컴퓨터네트워크 12주차 과제

- WebRTC -

제출일자	21.12.01
분 반	02
이 름	**배시현**
학 번	201702022

1)과제목표

이번 과제는 WebRTC를 사용하여 웹에서 실시간 커뮤니케이션 기능을 구현하는 것이다. Peerjs를 같이 사용하여서 웹에 화상통화앱을 만들어본다.

2)코드설명

1.server.js

```
JS server.js > ...
1   const express = require('express');
2   const app = express();
3   const server = require('http').Server(app);
4   const io = require('socket.io')(server);
5   const { v4: uuidV4 } = require('uuid');
6
7   app.set('view engine', 'ejs');
8   app.use(express.static('public'));
9
10  app.get('/', (req, res) => {
11    res.redirect(`/${uuidV4()}`);
12  });
13
14  app.get('/:room', (req, res) => {
15    res.render('room', { roomId: req.params.room });
16  });
17
18  io.on('connection', (socket) => {
19    socket.on('join-room', (roomId, userId) => {
20      socket.join(roomId);
21      socket.to(roomId).emit('user-connected', userId);
22
23      socket.on('disconnect', () => {
24        socket.to(roomId).emit('user-disconnected', userId);
25      });
26    });
27  });
28
29  server.listen(3000);
30
```

App.get('/', (req, res) => { 이부분으로 인하여 localhost:3000으로 접속시 랜덤한 roomid를 생성하여 redirection으로 다시 주소를 보낸다. 또한 ppt에 설명되어있는 방법으로 구현하기 위해 socket.io를 공부하였습니다.

Joining and leaving

You can call `join` to subscribe the socket to a given channel:

```
io.on("connection", socket => {
  socket.join("some room");
});
```

And then simply use `to` or `in` (they are the same) when broadcasting or emitting:

```
io.to("some room").emit("some event");
```

Socket.io 사이트에 있는 내용으로 해당내용을 바탕으로 공부했습니다.

2. public/script.js

```
navigator.mediaDevices.getUserMedia({
    video: true,
    audio: true,
  }).then(stream => {
    addVideoStream(myVideo, stream);

    myPeer.on('call', (call) => {
      call.answer(stream);
      const video = document.createElement('video');
      call.on('stream', (userVideoStream) => {
        addVideoStream(video, userVideoStream);
      });
    });

    socket.on('user-connected', (userId) => {
      connectToNewUser(userId, stream);
    });
  });

socket.on('user-disconnected', (userId) => {
  if (peers[userId]) peers[userId].close();
});

myPeer.on('open', (id) => {
  socket.emit('join-room', ROOM_ID, id);
});

function connectToNewUser(userId, stream) {
  const call = myPeer.call(userId, stream);
  const video = document.createElement('video');
  call.on('stream', (userVideoStream) => {
    addVideoStream(video, userVideoStream);
  });

  call.on('close', () => {
    video.remove();
  });

  peers[userId] = call;
}
```

```
function addVideoStream(video, stream) {
  video.srcObject = stream;
  video.addEventListener('loadedmetadata', () => {
    video.play();
  });
  videoGrid.append(video);
}
```

Script.js 또한 피피티에 나와있는대로 구현했다. 처음 addVideoStream함수를 작성하고 그 다음 Media calls를 참조하여 작성한다. 하지만 peesjs.com의 정보는 부족한 것 같아서 구글의 내용을 보고 분석 후 작성하였다. 그 다음 socket.on을 통해 user-connected를 설정하는데 그 안의 함수는 ConnnectToNewUser함수를 구현해야한다.

구현 후 테스트를 했을 때 여러 사용자가 입장하여도 비디오가 하나밖에 뜨지 않았다. 그래서 해결방법을 찾다가

```
const myPeer = new Peer(

);
```

해당 peer를 선언할 때 안에 내용을 삭제하였더니 제대로 실행되는 것을 확인하였다.

3. views/room.ejs

실습파일의 내용과 동일하다.

3)데모영상

https://youtu.be/JUGPqzrpBRg