Review

1 min

Congratulations! In this lesson you have built a strong conceptual foundation of what WebSockets are and how they are created. With the growing number of real-time applications, WebSockets introduced an alternative to repetitive HTTP

Preview: Docs Loading link description

requests

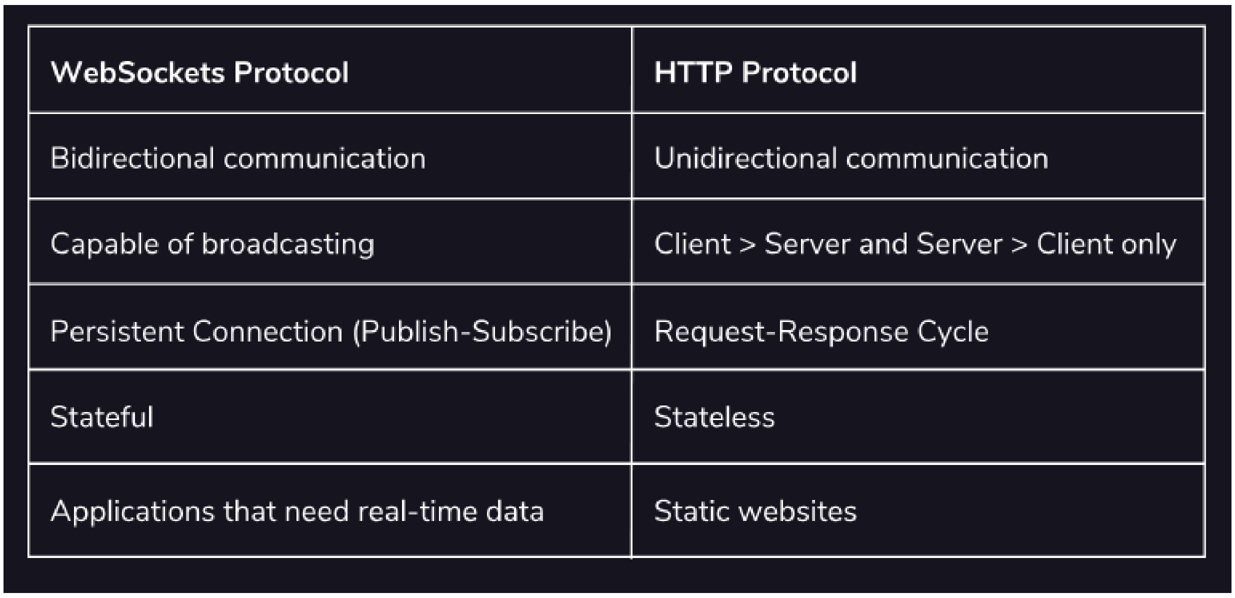
in order to build websites with continuous, bidirectional communication. Here's what else you learned:

- WebSockets allow for bidirectional communication between client and server allowing for a continuous flow of real-time data.
- Common communication patterns involve both client and server bi-directional communication along with broadcasting, which refers to a server being able to communicate with multiple clients.
- WebSocket connections are persistent giving them the advantage of lower overhead and being stateful compared to a stateless connection protocol like HTTP.
- The foundation for a WebSocket connection begins with an HTTP request and response called a *handshake* which aims to replace the HTTP protocol with a WebSocket protocol over a single TCP connection.
- A client can initiate a handshake with a special Upgrade header. A server can complete the handshake by sending back a response with the 101 Switching Protocols header.
- WebSockets are ideal for applications dealing with real-time data such as data trackers, multiplayer games, collaborative document editing, and social feeds and chat rooms.
- wss:// connections function just like ws:// ones, except that the initial handshake takes place over HTTPS instead of HTTP.

Now that you have a sense for what a WebSocket connection is and how they are formed we are ready to start building applications with WebSockets!

Instructions

The table to our right shows the contrast between WebSockets and traditional HTTP connections, revealing many of the benefits of using WebSockets for real-time applications.



WebSockets Protocol	HTTP Protocol
Bidirectional communication	Unidirectional communication
Capable of broadcasting	Client > Server and Server > Client only
Persistent Connection (Publish-Subscribe)	Request-Response Cycle
Stateful	Stateless
Applications that need real-time data	Static websites