JS kód magyarázata

🔷 1. Zeneadatok definiálása

```
const songsList = [
  {
    name: "Jazz In Paris",
    artist: "Media Right Productions",
    src: "assets/1.mp3",
    cover: "assets/1.jpg"
  },
  ...
];
```

Magyarázat:

- `const`: konstans változót hoz létre (értéke nem változik).
- `songsList`: egy tömb (listaszerű adattároló), amely objektumokat tartalmaz. Minden objektum egy dalt ír le.
- Az egyes objektumok mezői:
  - `name`: a dal címe.
  - `artist`: az előadó neve.
  - `src`: a zene fájl elérési útja.
  - `cover`: a borítókép elérési útja.

🔷 2. HTML elemek kiválasztása

```
const artistName = document.querySelector('.artist-name');
...
const playBtn = document.getElementById('play');
```

Magyarázat:

- `document.querySelector`: kiválaszt egy HTML elemet a megadott CSS szelektor alapján (pl. osztálynév).
- `document.getElementById`: kiválaszt egy elemet ID alapján.
- Ezeket a változókat arra használjuk, hogy a HTML felületet dinamikusan frissítsük (pl. megjelenítse az éppen játszott dalt).

🔷 3. Zenelejátszás alapállapot

```
let song = new Audio();
let currentSong = 0;
let playing = false;
```

Magyarázat:

- `new Audio()`: létrehoz egy új hangobjektumot (HTML5 `audio`).
- `currentSong`: tárolja, hogy hányadik dal van kiválasztva.
- `playing`: boolean érték, igaz vagy hamis, jelezve, hogy megy-e éppen a zene.

🔷 4. Eseménykezelés a betöltés után

```
document.addEventListener('DOMContentLoaded', () => {
    loadSong(currentSong);
    ...
});
```

Magyarázat:

- `DOMContentLoaded`: esemény, amely akkor történik, amikor a HTML teljesen betöltődött.
- Az itt definiált **függvények** az interakciókat kezelik:
    - `loadSong()`: betölti a dalt.
    - `updateProgress()`: frissíti az időt és a csúszkát.
    - `nextSong()`, `prevSong()`, `togglePlayPause()`: kezelik a gombokat.

- `seek()`: lehetővé teszi az idővonalon való kattintással ugrást.

🔷 5. Dal betöltése

```
function loadSong(index) {
    const { name, artist, src, cover: thumb } = songsList[index];
    ...
}
```

Magyarázat:

- **Objektumbontás (destructuring)**: egyszerre kinyerjük az objektum mezőit.
- **DOM manipuláció**: beállítjuk a zene címét, előadóját, fájlját és borítóját a HTML-ben.

🔷 6. Lejátszás közbeni frissítés

```
function updateProgress() {
    if (song.duration) {
        ...
    }
}
```

Magyarázat:

- Ellenőrzi, hogy mennyi ideje megy a zene, és vizuálisan frissíti a lejátszósávot.
- `song.currentTime`: az aktuális pozíció másodpercben.
- `song.duration`: a dal teljes hossza.
- `formatTime()` segédfüggvény átalakítja másodperceket `perc:mp` formátumra.

🔷 7. Lejátszás / szünet váltás

```
function togglePlayPause() {
    if (playing) {
```

```
        song.pause();
    } else {
        song.play();
    }
    ...
}
```

Magyarázat:

- Ha épp megy a zene, akkor megállítja, különben elindítja.
- A **CSS osztályokat** is váltogatja (`fa-play` , `fa-pause`) a gomb kinézetéhez.

🔷 8. Következő / előző dal kiválasztása

```
function nextSong() {
    currentSong = (currentSong + 1) % songsList.length;
    playMusic();
}
```

Magyarázat:

- Növeli vagy csökkenti a `currentSong` indexét.
- **Moduló (%) operátor**: körbeugrik, ha elérjük a lista végét.
- A `playMusic()` betölti és lejátssza az új dalt.

🔷 9. Idővonal kattintás kezelése

```
function seek(e) {
    const pos = (e.offsetX / prog.clientWidth) * song.duration;
    song.currentTime = pos;
}
```

Magyarázat:

- Az egérkattintás helyét arányosítja a teljes hosszal.
- Ezzel az idővonal bármely részére kattintva „odaugorhatunk” a zenében.

⬅️ END Összegzés

Ez a kód egy teljesen működőképes, kompakt zenelejátszó logikát tartalmaz:

🎵 Jellemzők:

- Dallista
- Lejátszás/szünet
- Következő/előző dal
- Idővonal frissítése és ugrás
- Dinamikus HTML frissítés

🧠 Szintaktikai alapfogalmak:

- `const`, `let`
- tömbök, objektumok
- függvények
- DOM kezelése (`querySelector`, `getElementById`)
- események (`addEventListener`)
- logikai ágak (`if`)
- időkezelés