP web traffic first.

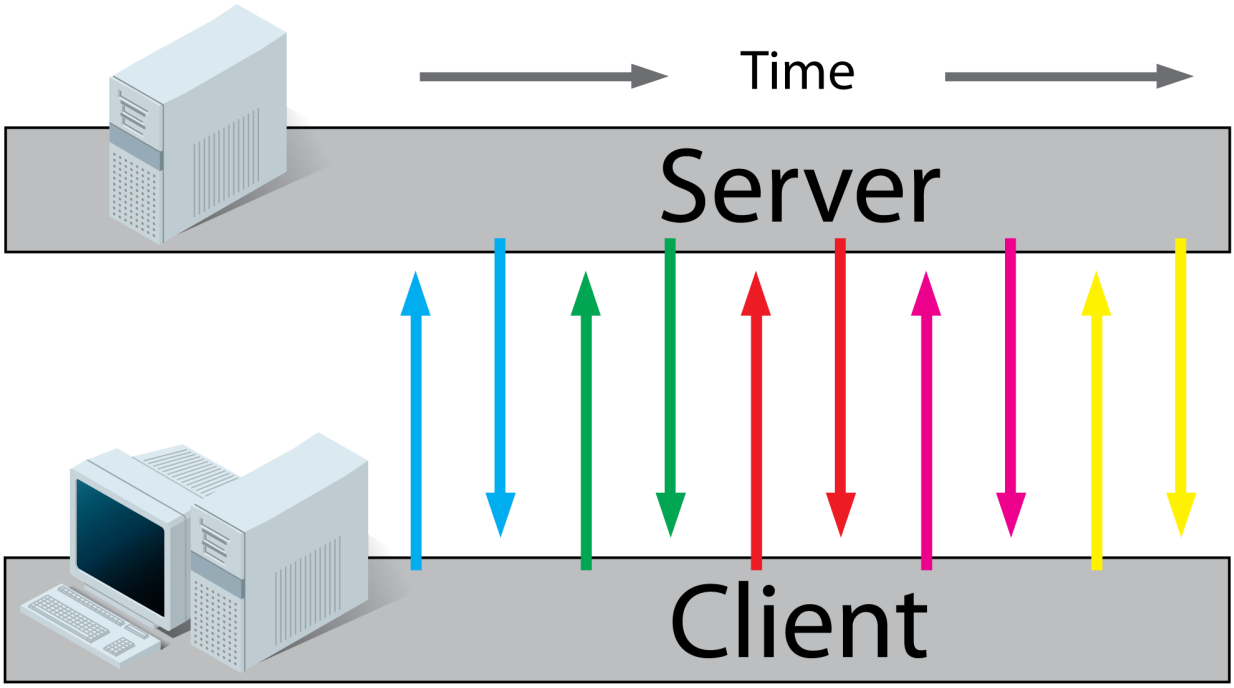
Regular HTTP:

1. A client requests a webpage from a server.
2. The server calculates the response
3. The server sends the response to the client.



Ajax Polling:

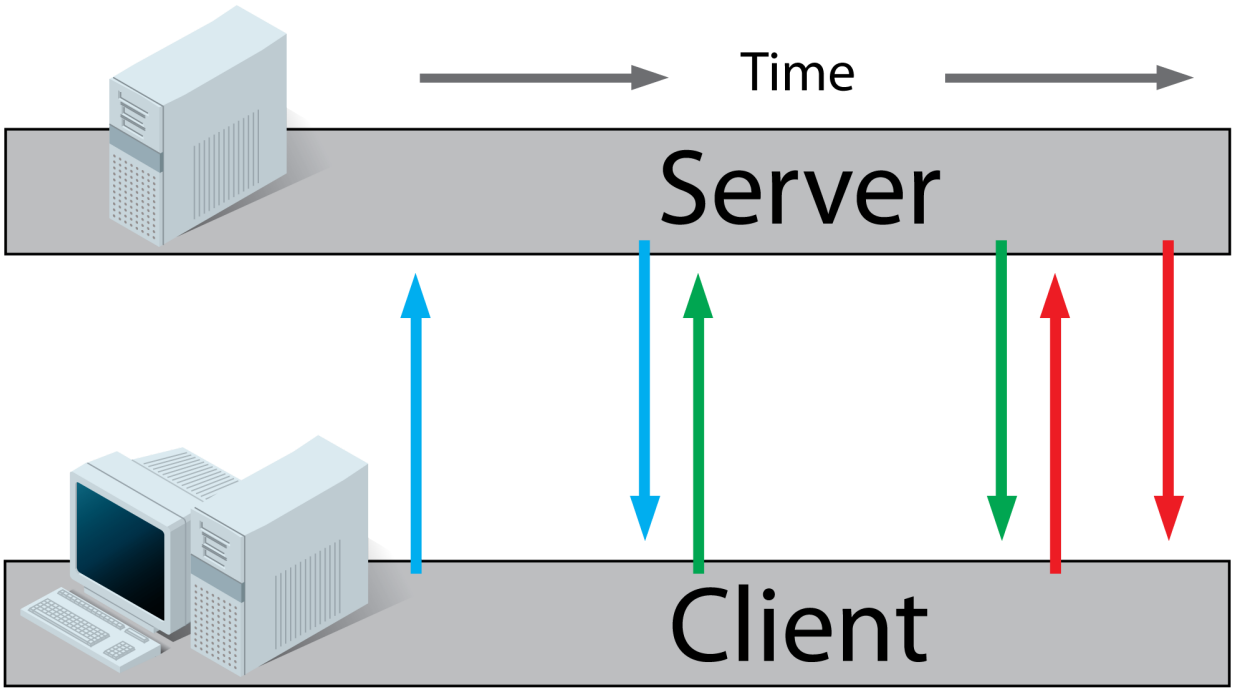
1. A client requests a webpage from a server using regular HTTP (see HTTP above).
2. The client receives the requested webpage and executes the JavaScript on the page which requests a file from the server at regular intervals (e.g. 0.5 seconds).
3. The server calculates each response and sends it back, just like normal HTTP traffic.



Ajax Long-Polling:

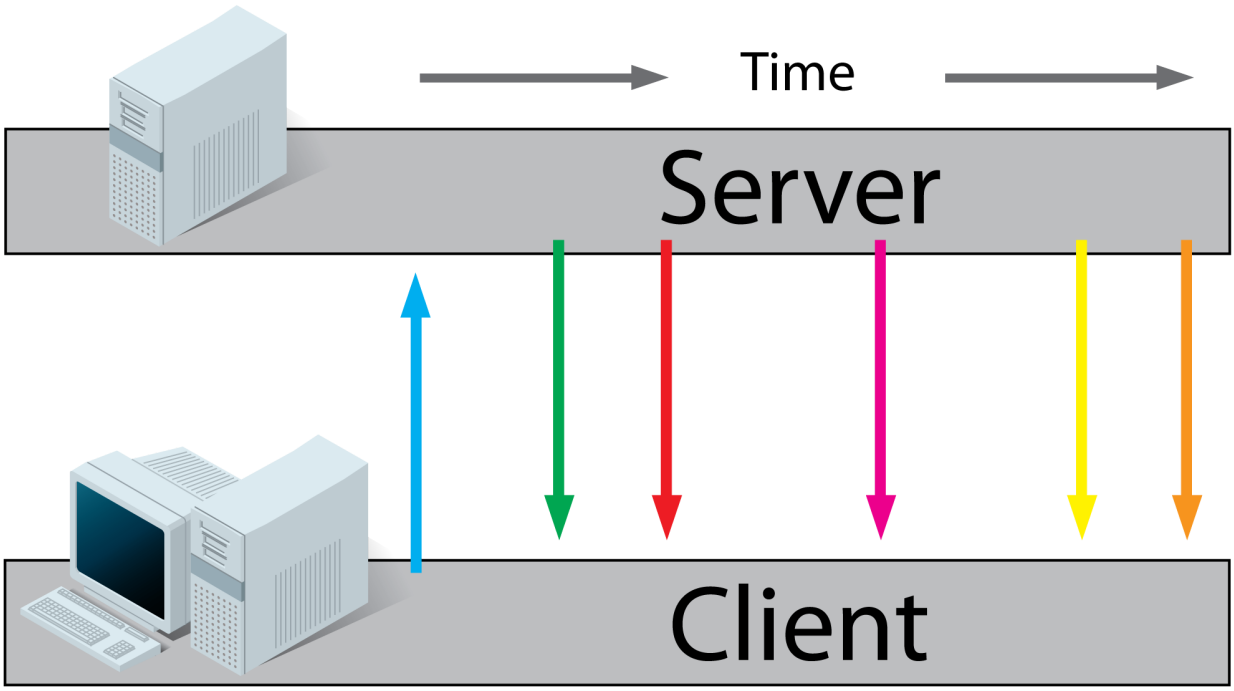
1. A client requests a webpage from a server using regular HTTP (see HTTP above).
2. The client receives the requested webpage and executes the JavaScript on the page which requests a file from the server.
3. The server does not immediately respond with the requested information but waits until there's **new** information available.

4. When there's new information available, the server responds with the new information.
5. The client receives the new information and immediately sends another request to the server, re-starting the process.



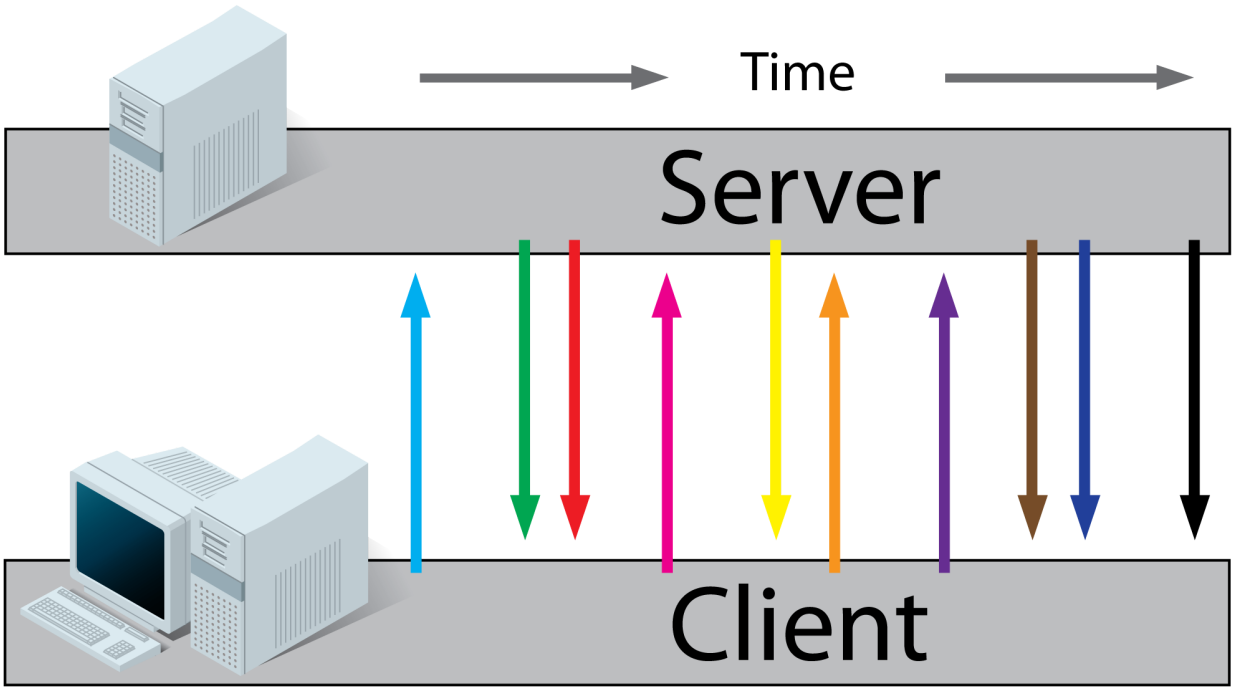
HTML5 Server Sent Events (SSE) / EventSource:

1. A client requests a webpage from a server using regular HTTP (see HTTP above).
2. The client receives the requested webpage and executes the JavaScript on the page which opens a connection to the server.
3. The server sends an event to the client when there's new information available.
    - Real-time traffic from server to client, mostly that's what you'll need
    - You'll want to use a server that has an event loop
    - Connections with servers from other domains are only possible with correct CORS settings
    - If you want to read more, I found these very useful: (article), (article), (article), (tutorial).



HTML5 Websockets:

1. A client requests a webpage from a server using regular http (see HTTP above).
2. The client receives the requested webpage and executes the JavaScript on the page which opens a connection with the server.
3. The server and the client can now send each other messages when new data (on either side) is available.
    - Real-time traffic from the server to the client **and** from the client to the server
    - You'll want to use a server that has an event loop
    - With WebSockets it is possible to connect with a server from another domain.
    - It is also possible to use a third party hosted websocket server, for example Pusher or others. This way you'll only have to implement the client side, which is very easy!
    - If you want to read more, I found these very useful: (article), (article) (tutorial).



Comet:

Comet is a collection of techniques prior to HTML5 which use streaming and long-polling to achieve real time applications. Read more on wikipedia or this article.

> Now, which one of them should I use for a realtime app (that I need to code). I have been hearing a lot about websockets (with socket.io [a node.js library]) but why not PHP ?

You can use PHP with WebSockets, check out Ratchet.

Share Improve this answer Follow

edited Feb 28 '19 at 9:26

answered Oct 12 '12 at 8:57

Tieme
53.3k 18 89 146

23 This is awesome! I am reading up on SSE and found this article, it's very nice - like I've now compared stuff, can you also include SSE here so we can also cross-check it's difference with Websocket? – index Nov 7 '12 at 7:34

1 @Tieme Oh was that it? I thought SSE meant Server-Sent Events. Anyway, thanks, I see it now. – index Nov 16 '12 at 3:28

5 You can accomplish the same with both solutions but the mechanism is different. Long-polling uses 'regular' http data, SSE uses a different underlying protocol and needs a different server setup compared to long-polling. – Tieme Oct 23 '13 at 7:53

2 Well you could use apache if you want. But a lot of people use Node.js because it has an event loop. But for Apache, see stackoverflow.com/questions/12203443/… – Tieme Nov 15 '13 at 9:45

2 @Tieme I know 2013 is a long time ago, but I'd like to point out SSE does not use a different protocol. I'ts just a variation of HTTP chunked encoding, and browsers accomodate it by setting TCP keepalive on the socket. It works with HTTP/2 as well, in contrast to websockets. – w00t Aug 2 '19 at 12:56

39

Tieme put a lot of effort into his excellent answer, but I think the core of the OP's question is how these technologies relate to PHP rather than how each technology works.

PHP is the most used language in web development besides the obvious client side HTML, CSS, and Javascript. Yet PHP has 2 major issues when it comes to real-time applications:

1. PHP started as a very basic CGI. PHP has progressed very far since its early stage, but it happened in small steps. PHP already had many millions of users by the time it became the embed-able and flexible C library that it is today, most of whom were dependent on its earlier model of execution, so it hasn't yet made a solid attempt to escape the CGI model internally. Even the command line interface invokes the PHP library (`libphp5.so` on Linux, `php5ts.dll` on Windows, etc) as if it still a CGI processing a GET/POST request. It still executes code as if it just has to build a "page" and then end its life cycle. As a result, it has very little support for multi-thread or event-driven programming (within PHP userspace), making it currently unpractical for real-time, multi-user applications.

Note that PHP does have extensions to provide event loops (such as `libevent`) and threads (such as `pthreads`) in PHP userspace, but very, very, few of the applications use these.

2. PHP still has significant issues with garbage collection. Although these issues have been consistently improving (likely its greatest step to end the life cycle as described above), even the best attempts at creating long-running PHP applications require being restarted on a regular basis. This also makes it unpractical for real-time applications.

PHP 7 will be a great step to fix these issues as well, and seems very promising as a platform for real-time applications.

Share Improve this answer

Follow

edited Jan 3 at 21:28

JoSSte

2,187 5 24 40

answered Oct 14 '14 at 7:11

JSON

1,648 17 26

2 One small correction: PHP was always written in C, as can be seen here: museum.php.net/php1 Also, "lesser used (but immensely more popular)" is rather self-contradictory; maybe what you mean is "more fashionable"? – IMSoP Dec 14 '14 at 22:16

1 @IMSoP - Thanks for the correction, I've been using PHP for over a decade and have always been under the impression that it's roots were in Perl. The PHP history page clearly supports that it was originally C as well. I'll edit my answer once I find a moment. – JSON Dec 15 '14 at 2:45

I'll remove the bit about Perl since it doesn't mix well with official documentation, but this is still a confusing area in PHP's early development. – JSON Feb 6 '15 at 7:23

PHP 7 seems very promising as a platform for real-time applications? What improvement/change in PHP7 for real-time applications? – I'll-Be-Back Aug 27 '15 at 20:25

1 though also svn.php.net/viewvc/phpdoc/en/trunk/appendices/… && web.archive.org/web/20090426061624/http://us3.php.net/… – eis Nov 1 '15 at 20:19

11

I have tried to make note about these and have collected and written examples from a **java perspective**.

HTTP for Java Developers

[Reverse Ajax - Old style]

[Async Handling on server side]

[Reverse Ajax - New style]

[Server Sent Events]

Putting it here for any java developer who is looking into the same subject.

Share Improve this answer Follow

edited Jun 20 '20 at 9:12
Community ♦
1 1

answered Apr 21 '17 at 14:47
John
9,835 20 62 106

@AlexanderDunn thank you for bringing it up. I'll correct it with updated links – John Dec 10 '19 at 14:18

1

You can easily use Node.JS in your web app only for real-time communication. Node.JS is really powerful when it's about WebSockets. Therefore "PHP Notifications via Node.js" would be a great concept.

See this example: [Creating a Real-Time Chat App with PHP and Node.js]

Share Improve this answer Follow

answered Apr 14 '19 at 16:38
Supun Kavinda
775 8 12

Highly active question. Earn 10 reputation in order to answer this question. The reputation requirement helps protect this question from spam and non-answer activity.