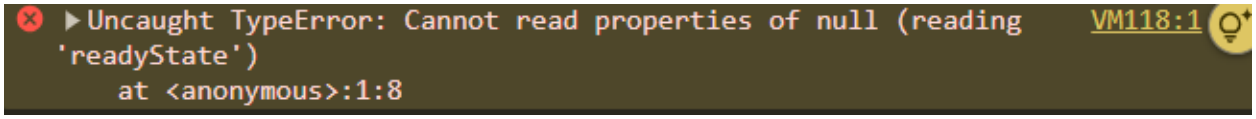
Before Opening the Connection

- Open the Console and type:

  ```
  socket.readyState
  ```



This happens before I created the connection

1. What number do you see? What does that value represent in the WebSocket lifecycle?

The number I see is a 1, and it represents the socket being created and the user being connected to it

Immediately After Clicking "Open Connection"

- Type again:

  ```
  socket.readyState
  ```

2. What value do you see now? Which constant does it match (CONNECTING, OPEN, CLOSING, or CLOSED)?

Connecting Code is 0

Open is 1

Connection closing is 2

Connection closed is 3

When the "Status: Connected ✅" Appears

- Check `socket.readyState` again.

3. What value is displayed? What does that tell you about the connection's ability to send and receive data?

The value that is displayed is 1, and that means the connection is open and is ready to communicate with the user

While the Connection Is Still Open

- Type several messages using the Send button.

4. Does `socket.readyState` change during normal message exchange? Why or why not?

The socket.readyState does not change during a normal messaging exchange because if it did change, that would represent another ready state, such as closing or closed.

After Clicking "Close Connection"

- Check `the socket.readyState` immediately after you click Close.

5. What value do you see while the connection is shutting down? What value do you see a few seconds later, after it fully closes?

Well, since the server I am running is fast, I only see the code of 3, meaning closed. However, if there are a lot of people on my server and I am shutting it down, they would see a ready state of 2, meaning closing.

Reflection Questions

1 . **How is a WebSocket connection different from an HTTP request?**

(Think about persistence, who initiates communication, and when the connection closes.)

Websockets differ from HTTP requests because they are Bi-directional, while HTTP requests are unidirectional. Websockets remain open until the client closes the connection, whereas HTTP connections close after the request-response cycle.

2. **Why are WebSockets ideal for real-time applications such as chats, dashboards, or multiplayer games?**

(Explain how the connection model supports continuous updates.)

Websockets are ideal for real-time applications because they are persistent, Bidirectional, and low-latency.

3. **Why is WebSocket communication considered *stateful*?**

(Relate this to how `readyState` and event handlers maintain ongoing connection context.)

WebSocket communication is stateful because the persistent connection requires both the client and server to maintain the connection's current context, as indicated by the readyState.

4. ***What do the `readyState` values reveal about the lifecycle of a WebSocket?***
(How does each state—CONNECTING, OPEN, CLOSING, CLOSED—help a developer manage program logic?)

The readyState values (CONNECTING (0), OPEN (1), CLOSING (2), CLOSED (3)) show the current status of the WebSocket connection.

5. ***If two users connect to the same WebSocket server at the same time, how might each user's `readyState` differ during setup or closing?***
(Think about concurrency and timing differences between independent client sessions.)

During setup and closing, the readyState of two users connected to the same server can differ. Each client has its own state machine. Network latency, server concurrency, and the specific timing of user or network actions cause state transitions.