Plagiarism Scan Report By SmallSEOTools

Report Generated on: Apr 28,2025

Plagiarized Content

0% Exact Plagiarized

12% Partial Plagiarized

Unique Content

Total Words: 1198 Total Characters: 6480 Plagiarized Sentences: 2.04 Unique Sentences: 14.96 (88%)

Content Checked for Plagiarism

ARTIKEL: "MENGENAL WEBSOCKET: KOMUNIKASI REAL-TIME DI WEB DAN CONTOH IMPLEMENTASINYA"

_____

Mengenal WebSocket: Solusi Komunikasi Real-Time di Web

Dalam era di mana kecepatan informasi menjadi kunci, kebutuhan akan komunikasi real-time di dunia web semakin meningkat. Mulai dari aplikasi chat hingga game online, semuanya memerlukan sistem komunikasi yang cepat dan e sien.

Salah satu teknologi yang memungkinkan komunikasi dua arah secara instan adalah WebSocket.

_____

Apa Itu WebSocket?

Berbeda dengan HTTP biasa yang berbasis request-response dan membuka koneksi baru setiap kali, WebSocket mempertahankan koneksi tetap terbuka, sehingga komunikasi data bisa terjadi secara real-time tanpa latensi tinggi.

Fitur HTTP WebSocket

Koneksi Baru setiap request Tetap terbuka

Latensi Tinggi Rendah

Cocok untuk Website biasa Aplikasi real-time

_____

Mengapa WebSocket Dibutuhkan?

Banyak aplikasi modern yang membutuhkan koneksi real-time seperti:

• Aplikasi chat online (WhatsApp Web, Telegram Web)
• Sistem noti kasi instan
• Game multiplayer berbasis browser
• Dashboard monitoring server secara live

Dengan HTTP biasa, komunikasi real-time sulit dicapai tanpa teknik tambahan seperti polling atau long polling, yang kurang e sien.

WebSocket memberikan solusi elegan untuk masalah ini.

_____

Eksperimen: Membangun Aplikasi Chat Modern Menggunakan WebSocket

Untuk memahami lebih dalam cara kerja WebSocket, saya melakukan eksperimen dengan membangun aplikasi chat sederhana.

Teknologi yang Digunakan:

• Server: Node.js menggunakan library ws

Langkah Eksperimen:

1. (Server.js)

```
const express = require('express');
const path = require('path');
const http = require('http');
const WebSocket = require('ws');

const app = express();
const port = 3000;

app.use(express.static(path.join(__dirname)));
const server = http.createServer(app);
const wss = new WebSocket.Server({ server });

const clients = new Map();
const history = [];

wss.on('connection', function connection(ws) {
console.log('Client connected');

ws.on('message', function incoming(data) {
let parsed;
try {
parsed = JSON.parse(data);
} catch (e) {
console.error('Invalid JSON:', data);
return;
}

if (parsed.type === 'join') {
ws.username = parsed.username;
clients.set(ws, parsed.username);
history.forEach(msg => {
ws.send(JSON.stringify(msg));
});
broadcast({ type: 'info', message: `${parsed.username} bergabung ke chat.` });
broadcastUserList();
}
else if (parsed.type === 'chat') {
const chatMsg = { type: 'chat', username: parsed.username, message: parsed.message };
history.push(chatMsg);
broadcast(chatMsg);
}
else if (parsed.type === 'changeUsername') {
const oldUsername = ws.username;
ws.username = parsed.username;
clients.set(ws, parsed.username);
broadcast({ type: 'info', message: `${oldUsername} mengganti nama menjadi ${parsed.username}.` });
broadcastUserList();
}
});

ws.on('close', function () {
if (ws.username) {
clients.delete(ws);
broadcast({ type: 'info', message: `${ws.username} keluar dari chat.` });
broadcastUserList();
}
});
});
```

```
function broadcast(message) {
const msgString = JSON.stringify(message);
wss.clients.forEach(function each(client) {
if (client.readyState === WebSocket.OPEN) {
client.send(msgString);
}
});
}

function broadcastUserList() {
const users = Array.from(clients.values());
const userListMessage = { type: 'userlist', users: users };
broadcast(userListMessage);
}

server.listen(port, () => {
console.log(`Server running at http://localhost:${port}`);
});
```

2. Membuat Client Chat Modern (index.html)

```
<!DOCTYPE html>
<html lang="id">
<head>
  <meta charset="UTF-8">
  <title>WebSocket Chat</title>
  <style>
    body {
      font-family: 'Roboto', sans-serif;
      margin: 0;
      padding: 0;
      height: 100vh;
      display: flex;
      overflow: hidden;
      background-color: #e0e0e0;
    }
    #sidebar {
      width: 220px;
      background-color: #f5f5f5;
      padding: 15px;
      box-shadow: 2px 0 5px rgba(0,0,0,0.1);
      overflow-y: auto;
    }
    #sidebar h3 {
      margin-top: 0;
      font-size: 18px;
      color: #333;
    }
    #userList li {
      margin: 10px 0;
      color: #555;
      font-size: 14px;
    }
    #messages {
      flex: 1;
      padding: 20px;
```

```
    overflow-y: auto;
    background-color: #ffffff;
    display: flex;
    flex-direction: column;
  }
  .message {
    max-width: 60%;
    margin-bottom: 15px;
    padding: 10px 15px;
    border-radius: 20px;
    word-wrap: break-word;
    animation: fadeIn 0.5s ease-in-out;
  }
  .incoming {
    background-color: #e5e5ea;
    align-self: flex-start;
    color: #000;
  }
  .outgoing {
    background-color: #0088cc;
    color: #fff;
    align-self: flex-end;
  }
  #input {
    padding: 10px;
    background: #fff;
    border-top: 1px solid #ccc;
    display: flex;
  }
  #inputField {
    flex: 1;
    padding: 10px;
    border: 1px solid #ccc;
    border-radius: 20px;
    margin-right: 10px;
  }
  #sendBtn {
    padding: 10px 20px;
    background-color: #0088cc;
    border: none;
    color: white;
    border-radius: 20px;
    cursor: pointer;
  }
  #sendBtn:hover {
    background-color: #0077b6;
  }
  @keyframes fadeIn {
    from { opacity: 0; transform: translateY(10px); }
    to { opacity: 1; transform: translateY(0); }
  }
</style>
```

```
</head>

<body>

<!-- Sidebar User List -->
<div id="sidebar">
  <h3>Online</h3>
  <ul id="userList" style="list-style: none; padding: 0;"></ul>
  <button id="changeUsernameBtn" style="margin-top: 10px; width: 100%; padding: 8px; background-color: #0088cc; color: white; border: none; border-radius: 20px; cursor: pointer;">Ganti Nama</button>
</div>

<!-- Main Chat Area -->
<div style="flex: 1; display: flex; flex-direction: column;">
  <div id="messages"></div>

  <div id="input" style="display: none;">
    <input id="inputField" type="text" placeholder="Ketik pesan...">
    <button id="sendBtn">Kirim</button>
  </div>
</div>

<!-- Modal Username -->
<div id="usernameModal" style="position: fixed; inset: 0; background: rgba(0,0,0,0.5); display: flex; align-items: center; justify-content: center;">
  <div style="background: white; padding: 20px; border-radius: 10px;">
    <h3>Masukkan Nama Anda</h3>
    <input id="usernameInput" type="text" placeholder="Nama..." style="padding: 10px; width: 100%;">
    <button id="startBtn" style="margin-top: 10px;">Mulai Chat</button>
  </div>
</div>

<script>
  let socket;
  let username;
  const messages = document.getElementById('messages');
  const inputField = document.getElementById('inputField');
  const sendBtn = document.getElementById('sendBtn');
  const userList = document.getElementById('userList');
  const usernameModal = document.getElementById('usernameModal');
  const startBtn = document.getElementById('startBtn');
  const changeUsernameBtn = document.getElementById('changeUsernameBtn');

  startBtn.addEventListener('click', function () {
    const input = document.getElementById('usernameInput');
    username = input.value.trim();
    if (username) {
      startChat();
      usernameModal.style.display = 'none';
      document.getElementById('input').style.display = 'flex';
    }
  });
```

```
changeUsernameBtn.addEventListener('click', function () {
  const newUsername = prompt('Masukkan nama baru:');
  if (newUsername && newUsername.trim() !== '') {
    username = newUsername.trim();
    socket.send(JSON.stringify({ type: 'changeUsername', username: username }));
  }
});

function startChat() {
  socket = new WebSocket('ws://localhost:3000');

  socket.addEventListener('open', function () {
    socket.send(JSON.stringify({ type: 'join', username: username }));
  });

  socket.addEventListener('message', async function (event) {
    let data = event.data;
    if (event.data instanceof Blob) {
      data = await event.data.text();
    }
    const parsed = JSON.parse(data);
    handleServerMessage(parsed);
  });

  sendBtn.addEventListener('click', sendMessage);
  inputField.addEventListener('keypress', function (e) {
    if (e.key === 'Enter') sendMessage();
  });
}

function sendMessage() {
  const message = inputField.value.trim();
  if (message) {
    socket.send(JSON.stringify({ type: 'chat', username: username, message: message }));
    inputField.value = '';
  }
}

function handleServerMessage(data) {
  if (data.type === 'chat' || data.type === 'info') {
    displayMessage(data);
  } else if (data.type === 'userlist') {
    updateUserList(data.users);
  }
}

function displayMessage(data) {
const wrapper = document.createElement('div');
wrapper.style.display = 'flex';
wrapper.style.alignItems = 'center';
wrapper.style.marginBottom = '10px';
wrapper.style.maxWidth = '70%';
```

```
  wrapper.style.animation = 'fadeIn 0.5s ease-in-out';

  const avatar = document.createElement('div');
  avatar.style.width = '40px';
  avatar.style.height = '40px';
  avatar.style.borderRadius = '50%';
  avatar.style.backgroundColor = '#0088cc';
  avatar.style.color = 'white';
  avatar.style.display = 'flex';
  avatar.style.alignItems = 'center';
  avatar.style.justifyContent = 'center';
  avatar.style.fontWeight = 'bold';
  avatar.style.margin = '0 8px';
  avatar.textContent = data.username ? data.username.charAt(0).toUpperCase() : '?';

  const msgDiv = document.createElement('div');
  msgDiv.className = 'message';

  if (data.type === 'chat') {
    msgDiv.className += data.username === username ? ' outgoing' : ' incoming';
    msgDiv.textContent = `${data.username}: ${data.message}`;
  } else if (data.type === 'info') {
    msgDiv.style.textAlign = 'center';
    msgDiv.style.color = '#888';
    msgDiv.textContent = data.message;
  }

  if (data.type === 'chat') {
    if (data.username === username) {
      wrapper.style.alignSelf = 'flex-end';
      wrapper.appendChild(msgDiv);
      wrapper.appendChild(avatar); // Avatar setelah bubble kalau outgoing
    } else {
      wrapper.style.alignSelf = 'flex-start';
      wrapper.appendChild(avatar);
      wrapper.appendChild(msgDiv); // Avatar sebelum bubble kalau incoming
    }
    messages.appendChild(wrapper);
  } else {
    messages.appendChild(msgDiv);
  }

  messages.scrollTop = messages.scrollHeight;
}

  function updateUserList(users) {
    userList.innerHTML = '';
    users.forEach(user => {
      const li = document.createElement('li');
      li.textContent = user;
      userList.appendChild(li);
    });
```

```
  }
</script>

</body>
</html>
```

Hasil Eksperimen

• Tampilan modern menggunakan tema biru-putih membuat pengalaman chatting lebih nyaman.

Analisis

Eksperimen menunjukkan bahwa:

• Pembuatan server dan client WebSocket cukup sederhana.

Kesimpulan

Melalui eksperimen ini, terbukti bahwa WebSocket adalah solusi tepat untuk kebutuhan komunikasi real-time di dunia web.Dengan WebSocket, kita dapat membangun aplikasi modern yang lebih responsif, e sien, dan menyenangkan bagi pengguna.

Referensi

• WebSocket API - MDN

• ws - Simple to use WebSocket server