고소실 9주차 보고서

20191571 김세영

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

1)Websocket

Websocket 은 HTML5 표준 기술로, 사용자의 브라우저와 서버 사이의 동적인 양방향 연결 채널을 구성한다. Websocket API를 통해 서버로 메세지를 보내고, 요청 없이 응답을 받아오는 것이 가능하다. 기존에는 클라이언트가 서버에게 요청하는 polling 방식이었지만, 이 방식은 단방향이고, 새로운 데이터가 없을 때도 매번 데이터의 업데이트 필요 유무를 확인하기 위해서 서버에 요청을 보내야하기 때문에 비효율적이다. websocket을 이용하며 웹서버와 브라우저가 지속적으로 연결된 TCP 라인을 통해 실시간으로 데이터를 주고받을 수 있게 되었다. 서버와 클라이언트 간의 WebSocket 연결은 HTTP 프로토콜을 통해 이루어지고, 만약 연결이 정상적으로 이루어진다면 서버와 클라이언트 간에 TCP/IP 기반의 WebSocket 연결이 이루어지고 일정 시간이 지나면 HTTP 연결은 자동으로 끊어진다. Websocket은 별도의 포트를 사용하지 않고 HTTP와 같은 80번 포트를 사용하고 있기 때문에 클라이언트인 웹 브라우저뿐만 아니라 웹 서버도 기능을 지원하고 있어야 한다.

2)Socket.io

Socket.io 역시 웹 서버와 클라이언트의 실시간 양방향 통신이 가능한 환경을 제공해주는 통신 기술이다. websocket은 html5 기술이기 때문에 오래된 버전의 웹 브라우저는 websocket을 지원하지 않지만 Socket.io는 사용할 수 있다. node.js 기반으로 만들어졌으며, JavaScript를 이용하여 거의 모든 웹 브라우저와 모바일 장치를 지원하는 실시간 웹 애플리케이션 지원 라이브러리이다. Socket.io는 WebSocket, FlashSocket, AJAX Long Polling, AJAX Multi part Streaming, IFrame, JSONP Polling을 하나의 API로 추상화한 것이며, 웹 브라우저와 웹 서버의 종류와 버전을 파악하여 가장 적합한 기술을 선택하여 사용한다. 통신 시작 시 각 브라우저에 대해 websocket, polling, streaming 방식들 중 최적의 방식을 찾아 메시지를 보내준다.

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

1)websocket

javascript에서 WebSocketClient라는 라이브러리를 사용하여 websocketAPI를 사용할 수 있다.

```
var oSocket = new WebSocket("ws://localhost:80");
```

new WebSocket() 메서드로 websocket 객체를 생성하여 웹서버와 연결한다. WebSocket 프로토콜을 나타내는 ws://는 URI 스키마(Scheme)를 사용한다. 암호화 소켓은 https:// 처럼 wss://를 사용한다. 그리고 생성된 WebSocket 인스턴스를 이용하여 소켓에 연결할 때(onopen), 소켓 연결을 종료할 때(onclose), 메시지를 받았을 때(onmessage) 등의 이벤트를 각각 정의할 수 있다. 서버에 메시지를 보내고 싶을 때에는 send() 메서드를 이용한다. close() 메서드를 사용하여 서버와의 연결을 해제한다. 현재 웹 소켓 상태는 readyState 속성을 이용하여 WebSocket 객체 안에 있는 상수 값과 비교하여 알 수 있다.

Value	State	Description
0	CONNECTING	소켓이 생성되었다. 연결이 아직 열려 있지 않다.
1	OPEN	연결이 열려 있고, 통신할 준비가 되었다.
2	CLOSING	연결이 닫히는 중이다.
3	CLOSED	연결이 닫혔거나 열 수 없었다.

2) socket.io

NPM(Node Package Management)을 이용하여 Socket.io를 웹 서버에 설치한다.

```
npm install socket.io
```

Node.js의 내장객체인 http 모듈을 이용하여 만들어진 서버를 이용한다.

```
var app = require('http').createServer(handler);
var io = require('socket.io')(app);

app.listen(80);
```

on 메소드를 이용하여 이벤트 핸들러를 등록할 수 있고, emit 메소드를 이용하여 클라이언트로 이벤트를 전달할 수 있다. 이벤트 이름은 이미 정해진 이름(connection)외에 개발자가 임의로 정할 수도 있다.

```
var socket = io('http://localhost');
socket.on('news', function (data) {
    console.log(data);
    socket.emit('my other event', {
        my: 'data'
    });
});
```

클라이언트는 Socket.io 패키지에 있는 클라이언트 스크립트를 이용하여 작성할 수 있다. io 함수의 인자로 서버의 도메인 주소나 IP 주소를 입력하면, 해당 클라이언트의 socket 객체를 반환 받을 수 있다. socket객체의 on 메소드를 이용하여 이벤트 핸들러를 등록할 수 있고, emit 메소드를 이용하여 서버단으로 이벤트를 전송이 가능하다.