Code Examples & Use Cases

This file contains practical examples of how to use and extend the WebSocket implementation.

🎯 Basic Usage Examples

Example 1: Simple Echo Test

Client Code (Browser Console)

```
// Connect to server
const ws = new WebSocket("ws://localhost:1338");

ws.onopen = () => {
  console.log("Connected!");
  ws.send("Hello, Server!");
};

ws.onmessage = (event) => {
  console.log("Received:", event.data);
};
```

Expected Output

```
Connected!
Received: Echo: Hello, Server!
```

Example 2: Multiple Messages

Client Code

```
const messages = ["Message 1", "Message 2", "Message 3", "Testing 123"];

ws.onopen = () => {
  messages.forEach((msg, index) => {
    setTimeout(() => {
      ws.send(msg);
      console.log(`Sent: ${msg}`);
    }, index * 1000); // Send every second
  });
};

ws.onmessage = (event) => {
  console.log(`Server replied: ${event.data}`);
};
```

Example 3: Measuring Latency

Client Code

```
let sentTime;

ws.onopen = () => {
  sentTime = performance.now();
  ws.send("PING");
};

ws.onmessage = (event) => {
  const latency = performance.now() - sentTime;
  console.log(`Round-trip latency: ${latency.toFixed(2)}ms`);
};
```

🔧 Server Extension Examples

Example 4: Broadcast to All Clients

server.mjs addition

```
// Store all connected clients
const clients = new Set();

function onSocketUpgrade(request, socket, head) {
  const { "sec-websocket-key": webSocketKey } = request.headers;
  console.log(`\n🔄 Upgrading to WebSocket connection`);

  const header = prepareHandshakeHeader(webSocketKey);
  socket.write(header);
  console.log(`✅ Handshake complete\n`);

  // Add client to set
  clients.add(socket);

  socket.on("readable", () => onSocketreadable(socket));

  socket.on("end", () => {
    console.log("👋 Client disconnected");
    clients.delete(socket); // Remove from set
  });

  socket.on("error", (err) => {
    console.error("❌ Socket error:", err.message);
    clients.delete(socket);
```

```
  });
}

// Broadcast function
function broadcast(message, excludeSocket = null) {
  for (const client of clients) {
    if (client !== excludeSocket && client.writable) {
      sendMessage(message, client);
    }
  }
}

// Modified message handler
function onSocketreadable(socket) {
  // ...existing frame parsing code...

  const receivedMessage = decoded.toString("utf8");
  console.log(`📩 Received: "${receivedMessage}"`);

  // Broadcast to all other clients
  broadcast(`[Broadcast] ${receivedMessage}`, socket);

  // Still send echo to sender
  sendMessage(`Echo: ${receivedMessage}`, socket);
}
```

Example 5: Room-Based Chat

server.mjs addition

```
// Room management
const rooms = new Map(); // roomName -> Set of sockets
const socketRooms = new Map(); // socket -> roomName

// Join room command: JOIN:roomName
// Message format: MSG:message
function onSocketreadable(socket) {
  // ...existing frame parsing code...

  const receivedMessage = decoded.toString("utf8");

  if (receivedMessage.startsWith("JOIN:")) {
    const roomName = receivedMessage.slice(5);
    joinRoom(socket, roomName);
    sendMessage(`Joined room: ${roomName}`, socket);
  } else if (receivedMessage.startsWith("MSG:")) {
    const message = receivedMessage.slice(4);
    const roomName = socketRooms.get(socket);
    if (roomName) {
      broadcastToRoom(roomName, message, socket);
```

```
    }
  } else {
    sendMessage(`Echo: ${receivedMessage}`, socket);
  }
}

function joinRoom(socket, roomName) {
  // Leave current room if any
  const currentRoom = socketRooms.get(socket);
  if (currentRoom) {
    rooms.get(currentRoom)?.delete(socket);
  }

  // Join new room
  if (!rooms.has(roomName)) {
    rooms.set(roomName, new Set());
  }
  rooms.get(roomName).add(socket);
  socketRooms.set(socket, roomName);

  console.log(`👤 Client joined room: ${roomName}`);
}

function broadcastToRoom(roomName, message, excludeSocket = null) {
  const room = rooms.get(roomName);
  if (!room) return;

  for (const client of room) {
    if (client !== excludeSocket && client.writable) {
      sendMessage(`[${roomName}] ${message}`, client);
    }
  }
}
```

Client Usage

```
// Join a room
ws.send("JOIN:general");

// Send message to room
ws.send("MSG:Hello everyone in general!");
```

Example 6: Authentication

server.mjs addition

```
// Simple token-based auth
const validTokens = new Set(["secret-token-123", "another-token-456"]);
```

```
const authenticatedSockets = new WeakSet();

function onSocketUpgrade(request, socket, head) {
  const token = request.headers["authorization"];

  // Check authentication
  if (!token || !validTokens.has(token)) {
    console.log("❌ Authentication failed");
    socket.write("HTTP/1.1 401 Unauthorized\r\n\r\n");
    socket.destroy();
    return;
  }

  console.log("✅ Authentication successful");
  authenticatedSockets.add(socket);

  // Continue with normal handshake
  const { "sec-websocket-key": webSocketKey } = request.headers;
  const header = prepareHandshakeHeader(webSocketKey);
  socket.write(header);

  socket.on("readable", () => onSocketreadable(socket));
}

function onSocketreadable(socket) {
  // Verify socket is authenticated
  if (!authenticatedSockets.has(socket)) {
    console.log("❌ Unauthenticated socket tried to send message");
    socket.destroy();
    return;
  }

  // ...existing code...
}
```

Client Usage

```
// WebSocket doesn't support custom headers directly
// Use query parameters instead
const ws = new WebSocket("ws://localhost:1338?token=secret-token-123");

// Or modify the request using a proxy/wrapper
```

Example 7: Rate Limiting

server.mjs addition

```
// Rate limiter: max 10 messages per second per client
const rateLimits = new WeakMap(); // socket -> {count, resetTime}

function checkRateLimit(socket) {
  const now = Date.now();
  const limit = rateLimits.get(socket);

  if (!limit || now > limit.resetTime) {
    // Reset counter
    rateLimits.set(socket, {
      count: 1,
      resetTime: now + 1000, // 1 second window
    });
    return true;
  }

  if (limit.count >= 10) {
    console.log("⚠️ Rate limit exceeded");
    return false;
  }

  limit.count++;
  return true;
}

function onSocketreadable(socket) {
  // Check rate limit
  if (!checkRateLimit(socket)) {
    sendMessage("Error: Rate limit exceeded. Max 10 messages/second",
socket);
    return;
  }

  // ...existing frame parsing code...
}
```

Example 8: Message Size Limits

server.mjs addition

```
const MAX_MESSAGE_SIZE = 1024 * 1024; // 1MB

function onSocketreadable(socket) {
  const [firstByte] = socket.read(1);
  if (!firstByte) return;

  const [secondByte] = socket.read(1);
  const lengthIndicator = secondByte & 0x7f;

  let messageLength = 0;
```

```
  if (lengthIndicator <= 125) {
    messageLength = lengthIndicator;
  } else if (lengthIndicator === 126) {
    messageLength = socket.read(2).readUInt16BE(0);
  } else {
    const upperBits = socket.read(4).readUInt32BE(0);
    const lowerBits = socket.read(4).readUInt32BE(0);
    messageLength = (upperBits << 32) + lowerBits;
  }

  // Check message size
  if (messageLength > MAX_MESSAGE_SIZE) {
    console.log(`❌ Message too large: ${messageLength} bytes`);
    sendMessage("Error: Message too large (max 1MB)", socket);
    socket.destroy();
    return;
  }

  // ...continue with normal processing...
}
```

Example 9: Heartbeat/Ping-Pong

server.mjs addition

```
// Send ping every 30 seconds
const pingInterval = 30000;
const clientPingTimers = new WeakMap();

function onSocketUpgrade(request, socket, head) {
  // ...existing handshake code...

  // Start ping timer
  const timer = setInterval(() => {
    if (socket.writable) {
      sendPing(socket);
    } else {
      clearInterval(timer);
    }
  }, pingInterval);

  clientPingTimers.set(socket, timer);

  socket.on("end", () => {
    clearInterval(timer);
  });
}

function sendPing(socket) {
```

```
  const firstByte = 0x89; // FIN + Ping opcode
  const secondByte = 0x00; // No payload

  const frame = Buffer.from([firstByte, secondByte]);
  socket.write(frame);
  console.log("🏓 Sent Ping");
}

// Pong is already handled in onSocketreadable with sendPong function
```

Example 10: JSON Message Protocol

server.mjs addition

```
function onSocketreadable(socket) {
  // ...existing frame parsing code...

  const receivedMessage = decoded.toString("utf8");

  try {
    // Try to parse as JSON
    const data = JSON.parse(receivedMessage);

    // Handle different message types
    switch (data.type) {
      case "echo":
        sendMessage(
          JSON.stringify({
            type: "echo_response",
            data: data.message,
            timestamp: Date.now(),
          }),
          socket
        );
        break;

      case "broadcast":
        broadcast(
          JSON.stringify({
            type: "broadcast",
            from: data.username,
            message: data.message,
            timestamp: Date.now(),
          }),
          socket
        );
        break;

      case "private":
        // Implement private messaging
```

```
          sendPrivateMessage(data.to, data.message, data.from);
          break;

        default:
          sendMessage(
            JSON.stringify({
              type: "error",
              message: "Unknown message type",
            }),
            socket
          );
      }
    } catch (err) {
      // Not JSON, treat as plain text
      sendMessage(`Echo: ${receivedMessage}`, socket);
    }
  }
```

Client Usage

```
// Send JSON messages
ws.send(
  JSON.stringify({
    type: "echo",
    message: "Hello!",
  })
);

ws.send(
  JSON.stringify({
    type: "broadcast",
    username: "Alice",
    message: "Hello everyone!",
  })
);

ws.onmessage = (event) => {
  const data = JSON.parse(event.data);
  console.log("Type:", data.type);
  console.log("Data:", data);
};
```

🧪 Testing Examples

Example 11: Load Testing Script

load-test.js

```
// Simple load test with multiple connections
const WebSocket = require("ws"); // Install with: npm install ws

const NUM_CLIENTS = 100;
const MESSAGES_PER_CLIENT = 10;

async function runLoadTest() {
  const clients = [];

  console.log(`Creating ${NUM_CLIENTS} connections...`);

  for (let i = 0; i < NUM_CLIENTS; i++) {
    const ws = new WebSocket("ws://localhost:1338");

    ws.on("open", () => {
      console.log(`Client ${i} connected`);

      // Send multiple messages
      for (let j = 0; j < MESSAGES_PER_CLIENT; j++) {
        ws.send(`Client ${i}, Message ${j}`);
      }
    });

    ws.on("message", (data) => {
      // Count responses
    });

    clients.push(ws);
  }

  console.log(`Total connections: ${NUM_CLIENTS}`);
  console.log(`Total messages: ${NUM_CLIENTS * MESSAGES_PER_CLIENT}`);
}

runLoadTest();
```

Example 12: Automated Test Suite

test.js

```
// Simple test suite
const WebSocket = require("ws");
const assert = require("assert");

async function runTests() {
  console.log("🧪 Starting tests...\n");

  // Test 1: Connection
  console.log("Test 1: Connection");
  const ws1 = new WebSocket("ws://localhost:1338");
```

```
  await new Promise((resolve) => {
    ws1.on("open", () => {
      console.log("✅ Connection successful\n");
      ws1.close();
      resolve();
    });
  });

  // Test 2: Echo
  console.log("Test 2: Echo message");
  const ws2 = new WebSocket("ws://localhost:1338");
  await new Promise((resolve) => {
    ws2.on("open", () => {
      ws2.send("Test message");
    });
    ws2.on("message", (data) => {
      assert(data.toString().includes("Test message"));
      console.log("✅ Echo working correctly\n");
      ws2.close();
      resolve();
    });
  });

  // Test 3: Large message
  console.log("Test 3: Large message");
  const ws3 = new WebSocket("ws://localhost:1338");
  const largeMessage = "x".repeat(1000);
  await new Promise((resolve) => {
    ws3.on("open", () => {
      ws3.send(largeMessage);
    });
    ws3.on("message", (data) => {
      assert(data.toString().includes(largeMessage));
      console.log("✅ Large message handled\n");
      ws3.close();
      resolve();
    });
  });

  console.log("🎉 All tests passed!");
}

runTests().catch(console.error);
```

🎨 UI Enhancement Examples

Example 13: Chat Application UI

enhanced-client.html

```html
<!DOCTYPE html>
<html>
  <head>
    <title>WebSocket Chat</title>
    <style>
      .message-bubble {
        padding: 10px;
        margin: 5px 0;
        border-radius: 15px;
        max-width: 70%;
      }
      .sent {
        background: #007bff;
        color: white;
        margin-left: auto;
        text-align: right;
      }
      .received {
        background: #e9ecef;
        color: black;
      }
    </style>
  </head>
  <body>
    <div id="chat-container">
      <div id="messages"></div>
      <input id="messageInput" placeholder="Type a message..." />
      <button onclick="sendMessage()">Send</button>
    </div>

    <script>
      const ws = new WebSocket("ws://localhost:1338");
      const messagesDiv = document.getElementById("messages");
      const messageInput = document.getElementById("messageInput");

      ws.onmessage = (event) => {
        addMessage(event.data, "received");
      };

      function sendMessage() {
        const msg = messageInput.value;
        if (msg.trim()) {
          ws.send(msg);
          addMessage(msg, "sent");
          messageInput.value = "";
        }
      }

      function addMessage(text, type) {
        const div = document.createElement("div");
        div.className = `message-bubble ${type}`;
        div.textContent = text;
        messagesDiv.appendChild(div);
```

```
          messagesDiv.scrollTop = messagesDiv.scrollHeight;
        }
      </script>
    </body>
  </html>
```

🔍 Debugging Examples

Example 14: Frame Inspector

```
// Log detailed frame information
function inspectFrame(buffer) {
  console.log("\n=== Frame Inspection ===");
  console.log("Hex:", buffer.toString("hex"));
  console.log("Bytes:", [...buffer]);

  const [firstByte, secondByte] = buffer;

  console.log("\nFirst Byte:", firstByte.toString(2).padStart(8, "0"));
  console.log("  FIN:", !!(firstByte & 0x80));
  console.log("  RSV1:", !!(firstByte & 0x40));
  console.log("  RSV2:", !!(firstByte & 0x20));
  console.log("  RSV3:", !!(firstByte & 0x10));
  console.log("  Opcode:", (firstByte & 0x0f).toString(16));

  console.log("\nSecond Byte:", secondByte.toString(2).padStart(8, "0"));
  console.log("  MASK:", !!(secondByte & 0x80));
  console.log("  Length:", secondByte & 0x7f);
  console.log("========================\n");
}
```

📚 More Resources

For more examples and use cases, check out:

- The main README.md for basic usage
- ARCHITECTURE.md for design patterns
- TESTING.md for test scenarios
- QUICK_REFERENCE.md for protocol details

Happy coding! 🚀