Remotion 개념과 활용법 — 학습 노트

> 이번 스터디 영상 작업과 직접 관련 없는 내용도 포함한, **Remotion 입문~실전 정리**입니다.
> "React를 알면 영상을 코드로 만들 수 있다"는 게 핵심. 한 번 익혀두면 자막 영상, 데이터 시각화 영상, 자동화 영상(매주 다른 데이터로 같은 템플릿 렌더) 등에 두루 씁니다.

---

1. Remotion이란?

Remotion은 **React 컴포넌트로 영상을 만드는 프레임워크**입니다. 영상을 "프레임의 연속"으로 보고, **"현재 몇 번째 프레임인지(frame)"에 따라 화면이 어떻게 그려질지**를 React로 정의하면, Remotion이 그걸 프레임마다 캡처해서 MP4로 합쳐줍니다.

- 영상 = 정지 화면(프레임)의 연속. 30fps면 1초 = 30프레임.
- 타임라인을 **마우스 드래그가 아니라 코드(숫자)로** 정의 → 수정·재현·자동화가 쉬움.
- HTML/CSS/SVG/Canvas/이미지/영상/오디오 다 넣을 수 있음 (브라우저가 그릴 수 있는 건 다 됨).

공식 사이트: https://www.remotion.dev/ **문서:** https://www.remotion.dev/docs/ **기본 개념:** https://www.remotion.dev/docs/the-fundamentals

> ⚠️ **라이선스 주의**: Remotion은 오픈소스지만 **회사 규모에 따라 유료 라이선스**가 필요할 수 있습니다. 개인·소규모는 보통 무료. 상업적 사용 전 https://www.remotion.dev/docs/license 확인.

---

2. 설치 & 프로젝트 시작

```
# 새 프로젝트 (템플릿 선택)
npx create-video@latest

# 기존 폴더에 직접 구성 시 핵심 패키지
npm i remotion @remotion/cli react react-dom
```

스크립트 (package.json):

```
{
  "scripts": {
    "start": "remotion studio src/index.ts",          // 미리보기 (브라우저 에디터)
    "render": "remotion render src/index.ts MyComp out/video.mp4"  // 렌더(MP4 출력)
  }
}
```

---

3. 핵심 개념 6가지

① Composition — "영상 한 편"의 정의

영상의 **id / 컴포넌트 / 길이 / fps / 해상도**를 등록합니다.

```
import { Composition } from "remotion";

<Composition
  id="MyVideo"              // 렌더 시 부르는 이름 (a-z, A-Z, 0-9, - 만 가능. _ 불가!)
  component={MyVideo}       // 그릴 React 컴포넌트
  durationInFrames={300}    // 길이 = 10초 (30fps 기준)
  fps={30}
  width={1920}
  height={1080}
/>
```

📄 https://www.remotion.dev/docs/composition

② useCurrentFrame — "지금 몇 프레임?"

모든 애니메이션의 출발점. 이 값에 따라 화면을 다르게 그리면 그게 곧 움직임입니다.

```
import { useCurrentFrame } from "remotion";
const frame = useCurrentFrame();   // 0, 1, 2, ... 매 프레임 증가
```

📄 https://www.remotion.dev/docs/use-current-frame

③ interpolate — "값을 부드럽게 매핑"

"프레임 0~30 동안 투명도를 0→1로" 같은 변환의 핵심 함수.

```
import { interpolate } from "remotion";

const opacity = interpolate(
  frame,            // 입력값 (현재 프레임)
  [0, 30],          // 입력 범위
  [0, 1],           // 출력 범위 → 0~30프레임 동안 0에서 1로
  { extrapolateRight: "clamp" }  // 범위 밖은 고정 (1 유지)
);
```

📄 https://www.remotion.dev/docs/interpolate

④ spring — "통통 튀는 자연스러운 애니메이션"

물리 기반 움직임. 등장 효과에 자주 씁니다.

```
import { spring, useVideoConfig } from "remotion";
const { fps } = useVideoConfig();

const sp = spring({ frame, fps, config: { damping: 12, mass: 0.8, stiffness: 100 } });
const scale = interpolate(sp, [0, 1], [0.9, 1]);  // 0.9 → 1로 통통 등장
```

📄 https://www.remotion.dev/docs/spring

⑤ Sequence — "타임라인에 배치"

"이 컴포넌트를 N번째 프레임부터 M프레임 동안 보여줘". **이번 작업에서 챕터를 시간축에 꽂은 핵심.**

```
import { Sequence } from "remotion";

<Sequence from={150} durationInFrames={300}>  {/* 5초 시점부터 10초간 */}
  <ChapterCard />
</Sequence>
```

- `from`만큼 시간이 흐른 뒤, **그 안의** `useCurrentFrame()` **은 다시 0부터 시작**합니다. (상대 시간) 📄 https://www.remotion.dev/docs/sequence

⑥ AbsoluteFill / Img / Audio / Video / staticFile — 화면 요소

```
import { AbsoluteFill, Img, Audio, OffthreadVideo, staticFile } from "remotion";

<AbsoluteFill style={{ backgroundColor: "#000" }}>   {/* 화면 꽉 채우는 div */}
  <Img src={staticFile("slides/a.png")} />            {/* public/ 폴더의 파일 */}
  <Audio src={staticFile("audio.mp3")} startFrom={0} />
  <OffthreadVideo src={staticFile("clip.mp4")} />     {/* 렌더에 안전한 비디오 */}
</AbsoluteFill>
```

- `staticFile()` = 프로젝트의 **`public/` 폴더** 기준 경로.
- 렌더에서 비디오는 `<Video>`보다 **`<OffthreadVideo>`** 가 안정적. 📄 https://www.remotion.dev/docs/staticfile · https://www.remotion.dev/docs/audio · https://www.remotion.dev/docs/offthreadvideo

---

4. 자주 쓰는 패턴 모음

페이드 인/아웃

```
const opacity = interpolate(
  frame,
  [0, 15, durationInFrames - 15, durationInFrames],
  [0, 1, 1, 0],
  { extrapolateLeft: "clamp", extrapolateRight: "clamp" }
);
```

초 → 프레임 변환 (타임코드 다룰 때 필수)

```
const FPS = 30;
const secToFrames = (sec: number) => Math.round(sec * FPS);
// 예: 2분 41초 시작 → secToFrames(161)
```

이미지 세로 스크롤 (이번 작업 핵심)

```
const progress = frame / (durationInFrames - 1);     // 0 → 1
const translateY = -scrollMax * Math.min(1, progress);
<div style={{ transform: `translateY(${translateY}px)` }}>...</div>
```

외부 데이터(JSON) 불러와 자막/차트 만들기

```
import data from "./utterances.json";    // 빌드에 포함됨
const current = data.find(d => sec >= d.startSec && sec < d.endSec);
```

한글 폰트

```
import { loadFont } from "@remotion/google-fonts/NotoSansKR";
const { fontFamily } = loadFont("normal", { weights: ["400", "700"] });
// 💡 weight를 적게 지정할수록 로딩 빠름 (이번에 4종 전부 불러서 느려졌음)
```

📄 https://www.remotion.dev/docs/google-fonts

---

5. 렌더링 (영상 출력)

```
# 기본 렌더
npx remotion render src/index.ts MyComp out/video.mp4

# 일부 프레임만 (데모/검수용) — 처음엔 짧게!
npx remotion render src/index.ts MyComp out/demo.mp4 --frames=0-899

# 동시 처리 수 조절 (CPU 코어에 맞춰)
npx remotion render ... --concurrency=4

# 로그 레벨
npx remotion render ... --log=warn
```

렌더 팁:

- **검수는 짧은 데모부터.** 긴 영상을 처음부터 풀 렌더하지 말 것.
- **미리보기 Studio와 렌더 동시 실행은 느림.** 렌더 시 Studio 끄기.
- 컴포지션 id는 `a-z A-Z 0-9 -` 만. **언더스코어(`_`) 쓰면 에러.**
- `remotion.config.ts` 로 기본 설정(이미지 포맷, concurrency 등) 지정 가능.

📄 렌더 문서: https://www.remotion.dev/docs/render 📄 CLI: https://www.remotion.dev/docs/cli 📄 설정: https://www.remotion.dev/docs/config

---

6. Remotion Studio (미리보기 에디터)

```
npm start   # = remotion studio
```

- 브라우저에서 열리는 **타임라인 + 미리보기 + props 편집기**.
- **Hot reload**: 코드를 저장하면 즉시 반영 (재렌더 불필요).
- 등록한 모든 Composition을 좌측에서 골라 스크럽하며 검수 가능.

📄 https://www.remotion.dev/docs/studio

---

7. 이번 영상에 적용한 구조 (요약 다이어그램)

```
StudyVideo (Composition, 214500프레임 = 119분)
│
├── <Audio audio.mp3 />                ← 원본 오디오 전체 깔기
├── <Sequence from=0 dur=150>          ← 인트로 5초
│     └── LogoIntro
│
├── CHAPTERS.map( 챕터 → )
│     <Sequence from=초→프레임 dur=구간>
│       └── ChapterRenderer
│             ├── ChapterCard (앞 1.5초)
│             ├── SlideScroll (슬라이드 자동 스크롤)  ── 또는 ── FillerScene (슬라이드 없는 챕터)
│             ├── SpeakerBadge (좌상단 발표자)
│             └── Subtitle (하단 자막, utterances.json 기반, 4줄 제한)
│
└── <Sequence 끝-150> Outro (아웃트로 5초)
```

8. 더 배우기 (추천 링크)

주제	링크
공식 문서 홈	https://www.remotion.dev/docs/
기본 개념 (꼭 읽기)	https://www.remotion.dev/docs/the-fundamentals
애니메이션 만들기	https://www.remotion.dev/docs/animating-properties
interpolate 가이드	https://www.remotion.dev/docs/interpolate
spring 가이드	https://www.remotion.dev/docs/spring
데이터 기반 영상(파라미터화)	https://www.remotion.dev/docs/parameterized-rendering
오디오 다루기	https://www.remotion.dev/docs/using-audio
자막(captions) 패키지	https://www.remotion.dev/docs/captions
트랜지션 효과	https://www.remotion.dev/docs/transitions/
예제 모음	https://www.remotion.dev/showcase
GitHub	https://github.com/remotion-dev/remotion
라이선스 (상업 사용 전 필독)	https://www.remotion.dev/docs/license

9. 한 줄 요약

> **"영상 = 프레임의 함수."** `useCurrentFrame()` 으로 현재 시간을 받아서, `interpolate` / `spring` 으로 화면 값을 바꾸고, `Sequence` 로 타임라인에 배치하면, Remotion이 그걸 프레임마다 그려 MP4로 합쳐준다. 나머지는 평범한 React.