Movie Star by Taha Efe

General Overview

The "Movie Star" web page is an interactive and visually appealing platform designed to showcase a diverse range of movies and series. It features a modern and user-friendly interface with a dynamic layout including a video trailer header, a responsive navigation bar, and various content sections with a functional search bar. The web page integrates multimedia elements like video and images seamlessly with textual content, providing an engaging user experience. The design focuses on user interactivity, with features like search functionality, content filtering, and dynamic content loading, ensuring a smooth and responsive experience for viewers.

The web page's aesthetic is clean and modern, utilizing a color scheme that highlights the visual elements of the movies and series showcased. It's evident that the design is carefully crafted to cater to a wide audience, appealing to movie enthusiasts and casual viewers alike. The layout is intuitive, allowing easy navigation through different sections of the site. The integration of APIs to fetch and display movie data demonstrates a sophisticated use of modern web development techniques, making the website not only visually appealing but also functionally robust.

The "Movie Star" project is accessible online through two primary links. The first is the live link, which directs users to the actual webpage, showcasing the project in action. This live version represents the culmination of the design and development efforts, allowing users to interact with the website's features in real-time.

Live Link

The second link is to the project's repository on GitHub. This repository provides a comprehensive view of the project's codebase, including all the files that make up the website.

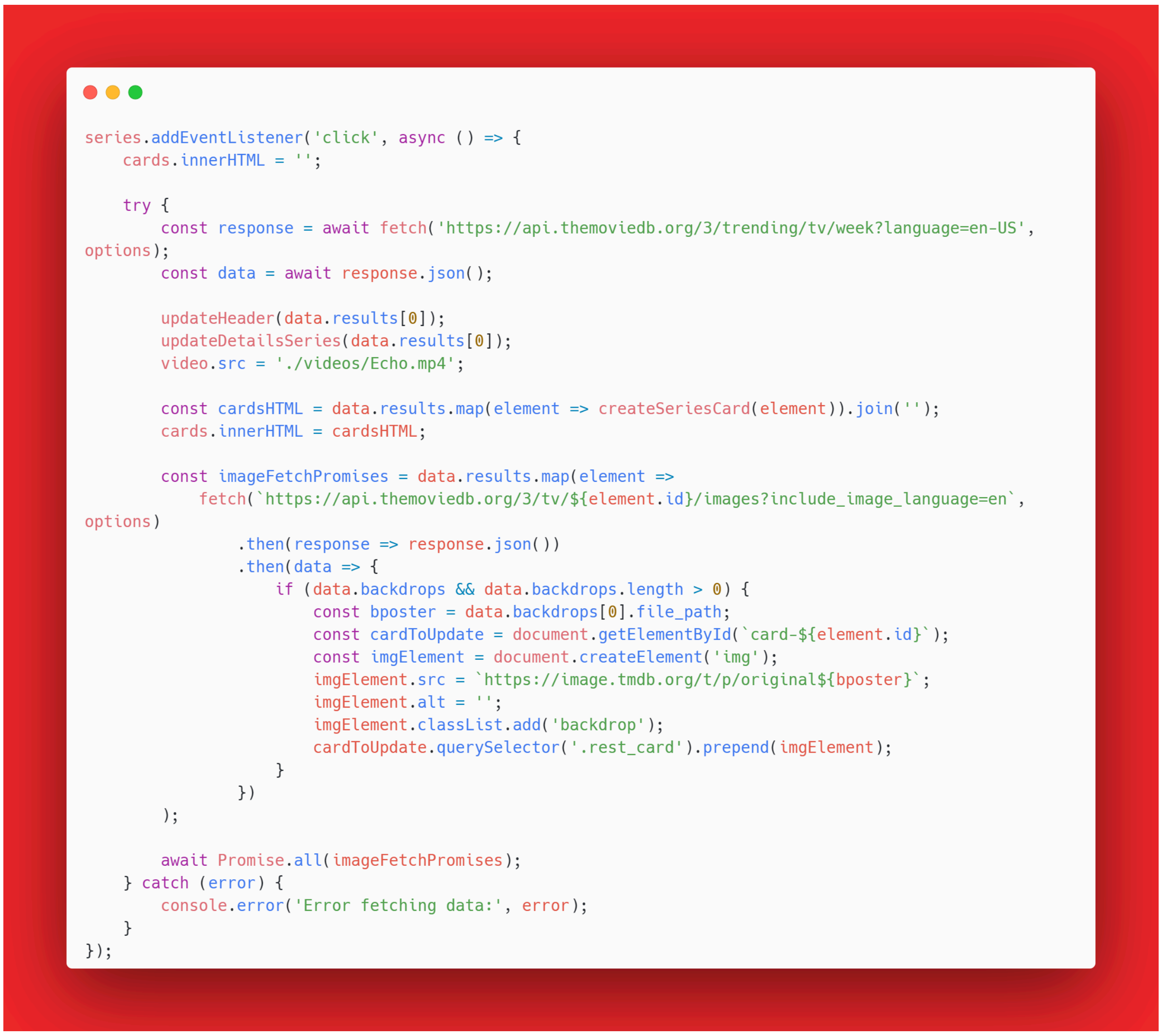
GitHub Repository Link

20202022016
TAHA EFE GÜMÜŞ

Code Snippets of Project

In this section, we delve into some of the most pivotal code snippets from the project. These snippets are instrumental in bringing the webpage to life, enabling dynamic interactions and functionality that are key to the user experience.

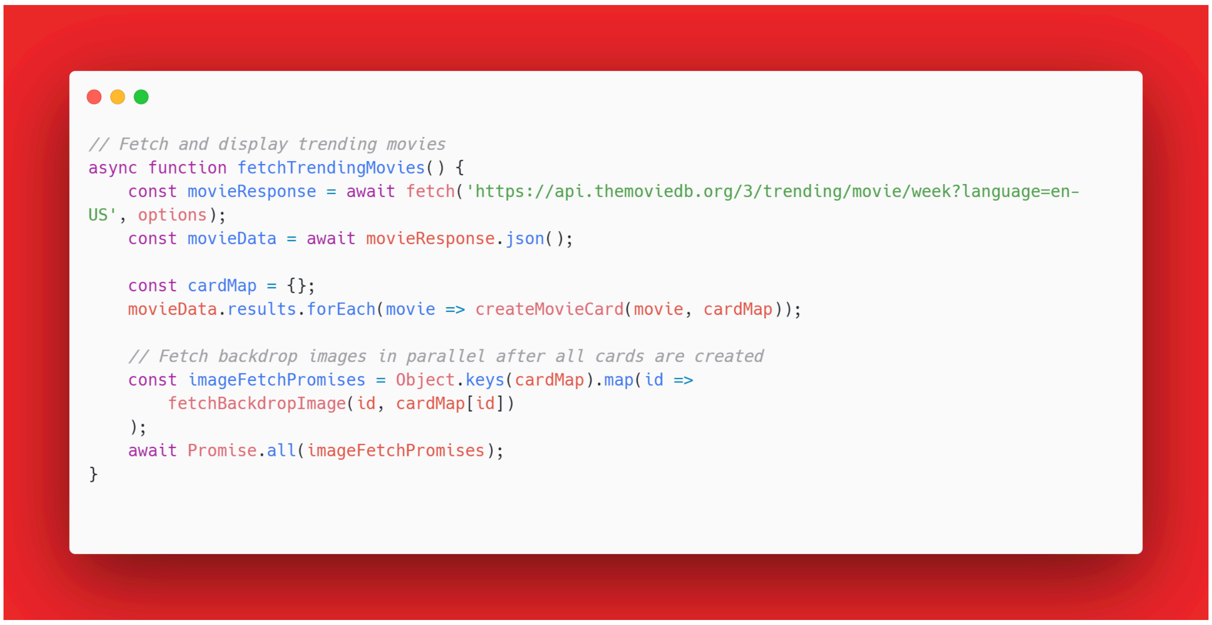
- Category Event Listeners



```javascript
series.addEventListener('click', async () => {
    cards.innerHTML = '';

    try {
        const response = await fetch('https://api.themoviedb.org/3/trending/tv/week?language=en-US',
options);
        const data = await response.json();

        updateHeader(data.results[0]);
        updateDetailsSeries(data.results[0]);
        video.src = './videos/Echo.mp4';

        const cardsHTML = data.results.map(element => createSeriesCard(element)).join('');
        cards.innerHTML = cardsHTML;

        const imageFetchPromises = data.results.map(element =>
            fetch(`https://api.themoviedb.org/3/tv/${element.id}/images?include_image_language=en`,
options)
                .then(response => response.json())
                .then(data => {
                    if (data.backdrops && data.backdrops.length > 0) {
                        const bposter = data.backdrops[0].file_path;
                        const cardToUpdate = document.getElementById(`card-${element.id}`);
                        const imgElement = document.createElement('img');
                        imgElement.src = `https://image.tmdb.org/t/p/original${bposter}`;
                        imgElement.alt = '';
                        imgElement.classList.add('backdrop');
                        cardToUpdate.querySelector('.rest_card').prepend(imgElement);
                    }
                })
        );

        await Promise.all(imageFetchPromises);
    } catch (error) {
        console.error('Error fetching data:', error);
    }
});
```

Explanation: The event listener in the project is a crucial JavaScript component that activates upon a click event on the series section. It initially clears existing content to make space for new series data. The script then asynchronously fetches trending series from an API, updating the webpage with new series information. It dynamically generates series cards, fetches additional images for each series, and updates the UI accordingly. This complex sequence of actions not only ensures up-to-date content display but also enhances user interaction by dynamically altering the webpage's content in response to user actions, demonstrating advanced JavaScript usage for responsive web design. Also, it has the same usage for the kids category.

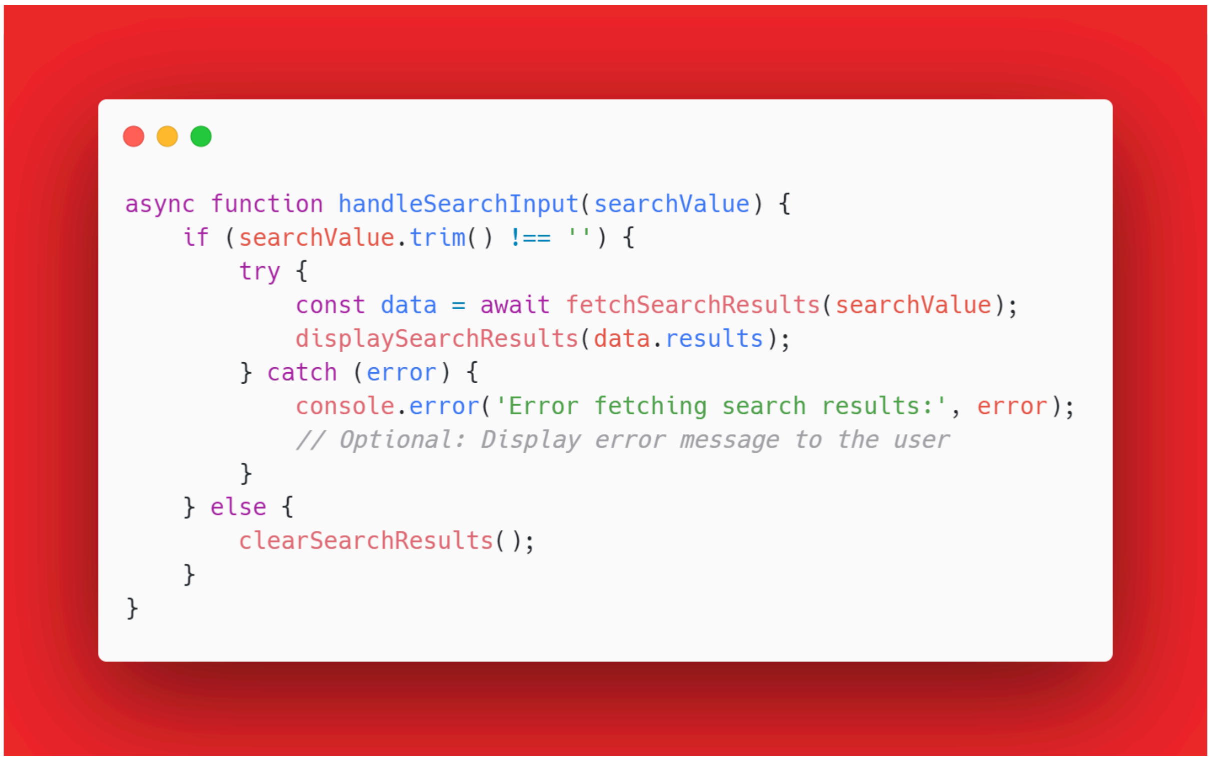
- Fetching and Displaying Trending Movies



```javascript
// Fetch and display trending movies
async function fetchTrendingMovies() {
    const movieResponse = await fetch('https://api.themoviedb.org/3/trending/movie/week?language=en-
US', options);
    const movieData = await movieResponse.json();

    const cardMap = {};
    movieData.results.forEach(movie => createMovieCard(movie, cardMap));

    // Fetch backdrop images in parallel after all cards are created
    const imageFetchPromises = Object.keys(cardMap).map(id =>
        fetchBackdropImage(id, cardMap[id])
    );
    await Promise.all(imageFetchPromises);
}
```

Explanation: The function fetchTrendingMovies uses asynchronous JavaScript to request trending movie data from an API. Upon receiving the data, it iterates through the results and calls createMovieCard for each movie to dynamically generate content on the page.

- Movie Search Functionality



```javascript
async function handleSearchInput(searchValue) {
    if (searchValue.trim() !== '') {
        try {
            const data = await fetchSearchResults(searchValue);
            displaySearchResults(data.results);
        } catch (error) {
            console.error('Error fetching search results:', error);
            // Optional: Display error message to the user
        }
    } else {
        clearSearchResults();
    }
}
```

Explanation: The function is triggered when the user inputs a search query. It checks if the input is not empty and then calls fetchSearchResults to get the relevant movie data from the API. Based on the results, it either displays the search results.

20202022016
TAHA EFE GÜMÜŞ