WebRTC

WebRTC 란? (개념정리)

요즘 코로나로 인해서 많은 회사들이 비대면으로 회의를 진행하고 있다. 나 같은 경우는 IT 동아리를 참여하고 있는데, 원래는 대면으로 진행했던 동아리였지만 코로나로 인해서 게더타운 이라는 서비스를 이용해서 비대면으로 매주 세션을 진행하고 있다. 이러한

https://doiler.tistory.com/17

WebRTC(Web Real-Time Communication) 란?

- **WebRTC**(Web Real-Time Communication)은 웹 애플리케이션과 사이트가 중간자 없이 브라우저 간에 오디오나 영상 미디어를 포착하고 마음대로 스트림할 뿐 아니라, 임의의 데이터도 교환할 수 있도록 하는 기술입니다. WebRTC를 구성하는 일련의 표준들은 플러그인이나 제 3자 소프트웨어 설치 없이 종단 간 데이터 공유와 화상 회의를 가능하게 합니다.

  이를 위하여 WebRTC는 상호 연관된 API와 프로토콜로 구성되어 함께 작동

  ⇒ **WebRTC는 웹 또는 앱(Android, IOS)에서 별도의 소프트웨어, 플러그인 없이 음성, 영상, 텍스트 같은 데이터를 브라우저끼리 주고 받을 수 있게 해주는 기술이라고 할 수 있다.**

- WebRTC 구성 방식
  - 두 단말이 서로 실시간 통신을 하기 위해서는 다음과 같은 사항이 필요
  1. 기기의 스트리밍 오디오 / 비디오 / 데이터를 가져올 수 있을 것
  2. 소통하고자 하는 기기의 IP주소와 포트 등 네트워크 통신을 위한 데이터
  3. 에러의 보고, 세션 초기화를 위한 신호 통신을 관리
  4. 서로 소통할 수 있는 해상도인지, 코덱은 알맞은지 등 capability 정보 교환
  5. 실제로 연결을 맺는다
  6. 이후 연결되는 동안 스트리밍 오디오 / 비디오 / 데이터를 주고받을 수 있어야 함.

  - 위 조건들을 충족하기 위해 WebRTC는 다음과 같은 API를 제공하고 있음
    - MediaStream : 카메라와 마이크 등의 데이터 스트림 접근
    - RTCPeerConnection : 암호화 및 대역폭 관리 및 오디오, 비디오 연결
    - RTCDataChannel : 일반적인 데이터의 P2P 통신

이 3 가지의 API를 통해 데이터 교환이 이루어지며 RTCPeerConnection들이 적절하게 데이터를 교혼할 수 있게 처리해주는 과정을 시그널링(Signaling)이라고 함.

- 시그널링을 수행하는 서버 = 시그널 서버
  - 전이중 통신을 지원하는 websocket으로 이를 구현하는 것이 가장 적합
  - 세션제어메세지, 네트워크 구성, 미디어 기능 정보 교환
    - 세션 제어 메세지 : 통신을 초기화하거나 닫고 오류를 보고
    - 네트워크 구성 : 외부세계에 컴퓨터의 IP 주소와 포트는 무엇인지 파악
    - 미디어 기능 : 브라우저와 통신하려는 브라우저에서 처리할 수 있는 코덱과 해상도가 무엇인지 파악
  - 시그널링은 P2P 스트리밍 시작 전에 성공적으로 완료되어야 함.

- 서버는 단지 웹 브라우저를 특정하기 위한 시그널링(Signaling) 과정으로만 쓰임
- 시그널링을 마친 뒤 실제 데이터는 P2P 혹은 중개 서버를 통해 주고받음

OpenVidu

- OpenVidu는 웹 또는 모바일 애플리케이션에 화상 통화를 쉽게 추가할 수 있는 플랫폼입니다.
- 애플리케이션에 매우 쉽게 통합할 수 있는 완전한 기술 스택을 제공합니다.
- 주요 목표는 개발자가 코드에 미치는 영향을 최소화하면서 매우 빠르게 앱에 실시간 커뮤니케이션을 추가할 수 있도록 하는 것입니다.

openvidu-hello-world

Hello World - OpenVidu Docs

https://docs.openvidu.io/en/stable/tutorials/openvidu-hello-world/

- **그룹 화상 통화를 할 수 있는 최소한의 기능**

- OpenVidu 애플리케이션 아키텍쳐에서 필요한 3가지 컴포넌트
  - OpenVidu deployment
  - server application
  - client application

- **첫 줄은 코드를 따라 다른 지점에서 필요할 변수를 선언합니다.**

```
var OV;
var session;

var mySessionId = document.getElementById("sessionId").value;
```

  - OV 는 OpenVidu object
  - session은 우리가 연결할 화상 통화

- **Initialize a new session and its events**

```
OV = new OpenVidu();
session = OV.initSession();

session.on("streamCreated", function (event) {
    session.subscribe(event.stream, "subscriber");
    });`
```

  - **OpenVidu 객체를 가져오고 이를 사용하여 Session 객체를 초기화합니다.**
  - streamCreated에서 event.stream 속성에서 사용할 수 있는 특정 스트림을 “subscriber”gka.

- 여기까지 하면 세션에 참여할 준비 됨.

  → 그러나 액세스 권한을 얻으려면 여전히 토큰이 필요하므로 서버 애플리케이션에 토큰을 요청해야 함

  → 그러면 서버 애플리케이션이 OpenVidu 배포에 대한 토큰 요청

- **Get an OpenVidu token**
  - 아래는 서버 애플리케이션에서 최종적으로 토큰을 검색하는 역할을 하는 코드
  - 튜토리얼에서는 jQuery.ajax() 메서드를 사용하여 필요한 HTTP 요청을 수행

```
var APPLICATION_SERVER_URL = "http://localhost:5000/";
// 백엔드 서버

function getToken(mySessionId) {
    return createSession(mySessionId).then(sessionId => createToken(sessionId));
}
// getToken 함수는 tptus ID를 매개변수로 받아와서 토큰 생성
// createSession함수 호출하여 세션 생성한 후, 생성된 세션 ID를 사용하여 createToken 함수를 호출

function createSession(sessionId) {
    return new Promise((resolve, reject) => {
        $.ajax({
            type: "POST",
            url: APPLICATION_SERVER_URL + "api/sessions",
```

```
            data: JSON.stringify({ customSessionId: sessionId }),
            headers: { "Content-Type": "application/json" },
            success: response => resolve(response), // The sessionId
            error: (error) => reject(error)
        });
    });
}
//

function createToken(sessionId) {
    return new Promise((resolve, reject) => {
        $.ajax({
            type: 'POST',
            url: APPLICATION_SERVER_URL + 'api/sessions/' + sessionId + '/connections',
            data: JSON.stringify({}),
            headers: { "Content-Type": "application/json" },
            success: (response) => resolve(response), // The token
            error: (error) => reject(error)
        });
    });
}
```

- POST로

- **Connect to the session using the token**

```
getToken(mySessionId).then(token => {

    session.connect(token)
        .then(() => {
            document.getElementById("session-header").innerText = mySessionId;
            document.getElementById("join").style.display = "none";
            document.getElementById("session").style.display = "block";

            var publisher = OV.initPublisher("publisher");
            session.publish(publisher);
        })
        .catch(error => {
            console.log("There was an error connecting to the session:", error.code, error.message);
        });
});
```

  - **변수 mySessionId는 토큰을 원하는 OpenVidu 세션**