

스타트업 개발자와 함께 공부하는 Node.js

12. Socket.IO

강의 내용은 강사가 별도로 명시하지 않는 한 비공개로 간주합니다.
녹음이나 사진 촬영을 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

웹 소켓

소개

- 웹 소켓(WebSocket)은 웹 애플리케이션 양방향 통신 프로토콜
- HTTP 기반
- 양방향 통신
- 지속적인 연결
- 저 지연 통신

웹 소켓

프로토콜

1. 핸드 셰이크
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+x0o=
```

웹 소켓

예제 - 서버

```
2 const WebSocket = require('ws');
3
4 const wss = new WebSocket.Server({ port: 8080 });
5
6 wss.on('connection', ws => {
7   console.log('클라이언트가 연결되었습니다. ');
8
9   // 클라이언트로부터 메시지를 받았을 때
10  ws.on('message', message => {
11    console.log(`받은 메시지: ${message}`);
12
13    // 클라이언트에게 메시지 전송
14    ws.send(`서버로부터 응답: ${message}`);
15  });
16
17  // 클라이언트가 연결을 닫았을 때
18  ws.on('close', () => {
19    console.log('클라이언트 연결이 종료되었습니다. ');
20  });
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`

웹 소켓

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 => {
23     const responses = document.getElementById('responses');
24     responses.innerHTML += `<p>${event.data}</p>`;
25   };
26
27   ws.onclose = () => {
28     console.log('웹 소켓 연결이 닫혔습니다.');
```

```
29   };
30
31   function sendMessage() {
32     const messageInput = document.getElementById('message');
33     const message = messageInput.value;
34     ws.send(message);
35     messageInput.value = '';
36   }
37 </script>
```

❑ [C]-[nodejs]-[project]-[12]-[ch12_01]

❑ index.html



Socket.IO

```
function todoitem(data) ;  
  var self = this  
  data = data || {}  
  // Set persisted properties  
  <html> <errorMessage = text - 200px>to , observable() ;  
  <body style="font-weight:bold;">HTML font code is here  
  <body style="background-color:yellowgreen">  
  <div - 200px> <todolistid = data.todolistid  
  <div - 200px> persisted properties  
  <errorMessage = ko , observable() ;
```

Socket.IO

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)

Socket.IO

예제 - 서버

```
2 const express = require('express');
3 const http = require('http');
4 const socketIo = require('socket.io');
5
6 const app = express();
7 const server = http.createServer(app);
8 const io = socketIo(server);
9
10 app.get('/', (req, res) => {
11   res.sendFile(__dirname + '/index.html');
12 });
13
14 io.on('connection', (socket) => {
15   console.log('a user connected');
16
17   socket.on('disconnect', () => {
18     console.log('user disconnected');
19   });
20
21   socket.on('chat message', (msg) => {
22     console.log('message: ' + msg);
23     io.emit('chat message', msg); // 모든 클라이언트에게 메시지 전송
24   });
25 });
26
27 server.listen(3000, () => {
28   console.log('listening on *:3000');
29 });
```

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

Socket.IO

예제 - 클라이언트

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

❑ [C]-[nodejs]-[project]-[12]-[ch12_02]

❑ index.html



실시간 채팅

시간 채팅

실시간 채팅

주요 요구사항

- ❑ 아이디와 비밀번호로 로그인
- ❑ 방 생성
- ❑ 방 참가
- ❑ 메시지 전송
- ❑ 사용자 연결 해제

실시간 채팅

프로젝트 생성

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

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

실시간 채팅

로그인

```
20 let users = {}; // 사용자 목록
21 let rooms = {}; // 채팅방 목록
22
23 io.on('connection', (socket) => {
24   console.log(`user connected =>
25     ${JSON.stringify(users)} : ${JSON.stringify(rooms)}`)
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) });
33     io.emit('update:users', Object.values(users));
34   });
```

```
70 // 로그인 버튼 클릭
71 loginButton.addEventListener('click', function() {
72   var username = usernameInput.value;
73   if (username) {
74     console.log(`login : ${username}`)
75     socket.emit('login', username); // 사용자 로그인 요청
76   }
77 });

100 // 리스닝 : 로그인 성공
101 socket.on('login:success', function(data) {
102   chatContainer.style.display = 'block';
103   usernameInput.disabled = true;
104   loginButton.disabled = true;
105   updateRooms(data.rooms); // 룸 정보 업데이트
106 });
```


실시간 채팅

방생성

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

```
79 // 방생성 버튼 클릭
80 createRoomButton.addEventListener('click', function() {
81     var room = roomInput.value;
82     if (room) {
83         socket.emit('create:room', room); // 방생성 요청
84     }
85 });

108 // 리스닝 : 룸 생성
109 socket.on('room:created', function(room) {
110     var item = document.createElement('li');
111     item.textContent = room;
112     roomsList.appendChild(item);
113     item.addEventListener('click', function() {
114         currentRoom = room // 룸 최신화
115         item.style = 'background-color:orange;';
116         socket.emit('join:room', room); // 룸에 들어옴을 알림
117     });
118 });
```

실시간 채팅

방 참가

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

```
123 // 리스닝 : 룸 조인
124 socket.on('room:joined', function(room) {
125     alert('Joined room: ' + room);
126 });

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

실시간 채팅

메시지 전송

```
60 // 메시지 전송
61 socket.on('chat:message', ({ room, message }) => {
62   io.to(room).emit('chat:message',
63     { user: users[socket.id], message });
64 });
65
```

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

실시간 채팅

사용자 연결 해제

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

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

133 // 리스닝 : 사용자 업데이트
134 socket.on('update:users', function(users) {
135     updateUsers(users);
136 });
```

The background is a dark blue gradient. In the center, there is a faint image of a person's hands typing on a laptop keyboard. To the left of the hands, there is a faint, semi-transparent image of a person's head in profile, facing right. Overlaid on this background are several lines of code in a light blue font, including HTML tags like <DropdownItem>, <Bolticon />, and JavaScript code like keydown, closeOnEscape, and onKeyDown. The text '백문이 불여일타' is written in white, bold Korean characters in the center of the image. There are two large white L-shaped brackets: one in the top left corner and one in the bottom right corner.

백문이 불여일타