Web Sockets Js

→ It allows for real time bi directional communication b/w a client (Browser) and a server.

↳ Useful for building Interactive web apps that require instant data updates.

↳ They provide → Full duplex communication channel over → single → Tcp Connection

↳ Unlike HTTP → Enable both (client, Server) → to send message → each other any time

← without ← waiting for request from client

↳ It → uses → protocol → starts with → [ws://] → Unencrypted

↳ [wss://] → Encrypted

Instead of → http:// / https://

↳ Create Web Socket Connection

Create → new → Web Socket object → passing → Web Socket Server (parameter as URL)

```
<script>
    Const Socket = new WebSocket("ws://example.com/socket");
    console.log(Socket);
</script>
```

Console o/p

```
WebSocket : {url: "ws://___",
             readyState : 0,
             bufferedAmount : 0,
             onopen : null,
             onerror : null
             ...
            }
```

WebSocket Events

These Objects → emit → various → events → to handle → diff stage of the connection

- open → Triggered when conn is established successfully.
- message → Fired when server sends a message.
- error → Triggered when error occurs.
- close → Fired when connection is closed.

MessageEvent { isTrusted: true, data: { "error": "Unknown api key" }, origin: 'ws://...', lastEventId: '', source: null, ... }

- Sending Data: To send data to server → use → send() (WebSocket object)

Data can → String
→ Blob
→ ArrayBuffer

- Server side WebSocket Implementation:

On server side → need → implement → WebSocket server → listens → for WebSocket connections

handle message from client.

- Error Handling
- Security Considerations
- Use cases
  - Real Time chat application
  - Online gaming
  - Live Notifications
  - Collaborative Tools
  - Any scenario → Immediate Data update are required.

WebSockets in JavaScript allow for real-time, bidirectional communication between a client (usually a web browser) and a server.

They are particularly useful for building interactive web applications that require instant data updates.

Creating a WebSocket Connection

To establish a WebSocket connection in JavaScript, create a new WebSocket object, passing the WebSocket server URL as a parameter.

```
const socket = new WebSocket('ws://example.com/socket');
```

WebSocket Events

WebSocket objects emit various events to handle different stages of the connection.

open: Triggered when the connection is successfully established

```
socket.addEventListener('open', (event) => {
  // Connection is open.
});

socket.addEventListener('message', (event) => {
  const message = event.data;
  // Handle incoming message.
});
```

message: Fired when the server sends a message

error: Triggered when an error occurs

```
socket.addEventListener('error', (event) => {
  console.error('WebSocket error:', event);
});

socket.addEventListener('close', (event) => {
  if (event.wasClean) {
    console.log(`Connection closed cleanly,
code=${event.code}, reason=${event.reason}`);
  } else {
    console.error('Connection abruptly closed');
  }
});
```

close: Fired when the connection is closed

Sending Data

To send data to the server, use the **send** method of the WebSocket object. Data can be a string, ArrayBuffer, or Blob.

```
socket.send('Hello, server!');
```

Closing the Connection

To close the WebSocket connection, call the **close** method on the WebSocket object.

```
socket.close();
```

Server-Side WebSocket Implementation

On the server side, you need to implement a WebSocket server that **listens** for WebSocket connections and **handles** messages from clients. Popular libraries for this purpose include ws for Node.js and libraries for various languages like Python, Ruby, and Java.

Error Handling

Be sure to handle errors gracefully by listening for the error event and providing appropriate feedback to the user.

9/10

Security Considerations

When using WebSockets, consider security measures like encrypting the connection using `wss://`, implementing authentication, and validating incoming messages to prevent vulnerabilities like WebSocket-based attacks.

Use Cases

WebSockets are suitable for real-time chat applications, online gaming, live notifications, collaborative tools, and any scenario where immediate data updates are required.

WebSockets in JavaScript

Q: How do WebSockets enable real-time communication in web applications?

A: WebSockets provide a persistent connection between a client and server, allowing for full-duplex communication.

```
const socket = new WebSocket('ws://example.com/ws');

socket.onmessage = (event) => {
    console.log('Message from server ', event.data);
};

socket.send('Hello server!');
```

```
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Web Sockets</title>
</head>

<body>
    <script>
        const socket = new WebSocket("wss://demo.piesocket.com/v4/channel_123?api_key=VCXCEuvhGcBDP7XhiJJUDvR1e1D3eiVjgZ9VRiaV&notify_self");
        console.log(socket);

        socket.addEventListener('open', (event) => {
            //Connection is open
        });

        socket.addEventListener('message', (event) => {
            const message = event.data;
            console.log(event);
            console.log(message);
            //Handle Incoming Messages
        });

        socket.addEventListener('error', (event) => {
            console.log("Websocket Error : " + event);
        });

        socket.addEventListener('close', (event) => {
            if (event.wasClean) {
                console.log(
                    "Connection Closed successfully!..",
                    code = `${event.code}`,
                    reason = `${event.reason}`
                );
            } else {
                console.log("Connection abrutply closed!..");
            }
        });
    </script>
</body>

</html>
```

websocket_SendMessages.html:13

WebSocket {url: 'wss://demo.piesocket.com/v4/channel_123?api_key=
VCXCEuvhGcBDP7XhiJJUDvR1e1D3eiVjgZ9VRiaV&notify_self', readyStat
e: 0, bufferedAmount: 0, onopen: null, onerror: null, …}

Uncaught DOMException: Failed to execute 'send' on 'WebSocket': Still in CONNECTING state. websocket_SendMessages.html:22

at http://127.0.0.1:5500/JS_WebSockets/websocket_SendMessages.html:22:16

WebSocket connection to 'wss://demo.piesocket.com/v4/channel_123?api_key=VCXCEuvhGcBDP7XhiJJUDvR1' failed: websocket_SendMessages.html:12

>

websocket_CloseConnection.html:13

WebSocket {url: 'wss://demo.piesocket.com/v4/channel_123?api_key=VCXCEuvhGcBDP7XhiJJUDvR1e1D3eiVjgZ9VRiaV&notify_self', readyState: 0, bufferedAmount: 0, onopen: null, onerror: null, …}

Uncaught DOMException: Failed to execute 'send' on 'WebSocket': Still in CONNECTING state.
    at http://127.0.0.1:5500/JS_WebSockets/websocket_CloseConnection.html:22:16

websocket_CloseConnection.html:22

WebSocket connection to 'wss://demo.piesocket.com/v4/channel_123?api_key=VCXCEuvhGcBDP7XhiJJUDvR1' failed:

websocket_CloseConnection.html:12