K-Digital Training KDT 풀스택 웹 개발자 양성 부트캠프 3기

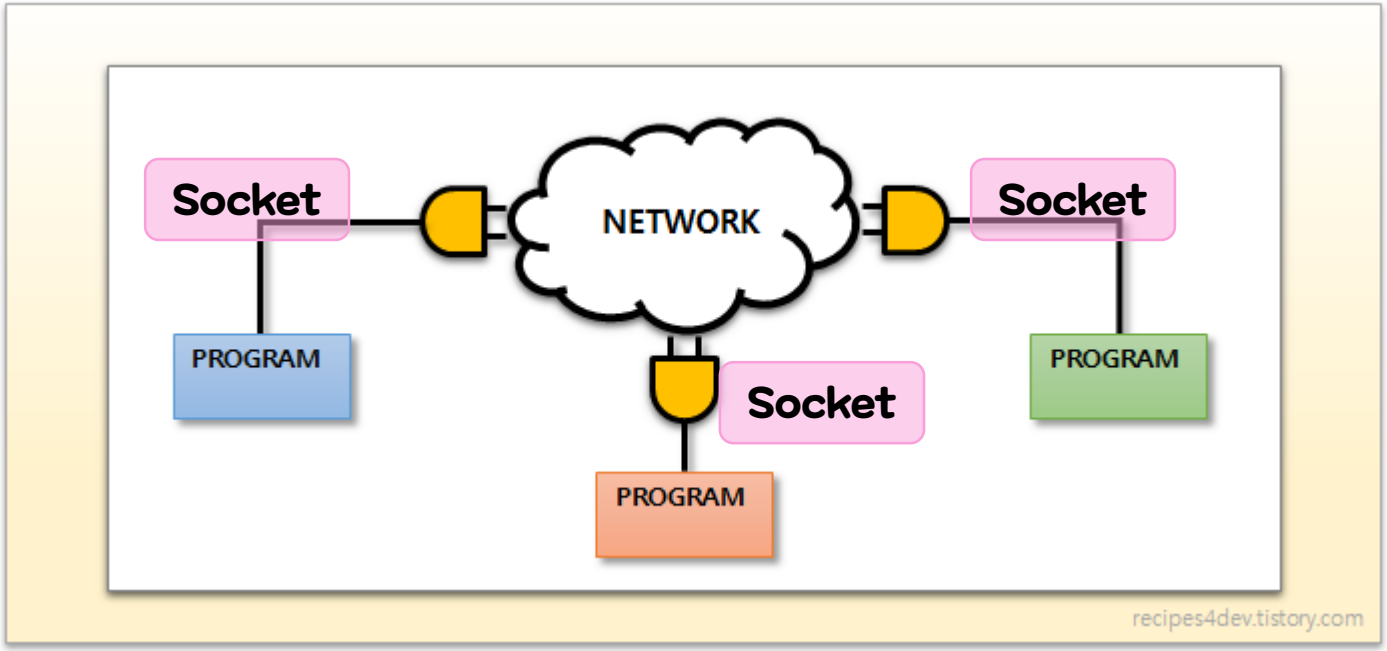
Socket

WITH 팀 뤼쳐드

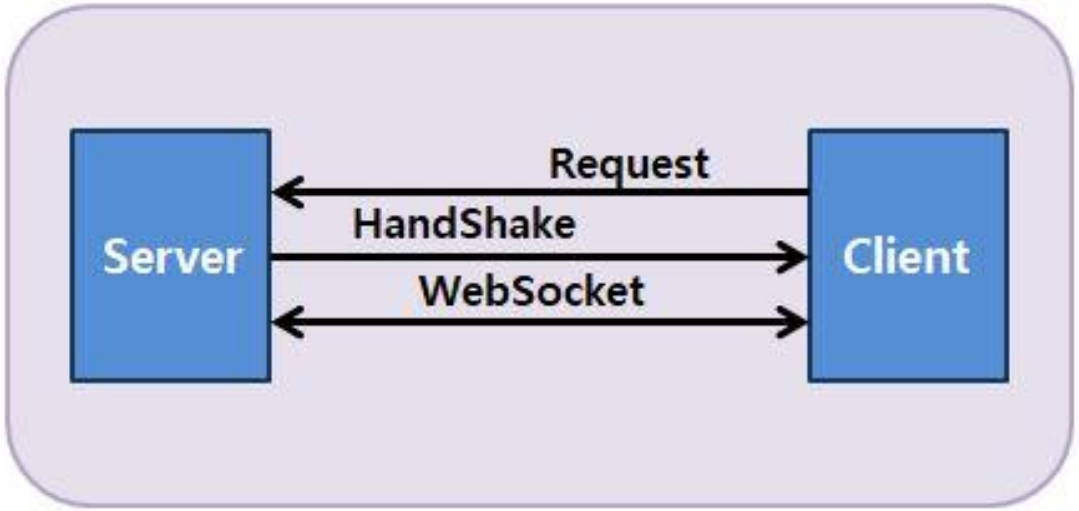
소켓

소켓(Socket)

- 프로세스가 네트워크로 **데이터를 내보내거나 데이터를 받기 위한** 실제적인 창구역할을 하는 것



소켓(Socket)



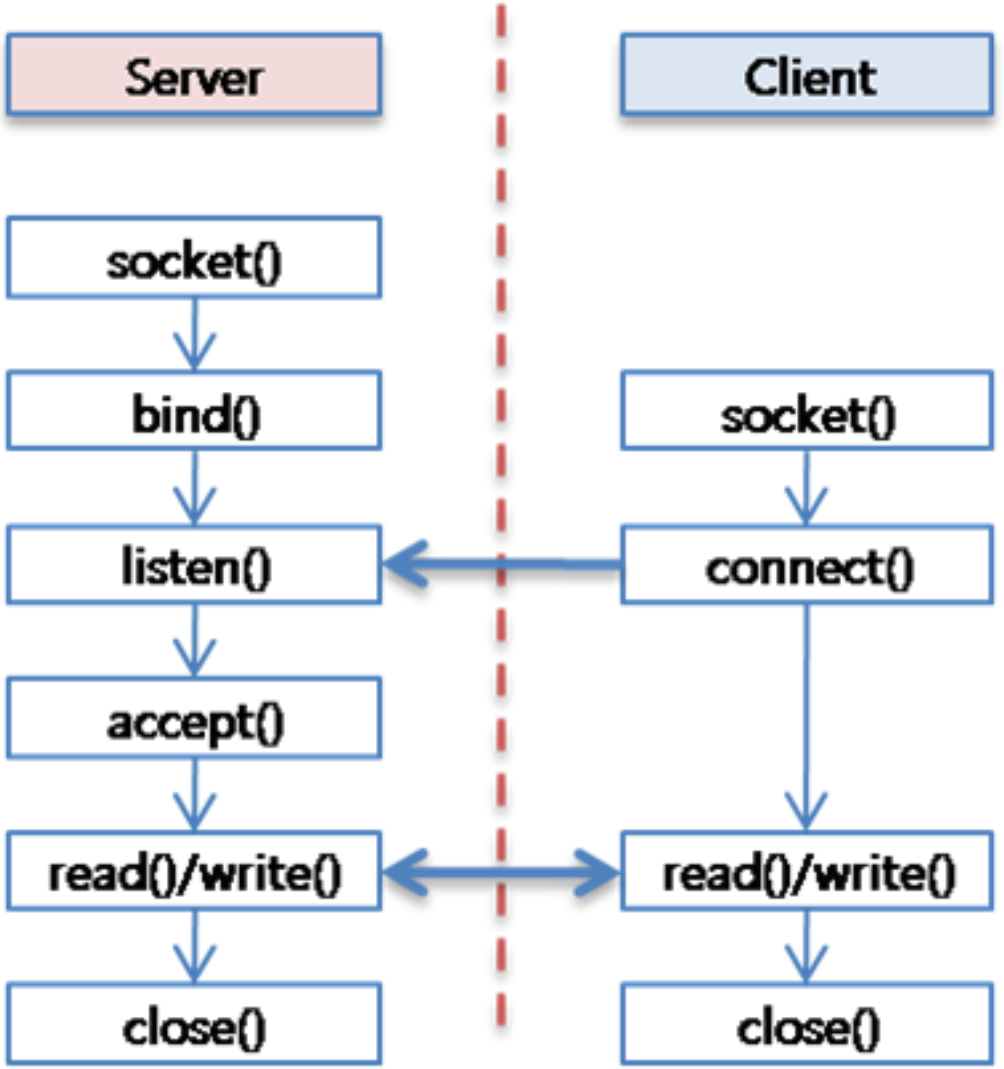
- **서버와 클라이언트를 연결**해주는

  도구로써 인터페이스 역할을 하는 것

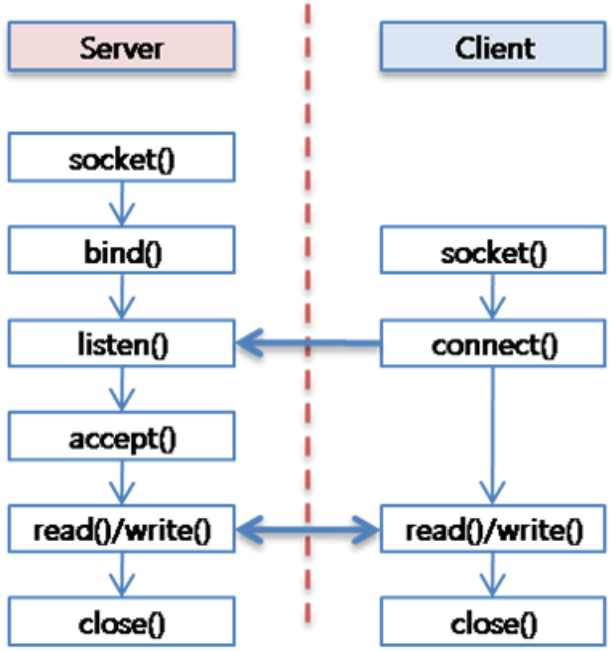
  - 서버 : 클라이언트 소켓의 연결 요청을 대기하고, 연결 요청이 오면 클라이언트 소켓을 생성해 통신을 가능하게 하는 것
  - 클라이언트 : 실제로 데이터 송수신이 일어나는 곳

- 소켓은 프로토콜, IP 주소, 포트 넘버로 정의된다.

소켓 흐름

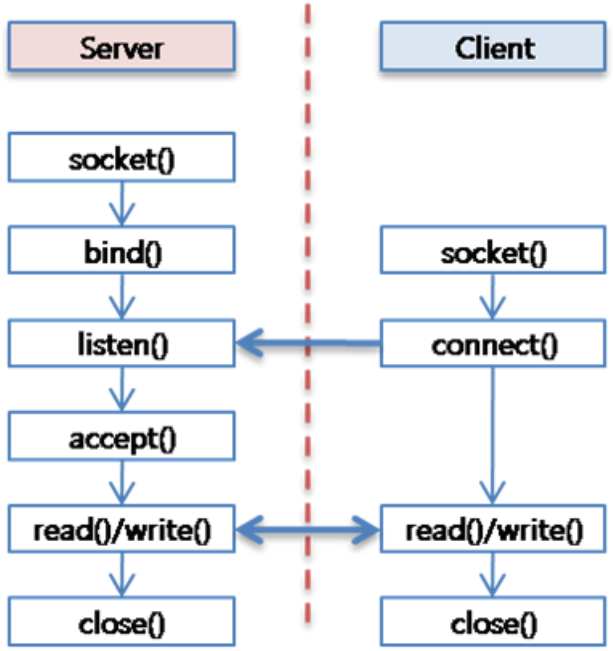

Server
Client
socket()
bind()
listen()
accept()
read()/write()
close()
socket()
connect()
read()/write()
close()

소켓 흐름



- **서버(Server)**
  - **socket()** : Socket 생성 함수
  - **bind()** : ip와 port 번호 설정 함수
  - **listen()** : 클라이언트의 요청에 수신 대기열을 만드는 함수
  - **accept()** : 클라이언트와의 연결을 기다리는 함수

소켓 흐름



- **클라이언트(client)**
  - **socket()** : 소켓을 여는 함수
  - **connect()** : 통신할 서버의 설정된 ip와 port 번호에 통신을 시도하는 함수
  - 통신 시도 시, 서버가 accept()함수를 이용해 클라이언트의 socket descriptor를 반환
  - 이를 통해 클라이언트와 서버가 서로 read() write()를 반복하며 통신

Socket.io

WebSocket이란?

- **WebSocket** : 양방향 소통을 위한 프로토콜(약속)
  - HTML5 웹 표준 기술
  - 빠르게 작동하며 통신할 때 아주 적은 데이터를 이용한다.
  - 이벤트를 단순히 듣고, 보내는 것만 가능하다.

- **Socket.io** : 양방향 통신을 하기 위해 웹 소켓 기술을 활용하는 라이브러리
  - 표준 기술이 아닌 **라이브러리**
  - 방 개념을 이용해 일부 클라이언트에게만 데이터를 전송하는 브로드캐스팅이 가능하다.

Socket.io

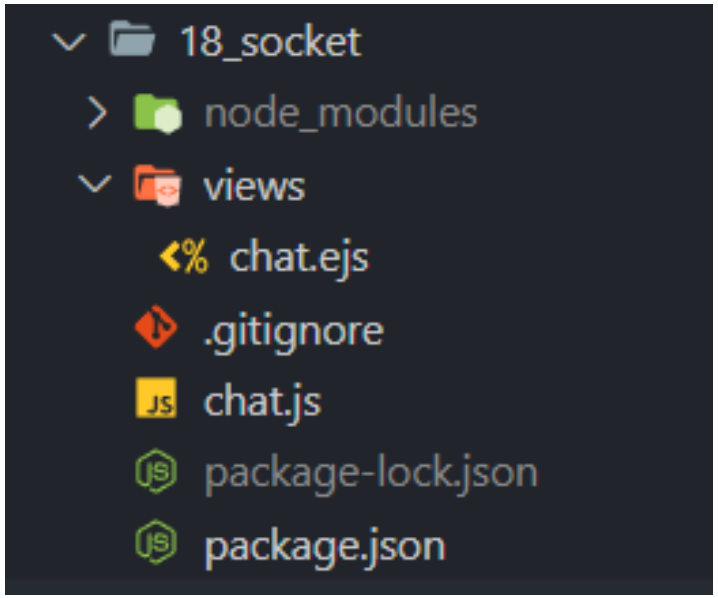
socket.io

- 웹 소켓을 기반으로 실시간 웹 애플리케이션을 위한 **Javascript 라이브러리**
- 웹 클라이언트와 서버 간의 실시간 **양방향 통신을 가능**하게 해주는 패키지
- 특징
  - 1) 이벤트 기반
  - 2) 서버 소켓과 클라이언트 소켓을 연결해 실시간 양방향 통신을 도와준다.

프로젝트 생성하기

나만의 **채팅 프로젝트**를 만들어보자!!

18_socket 프로젝트 구조




```
mkdir 18_socket # 폴더 생성
cd 18_socket # 폴더 이동

npm init -y # 프로젝트 시작 명령어 (-y 옵션: package.json 기본 값으로 생성)
# package.json에서 "main" 값을 index.js에서 chat.js로 변경 (진입점 파일명)

npm install express ejs

# chat.js 파일 생성
# .gitignore 파일 생성
```

Socket.io 사용하기

Socket.io 불러오기 (client)

https://socket.io/docs/v4/client-installation/

```
<script
  src="https://cdn.socket.io/4.5.3/socket.io.min.js"
  integrity="sha384-WPFUvHkB1aHA5TDSZi6xtDgkF0wXJcIIxXhC6h8OT8EH3fC5PWro5pWJ1THjcfEi"
  crossorigin="anonymous"
></script>
```

```
let socket = io.connect();
```

views/chat.ejs

Socket.io 불러오기 (server)


```
const express = require('express');
const app = express();
const http = require('http').Server(app);
const io = require('socket.io')(http);
```

```
io.on('connection', (socket) => {
  console.log('Server Socket Connected >> ', socket.id);
});
```

chat.js

실습42. Socket 연습

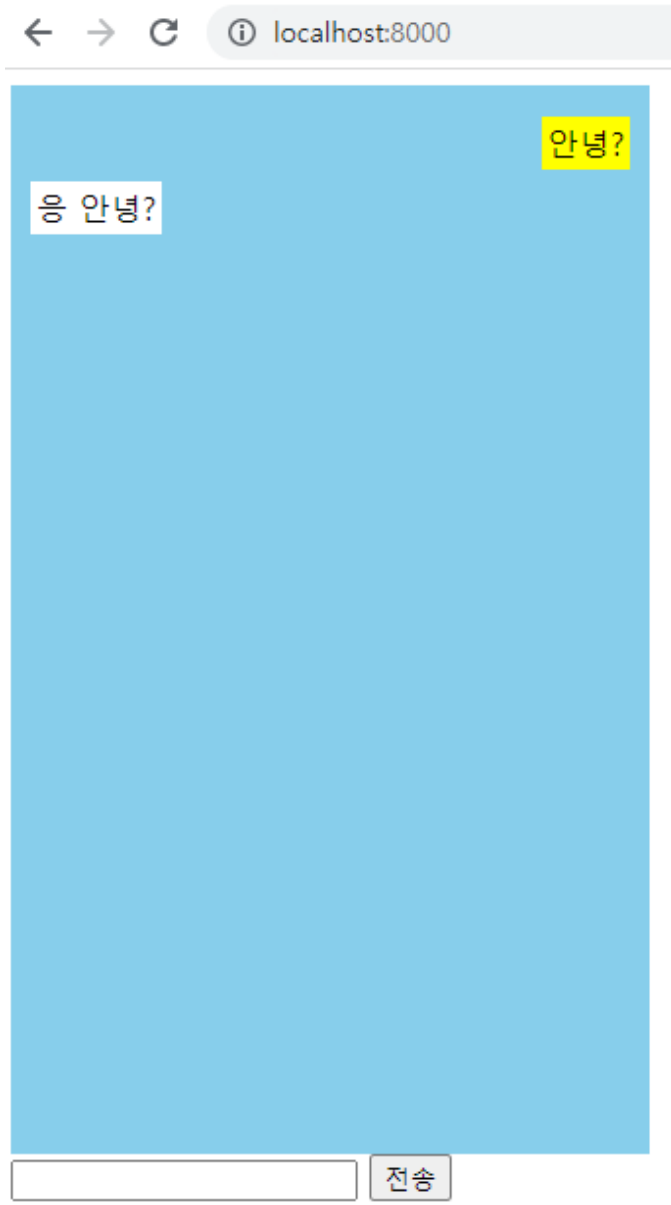
- 각 버튼을 누를 때마다 서버로 메시지 전송
- 서버
  - **클라이언트로부터 받은 메시지**를 console 에 찍고, 그에 대한 응답을 보내기
- 클라이언트
  - 버튼을 누를 때 서버로 메시지 보내기
  - **서버로부터 받은 응답 메시지**를 화면에 보여주기


```
// 이벤트 명을 지정하고 메세지를 보냄
socket.emit('전송할 이벤트 명', msg);

// 해당 이벤트를 받고 콜백함수를 실행
socket.on('받을 이벤트 명', (msg) => { ... });
```

실습43. 채팅창 UI 만들기

- 내가 보낸 채팅은 오른쪽에
- 다른 사람이 보낸 채팅은 왼쪽에 표시될 수 있도록



emit() from server

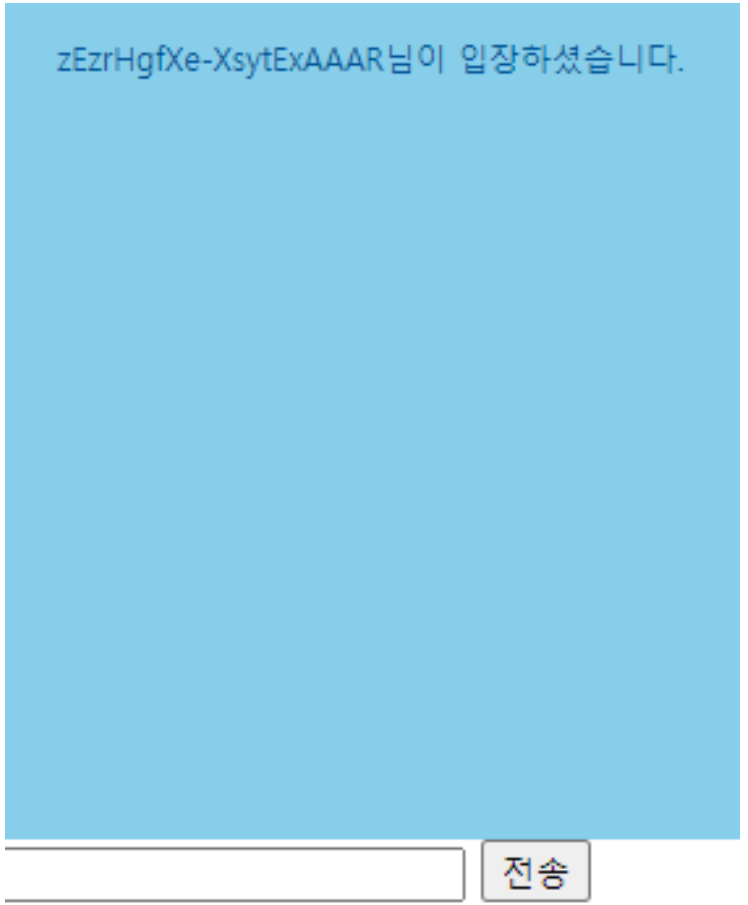

```
// 메시지를 전송한 클라이언트에게만 메시지를 전송한다.
socket.emit( "이벤트명", 데이터 );

// 서버에 접속된 모든 클라이언트에게 메시지를 전송한다.
io.emit( "이벤트명", 데이터 );
```

실습44. 채팅창 입장 안내 문구

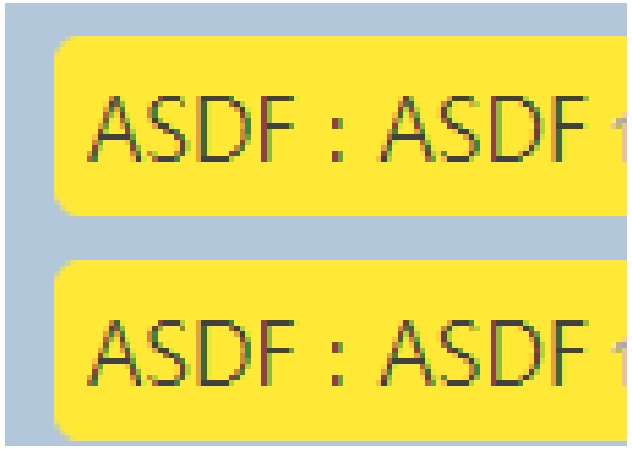
- Socket에 담겨 있는 id 정보를 이용하여 000 님이 입장했습니다. 라는 메시지 보여주기

```
io.on( "connection", function ( socket ){
    // 최초 입장했을 때
    console.log( "Server Socket Connected", socket.id );
    io.emit( "notice", `${socket.id}님이 입장하셨습니다.`);
});
```



실습45. 채팅창 메세지 전송

- 메시지에 누가 작성한 메시지인지 작성자 이름 받고 이름 보여주기
- 아래와 같은 형식으로 보여줘도 OK
  - 작성자 : MESSAGE



emit() from server2


```
// 소켓아이디에 해당하는 클라이언트에게 메시지 전송
io.to(소켓 아이디).emit( "이벤트명", 데이터 );
```

실습46. DM기능 추가하기

- 특정 사람에게만 메시지를 보낼 수 있도록 DM 기능 추가해보기
- DM 메시지는 일반 메시지와 다르게 ui 상으로 변화 주기