HTTP Push

In this lesson, we will learn about the HTTP Push mechanism.

We'll cover the following

Time To Live (TTL)

In the regular client-server communication, which is *HTTP PULL*, there is a *Time to Live (TTL)* for every request. It could be 30 secs to 60 secs, varies from browser to browser.

If the client doesn't receive a response from the server within the TTL, the browser kills the connection & the client has to re-send the request hoping it would receive the data from the server before the TTL ends this time.

Open connections consume resources & there is a limit to the number of open connections a server can handle at one point in time. If the connections don't close & new ones are being introduced, over time, the server will run out of memory. Hence, the TTL is used in client-server communication.

But what if we are certain that the response will take more time than the TTL set by the browser?

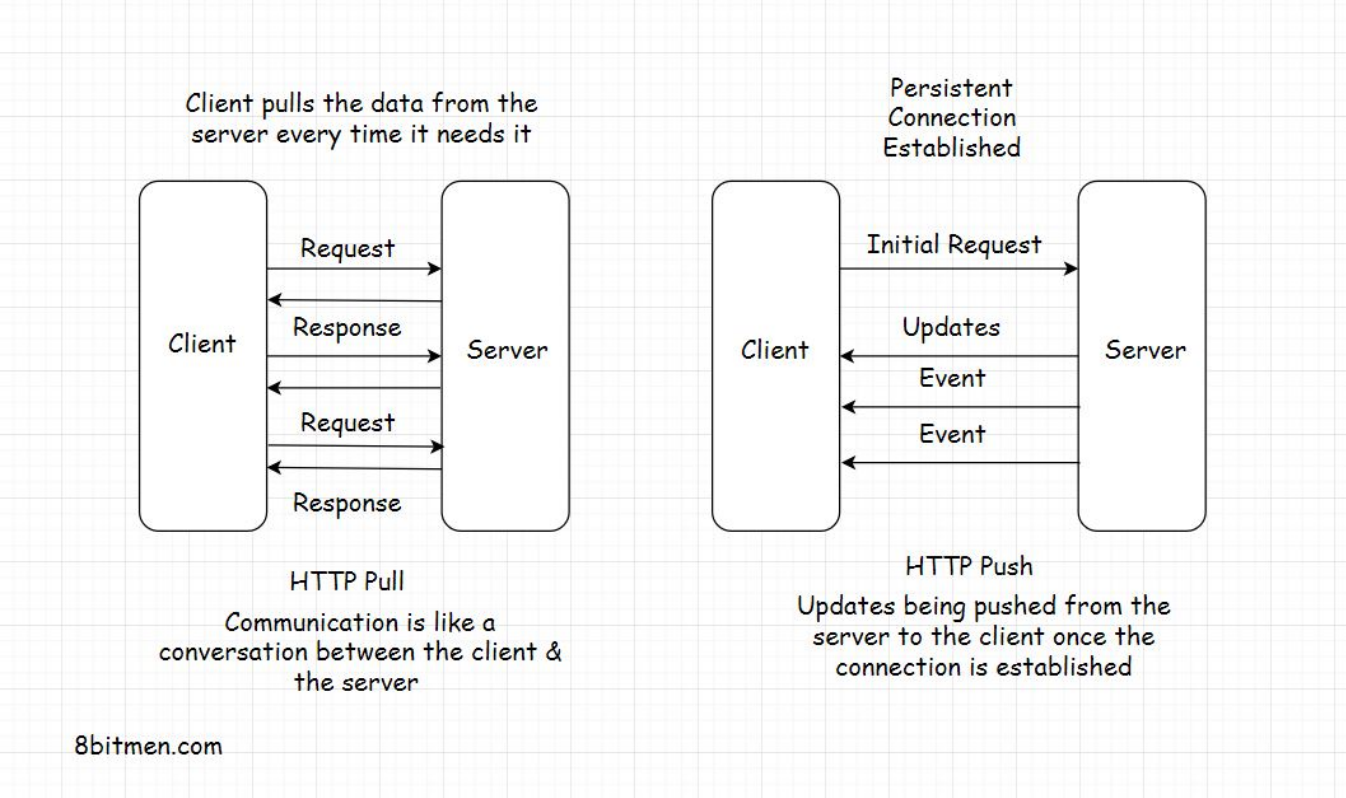
Persistent Connection

In this case, we need a *Persistent Connection* between the client and the server.

> A persistent connection is a network connection between the client & the

> server that remains open for further requests & the responses, as
>
> opposed to being closed after a single communication.

It facilitates HTTP Push-based communication between the client and the server.



Heartbeat Interceptors

Now you might be wondering how is a persistent connection possible if the browser kills the open connections to the server every X seconds?

The connection between the client and the server stays open with the help of *Heartbeat Interceptors*.

> These are just blank request responses between the client and the server to prevent the browser from killing the connection.

Isn't this resource-intensive?

Resource Intensive

Yes, it is. Persistent connections consume a lot of resources in comparison to the HTTP Pull behaviour. But there are use cases where establishing a

the HTTP Pull behaviour. But there are use cases where establishing a persistent connection is vital to the feature of an application.

For instance, a browser-based multiplayer game has a pretty large amount of request-response activity within a certain time in comparison to a regular web application.

It would be apt to establish a persistent connection between the client and the server from a user experience standpoint.

Long opened connections can be implemented by multiple techniques such as *Ajax Long Polling*, *Web Sockets*, *Server-Sent Events* etc.

Let's have a look into each of them.