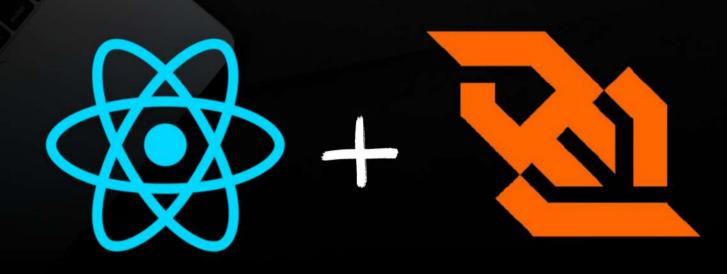
REACT JS



WEBSOCKETS USING REACT





WHAT ARE WEBSOCKETS?

WebSockets are a protocol used for real-time, interactive communication between a client (like a web browser) and a server. Unlike HTTP, a one-way request-response model, WebSockets use a bidirectional, full-duplex connection. This means that once a WebSocket connection is established, the client and the server can send messages to each other at any time, without repeatedly requesting information or polling the server.

Here's a brief rundown:

- Initiation: Starts with a handshake using ws:/ or wss:/ in the URL.
- Communication: The connection stays open for continuous, two-way data exchange, more efficient than HTTP.
- **Lifecycle**: Remains active until either side closes it; both are notified when it terminates.



SETTING UP A WEBSOCKET SERVER IN NODE

```
server.ts
import WebSocket, { WebSocketServer } from 'ws'; **
const wss = new WebSocketServer({ port: 8080 });
wss.on('connection', (ws: WebSocket) => {
  console.log('New client connected');
  ws.on('message', (message: string) => {
    console.log(`Received message: ${message}`);
    ws.send(`Echo: ${message}`);
  });
  ws.on('close', () => {
    console.log('Client disconnected');
 });
});
```



ASHFAQUE AHMED

Explanation:

- WebSocketServer creates a WebSocket server on port 8080.
- WebSocket is a type for the socket.
- connection event triggers when a client connects.
- message event handles incoming messages and sends an echo back to the client.
- close event handles disconnections.



1NTEGRATING WEBSOCKET IN REACT

```
...
                    WebSocketComponent.tsx
import React, { useEffect, useState } from 'react';
const WebSocketComponent: React.FC = () => {
  const [message, setMessage] = useState<string>('');
  const [ws, setWs] = useState<WebSocket | null>(null);
 useEffect(() => {
    const socket = new WebSocket('ws://localhost:8080');
    setWs(socket);
    socket.onmessage = (event) => {
      setMessage(event.data);
   };
    return () => {
      socket.close();
   };
  }, []);
  const sendMessage = () => {
    if (ws) {
      ws.send('Hello Server!');
  };
 return (
    <div>
      <button onClick={sendMessage}>Send Message</putton>
      Message from server: {message}
    </div>
  );
};
export default WebSocketComponent;
```

Explanation:

- useEffect sets up the WebSocket connection when the component mounts.
- ws.onmessage handles incoming messages.
- sendMessage function sends a message to the server.
- Cleanup: WebSocket is closed when the component unmounts, return in useEffect is run when a component unmounts.

04 HANDLING ERRORS AND RECONNECTION

```
...
                       WebSocketComponent.tsx
import React, { useEffect, useState } from 'react';
const WebSocketComponent: React.FC = () => {
  const [message, setMessage] = useState<string>('');
  const [ws, setWs] = useState<WebSocket | null>(null);
  const [error, setError] = useState<string>('');
  useEffect(() => {
    const connect = () => {
      const socket = new WebSocket('ws://localhost:8080');
      setWs(socket);
      socket.onmessage = (event) => {
        setMessage(event.data);
      };
      socket.onerror = (event) => {
       setError('WebSocket error');
      };
      socket.onclose = () => {
        setError('WebSocket closed, trying to reconnect...');
        setTimeout(connect, 3000); // Reconnect after 3 seconds
      };
    };
   connect();
    return () => {
      if (ws) {
        ws.close();
  }, [ws]);
  const sendMessage = () => {
    if (ws) {
     ws.send('Hello Server!');
  };
  return (
    <div>
      <button onClick={sendMessage}>Send Message</button>
      Message from server: {message}
      Error: {error}
    </div>
  );
};
export default WebSocketComponent;
```

Explanation:

- Error Handling: socket.onerror captures WebSocket errors.
- Reconnection: socket.onclose triggers reconnection logic after a short delay.



THANK YOU FOR READING

FOLLOW FOR MORE POSTS ON REACT...

FOR SERVICES: WHATSAPP 03422076556

