WebSockets in Node.js

1. Need for WebSockets

In real-time applications like chat apps, live notifications, or online gaming, the server often needs to send updates to the client instantly. Traditionally, this was done using **HTTP polling**:

- The client repeatedly sends requests to the server asking "Any new messages?".
- The server responds with the data if available.

Problems with polling: - High latency: The client may not get the message immediately. - Inefficient: Many requests may return empty responses. - High server load: Repeated requests for the same data.

WebSockets solve these problems by allowing a **persistent, bidirectional connection** between client and server. This enables the server to push updates instantly without waiting for the client to request.

2. Bidirectional Communication

Bidirectional communication means: - Client can send data to server anytime. - Server can send data to client anytime.

Example in a chat app: - Client → Server: "Send message to user X" - Server → Client: "You have a new message from user Y"

This is essential for real-time features.

3. How WebSockets Work

1. Client sends an **HTTP request with** `Upgrade` **header** to switch protocols to WebSocket.
2. Server responds with `101 Switching Protocols` if it supports WebSocket.
3. After the handshake, both client and server can send messages freely.

4. Using WebSockets in Node.js

Using `ws` **Module**

Installation:

```
npm install ws
```

Server (server.js):

```
const WebSocket = require('ws');
const wss = new WebSocket.Server({ port: 8080 });
```

```
wss.on('connection', (ws) => {
  console.log('New client connected');

  ws.on('message', (message) => {
    console.log(`Received: ${message}`);
    // Broadcast to all clients
    wss.clients.forEach((client) => {
      if (client.readyState === WebSocket.OPEN) {
        client.send(`Server: ${message}`);
      }
    });
  });

  ws.send('Welcome to the chat!');
});
```

Client (HTML):

```
<!DOCTYPE html>
<html>
<body>
  <input id="msg" placeholder="Type message" />
  <button onclick="sendMessage()">Send</button>
  <ul id="messages"></ul>

  <script>
    const ws = new WebSocket('ws://localhost:8080');
    const messages = document.getElementById('messages');

    ws.onmessage = (event) => {
      const li = document.createElement('li');
      li.textContent = event.data;
      messages.appendChild(li);
    };

    function sendMessage() {
      const input = document.getElementById('msg');
      ws.send(input.value);
      input.value = '';
    }
  </script>
</body>
</html>
```

Using Headers Upgrade Manually

WebSocket starts as a normal HTTP request:

```
GET /chat HTTP/1.1
Host: server.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: <base64>
Sec-WebSocket-Version: 13
```

Server responds with:

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: <calculated-key>
```

After this handshake, the connection is upgraded to WebSocket and both client and server can send messages freely.

Summary: - WebSockets eliminate the need for polling. - Enable real-time bidirectional communication. - Essential for chat apps, live notifications, and gaming. - Node.js makes it easy with the `ws` library or manual HTTP Upgrade handling.