# 9주차 과제

**20161663 허재성**

## 1. websocket과 socket.io 통신 방식에 대해 자세히 조사한 후 이에 대해 기술

Websocket은 하나의 TCP 접속에 전이중 통신(Full Duplex) 채널을 제공하는 컴퓨터 통신 프로토콜이다. 전이중 통신이란 두 대의 단말기가 데이터를 송수신하기 위해 각각 독립된 회선을 사용하여 송신과 수신이 한 번에 이루어질 수 있는 것을 의미한다. HTTP는 반이중 통신(Half Duplex) 채널을 제공하여 한 쪽이 송신할 때 수신을 동시에 할 수 없다. 하지만 websocket은 http 프로토콜과 호환이 된다. Websocket은 웹 브라우저와 웹 서버간의 통신을 가능하게 하며 서버와의 실시간 데이터 전송을 용이하게 한다. 서버가 내용을 클라이언트에 보내는 방식을 사용한다. 연결이 유지된 상태에서는 메시지들이 오갈 수 있게 되어 있어 양방향 대화가 클라이언트와 서버 사이에 발생 가능하다.

Socket.io는 실시간 웹 애플리케이션을 위한 Javascript 라이브러리로 WebSocket을 기반으로 웹 클라이언트와 서버 간에 실시간으로 양뱡향 통신이 가능하도록 한다. Socket.io는 브라우저에서의 실행을 담당하는 클라이언트 라이브러리와 Node.js 용 서버 라이브러리 두 부분으로 이루어져 있다. WebSocket을 기반으로 하지만 편의 기능이 많이 추가되어 개발이 편하다.

만약 Http 프로토콜을 사용할 경우 클라이언트에서는 주기적으로 서버에 데이터의 업데이트가 있는지 확인하는 요청을 보내야 하며 서버는 데이터 업데이트가 있을 경우 해당 내용을 클라이언트에 응답하는 폴링 과정이 필요하다. 폴링 과정은 데이터 업데이트가 없을 때도 클라이언트가 서버에 요청을 해야 해서 비효율적이다. 하지만 WebSocket 프로토콜은 양방향 통신을 지원하므로 데이터가 업데이트되면 서버가 직접 알려주므로 효율적이다.

## 2. websocket/socket.io 통신 방식을 사용해서 실제 채팅 프로그램을 구현하게 됐을 때의 전체적인 구현 방식 기술

8주차와 비슷하게 ChatTemplate 컴포넌트의 getChatLog 함수에서 채팅 데이터를 가져오는 기능을 구현하였는데 8주차와는 다르게 서버에 저장된 채팅을 가져오도록 구현하였다. 서버 주소 http://localhost:8001에서 axios.get 함수를 사용하여 서버의 채팅 데이터가 현재 채팅 데이터보다 많아서 업데이트가 필요할 경우 채팅을 setChats로 갱신하였다. MessengerTemplate 컴포넌트에서는 onMsgSubmit 함수가 변경되었는데 username과 message를 올바르게 입력된 상태에서 전송 버튼을 누르면 axios.post 함수를 이용해 위의 서버로 채팅 데이터를 생성해 전송하고 서버에는 새로운 채팅 데이터가 추가되어 저장된다. 채팅이 서버에 저장되어 있으므로 새로고침하거나 client를 종료해도 서버를 종료하지 않는 한 서버에 저장되어 있던 채팅 내용을 그대로 가져올 수 있다. 마지막으로 ChatSearch라는 컴포넌트를 추가하여 채팅 내용 중 검색 내용을 포함한 메시지를 포함한 부분을 빨간색으로 보여주는 기능을 추가했는데, 새로 추가한 컴포넌트의 입력 창에 검색 keyword를 입력 후 버튼을 누르면 ChatTemplate에 작성한 onSearchKeySubmit 함수가 호출되어 서버로부터 가져온 chats 배열의 순회하여 조사해 해당 메시지를 찾는다. 메시지를 찾을 경우 해당 메시지를 가진 chats 배열의 원소의 searched를 true로 설정하면 붉은색으로 출력된다. 이 때 keyword를 검색하자마자 해당 메시지가 붉게 보이려면 검색되자 마자 리렌더링이 되어야 하는데 searched를 true로 설정할 때 단순 대입연산자만 사용하면 리렌더링이 되지 않으므로 setState인 setChats를 사용하여 searched를 변경해줘야 정상적으로 리렌더링이 되어 검색 결과가 바로 붉은 색으로 표시된다.