**Short polling vs long polling vs Web sockets**

**Short Polling**

*Short polling is an AJAX-based timer that calls at fixed delays.*

In Short Polling Technique:

1. Client makes a request to the server
2. Server can respond in two ways:
   * It sends an empty response
   * It sends data object in its body (JSON Object)
3. As soon as a client receives the response from the server, it will wait for a couple of seconds and repeat the above process.

**Some challenges in short-polling:**

Making repeated requests to the server wastes resources as each new incoming connection must be established, the HTTP headers must be passed, a query for new data must be performed, and a response (usually with no new data to offer) must be generated and delivered. The connection must be closed and any resources cleaned up.

**Long Polling**

Long Polling works differently from short polling in the following way:

1. Client makes a request to the server
2. Server can respond in two ways:
   * If it has some new data available, it can respond right away.
   * If it doesn't have anything new data, it will keep that connection open for a period of time and when it receives new data it will respond back with updated data.

In short, it is a mechanism where the client continuously asks the server for new information using regular HTTP requests & the server stalls its answer when it has nothing new to report.

As long as the client makes sure it constantly has a polling request open, it will receive information from the server quickly after it becomes available.

To prevent connections from timing out (being aborted because of a lack of activity), long polling techniques usually set a maximum time for each request, after which the server will respond anyway, even though it has nothing to repeat, after which the client will start a new request.

Periodically restarting the request also makes the technique more robust, allowing clients to recover from temporary connection failures or server problems.

A busy server that is using long-polling may have thousands of waiting requests and thus TCP connections open. **NodeJS**, which makes it easy to manage many connections without creating a separate thread of control for each one is a good fit for such a system.

**Some challenges in long-polling:**

* Message ordering and delivery guarantees: Message ordering cannot be guaranteed if the same client opens multiple connections to the server.
* If the client was not able to receive the message then there will be possible message loss.
* Performance and scaling
* Device support and fallbacks

**Web Sockets**

*WebSocket is a computer communication protocol that provides full-duplex communication channels over a single TCP connection.*

The WebSocket protocol enables interaction between a client and a web server with lesser overheads, providing real-time data transfer from and to the server. WebSockets keeps the connection open, allowing messages to be passed back and forth between the client and the server. In this way, a two-way ongoing conversation can take place between the client and the server.

**Some advantages of Web Sockets over long-polling:**

* WebSockets keeps a unique connection open while eliminating latency problems that arise with Long Polling.
* Long polling is much more resource-intensive on servers whereas WebSockets have an extremely lightweight footprint on servers.
* WebSockets pass through most firewalls without any reconfiguration.
* Good security model (origin-based security model).

