Bideo eta Audioa. Ariketak

Hemengo kode-adibidea jarraituz:
https://codesandbox.io/s/bideo-ariketa-1-bh3cg

1) Controls atributua ezabatu <video> etiketatik. Jarraian 3 botoi prestatu bideoaren azpian agertzeko: Jo, Eten, Kaptura

Jo Eten Kaptura

- Jo botoian sakatzean, bideoa martxan jarriko da
- Eten botoian sakatzean, bideoa eten egingo da. Berrriro martxan jartzeko, "Jo" botoian sakatu
- Kaptura botoian sakatzean, bideoaren uneko fotogramaren kaptura egingo da. Kaptura <canvas> etiketan bistaratu behar da, dimentsioak aldatuz - txikiagoa ikusi behar da bideoa baino, zuk aukeratu tamaina - Ikus beheko irudia adibide gisa:

Argibidea: drawImage() metodoak lehenengo parametroan bideo baten erreferentzia onartzen du.

Jo Eten Kaptura

Soluzioa: https://codesandbox.io/s/bideoariketak-soluzioa-i3kp8

2) Aurreko ariketa moldatu bideoaren kapturak automatikoki egin daitezen, alegia canvas-en bideoaren fotogramak agertuko dira bideoa martxan dagoen bitartean, automatikoki.
Lerro hau lagungarria egingo zaizu:

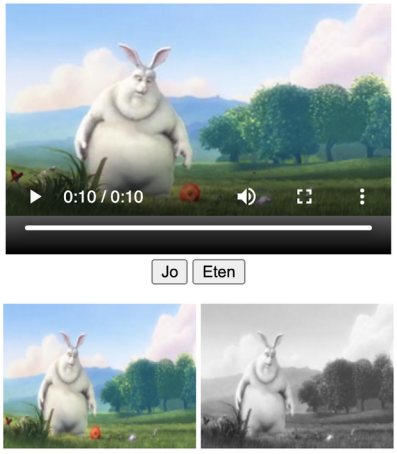
bideoa.addEventListener("play", function () { .... zure kodea ... }

Bideoa amaitzen denean edo bideoa etetean ("pause" egitean), canvas osagaian fotogramak kopiatzeari utzi.

Beste lerro hau lagungarria egingo zaizu baita ere:
bideoa.addEventListener("pause", function () { ... zure kodea hemen ...}

Soluzioa: https://codesandbox.io/s/bideosoluzioa2-jiqyn

3) Hurrengo ariketan <video> eta <canvas> etiketak uztartuko ditugu. Helburua hau lortzea da:



Goiko aldean bideo bat dugu. Bere azpian bi canvas txiki. Ezkerrekoan orain arte egin duguna. Eskubikoan txuri-beltzez margotuko dugun beste canvas bat. Prozesua honakoa da:

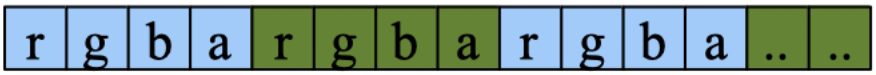
Bideotik frame bat atera eta ezkerreko canvas osagaian (buffer deritzona) utzi:

```
buffer.drawImage(bideoa, 0, 0, 160, 120);
```

Jarraian, bufferetik frame baten informazioa erauzi, array gisa

```
let frame = buffer.getImageData(0, 0, 320, 120);
```

frame array-ak horrelako itxura izango du:



alegia, pixel bakoitzak 4 balio izango ditu: R (red edo gorri kolorea), G (green edo berdea), B (blue edo urdina), A (alpha edo gardentasun maila). Adibidez, pixel guztiz gorria badugu erdi gardena, orduan bere balioa (255, 0, 0, 100) izango da.

Zenbat pixel ditugun jakiteko, zatiketa sinple bat egin dezakegu:

```
let length = frame.data.length / 4;
```

Ondoren, pixel bakoitzeko, bere posizioa eta (r,g,b,a) balioak aterako ditugu eta horiekin txuribeltzez() izeneko funtzio bati deituko diogu, kolorez dagoen pixel bat txuribeltz bihurtzeko:

```
for (let i = 0; i < length; i++) {
      let r = frame.data[i * 4 + 0];
      let g = frame.data[i * 4 + 1];
      let b = frame.data[i * 4 + 2];

      txuribeltzez(i, r, g, b, frame.data);
    }
```

txuribeltzez() funtzioa erraza da, kolore-osagai bakoitzaren balioak batu eta batezbestekoa lortuko du hasieran (grisa kolorea lortzeko). Ondoren pixel horri grisa kolorea esleitzen dio:

```
function txuribeltzez(pos, r, g, b, data) {
  var grisa = (r + g + b) / 3;
  data[pos * 4 + 0] = grisa;
  data[pos * 4 + 1] = grisa;
  data[pos * 4 + 2] = grisa;
}
```

Egitea falta zaigun bakarra txuribeltzez lortu dugun frame array-a canvas-ean margotzea da:

```
txuribeltza.putImageData(frame, 0, 0);
```

Soluzioa hemen ikus dezakezu: https://juananpe.github.io/txuribeltz/

4) Audio ariketa batekin jarraituko dugu. Hiru fitxategiz osatutako array bat dugu:

```
  ['audio1.mp3', 'audio2.mp3', 'audio3.mp3']
```

Programa ezazu script bat hiru audio horiek jotzeko, bata bestearen atzetik jo itzazu. Bukatzean, lehenengoarekin jarraitu, amaigabeko begizta batean.