고급소프트웨어실습I

9주차 보고서

20171646 박태윤

**websocket과 socket.io 통신 방식에 대해 자세히 조사한 후 이에 대 해 기술**

websocket : 웹 서버와 클라이언트의 실시간 양방향 연결 채널을 구성하는 컴퓨터 통신 프로토콜입니다. websocket을 이용하면 하나의 HTTP접속만으로 양방향 메시지를 주고받을 수 있습니다. 기존에 존재하던 양방향 통신 방식인 Ajax, iframe과 같은 방식은 polling방식으로 새로운 데이터를 수신하려면 클라이언트가 서버에게 요청을 해야만 하는 과정을 거칩니다. 이러한 방식은 실시간으로 새로운 내용이 반영이 되어야 하는 서비스에서는 별로 좋지 못하며 새로 추가된 내용이 없더라도 매 request마다 응답을 줘야 하기 때문에 불필요한 overhead가 발생할 수 있습니다. 따라서 실시간이 요구되는 응용프로그램에는 조금 더 효과적인 websocket을 사용합니다. 기존의 통신 방식과는 달리 websocket은 클라이언트 쪽에서 서버에게 request없이도 response를 받을 수 있는 방식으로 동작을 합니다. 즉, 새로운 내용이 서버에 입력이 되었을 때 refresh를 하는 등의 불필요한 처리 없이 실시간으로 클라이언트 쪽에서의 내용이 업데이트가 됩니다. websocket은 response를 받은 후에도 커넥션이 살아있습니다. 즉 서버와 클라이언트 간의 연결이 일정 시간 동안 계속 유지가 되는데, 이를 통해 실시간으로 서버와 클라이언트 간의 양방향 통신 또는 데이터 전송이 가능하도록 합니다. websocket연결은 HTTP프로토콜을 통해 이루어지는데, HTTP에서 WebSocket으로의 프로토콜 전환을 뜻하는 handshake과정이 성공하면 새로운 소켓이 만들어지고 이를 통해 통신을 하는 방식이다. 연결이 계속 유지가 되는 stateful한 방식을 통해 실시간 양방향 통신을 할 수 있다는 장점이 있지만 비정상적으로 연결이 끊겼을 경우 이를 처리해야 하며 연결을 계속 유지해야 한다는 점에서 비용이 지속적으로 든다는 단점 또한 존재한다.

socket.io : 실시간 양방향 통신을 가능하게 하는 websocket은 HTML5의 기술이기 때문에 오래된 version의 웹 브라우저에서는 지원하지 않는 문제점을 가지고 있습니다. 이러한 문제점 때문에 JavaScript를 이용하여 브라우저 종류에 상관없이 실시간 양방향 통신 웹을 구현할 수 있도록 하는 기술인 socket.io가 등장하였습니다. 즉 websocket을 지원하는 브라우저라면 websocket방식을, 그렇지 않은 브라우저라면 socket.io를 통해 실시간 통신 동작을 하게 됩니다. 이는 브라우저와 웹 서버의 종류와 버전을 파악하여 websocket이외의 Ajax long polling이나 FlashSocket등 여러 통신 방법 중 하나를 선택하여 마치 websocket처럼 동작할 수 있게끔 하는 식으로 구현이 됩니다. 이는 여러 통신 방법들을 하나의 API로 추상화하는 방식으로 가능하게 합니다.

**실제로 websocket/socket.io 통신 방식을 사용해서 실제 채팅 프로 그램을 구현하게 됐을 때의 전체적인 구현 방식 기술**

websocket: 실습에서 사용했던 Node.js에서 websocket방식을 이용하려면 ws패키지를 이용하면 됩니다. 우선 서버를 구현하는데, require(‘ws’).Server를 이용하여 websocket server를 생성하고, 포트 넘버를 할당 받으면 이에 대해 on함수를 이용하여 클라이언트의 접속이나 메시지 등과 같은 이벤트에 대한 동작을 수행할 수 있습니다. 클라이언트는 앞서 만든 서버와 포트 넘버에 대응되는 url을 통해 WebSocket객체를 만들고 역시 on함수를 이용하여 서버에서 넘어온 이벤트에 대한 동작을 정의할 수 있습니다. 이벤트는 listen함수를 통해 받아올 수 있으며, 이 받아온 이벤트에 대해 on함수를 이용하여 클라이언트가 접속한 경우, 메시지를 보낸 경우 또는 클라이언트 쪽에서는 서버에서 메시지를 보낸 경우 등에 각각 대응되는 동작을 시킬 수 있습니다. 만일 메시지를 보내고 싶다면 send함수를 이용하여 서버에 보내줄 수 있으며 console.log함수 등을 이용하여 받은 메시지를 화면에 출력하는 등의 동작을 할 수 있습니다.

socket.io: socket.io에서는 서버 측에서는‘socket.io’패키지가, 클라이언트 쪽에서는 ‘socket.io-client’패키지가 필요합니다. 이 역시 실습에서 사용된 Node.js를 통해 구현을 하면 websocket방식과 달리 http서버를 만들어서도 동작 시킬 수 있습니다. 이는 우선 require(‘express’)를 통해 express객체를 생성한 후 require(‘http’).createServer(express 객체)를 통해 http서버를 만들 수 있습니다. 이렇게 만든 http서버를 실시간 양방향 통신이 가능한 socket.io 서버로 만들 수 있는데, 이는 require(‘socket.io’)를 통해 가능합니다. 이벤트에 대한 동작 정의는 앞선 websocket과 마찬가지로 on함수를 이용하여 정의할 수 있으며 이벤트 또한 역시 listen함수를 이용하여 받아올 수 있습니다.