Lecture 14. Web Sockets in Node js

13 April 2025 18:58

WebSockets are used for real-time communication between the client and the server. It establishes a long-lived connection over a single TCP connection, allowing both the client and the server to send data to each other at any time.
In Node.js, the **ws** library is a popular choice for implementing WebSockets.

Establishment of an complete Web Socket Server :-

Step 1: Set Up a New Node.js Project

First, initialize a new Node.js project and install the necessary dependencies.

1. **Create a project directory**:

   bash
   Copy code
   mkdir websocket-app
   cd websocket-app
2. **Initialize a Node.js project**:

   bash
   Copy code
   npm init -y
3. **Install the ws package** (a popular WebSocket library for Node.js):

   bash
   Copy code
   npm install ws

Step 2: Create a WebSocket Server

1. **Create a file named server.js** in the project root.
2. **Set up the WebSocket server** using the ws library:

   javascript
   Copy code

```
const WebSocket = require('ws');  // Import the 'ws' library

const wss = new WebSocket.Server({ port: 8080 });  // Create a WebSocket server

// Event listener for when a client connects
wss.on('connection', (ws) => {
   console.log('A client connected');

   // Event listener for receiving messages from the client
   ws.on('message', (message) => {
      console.log('received: %s', message);
   });

// Send a message to the client
   ws.send('Welcome to the WebSocket server!');
});

console.log('WebSocket server running on ws://localhost:8080');
```

Step 3: Create the WebSocket Client

1. **Create a file named index.html** inside the public folder.
2. **Set up a basic HTML structure** and connect to the WebSocket server:

```
html
Copy code
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>WebSocket Client</title>
</head>
<body>
  <h1>WebSocket Client</h1>
  <button id="sendMessageButton">Send Message</button>
  <div id="messages"></div>

<script>
    // Establish a WebSocket connection to the server
    const socket = new WebSocket('ws://localhost:8080');

// Display a welcome message when connected
    socket.addEventListener('open', () => {
      console.log('Connected to the WebSocket server');
    });

// Handle incoming messages from the server
    socket.addEventListener('message', (event) => {
      const messagesDiv = document.getElementById('messages');
      messagesDiv.innerHTML += `<p>Server: ${event.data}</p>`;
    });

// Send a message when the button is clicked
    document.getElementById('sendMessageButton').addEventListener('click', () => {
      socket.send('Hello from the client!');
    });
  </script>
</body>
</html>
```

Step 4: Run the WebSocket Server

1. **Run the WebSocket server** by executing the following command in your terminal:

```
bash
Copy code
node server.js
```

This will start the WebSocket server on ws://localhost:8080.

Step 5: Open the WebSocket Client

1. Open the index.html file in your browser. This will establish a WebSocket connection to the server.
2. Once connected, the client will display a welcome message from the server. Clicking the "Send Message" button will send a message from the client to the server, which will be displayed on the server's console.

Step 6: Test the Communication

- When you open multiple browser tabs with the client, each tab will connect to the WebSocket server, and any message sent from one tab will be broadcasted to all connected clients.

Step 7: (Optional) Broadcast Messages to All Clients

If you want to send messages to all connected clients, you can iterate over all WebSocket connections. Modify the server code to broadcast messages:

javascript
Copy code

```
const WebSocket = require('ws');
const wss = new WebSocket.Server({ port: 8080 });
let clients = [];  // Store the connected clients
wss.on('connection', (ws) => {
   console.log('A client connected');
   clients.push(ws);  // Add the client to the list of connected clients
ws.on('message', (message) => {
      console.log('received: %s', message);
// Broadcast the message to all connected clients
      clients.forEach(client => {
         if (client !== ws && client.readyState === WebSocket.OPEN) {
            client.send(`New message: ${message}`);
         }
      });
   });
ws.on('close', () => {
      // Remove the client when they disconnect
      clients = clients.filter(client => client !== ws);
   });
ws.send('Welcome to the WebSocket server!');
});
console.log('WebSocket server running on ws://localhost:8080');
```

Step 8: Close the WebSocket Connection

To close the connection gracefully, you can call the close() method on the WebSocket connection, either from the client or server:

On the Server:

javascript
Copy code

```
ws.close();  // Close the WebSocket connection for the specific client
```

On the Client:

javascript
Copy code

```
socket.close();  // Close the WebSocket connection from the client side
```