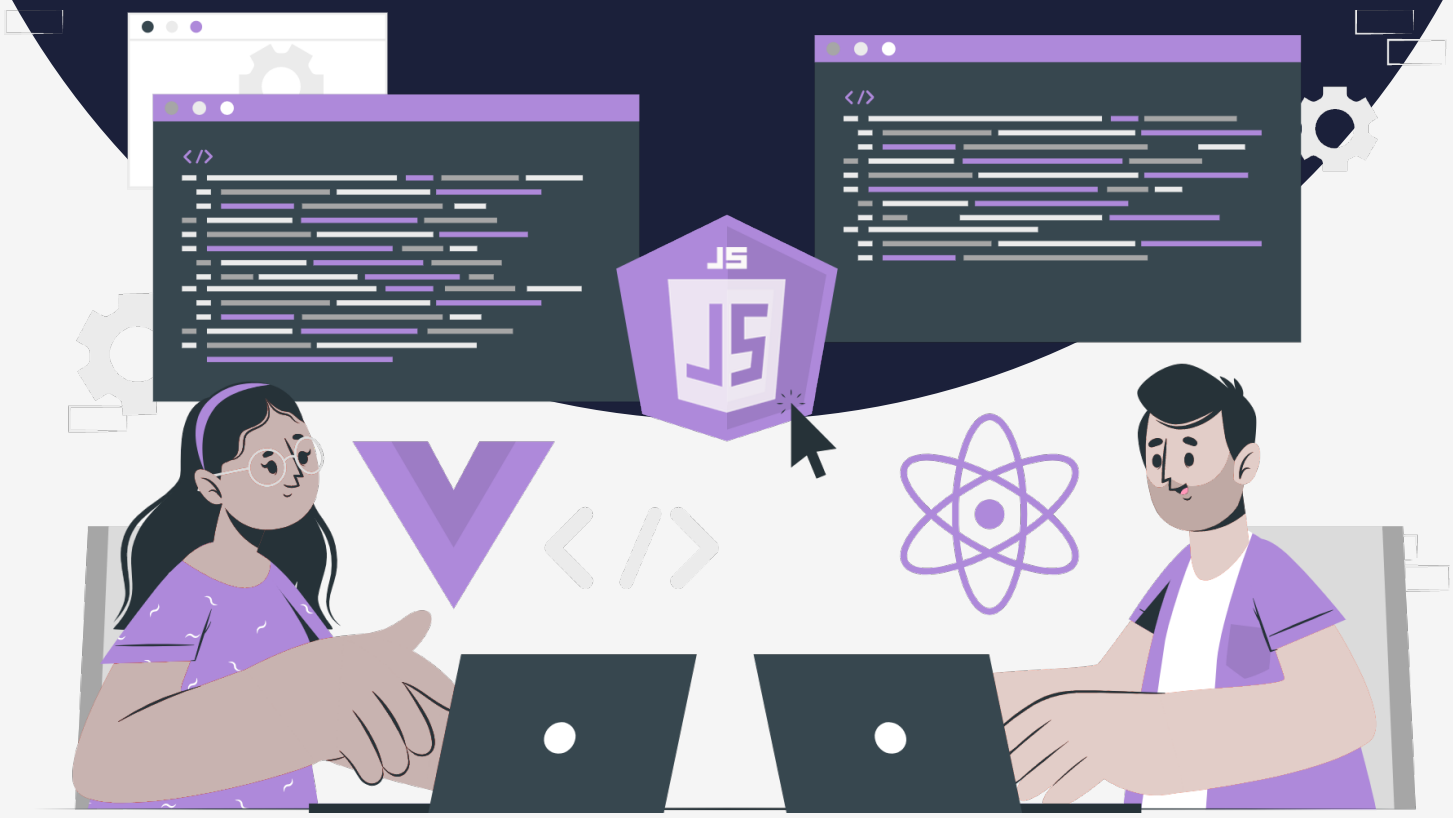
Lesson:

Web Socket



PW SKILLS

Topics Covered

1. Introduction.
2. Transmission Modes.
3. Web Sockets.
4. Why were web sockets created?
5. Working of Web Sockets.
6. Advantages of Web Sockets.
7. Disadvantages of web sockets.

WebSockets is like a superpower for web developers! They are a technology that allows real-time communication between web browsers and servers, enabling developers to create highly interactive web applications. With WebSockets, you can build applications that respond instantly to user actions, display live data feeds, and much more. Think of WebSockets as a magical portal that connects your web browser to the server, allowing you to communicate in real time without having to constantly refresh the page. It's an exciting technology that is revolutionizing how we interact with the web, and once you start using it, you'll wonder how you ever built web applications without it!

Transmission Modes.
Before understanding web sockets you need to understand the transmission modes.
Transmission modes refer to the direction of data flow between two communication devices in a computer network.

There are three types of transmission modes:

1. Simplex: In simplex mode, data can only flow in one direction. One device is the sender and the other is the receiver. The receiver can only receive data, and the sender can only transmit data. Examples of simplex mode communication include a radio broadcast or a television transmission.
2. Half-Duplex: In half-duplex mode, data can flow in both directions, but not simultaneously. Only one device can transmit data at a time, while the other device receives it. The devices take turns transmitting and receiving data. Examples of half-duplex mode communication include a walkie-talkie or a two-way radio.
3. Full-Duplex: In full-duplex mode, data can flow in both directions simultaneously. Both devices can transmit and receive data at the same time, without waiting for each other. Examples of full-duplex mode communication include a telephone call or a video conference.

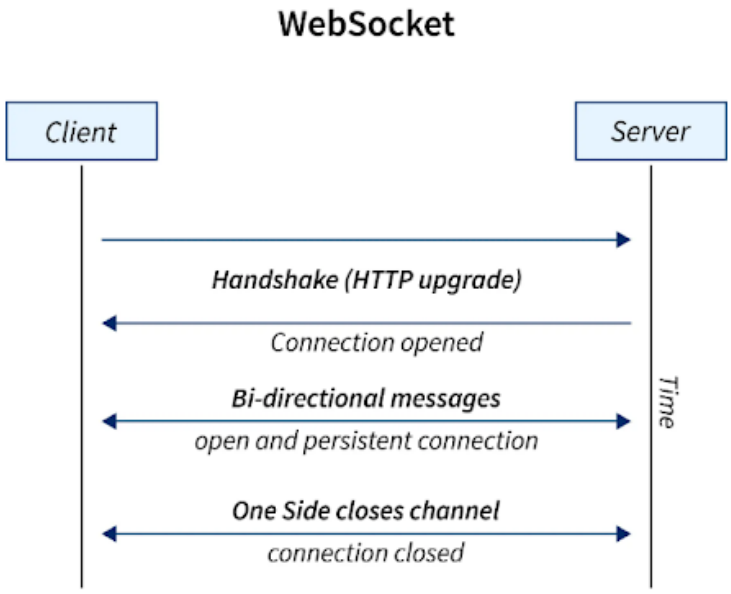
In computer networks, transmission modes are important because they determine the type of communication that can take place between two devices. Choosing the appropriate transmission mode is crucial for ensuring efficient and effective communication.

Web Sockets.
A WebSocket is a communication protocol that enables full-duplex and bidirectional data transfer between a client and a server over the internet. Unlike the traditional HTTP protocol, which is unidirectional and stateless, WebSockets enable real-time communication, allowing data to flow back and forth between the server and the client in real time.

Once a WebSocket connection is established, it is kept open until either the client or the server terminates it. This means that the connection remains "stateful", allowing data to be transmitted in real-time between the two endpoints.

When the connection is closed, either by the client or the server, the connection is terminated from both ends. This means that both the client and the server can initiate the closure of the connection.

WebSockets have become an important tool in internet architecture, as they enable real-time communication between clients and servers, and they are used in various applications, such as online gaming, chat applications, and financial trading systems, to name a few. Their ability to provide real-time, bi-directional communication has made them an essential component of modern web development.



Why were web sockets created?

The traditional HTTP protocol used in web development is designed for client-server communication, where the client sends a request to the server and the server responds with a static or dynamic resource. However, this communication is unidirectional and lacks real-time interaction between the client and server.

With the increasing demand for real-time web applications, there was a need for a new technology that allowed for bidirectional, real-time communication between the client and server. This is where WebSockets come in.

WebSockets provide a persistent, low-latency connection between the client and server, allowing for real-time, bidirectional communication. They use a message-based format to send and receive data and can transmit data in either text or binary format. This makes it possible to create real-time web applications such as chat applications, multiplayer games, and collaborative editing tools.

Before WebSockets, real-time communication in web applications was achieved using techniques such as polling or long-polling, where the client would repeatedly send requests to the server to check for new data. This approach was inefficient and put a strain on server resources. With WebSockets, the connection between the client and server is kept open, eliminating the need for repeated requests and reducing server load.

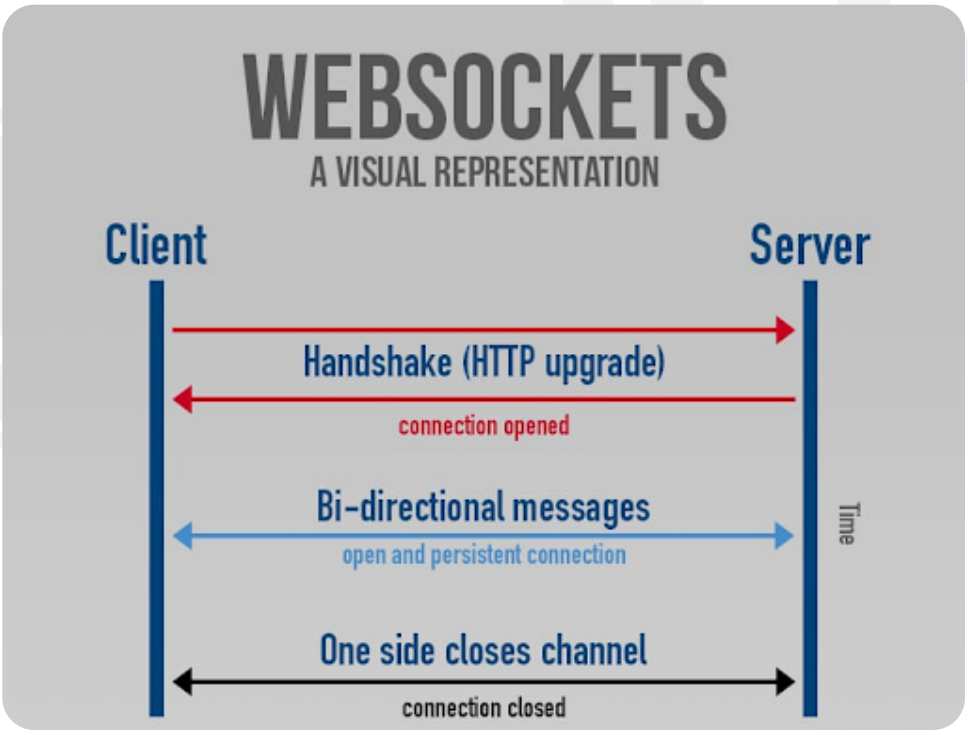
Thus, WebSockets were developed to address the need for bidirectional, real-time communication in web applications, and provide a more efficient and effective way to create real-time web applications.

Working of Web Sockets.

WebSockets allow for real-time, bidirectional communication between a client and a server. The WebSocket protocol basically follows the below-mentioned steps:

1. The client sends a WebSocket handshake request to the server over an HTTP or HTTPS connection. This handshake request contains the URL of the WebSocket server and a set of headers that describe the WebSocket protocol and other parameters.
2. If the server accepts the WebSocket handshake request, it responds with a WebSocket handshake response. This response includes a set of headers that describe the WebSocket protocol and other parameters. Once the client receives this response, the WebSocket connection is established.
3. Once the WebSocket connection is established, the client and server can send and receive data in real-time. The client can send data to the server using a WebSocket send() method, and the server can send data to the client using a WebSocket send() method.
4. The WebSocket connection remains open until either the client or server closes the connection. Once the connection is closed, the WebSocket object is disposed of and can no longer be used.

The WebSocket protocol uses a message-based format to send and receive data. Messages can be in text or binary format, and can be of any length. The WebSocket protocol also includes mechanisms for error handling, authentication, and compression.



Advantages of Web Sockets.

WebSockets provide a number of advantages over traditional HTTP-based communication protocols. Here are some of the key advantages of WebSockets:

1. Real-time communication: WebSockets provide a persistent, low-latency connection between the client and server, allowing for real-time, bidirectional communication. This makes it possible to create real-time web applications such as chat applications, multiplayer games, and collaborative editing tools.
2. Reduced network overhead: WebSockets use a persistent connection, eliminating the need for repeated requests and reducing network overhead. This makes WebSockets more efficient than traditional HTTP-based communication protocols such as polling or long-polling.
3. Scalability: WebSockets allow for multiple connections to be open at the same time, allowing for a high degree of scalability. This is especially important for applications that require real-time, bidirectional communication between many clients and servers.
4. Reduced server load: WebSockets reduce the server load by eliminating the need for repeated requests and reducing network overhead. This makes WebSockets more efficient than traditional HTTP-based communication protocols, reducing server load and improving application performance.
5. Flexibility: WebSockets can transmit data in either text or binary format, making them highly flexible and adaptable to a wide range of applications.
6. Security: WebSockets use the same security protocols as HTTP, such as SSL/TLS, to ensure secure communication between the client and server.

Disadvantages of web sockets.

Although WebSockets provide many advantages over traditional HTTP-based communication protocols, they also have some disadvantages. Here are some of the key disadvantages of WebSockets:

1. Browser compatibility: Although WebSockets are supported by most modern browsers, there are still some older browsers that do not support WebSockets. This can make it difficult to create applications that are compatible with all browsers.
2. Firewall and proxy issues: Some firewalls and proxies may block WebSocket traffic, making it difficult to establish WebSocket connections between clients and servers. This can limit the usability and accessibility of WebSockets.
3. Server resource utilization: WebSockets can consume more server resources than traditional HTTP-based communication protocols, especially if many clients are connected simultaneously. This can impact server performance and scalability.