

## Copilot

Thank you for providing additional details! I see the issue now. The problem lies in the asynchronous nature of the fetch API and the way you're handling the response.

Here's what's happening in your code:

1. In the `httpService` class, the `getAllMovies` method initiates a network request using `fetch`. However, the actual data retrieval happens asynchronously. The `fetch` call returns a promise, which resolves with the response once it's available.
2. Inside the `.then()` block, you're handling the response by calling `response.json()`. Again, this method returns a promise that resolves with the parsed JSON data. However, you're not doing anything with the movies data inside this `.then()` block.
3. Since the data retrieval is asynchronous, the `movies` variable in your `app.js` is not immediately populated with the movie data. As a result, when you try to iterate over `movies` using `forEach`, it's still an empty array (or undefined).

To fix this, you need to ensure that the `movies` array is populated before attempting to iterate over it. Here's how you can modify your code:

```
// httpService.js
async getAllMovies() {
  try {
    const response = await fetch(this.urlBase);
    if (!response.ok) {
      throw new Error('Network response was not ok');
    }
    const movies = await response.json();
    return movies; // Return the movies array
  } catch (error) {
    console.error('Error:', error);
    return []; // Return an empty array in case of an error
  }
}

// app.js
(async () => {
  try {
    const httpService = new HttpService(); // Assuming you've instantiated the service
    const movies = await httpService.getAllMovies();
    if (Array.isArray(movies)) {
      movies.forEach(movie => {
        console.log(movie);
      });
    } else {
      console.error('Movies data is not an array.');
```

Here's what changed:

1. In `httpService.js`, I've made the `getAllMovies` method `async` and used `await` to handle the asynchronous operations. It now returns the `movies` array.
2. In `app.js`, I've wrapped the code in an immediately invoked `async` function to use `await` for fetching the movies. If the `movies` data is not an array (e.g., due to an error), it logs an error message.

Make sure to adjust the code according to your actual implementation, but this should give you a better understanding of how to handle asynchronous data retrieval in JavaScript. 🎬🏠