Movie Rating System

Callback functions

Exercise 1:

Create a function named `processMovies`. This function should accept two parameters: an array of movie objects and a callback function.

The structure of a movie object in the array should be like this:

```
{
  id: 5,
  title: "Movie title",
  genres: ["Genre1", "Genre2"],
  ratings: [8, 7, 10]
}
```

Your `processMovies` function should perform the following tasks:

1. Check if the first argument is an array. If not, throw an error with the message 'Expected an array of movies'.
2. For each object in the array, check if it contains the properties `id`, `title`, `genres`, and `ratings`. If any object does not contain these properties, throw an error with the message `Movie with ID ${movie.id} is not in the right structure` (the id property will always be).
3. If the object structure is correct, clone the array to avoid mutating the original one: `const newMovies = JSON.parse(JSON.stringify(movies));`.
4. Loop over each object in the cloned array, and pass each object to the callback function.
5. Return the new array.

Exercise 2:

Create another function named `processMovie`. This function should accept two parameters: a movie object and a callback function.

The structure of a movie object should be like the one described in Exercise 1.

Your `processMovie` function should perform the following tasks:

1. Check if the first argument is an object. If not, throw an error with the message 'Expected a movie object'.
2. Check if the object contains the properties `id`, `title`, `genres`, and `ratings`. If it does not, throw an error with the message 'Movie is not in the right structure'.
3. If the object structure is correct, clone the object to avoid mutating the original one.
4. Apply the callback function to the cloned object.
5. Return the resulting object after applying the callback function.

For each of these functions, remember to pass a callback function that will be applied to each object in the respective exercise.

For instance, you could use the `processMovies` function with a callback that counts the number of ratings for each movie and adds it as a new property to each movie object in the array.

```
// Test for processMovie function
const movieToProcess = [
  {
    id: 1,
    title: "Movie 1",
    genres: ["Action", "Drama"],
    ratings: [9, 7, 10]
  },
  {
    id: 2,
    title: "Movie 2",
    genres: ["Horror", "Thriller"],
    ratings: [5, 6, 7]
  },
];

const updatedMovie = processMovies(moviesToProcess, movie => {
  // Add a property of a count of the number of ratings for each movie
  movie.ratingCount = movie.ratings.length;

  return movie;
});

console.log(updatedMovie);
```

The `processMovie` function will return a new array with movie objects with a `ratingCount` property, without modifying the original `moviesToProcess` array.