Music Player

Loops

1. **Find Songs by Artist**
    - **Task:** Write a function that finds all songs by a specific artist in the given array and returns a new array with the matching songs.
    - **Inputs:** An array of song objects, an artist name.
    - **Example:**
        - Input:
          `[{ title: "Song 1", artist: "Artist 1" }, { title: "Song 2", artist: "Artist 2" }, { title: "Song 3", artist: "Artist 1" }]`
          , artist: "Artist 1"
        - Output:
          `[{ title: "Song 1", artist: "Artist 1" }, { title: "Song 3", artist: "Artist 1" }]`
    - **Tip:** Use a loop to iterate over the songs, check if the artist matches the specified artist, and add matching songs to the new array.
2. **Shuffle Songs**
    - **Task:** Write a function that shuffles the order of songs in the given array.
    - **Input:** An array of song objects.
    - **Example:**
        - Input:
          `[{ title: "Song 1" }, { title: "Song 2" }, { title: "Song 3" }]`
        - Output (one possible output):
          `[{ title: "Song 2" }, { title: "Song 3" }, { title: "Song 1" }]`
    - **Tip:** Use a loop to iterate over the array and swap the position of songs randomly using the Fisher-Yates algorithm.
3. **Calculate Average Rating**
    - **Task:** Write a function that calculates the average rating of songs in the given array and returns the result.
    - **Input:** An array of song objects.

- **Example:**
  - Input:
    ```
    [{ title: "Song 1", rating: 8 }, { title: "Song 2", rating: 7 }, { title: "Song 3", rating: 9 }]
    ```
  - Output: `8`
- **Tip:** Use a loop to iterate over the songs, sum up the ratings, and divide the sum by the number of songs to calculate the average.

4. **Matrix Operations: Multiply by Scalar**

- **Task:** Write a function that multiplies each element of a given matrix by a scalar value.
- **Inputs:** A matrix (2D array) and a scalar value.
- **Example:**
  - Input: `[[1, 2], [3, 4]]`, scalar: 2
  - Output: `[[2, 4], [6, 8]]`
- **Tip:** Use nested loops to iterate over the matrix and multiply each element by the scalar value.

5. **Matrix Operations: Transpose**

- **Task:** Write a function that transposes a given matrix, swapping its rows and columns.
- **Input:** A matrix (2D array).
- **Example:**
  - Input: `[[1, 2, 3], [4, 5, 6]]`
  - Output: `[[1, 4], [2, 5], [3, 6]]`
- **Tip:** Use nested loops to iterate over the matrix and construct a new matrix where the rows become columns and vice versa.

6. **Count Songs by Artist**

- **Task:** Write a function that counts the number of songs by each artist in the given array and returns an object with the artist as the key and the count as the value.
- **Input:** An array of song objects.
- **Example:**
  - Input:
    ```
    [{ title: "Song 1", artist: "Artist 1" }, { title: "Song 2", artist: "Artist 2" }, { title: "Song 3", artist: "Artist 1" }]
    ```
  - Output: `{ "Artist 1": 2, "Artist 2": 1 }`

7. **Playlist Duration**

- **Task:** Write a function that calculates the total duration of each playlist. The playlists are represented as an array of objects, each containing a title and an array of songs. Each song is an object with a title and a duration.
- **Inputs:** An array of playlist objects.
- **Example:**
  - Input:

```
[
   {
     title: "Playlist 1",
     songs: [
       { title: "Song 1", duration: 2.5 },
       { title: "Song 2", duration: 3 },
       { title: "Song 3", duration: 4 }
     ]
   },
   {
     title: "Playlist 2",
     songs: [
       { title: "Song 1", duration: 5 },
       { title: "Song 2", duration: 2 },
       { title: "Song 3", duration: 2.5 }
     ]
   }
]
```

  - Output: `["Playlist 1: 9.5 minutes", "Playlist 2: 9.5 minutes"]`
- **Tip:** Use a loop to iterate over the playlists. Inside that loop, use another loop to iterate over the `songs` array of each playlist and sum up their durations. Keep track of the total duration for each playlist. The result should be an array of strings, where each string contains the title of a playlist and its total duration.

Bonus

8. **Remove Duplicate Songs**

- **Task:** Write a function that removes duplicate songs from the given array and returns a new array with unique songs.
- **Input:** An array of song objects.
- **Example:**

- Input:
  `[{ title: "Song 1" }, { title: "Song 2" }, { title: "Song 1" }, { title: "Song 3" }, { title: "Song 2" }]`
- Output:
  `[{ title: "Song 1" }, { title: "Song 2" }, { title: "Song 3" }]`

9. **Calculate Total Playtime**

   - **Task:** Write a function that calculates the total playtime of all songs in the given array and returns the result in minutes.
   - **Input:** An array of song objects.
   - **Example:**
     - Input:
       `[{ title: "Song 1", duration: "3:30" }, { title: "Song 2", duration: "4:15" }, { title: "Song 3", duration: "2:50" }]`
     - Output: `600` (10 minutes)
     - Here is a helper function that convert duration to seconds:

```
function getSongDurationInSeconds(duration) {
    const [minutes, seconds] = duration.split(':');
    return Number(minutes) * 60 + Number(seconds);
}
```