

스타트업 개발자와 함께 공부하는

Node.js

12. Socket.IO

강의 내용은 강사가 별도로 명시하지 않는 한 비공개로 간주합니다. 녹음이나 사진 촬영를 허락하지 않으며 콘텐츠를 블로그, SNS 등에 게시하거나 공개적으로 공유하지 마세요.

콘텐츠 공유 가능 여부에 대해 궁금한 점이 있는 경우 강사에게 문의하시기 바랍니다.







목차

- 1. 웹 소켓
- 2. Socket.IO
- 3. 실시간 채팅





소개

- □ 웹 소켓(WebSocket)은 웹 애플리케이션 양방향 통신 프로토콜
- ☐ HTTP 기반
- □ 양방향 통신
- □ 지속적인 연결
- □ 저지연통신



프로토콜

- 1. 핸드셰이크
- 2. 응답
- 3. 데이터 프레임

GET /chat HTTP/1.1

Host: server.example.com

Upgrade: websocket
Connection: Upgrade

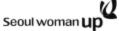
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==

Sec-WebSocket-Version: 13

HTTP/1.1 101 Switching Protocols

Upgrade: websocket
Connection: Upgrade

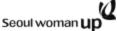
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=



예제 - 서버

```
2 const WebSocket = require('ws');
4 const wss = new WebSocket.Server({ port: 8080 });
6 wss.on('connection', ws => {
     console.log('클라이언트가 연결되었습니다.');
    // 클라이언트로부터 메시지를 받았을 때
    ws.on('message', message => {
      console.log('받은 메시지: ${message}');
11
12
13
      // 클라이언트에게 메시지 전송
      ws.send(`서버로부터 응답: ${message}`);
15
    });
16
    // 클라이언트가 연결을 닫았을 때
    ws.on('close', () => {
      console.log('클라이언트 연결이 종료되었습니다.');
19
    });
21 });
22
23 console.log('웹 소켓 서버가 ws://localhost:8080 에서 실행 중입니다.');
```

- ☐ [C]-[nodejs]-[project]-[12]-[ch12_01]
- □ npm init -y
- □ npm i ws
- □ server.js



예제 - 클라이언트

```
15
     <script>
16
       const ws = new WebSocket('ws://localhost:8080');
17
18
       ws.onopen = () => {
19
         console.log('웹 소켓 연결이 열렸습니다.');
20
21
22
       ws.onmessage = event => {
         const responses = document.getElementById('responses');
23
24
         responses.innerHTML += `${event.data}`;
25
26
       ws.onclose = () => {
27
28
         console.log('웹 소켓 연결이 닫혔습니다.');
29
       };
30
31
       function sendMessage() {
32
         const messageInput = document.getElementById('message');
33
         const message = messageInput.value;
         ws.send(message);
34
         messageInput.value = '';
35
36
37
     </script>
```

- □ [C]-[nodejs]-[project]-[12]-[ch12_01]
- ☐ index.html





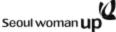
소개

- □ 실시간 웹 애플리케이션을 위한 자바스크립트 라이브러리
- □ 브라우저와 서버, 서버와 서버 간에 통신
- □ 양방향 실시간 통신
- □ 자동 폴백
- □ 간단한 이벤트 기반 API
- □ 방(Rooms)와 네임스페이스(Namespace)

예제 - 서버

```
2 const express = require('express');
 3 const http = require('http');
 4 const socketIo = require('socket.io');
 6 const app = express();
 7 const server = http.createServer(app);
 8 const io = socketIo(server);
10 app.get('/', (req, res) => {
11    res.sendFile(__dirname + '/index.html');
12 });
13
14 io.on('connection', (socket) => {
     console.log('a user connected');
16
     socket.on('disconnect', () => {
17
       console.log('user disconnected');
18
19
     });
20
     socket.on('chat message', (msg) => {
21
22
       console.log('message: ' + msg);
       io.emit('chat message', msg); // 모든 클라이언트에게 메시지 전송
24
    });
25 });
26
27 server.listen(3000, () => {
     console.log('listening on *:3000');
29 });
```

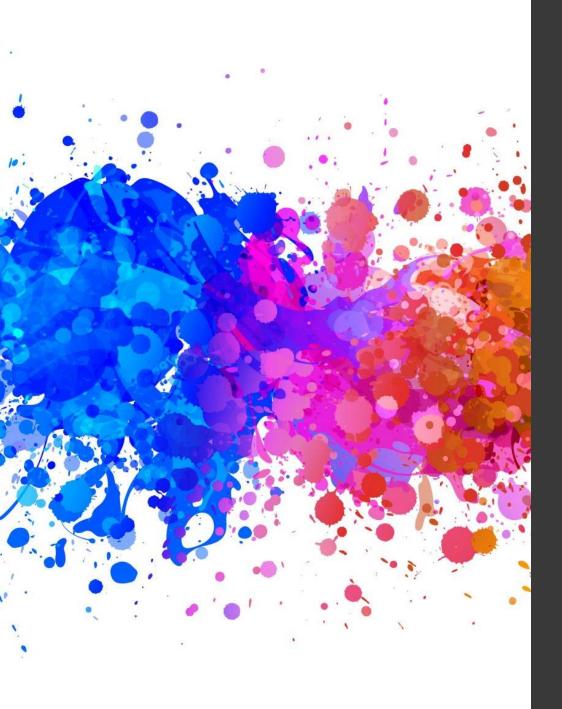
- ☐ [C]-[nodejs]-[project]-[12]-[ch12_02]
- □ npm init -y
- ☐ npm i express socket.io
- □ server.js



예제 - 클라이언트

```
var socket = io();
18
19
         document.getElementById('form').addEventListener('submit', function(e) {
20
21
           e.preventDefault(); // 페이지 리로드 방지
           var input = document.getElementById('input');
22
           if (input.value) {
23
             socket.emit('chat message', input.value);
24
             input.value = '';
25
26
         });
27
28
29
         socket.on('chat message', function(msg) {
           var item = document.createElement('li');
           item.textContent = msg;
31
           document.getElementById('messages').appendChild(item);
32
           window.scrollTo(0, document.body.scrollHeight);
33
34
```

- ☐ [C]-[nodejs]-[project]-[12]-[ch12_02]
- ☐ index.html





주요 요구사항

- □ 아이디와 비밀번호로 로그인
- □ 방생성
- □ 방참가
- □ 메시지 전송
- □ 사용자 연결 해제



프로젝트 생성

```
npm init -y
npm i express socket.io nodemon
```

- ☐ [C]-[nodejs]-[project]-[12]-[ch12_02]
- □ 프로젝트 생성
- □ 의존성 설치

로그인

```
20 let users = {}; // 사용자 목록
21 let rooms = {}; // 채팅방 목록
22
23 io.on('connection', (socket) => {
     console.log(`user connected =>
24
       ${JSON.stringify(users)} : ${JSON.stringify(rooms)}`)
25
26
27
     // 사용자 로그인
28
     socket.on('login', (username) => {
29
       users[socket.id] = username;
30
       console.log(`login => ${username}`)
31
       socket.emit('login:success',
32
         { username, rooms: Object.keys(rooms) });
       io.emit('update:users', Object.values(users));
33
34
     });
```

```
// 로그인 버튼 클릭
70
       loginButton.addEventListener('click', function() {
71
         var username = usernameInput.value;
72
         if (username) {
73
           console.log(`login : ${username}`)
74
75
           socket.emit('login', username); // 사용자 로그인 요청
76
77
        // 리스닝 : 로그인 성공
100
101
        socket.on('login:success', function(data) {
          chatContainer.style.display = 'block';
102
          usernameInput.disabled = true;
103
          loginButton.disabled = true;
104
          updateRooms(data.rooms); // 룸 정보 업데이트
105
       });
106
```

방생성

```
// 채팅방 생성
36
37
     socket.on('create:room', (room) => {
38
       if (!rooms[room]) {
         rooms[room] = [];
39
         socket.join(room);
40
         socket.emit('room:created', room);
41
         io.emit('update:rooms', Object.keys(rooms));
42
       } else {
43
         socket.emit('room:exists', room);
44
45
46
```

```
79
       // 방생성 버튼 클릭
       createRoomButton.addEventListener('click', function() {
80
81
         var room = roomInput.value;
         if (room) {
82
83
           socket.emit('create:room', room); // 방생성 요청
84
85
       });
        // 리스닝 : 룸 생성
108
109
        socket.on('room:created', function(room) {
110
          var item = document.createElement('li');
111
          item textContent = room;
112
          roomsList.appendChild(item);
          item.addEventListener('click', function() {
113
114
            currentRoom = room // 룸 최신화
115
            item.style = 'background-color:orange;'
            socket.emit('join:room', room); // 룸에 들어옴을 알림
116
117
          });
118
        });
```



방 참가

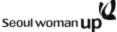
```
// 채팅방 참가
48
49
     socket.on('join:room', (room) => {
       if (rooms[room]) {
50
51
         socket.join(room);
52
         socket.emit('room:joined', room);
53
         socket.to(room).emit('user:joined',
54
           users[socket.id]);
       } else {
55
         socket.emit('room:notfound', room);
56
57
58
     });
```

```
// 리스닝 : 룸 조인
123
        socket.on('room:joined', function(room) {
124
         alert('Joined room: ' + room);
125
126
       });
        // 리스닝 : 사용자 방 조인
147
        socket.on('user:joined', function(user) {
148
           var item = document.createElement('li');
149
150
           item.textContent = user + ' joined the room';
          chatList.appendChild(item);
151
          window.scrollTo(0, document.body.scrollHeight);
152
153
        });
```



메시지 전송

```
// 리스닝 : 채팅 메시지
138
139
        socket.on('chat:message', function(data) {
          console.log(`received message => ${data}`);
140
141
          var item = document.createElement('li');
          item.textContent = data.user + ': ' + data.message;
142
143
          chatList.appendChild(item);
          window.scrollTo(0, document.body.scrollHeight);
144
145
        });
```



사용자 연결 해제

```
// 사용자 연결 해제
66
     socket.on('disconnect', () => {
67
       if (users[socket.id]) {
68
         io.emit('user:left', users[socket.id]);
69
         delete users[socket.id];
70
         io.emit('update:users', Object.values(users));
71
72
73
     });
74 });
```

```
155
        // 리스닝 : 사용자 방 나감
156
        socket.on('user:left', function(user) {
          var item = document.createElement('li');
157
158
          item.textContent = user + ' left the chat';
159
          chatList.appendChild(item);
160
          window.scrollTo(0, document.body.scrollHeight);
161
        });
         // 리스닝 : 사용자 업데이트
133
         socket.on('update:users', function(users) {
134
           updateUsers(users);
135
136
         });
```

