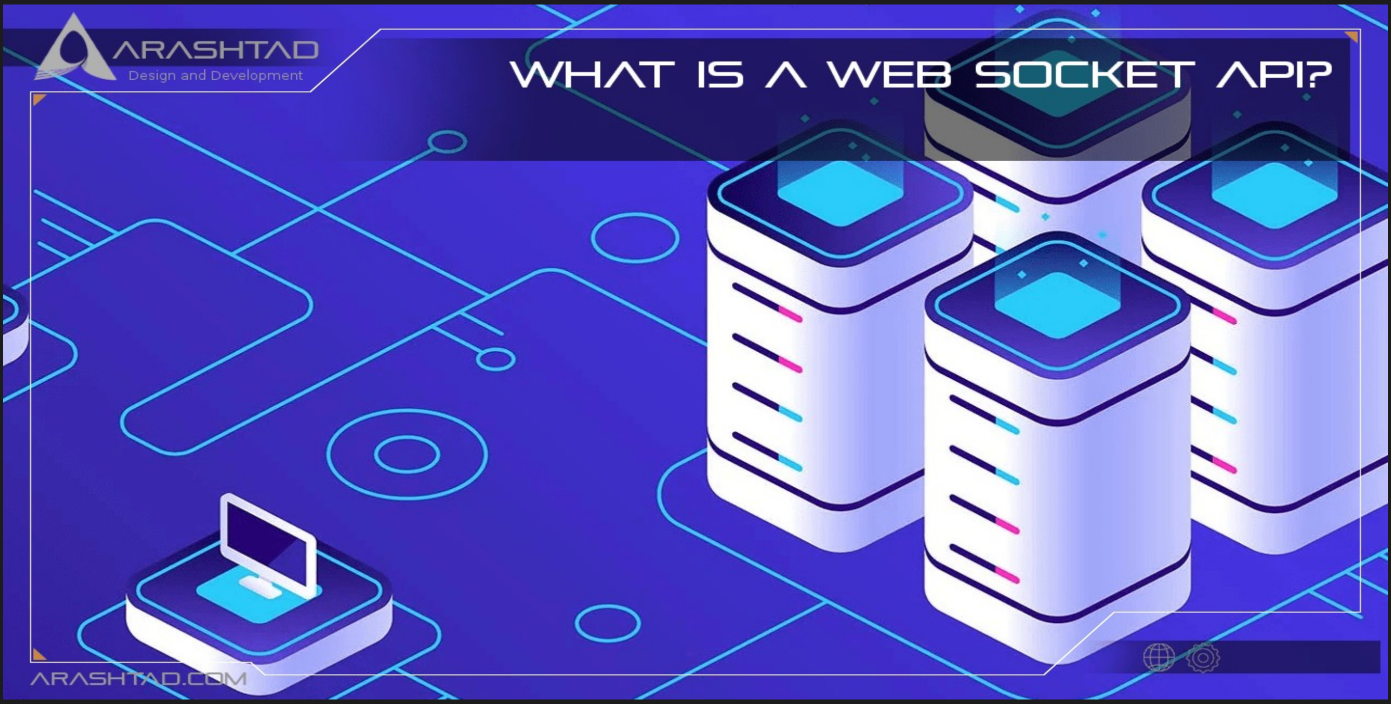
Title	WHAT IS A WEB SOCKET API?
Description	Intro
Date	2022 13 August
Author	Arashtad
Author URI	https://Arashtad.com



A WebSocket API is a modern technology that can establish a two-way interactive communication session between a user's browser and a server. By using this API, you can send messages to a server and receive event-driven responses without polling the server. Most commonly, WebSocket is a duplex protocol that is used in client-server communications. Client-server communication is bidirectional, which means it goes back and forth between the client and the server.

WHAT IS A WEB SOCKET?

A WebSocket is a computer communications protocol that provides full-duplex communication channels over a TCP connection. Through the use of the WebSocket, the connection lasts until one party decides to end it. A connection that is broken at one end prevents the other party from communicating with the first party. To initiate a connection, WebSocket needs HTTP support. When it comes to perfect streaming of data and other unsynchronized traffic, it serves as the spine for advanced web application development.

In contrast to half-duplex alternatives such as HTTP polling, WebSockets deliver real-time data transfer from and to a web server with low overhead. Messages can be passed back and forth while maintaining the connection, as the server has a standardized way to send content to the client without being requested by the client first. In this way, The client and server can have an ongoing two-way conversation. For the environments where non-web Internet connections are blocked using a firewall, communications are usually done over TCP port number 443 (or 80 for unsecured connections). WS (WebSocket) and WSS (WebSocket Secure) are two new uniform resource identifiers (URIs) that identify unencrypted and encrypted connections, respectively, in the WebSocket protocol specification. Aside from the scheme name and fragment (i.e. # is not supported), all other URI components follow URI generic syntax.

WHEN DO WE NEED A WEB SOCKET API?

To maximize the potential of WebSockets, one must be fully aware of their utility and avoid bad scenarios to take full advantage of them. The followings are the use cases of the web socket:
1. Developing a real-time trading web application:
WebSocket is most commonly used in real-time application development, allowing the client to view data continuously. Due to the continuous transmission of this data by the backend server, WebSocket allows this data to be transmitted or pushed continuously in the already open connection. With WebSockets, data is transmitted quickly and the performance of the application is boosted. A real-life example of such a WebSocket utility is in the bitcoin trading website. WebSockets play an important role in data transfer between a backend server and a client.
2. Developing Messaging Apps:
For operations such as one-time exchanges and publishing/broadcasting messages, chat application developers use WebSocket. Communication becomes simple and quick when WebSocket connections are used for sending and receiving messages.
3. Game Development:
The server must receive the data unremittingly during application development, without requesting UI updates. An application using WebSockets can achieve this without disrupting its UI.

Don’t forget to keep your operational hassles at bay by knowing the cases where WebSocket should not be used, now that you know where it should not be used. If old data fetching is needed or if data is needed only once, then WebSocket shouldn’t be used. Instead, HTTP protocols should be used in these circumstances.

WEB SOCKET PROTOCOL

In WebSocket protocol, various discrete chunks of data are carried out with each data chuck; the protocol also implements various frame types, portions of data, and payload lengths to ensure its proper functioning. To comprehend WebSocket protocol in detail, one must understand its building blocks. the foremost bits are listed below. A WebSocket's Fin Bit is the fundamental bit. it is automatically generated when a connection starts. the RSV1, RSV2, and RSV3 bits are a bit reserved for future opportunities. The opcode consists of a number that describes how the payload data of a specific frame is understood. The most common opcode values are 0x00, 0x0, 0x02, 0x0a, 0x08, and so on.
A mask bit is activated when one bit is set to 1. For all the payload data, WebSocket requires that the client select a random key. A masking key, when combined with payload data, assists in XORing payload data; in addition to preventing cache misinterpretation or cache poisoning, masking is important from the application API security perspective. Let's understand its crucial components in detail now:

Payload len
Describes the total length of the encoded payload data in WebSocket; Payload len is displayed when the encoded data length does not exceed 126 bytes; Once the payload data length exceeds 126 bytes, additional fields are displayed.
Masking-key
Client frames are masked with a 32-bit value. When the mask bit is 1, the masking key appears. when it's 0, the masking key doesn't appear.
Payload data
A payload is any kind of data related to an application or extension. This data is used during the early handshakes between the client and server.

WEB SOCKET API VS RESTFUL API

RESTful API and Web Socket API can be compared in so many different ways. RESTful API is stateless and thus we have no data storage whereas web socket API is stateful and data can be stored. RESTful API is one directional while the web socket API is bi-directional. Moreover, REST API follows the request-response model, whereas the Web socket API uses the Full Duplex model. Furthermore, HTTP request in REST API contains headers like head section, and title section. On the other hand, Web Socket API has no overhead and it is suitable for real-time applications. A new TCP connection will be set up for each HTTP request in a REST API, whereas in the Web Socket API, Only a single TCP connection is enough. To retrieve data in a RESTful API, it all depends upon the HTTP methods that are being used. While, in the web socket API, it depends upon the IP address and port number to retrieve the data. More to add, Web socket API is much quicker than the REST API.

DIFFERENCES BETWEEN WEBSOCKET AND HTTP:

Due to the fact that HTTP and WebSocket are both used for application communication, people often get confused and find it difficult to decide which to use. Take a look at the below-mentioned text and gain a better understanding of HTTP and WebSocket. As already mentioned, WebSocket is a bidirectional and framed protocol. whereas HTTP is a unidirectional protocol that acts on top of TCP. Due to WebSocket's capability to transmit data continuously, it's primarily used for developing real-time applications. HTTP, however, is stateless and can be used to build RESTful and SOAP applications. WebSocket is a protocol that allows communication at both ends, making it a faster protocol. HTTP, on the other hand, is a stateless protocol that is used to develop RESTful and SOAP applications. Soap can still be implemented via HTTP, but REST is widely used. HTTP must construct separate connections for separate requests. once the request is completed, the connection automatically breaks. WebSocket uses a unified TCP connection, which must be terminated by one party. Until it happens, the connection remains active.

HOW DO WEB SOCKET APIS WORK?

For a quick overview, WebSocket handshakes begin with WS or WSS, which are equivalent to HTTP or HTTPS, respectively. Using this scheme, both servers and clients are expected to follow the standard WebSocket connection protocol. A WebSocket connection is established by upgrading the HTTP request, which includes headers such as Connection: Upgrade, Upgrade WebSocket, Sec-WebSocket-Key, etc.

CONCLUSION

In this article, you learned about all the details of web socket API. What it is, How it works, its differences with HTTP and REST API, and its use cases. Web Socket API is very useful when it comes to an interactive and dynamic application that needs a quick response to the client. Examples of this kind of application could be in designing web-based games, trading websites, chat applications, and so on.

FOLLOW US