

Part3. 웹뷰와 RN 앱 통신 학습

유튜브 SDK 활용 학습 앱 프로젝트

프로젝트2 - 유튜브 영상 구간 반복 학습기

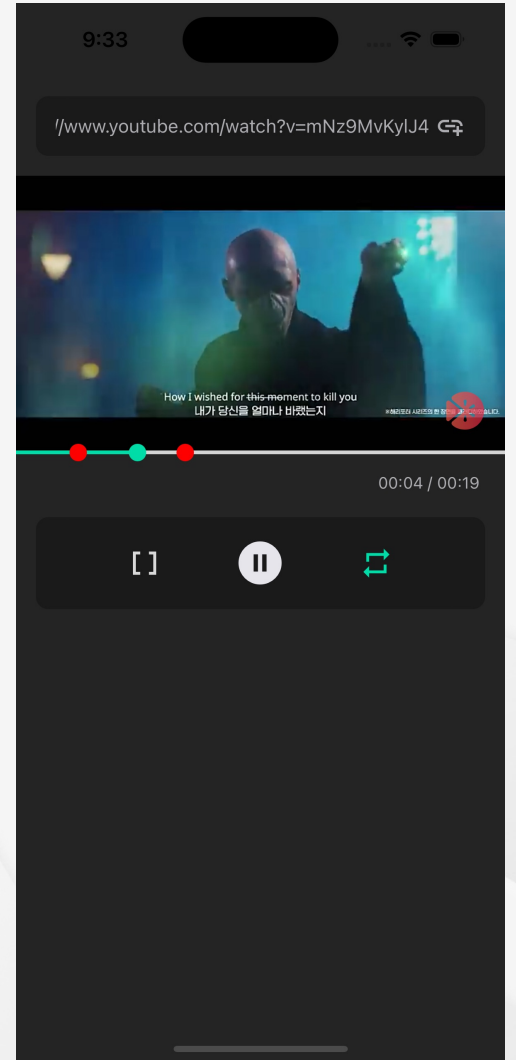
학습 목표 및 구성

학습 목표

- 01** Static HTML을 이용해서 웹 사이트를 로딩 할 수 있다
- 02** 웹뷰를 이용하여 웹용 API를 앱에서 사용할 수 있다
- 03** 웹뷰를 이용하여 앱과 웹간 통신을 구현할 수 있다
- 04** React Native에서 터치, 드래그 등 제스처를 처리 할 수 있다
- 05** 웹뷰를 디버깅 할 수 있다

유튜브 영상 구간 반복 학습기

- 링크 입력 컴포넌트
- 유튜브 영상 로드
- 재생 / 멈춤 기능
- 영상 재생 시간 정보
- Seek Bar
- 구간 반복 기능



프로젝트2 - 유튜브 영상 구간 반복 학습기 (React Native)

프로젝트 셋업

프로젝트 초기화

- `npx react-native@0.73.6 init LoopTube --version 0.73.6`

패키지 설치

- react-native-vector-icons
- react-native-webview

Figma

- 웹 기반 UI/UX 디자인 도구
- **실시간 협업:** 여러 사용자가 동시에 작업 가능
- **디자인-개발 협업:** 디자인 스펙 및 요구사항 명확화
- **코드와의 연계:** CSS, iOS, Android 코드 추출 가능



프로젝트2 - 유튜브 영상 구간 반복 학습기 (React Native)

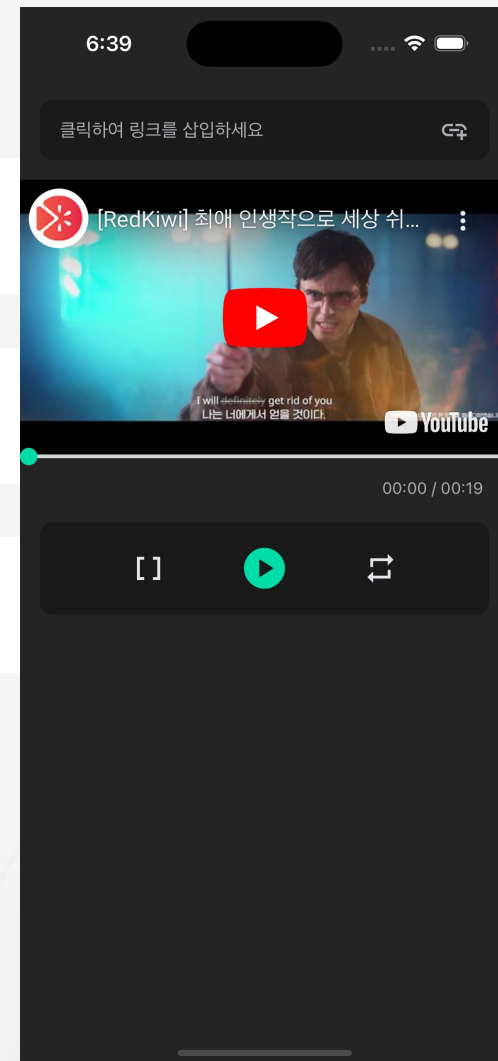
유튜브 영상 로드

학습 목표

01 Static HTML을 이용해서 유튜브 API를 이용할 수 있다

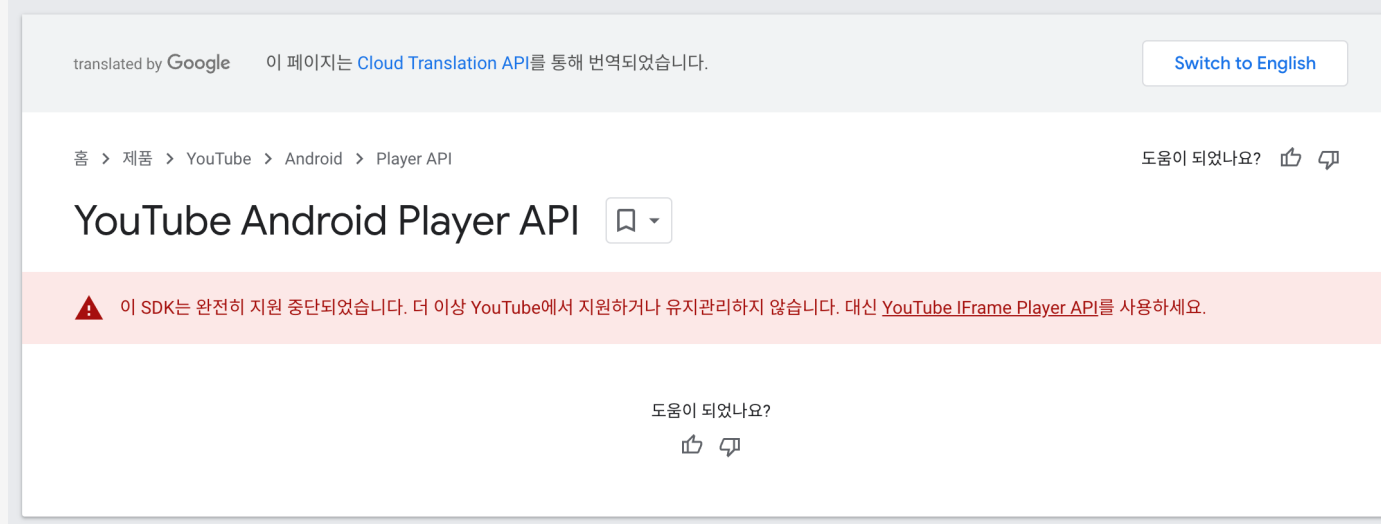
02 React Native TextInput을 사용할 수 있다

03 URL의 쿼리 스트링을 파싱 할 수 있다



YouTube Player API

- 공식적인 API는 웹용 YouTube IFrame Player API만 제공하고 있음
- 이러한 경우 웹뷰를 이용해서 React Native에서 YouTube IFrame API를 사용할 수 있음
- https://developers.google.com/youtube/iframe_api_reference



프로젝트2 - 유튜브 영상 구간 반복 학습기 (React Native)

웹뷰 최적화 하기

학습 목표

01

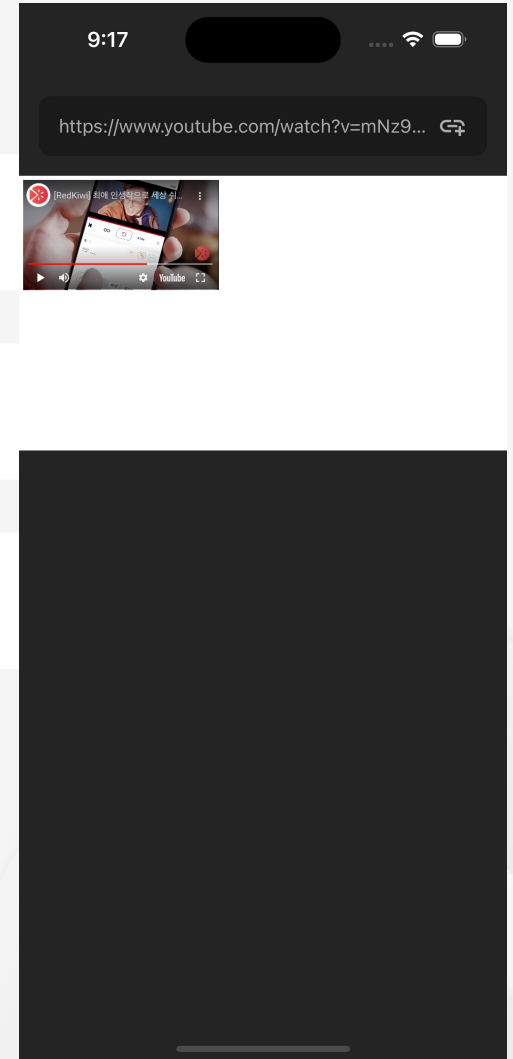
웹뷰의 뷰포트 크기를 디바이스에 최적화 할 수 있다

02

영상이 전체 화면으로 재생되는 것을 컨트롤 할 수 있다

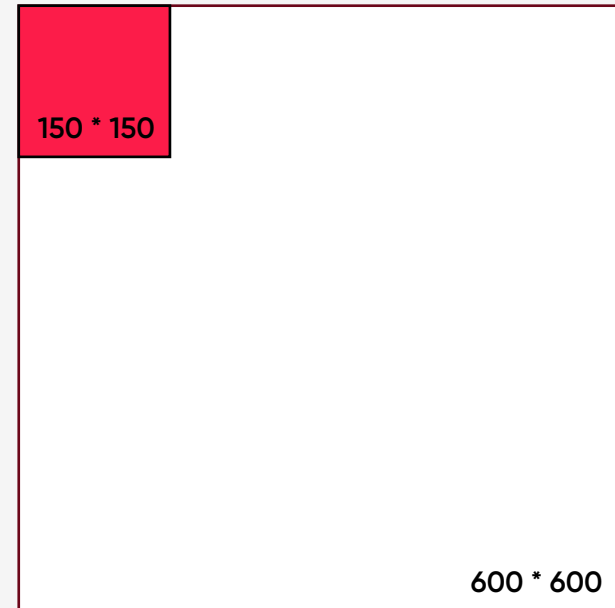
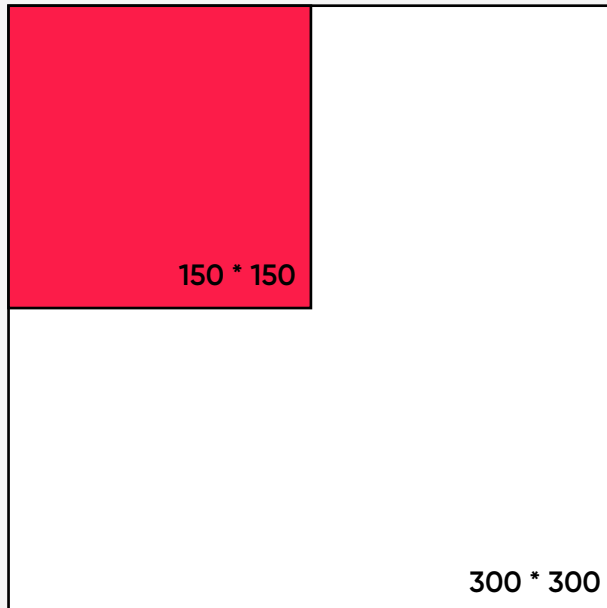
03

영상이 자동 재생되는 것을 컨트롤 할 수 있다



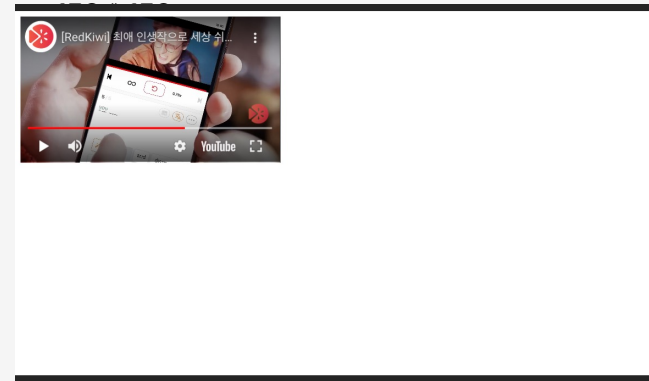
뷰포트

- 웹사이트를 볼 때 사용하는 화면의 영역



뷰포트

- 뷰포트를 디바이스의 크기와 일치시켜야 함



프로젝트2 - 유튜브 영상 구간 반복 학습기 (React Native)

재생 / 멈춤 기능 구현하기

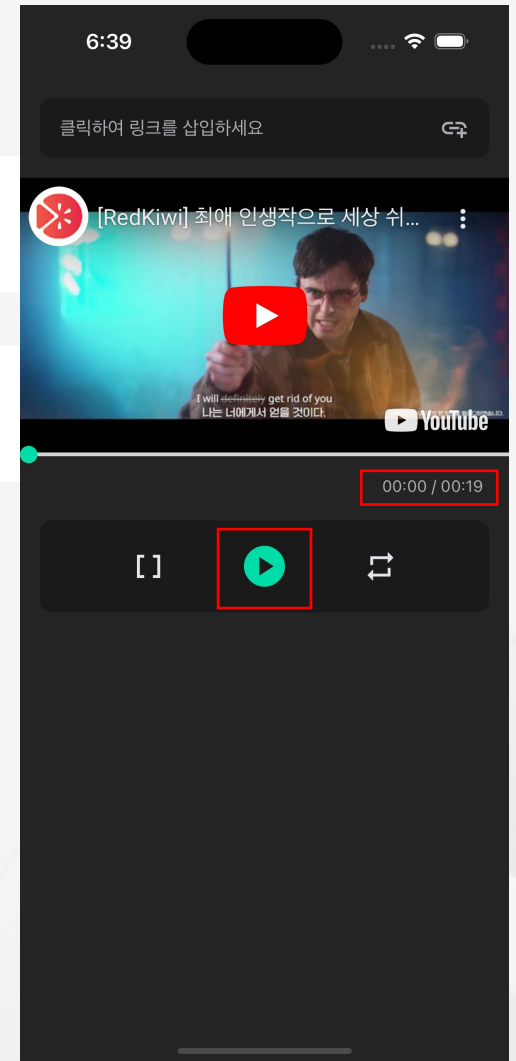
학습 목표

01

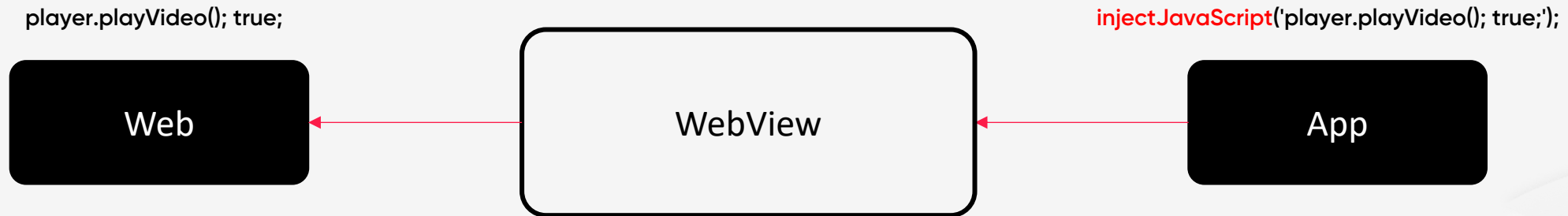
앱과 웹 간에 통신을 할 수 있다

02

웹에서 앱으로 여러 메시지를 보내고 이를 앱에서 처리할 수 있다



앱에서 웹으로 메시지 보내기

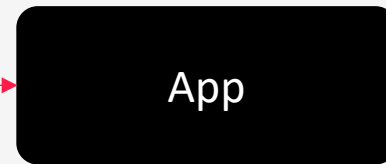


웹에서 앱으로 메시지 보내기

```
window.ReactNativeWebView.postMessage(playerState);
```



```
onMessage(playerState);
```

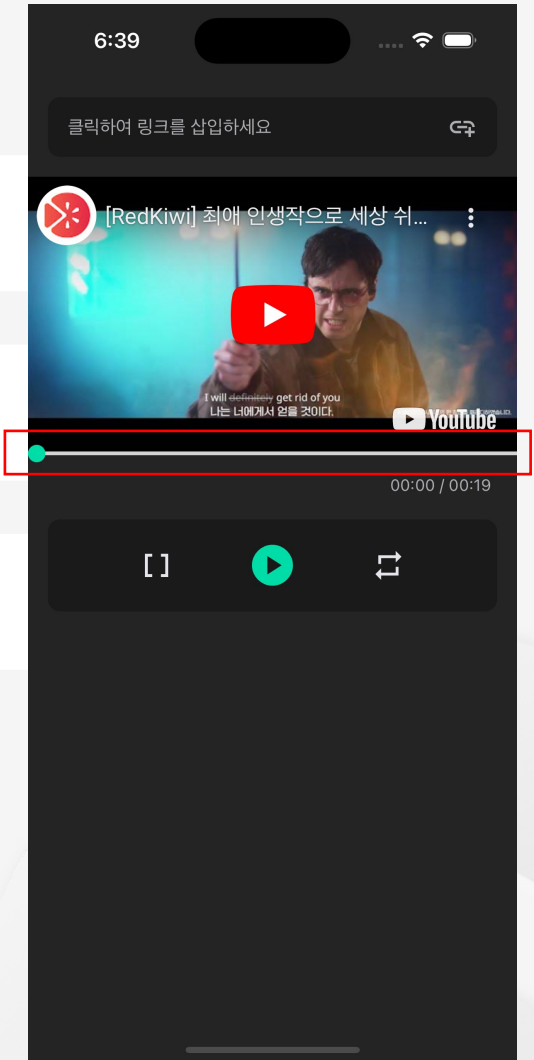


프로젝트2 - 유튜브 영상 구간 반복 학습기 (React Native)

Seek bar 구현하기

학습 목표

- 01 React Native의 Animation을 이용해서 SeekBar에 현재 재생 시점을 표시할 수 있다
- 02 React Native의 PanResponder를 이용해서 SeekBar를 컨트롤 가능하게 만들 수 있다
- 03 SeekBar로 웹과 통신하여 영상을 컨트롤 할 수 있다



PanResponder

- React Native에서 터치와 제스처 이벤트를 쉽게 처리할 수 있도록 도와주는 유틸리티
- 사용자가 화면을 터치하고 이동하는 동작을 감지하고 이에 대한 반응을 정의할 수 있음
- 주요 콜백 함수
 - onStartShouldSetPanResponder
 - onMoveShouldSetPanResponder
 - onPanResponderGrant
 - onPanResponderMove
 - onPanResponderRelease

onStartShouldSetPanResponder

- 사용자가 화면을 터치했을 때 PanResponder가 활성화될지 여부를 결정
- 반환값이 true일 경우, PanResponder는 해당 터치 이벤트를 처리
- 주로 사용자가 화면을 터치하는 순간에 어떤 조건을 체크하여 PanResponder를 활성화할지 결정하는 데 사용

onMoveShouldSetPanResponder

- 사용자가 터치한 상태에서 이동할 때 PanResponder가 활성화될지 여부를 결정
- 반환값이 true일 경우, PanResponder는 해당 이동 이벤트를 처리
- 사용자가 화면을 터치한 채로 이동하는 상황에서 PanResponder가 활성화될지 여부를 결정

onPanResponderGrant

- PanResponder가 활성화되고 제스처가 시작될 때 호출
- 여기서 제스처 시작 시 필요한 초기 설정을 할 수 있음
- 예를 들어, 사용자가 드래그를 시작할 때 필요한 초기화 작업을 수행
- SeekBar에서는 드래그를 시작할 때 영상을 일시 정지 할 수 있음

onPanResponderMove

- 사용자가 터치한 상태에서 이동할 때 호출
- gestureState 객체를 통해 터치 이동 정보를 받아 처리
- 사용자의 이동 경로를 추적하거나 UI 요소를 드래그할 때 사용
- SeekBar에서는 사용자의 이동 위치를 기반으로 새로운 비디오 재생 시간을 계산하고, 이를 SeekBar에 반영

gestureState

stateID: 각 터치 시퀀스에 대한 고유 식별자. 여러 손가락으로 터치하는 멀티터치 제스처를 처리할 때 유용

moveX: 마지막으로 기록된 이동 위치의 x 좌표

moveY: 마지막으로 기록된 이동 위치의 y 좌표

x0: 터치가 시작된 위치의 x 좌표

y0: 터치가 시작된 위치의 y 좌표

dx: 제스처 동안의 총 이동 거리의 x 축 변화량 ($\text{moveX} - \text{x0}$)

dy: 제스처 동안의 총 이동 거리의 y 축 변화량 ($\text{moveY} - \text{y0}$)

vx: x 축의 현재 제스처 이동 속도

vy: y 축의 현재 제스처 이동 속도

numberActiveTouches: 현재 화면에 접촉 중인 손가락의 수. 멀티터치를 감지하는 데 유용.

onPanResponderRelease

- 사용자가 터치를 해제할 때 호출
- 제스처가 끝날 때 필요한 작업을 수행
- 예를 들어, 사용자가 드래그를 마쳤을 때 최종 위치를 설정하는 작업을 수행.
- SeekBar에서는 최종 이동 위치를 기반으로 비디오 재생 시간을 설정하고, 비디오를 재생

프로젝트2 - 유튜브 영상 구간 반복 학습기 (React Native)

구간 반복 기능 구현하기

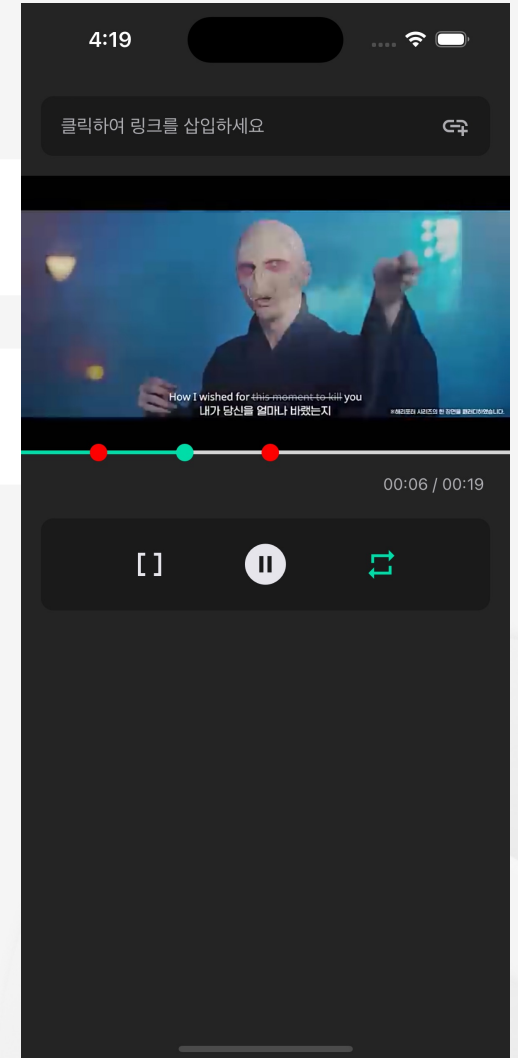
학습 목표

01

영상의 구간을 설정하여 반복하는 기능을 구현한다

02

웹뷰를 디버깅 할 수 있다



웹뷰 디버깅 하기

- 웹뷰에서 발생하는 에러 또는 콘솔 로그 같은 것들은 메트로 서버에 나타나지 않음
- 웹뷰에서 `webViewDebuggingEnabled` 활성화
- iOS -> Safari 개발자 도구 사용
- Android -> Chrome 개발자 도구 사용
- <https://github.com/react-native-webview/react-native-webview/blob/master/docs/Debugging.md#debugging-webview-contents>

프로젝트2 - 유튜브 영상 구간 반복 학습기 (Expo)

프로젝트 셋업

프로젝트 초기화

- `yarn create expo --template expo-template-blank-typescript@50`
- Eslint
 - <https://docs.expo.dev/guides/using-eslint/>
- Prettier
 - <https://docs.expo.dev/guides/using-eslint/#prettier>

패키지 설치

- react-native-vector-icons
 - 따로 설치 할 필요 없이 @expo/vector-icons 사용
- react-native-webview
 - `npx expo install react-native-webview`

Figma

- 웹 기반 UI/UX 디자인 도구
- **실시간 협업:** 여러 사용자가 동시에 작업 가능
- **디자인-개발 협업:** 디자인 스펙 및 요구사항 명확화
- **코드와의 연계:** CSS, iOS, Android 코드 추출 가능



프로젝트2 - 유튜브 영상 구간 반복 학습기 (Expo)

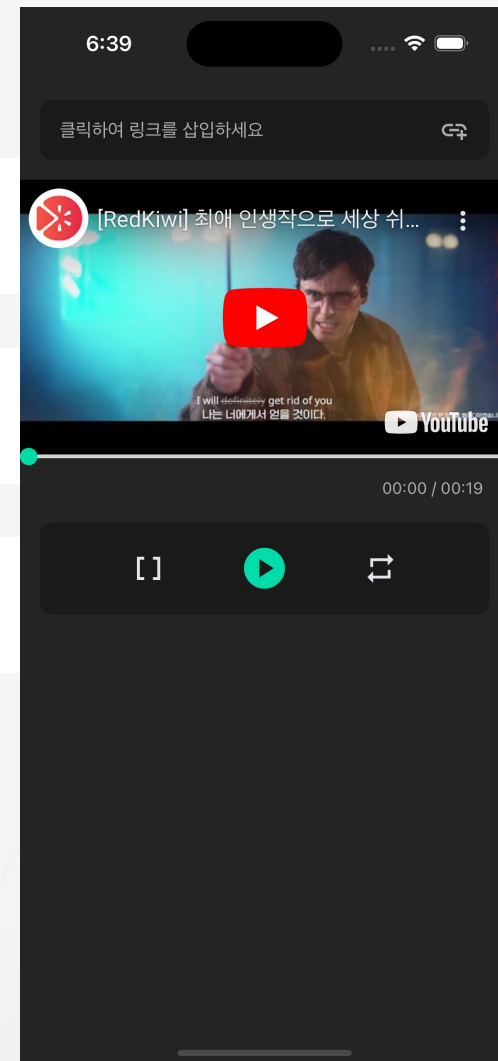
유튜브 영상 로드

학습 목표

01 Static HTML을 이용해서 유튜브 API를 이용할 수 있다

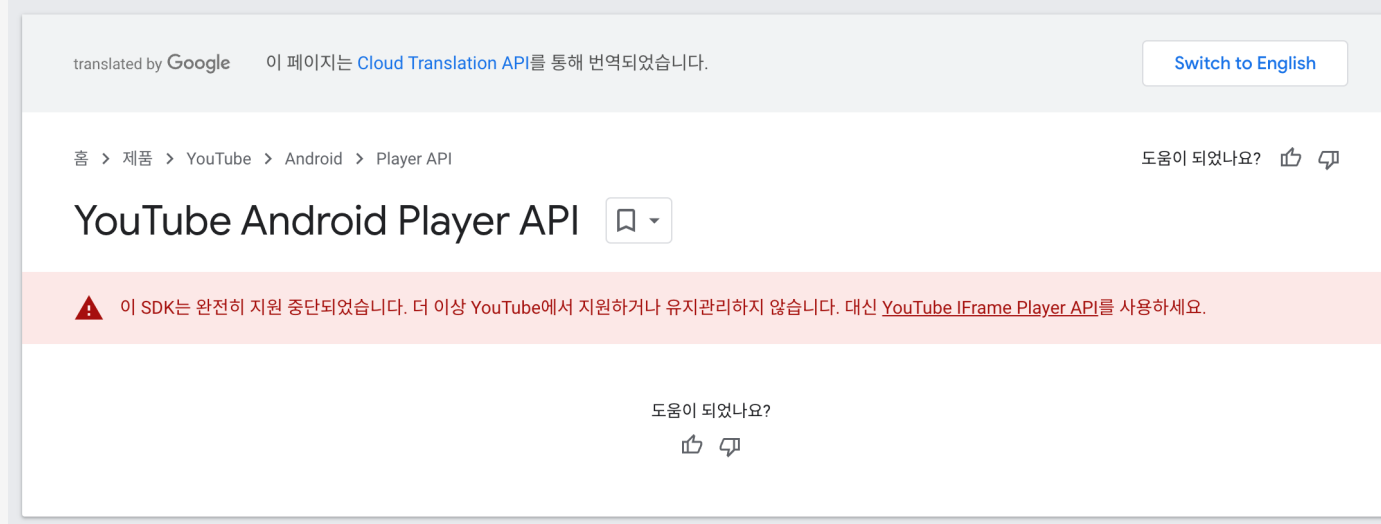
02 React Native TextInput을 사용할 수 있다

03 URL의 쿼리 스트링을 파싱 할 수 있다



YouTube Player API

- 공식적인 API는 웹용 YouTube IFrame Player API만 제공하고 있음
- 이러한 경우 웹뷰를 이용해서 React Native에서 YouTube IFrame API를 사용할 수 있음
- https://developers.google.com/youtube/iframe_api_reference



프로젝트2 - 유튜브 영상 구간 반복 학습기 (Expo)

웹뷰 최적화 하기

학습 목표

01

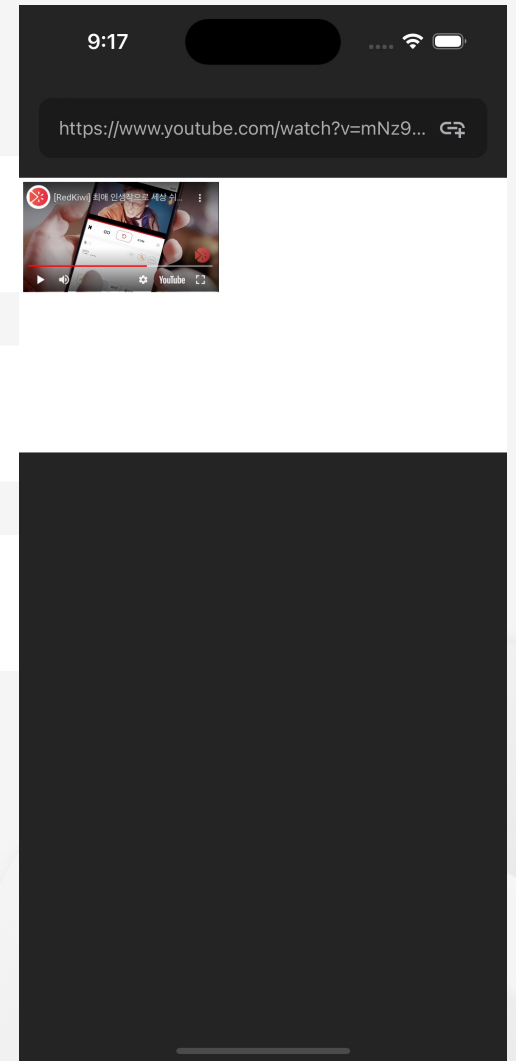
웹뷰의 뷰포트 크기를 디바이스에 최적화 할 수 있다

02

영상이 전체 화면으로 재생되는 것을 컨트롤 할 수 있다

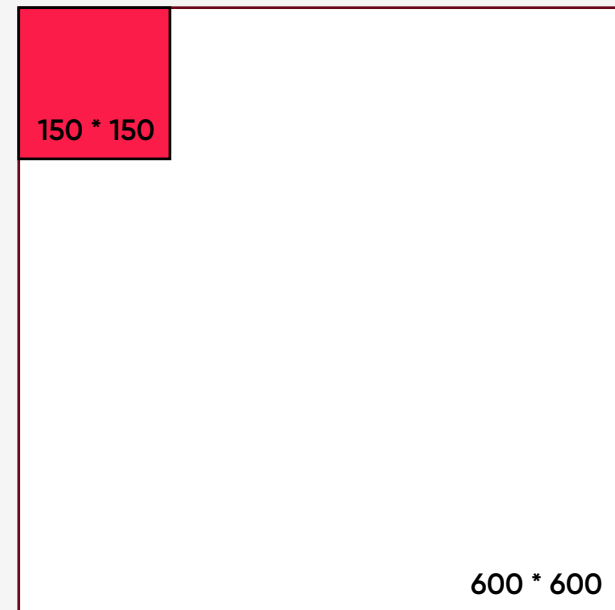
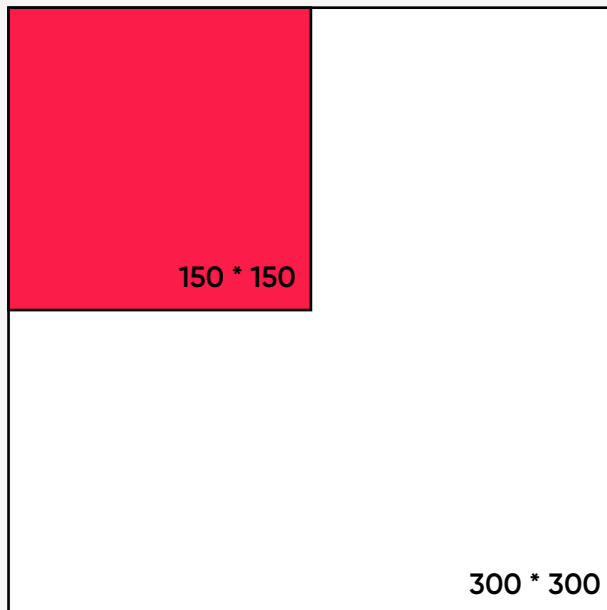
03

영상이 자동 재생되는 것을 컨트롤 할 수 있다



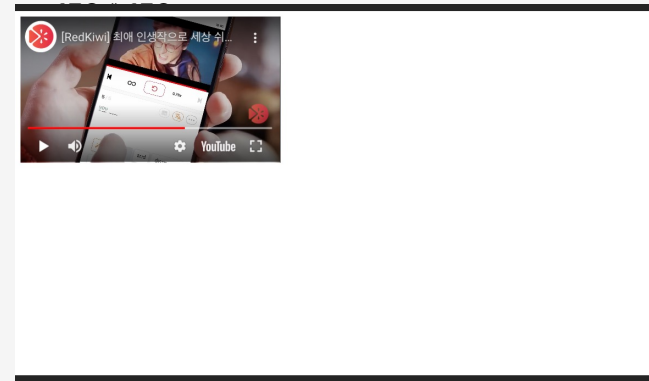
뷰포트

- 웹사이트를 볼 때 사용하는 화면의 영역



뷰포트

- 뷰포트를 디바이스의 크기와 일치시켜야 함



프로젝트2 - 유튜브 영상 구간 반복 학습기 (Expo)

재생 / 멈춤 기능 구현하기

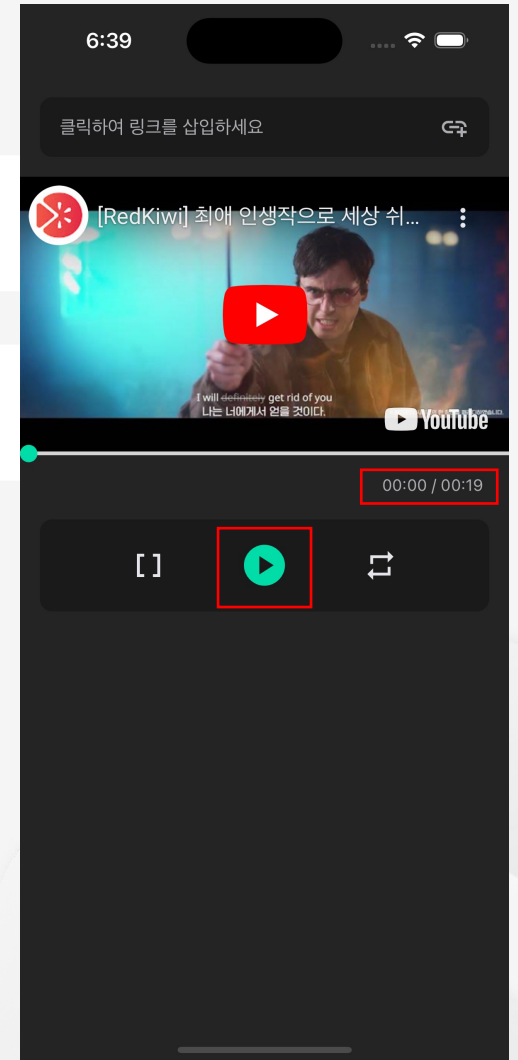
학습 목표

01

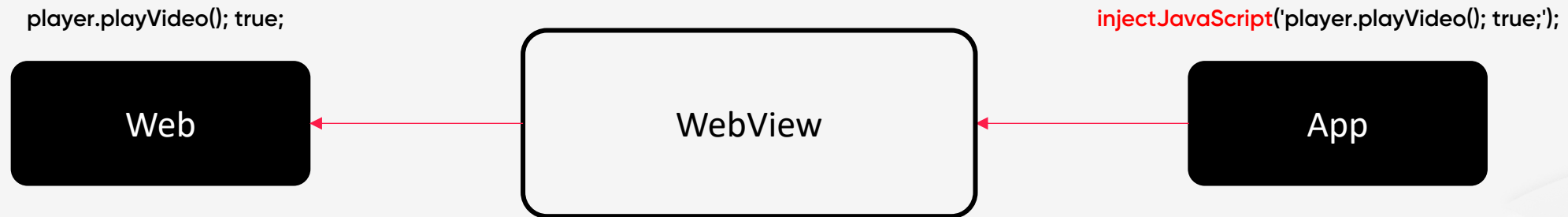
앱과 웹 간에 통신을 할 수 있다

02

웹에서 앱으로 여러 메시지를 보내고 이를 앱에서 처리할 수 있다



앱에서 웹으로 메시지 보내기

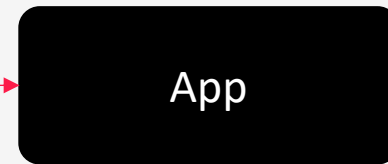


웹에서 앱으로 메시지 보내기

`window.ReactNativeWebView.postMessage(playerState);`



`onMessage(playerState);`

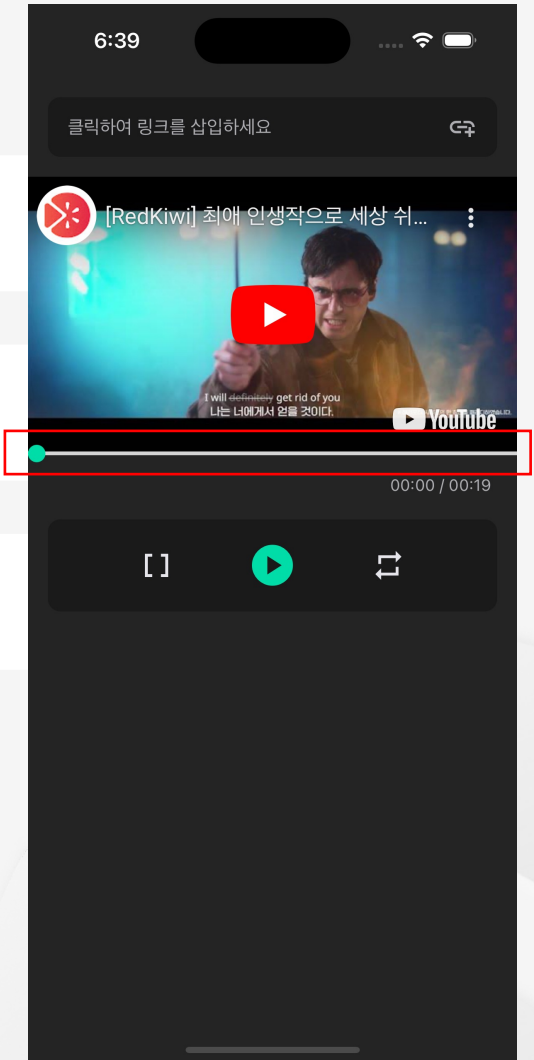


프로젝트2 - 유튜브 영상 구간 반복 학습기 (Expo)

Seek bar 구현하기

학습 목표

- 01 React Native의 Animation을 이용해서 SeekBar에 현재 재생 시점을 표시할 수 있다
- 02 React Native의 PanResponder를 이용해서 SeekBar를 컨트롤 가능하게 만들 수 있다
- 03 SeekBar로 웹과 통신하여 영상을 컨트롤 할 수 있다



PanResponder

- React Native에서 터치와 제스처 이벤트를 쉽게 처리할 수 있도록 도와주는 유틸리티
- 사용자가 화면을 터치하고 이동하는 동작을 감지하고 이에 대한 반응을 정의할 수 있음
- 주요 콜백 함수
 - onStartShouldSetPanResponder
 - onMoveShouldSetPanResponder
 - onPanResponderGrant
 - onPanResponderMove
 - onPanResponderRelease

onStartShouldSetPanResponder

- 사용자가 화면을 터치했을 때 PanResponder가 활성화될지 여부를 결정
- 반환값이 true일 경우, PanResponder는 해당 터치 이벤트를 처리
- 주로 사용자가 화면을 터치하는 순간에 어떤 조건을 체크하여 PanResponder를 활성화할지 결정하는 데 사용

onMoveShouldSetPanResponder

- 사용자가 터치한 상태에서 이동할 때 PanResponder가 활성화될지 여부를 결정
- 반환값이 true일 경우, PanResponder는 해당 이동 이벤트를 처리
- 사용자가 화면을 터치한 채로 이동하는 상황에서 PanResponder가 활성화될지 여부를 결정

onPanResponderGrant

- PanResponder가 활성화되고 제스처가 시작될 때 호출
- 여기서 제스처 시작 시 필요한 초기 설정을 할 수 있음
- 예를 들어, 사용자가 드래그를 시작할 때 필요한 초기화 작업을 수행
- SeekBar에서는 드래그를 시작할 때 영상을 일시 정지 할 수 있음

onPanResponderMove

- 사용자가 터치한 상태에서 이동할 때 호출
- gestureState 객체를 통해 터치 이동 정보를 받아 처리
- 사용자의 이동 경로를 추적하거나 UI 요소를 드래그할 때 사용
- SeekBar에서는 사용자의 이동 위치를 기반으로 새로운 비디오 재생 시간을 계산하고, 이를 SeekBar에 반영

gestureState

stateID: 각 터치 시퀀스에 대한 고유 식별자. 여러 손가락으로 터치하는 멀티터치 제스처를 처리할 때 유용

moveX: 마지막으로 기록된 이동 위치의 x 좌표

moveY: 마지막으로 기록된 이동 위치의 y 좌표

x0: 터치가 시작된 위치의 x 좌표

y0: 터치가 시작된 위치의 y 좌표

dx: 제스처 동안의 총 이동 거리의 x 축 변화량 ($\text{moveX} - \text{x0}$)

dy: 제스처 동안의 총 이동 거리의 y 축 변화량 ($\text{moveY} - \text{y0}$)

vx: x 축의 현재 제스처 이동 속도

vy: y 축의 현재 제스처 이동 속도

numberActiveTouches: 현재 화면에 접촉 중인 손가락의 수. 멀티터치를 감지하는 데 유용.

onPanResponderRelease

- 사용자가 터치를 해제할 때 호출
- 제스처가 끝날 때 필요한 작업을 수행
- 예를 들어, 사용자가 드래그를 마쳤을 때 최종 위치를 설정하는 작업을 수행.
- SeekBar에서는 최종 이동 위치를 기반으로 비디오 재생 시간을 설정하고, 비디오를 재생

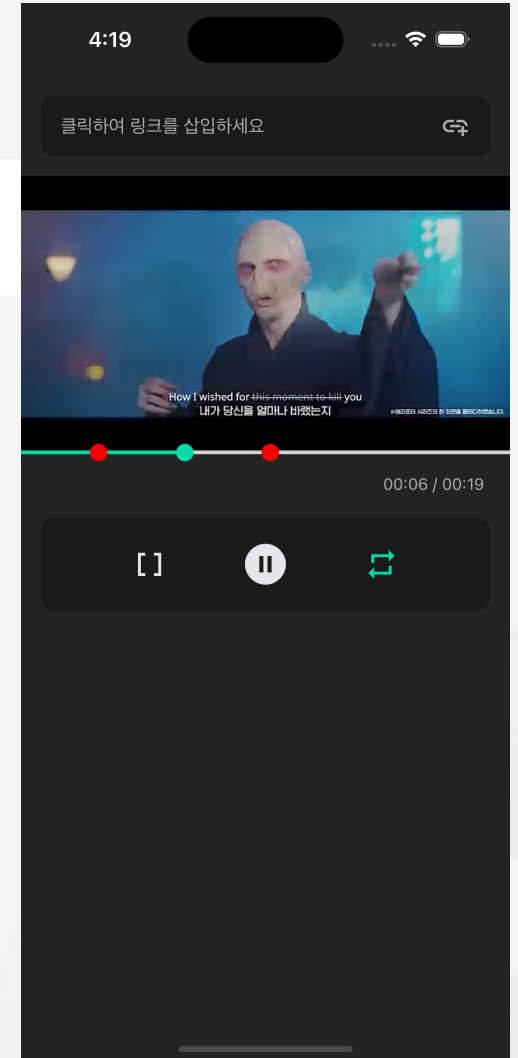
프로젝트2 - 유튜브 영상 구간 반복 학습기 (Expo)

구간 반복 기능 구현하기

학습 목표

01

영상의 구간을 설정하여 반복하는 기능을 구현한다

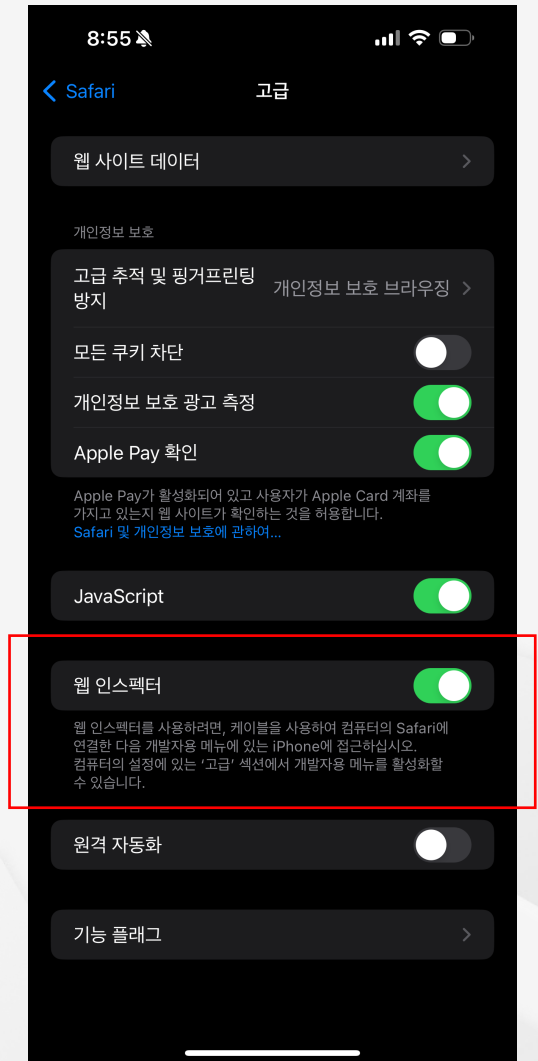


웹뷰 디버깅 하기

- 웹뷰에서 발생하는 에러 또는 콘솔 로그 같은 것들은 메트로 서버에 나타나지 않음
- 웹뷰에서 webViewDebuggingEnabled 활성화
- 컴퓨터와 기기 연결하기
- iOS -> Safari 개발자 도구 사용
- Android -> Chrome 개발자 도구 사용
- <https://github.com/react-native-webview/react-native-webview/blob/master/docs/Debugging.md#debugging-webview-contents>

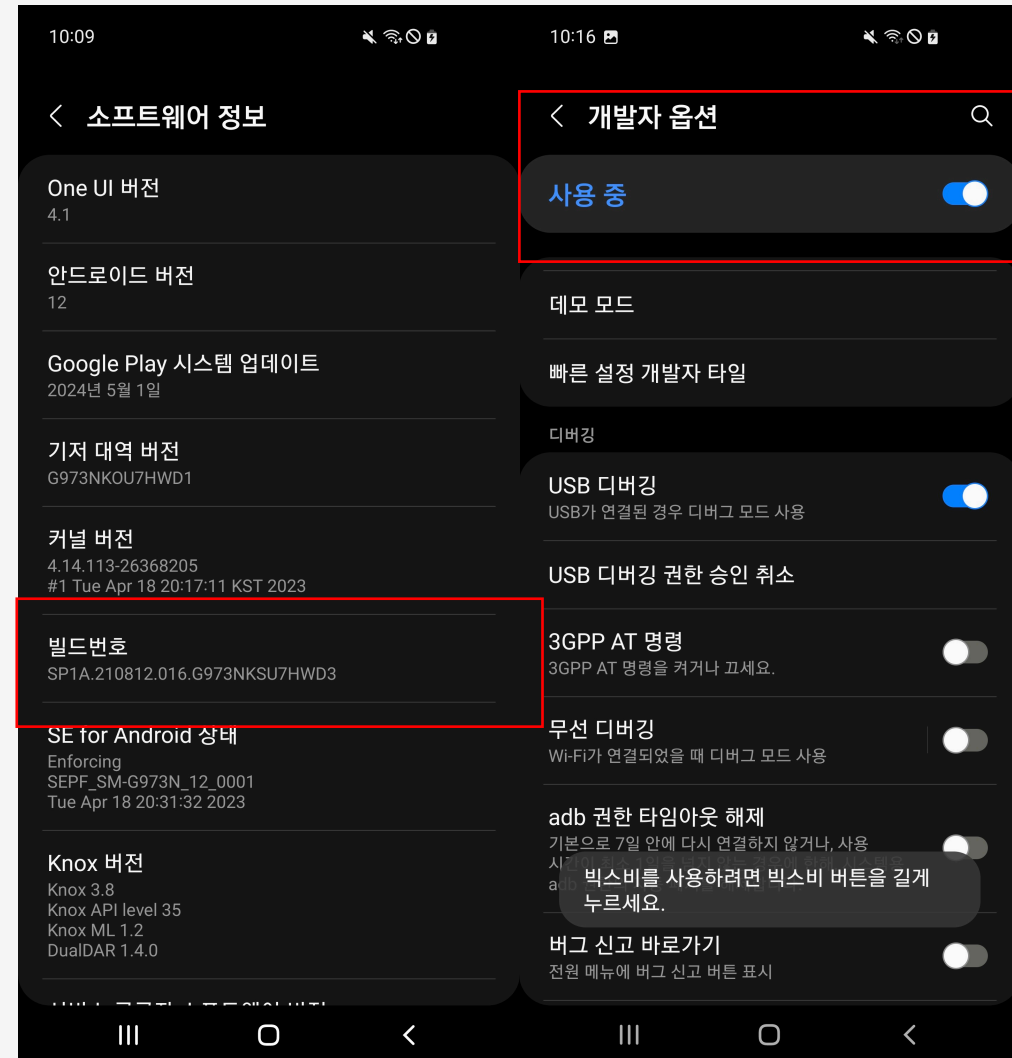
웹뷰 디버깅 하기 - iOS

- 컴퓨터와 아이폰 기기를 USB로 연결하기
- 설정 > Safari > 고급 > 웹 인스펙터 활성화



웹뷰 디버깅 하기 - Android 기기 연결하기

- 컴퓨터와 안드로이드 기기를 USB로 연결하기
- 개발자 옵션 활성화: 설정 > 휴대전화 정보 > 소프트웨어 정보 > 빌드 번호를 계속 탭하기

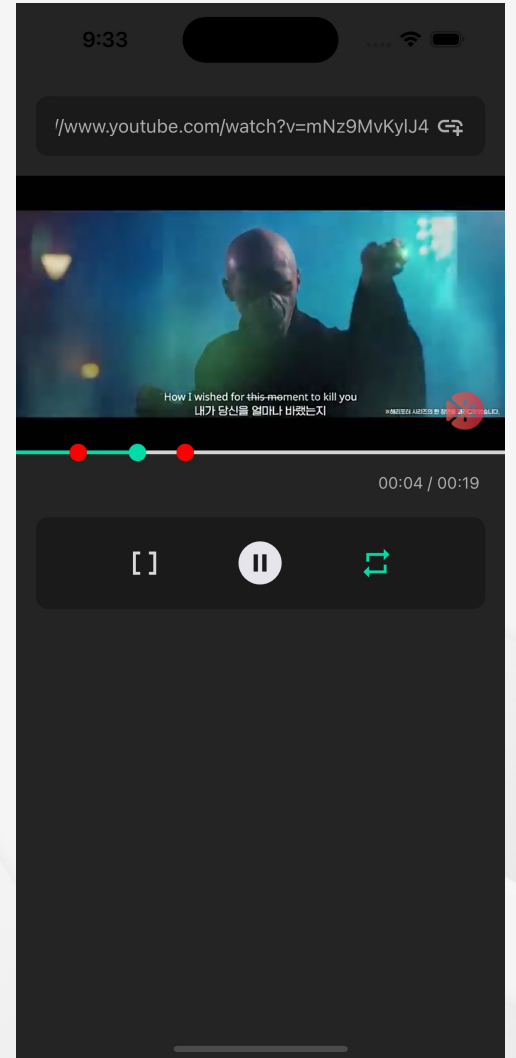


프로젝트2 - 유튜브 영상 구간 반복 학습기

학습 리뷰 및 예상 면접 질문

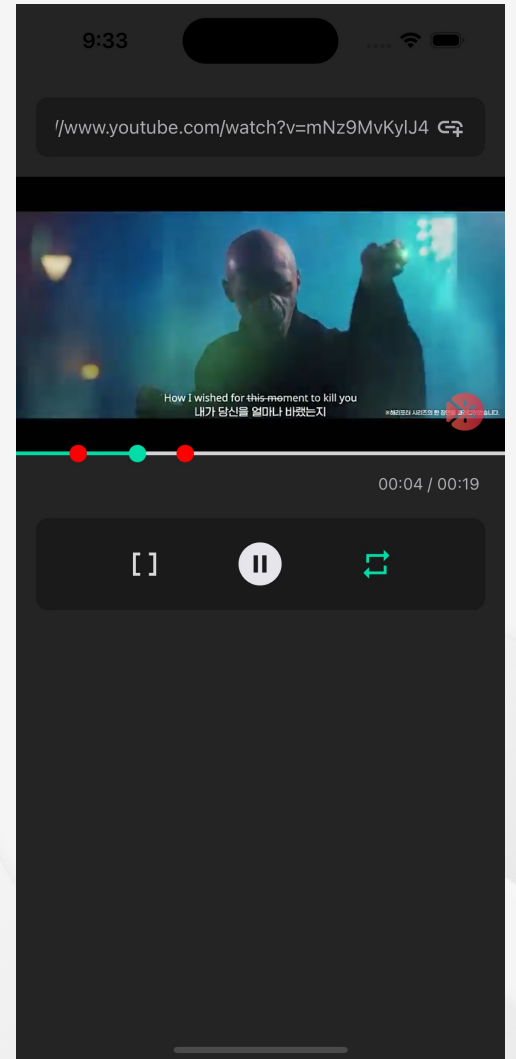
링크 컴포넌트 구현

- TextInput를 이용한 입력 컴포넌트 구현
- query-string 패키지를 이용한 유튜브 id 파싱



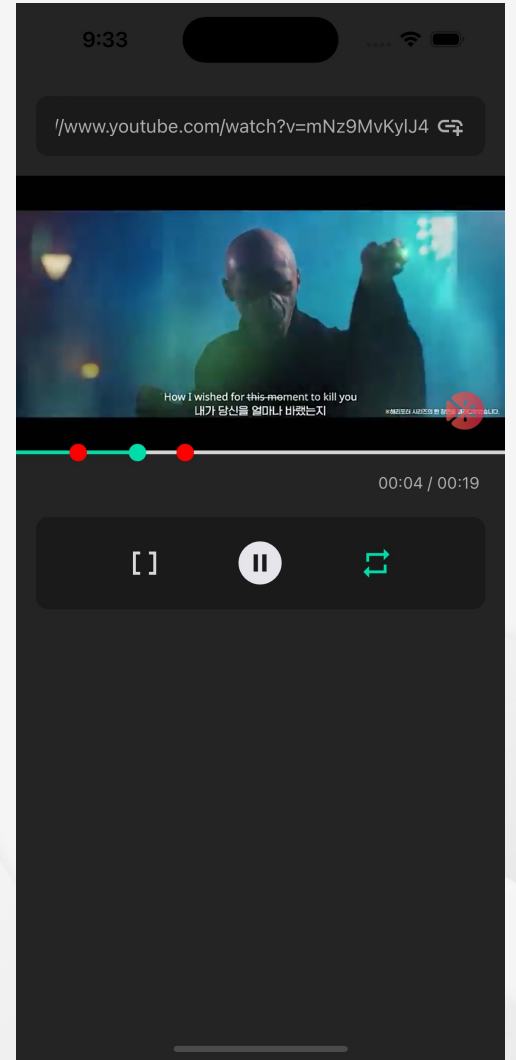
유튜브 영상 로드

- Static HTML을 이용한 유튜브 웹 API 사용
- 뷰포트 크기 설정하기
- 영상 전체 화면 재생 비활성화
 - `allowsInlineMediaPlayback`
- 영상 자동 재생 활성화
 - `mediaPlaybackRequiresUserAction=false`



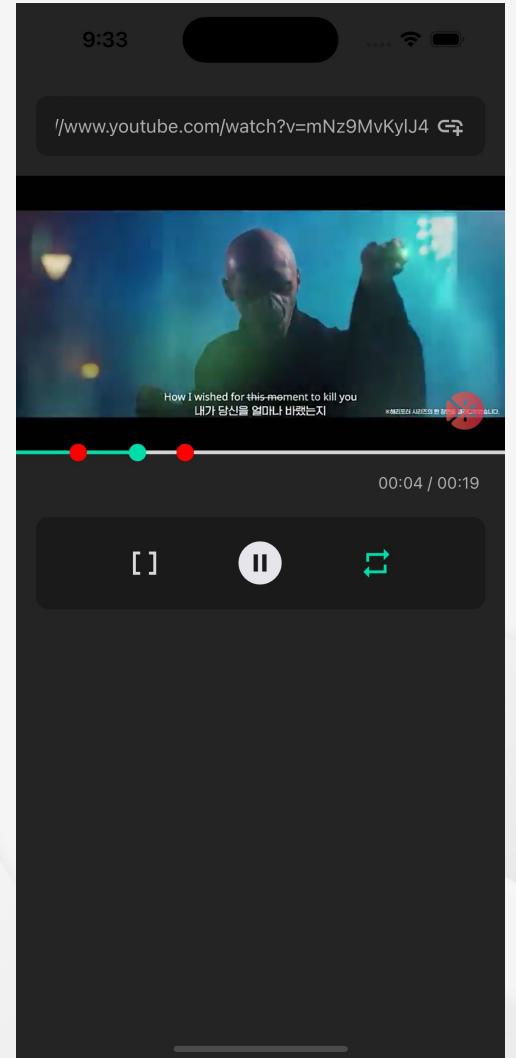
재생 / 멈춤 기능 구현하기

- injectJavascript 함수를 이용해 앱에서 웹으로 통신하기
- window.ReactNativeWebView.postMessage / onMessage를 통해서 웹에서 앱으로 통신하기
- 웹에서 보내는 다양한 메시지 타입 제어하기



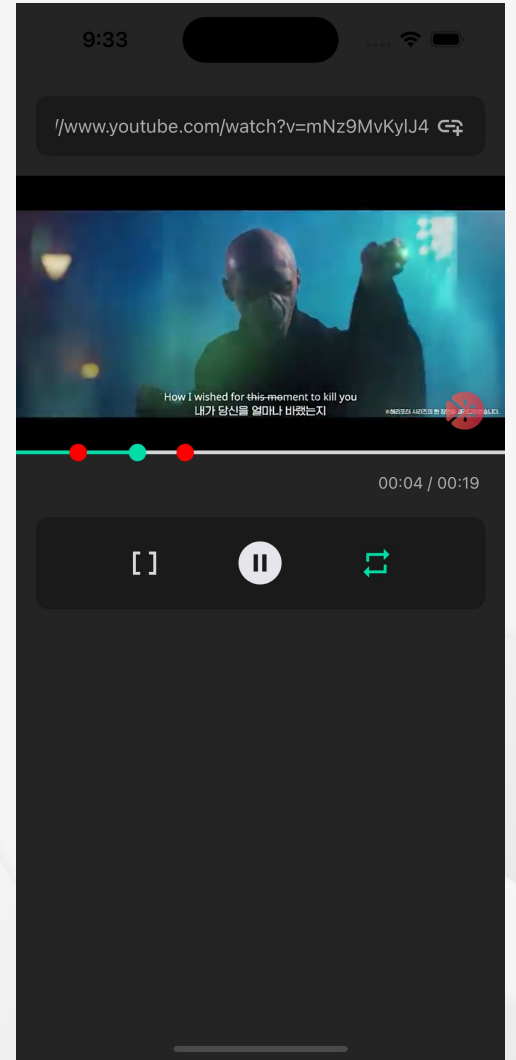
Seek Bar 구현하기

- PanResponder를 이용하여 사용자 제스처 처리
- Animated.timing을 이용하여 부드러운 애니메이션 구현



구간 반복 기능 구현하기

- 앱과 웹이 서로 통신하여 반복 기능 구현하기
- 웹뷰 디버깅 하기



질문1. 앱과 웹 간의 통신을 어떻게 구현했는지 설명해주세요.

- 앱과 웹뷰 간의 통신은 injectJavaScript와 postMessage를 통해 이루어집니다. 앱에서 웹으로 명령어를 실행하기 위해서는 웹뷰의 injectJavaScript를 통해 웹에서 특정 코드를 실행할 수 있습니다. 반대로, 웹에서 앱으로 메시지를 보낼 때는 postMessage를 통해 전송하고, 앱에서는 웹뷰의 onMessage를 통해 해당 메시지를 수신받을 수 있습니다.
- 이때 플레이어 상태 변화나 현재 재생 시간 등 다양한 타입의 메시지를 앱으로 보내게 되는데, 단순히 메시지를 보내면 앱 쪽에서 어떤 타입의 메시지인지 구분하기 어렵습니다. 이를 해결하기 위해, 메시지의 타입과 값을 JSON 문자열로 만들어 전송하고, 앱 쪽에서 이를 파싱하여 어떤 타입의 메시지인지를 구분하게 하였습니다.

질문2. 웹뷰에서 영상을 재생할 때 어떤 최적화 작업을 하셨나요?

- 웹뷰의 뷰포트와 스크린의 사이즈를 일치시켜 영상의 크기가 화면에 잘 맞도록 구현해야 했습니다. 이를 위해 뷰포트 설정을 적절히 조정했습니다. 또한, body에 있는 기본 마진과 패딩을 없애서 웹뷰에 영상만 나오도록 구현했습니다.
- 또한, 영상이 인라인으로 재생되지 않고 전체 화면으로 나오는 문제를 해결하기 위해, 웹뷰의 `allowsInlineMediaPlayback` 속성을 설정했습니다. 이를 통해 영상이 인라인으로 재생되도록 했습니다. 마지막으로, 영상이 자동으로 재생되지 않는 경우에는 `mediaPlaybackRequiresUserAction`을 `false`로 설정하여 사용자의 액션 없이도 영상이 자동으로 재생되도록 했습니다.

질문3. React에서 state를 설정할 때, setState에 값을 직접 지정하는 것과 함수를 이용해서 업데이트하는 것의 차이점?

- 값을 직접 지정하는 방법은 setState에 새로운 값을 직접 넣는 것입니다. 이 방법은 현재 state에 기반하여 새로운 state 값을 계산할 필요가 없을 때 사용됩니다. 예를 들어, 네트워크 요청이 완료되어 데이터를 state에 저장하거나 사용자가 버튼을 눌렀을 때 특정 값을 설정하는 경우가 해당됩니다. 이런 상황에서는 단순히 setState에 새 값을 넣어주면 됩니다.
- 함수를 이용하는 방법은 이전 state 값에 의존하여 새로운 state 값을 계산해야 할 때 사용합니다. 예를 들어, 현재 카운터 값을 1 증가시키는 경우가 있습니다. 이때 setState에 현재 state 변수를 가져와서 +1 하는 로직을 구현하면, useEffect와 같은 의존성 배열에 해당 state가 추가됩니다. 이 경우 useEffect 같은 훅들이 무한으로 호출될 수 있습니다. 따라서, 이전 state 값을 기반으로 새로운 값을 설정해야 할 때는 함수를 사용하는 것이 안전합니다. 이 방법에서는 setState에 함수를 전달하고, 이 함수는 이전 state를 인수로 받아 새로운 state를 반환합니다. 이렇게 하면 의존성 배열에 state를 추가할 필요가 없기 때문에 안전하게 업데이트할 수 있습니다.