Channel Islands
CALIFORNIA STATE UNIVERSITY

Movie Picker

Shane McDonough • Dr. Brian Thoms • COMP 499

Introduction

The purpose of this web application is to help you find a movie or TV show when you are unsure what to watch. It provides the title, the release date, the TMDB user rating, and the genres it is a part of. Additionally, it provides a description of the movie or TV show and shows which services can be used to view the movie or TV show. The same movie or TV show will not be shown to the user again after it is shown once until the webpage is refreshed. Also, on the left side of the screen, the user has the ability to filter the movie or TV show in multiple categories. The user has the ability to choose whether they'd rather see a movie or a TV show in addition to having the ability to filter by genre, provider, monetization type, rating, and region. Filtering by genre allows the user to include or exclude genres in the media. Filtering by provider allows the user to include platforms the media must be available on. Filtering by monetization type allows the user to decide whether they are willing to rent or buy a movie or TV show or just see if it's available on a streaming service they already have. Filtering by rating allows the user to select a minimum rating and a maximum rating. and finally filtering by watch region allows the user to show only media available in a certain region. Any and all of these filters can be combined with each other to limit the media to only what the user wants to see. Finally, a user can click a "more like this" button on any movie or TV show which sets the genre filters to include all of the genres in used in the current media.

Methodology

I based my programming strategy off of the agile methodology used in COMP 350 software engineering. I created a GitHub Project board of user stories that contained new features to add, bugs to fix, or general features to implement. All of these stories start in the "todo" section of the board. Once I begin working on the user story, I move them to the "in progress" column of the board and create an issue connected to the GitHub repository. Then once I complete my changes, I commit them and use a closing keyword in the message to close the issue which automatically moves it into the "done" column. This also links the code changes to the completed story. Throughout this processes I very infrequently had more than one user story in progress at a time, so I did not need to create branches for each user story because I completed this project alone.

To start out I created a static webpage using HTML and CSS. I then used JavaScript to call on the REST API for TMDB. I use this API get movie and TV show information and use search parameters in order to refine the search. The API gives an array of twenty movies or TV shows, and a random one is selected and shown to the user. Once a movie or TV show is shown to the user its ID is stored in a set and excluded from following searches. If the array contains nothing that hasn't already been displayed, a page count is incremented and the pool of the next twenty results are selected. This ensures that a user will only see a piece of media once instead of encountering it repeatedly.

Once a movie has been selected, I use jQuery to manipulate the DOM and dynamically add information about the movie or TV show. This information includes the title, the poster, the description, the rating, the release date, the providers where it is available, the genres corresponding to the media, the list of all genres, the list of all providers, and the list of all regions.

Languages, Frameworks, and Techniques

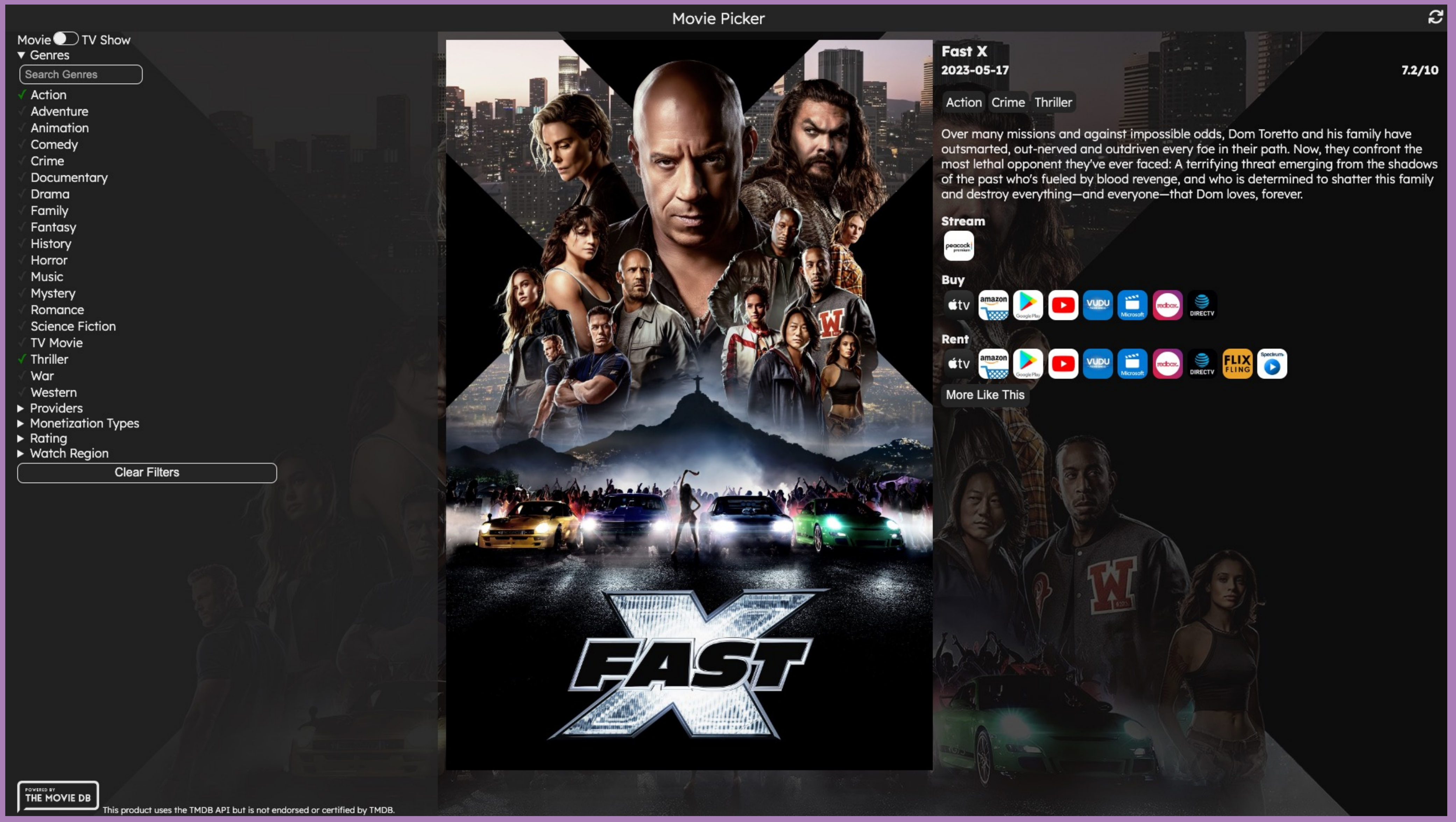
- HTML
- CSS
- JavaScript
- jQuery
- Git
- GitHub
- GitHub Projects
- Dom manipulation
- TMDB REST API
- Functional method chaining
- String interpolation
- Asynchronous code

Links

Project
https://shane.cikeys.com/MoviePicker/

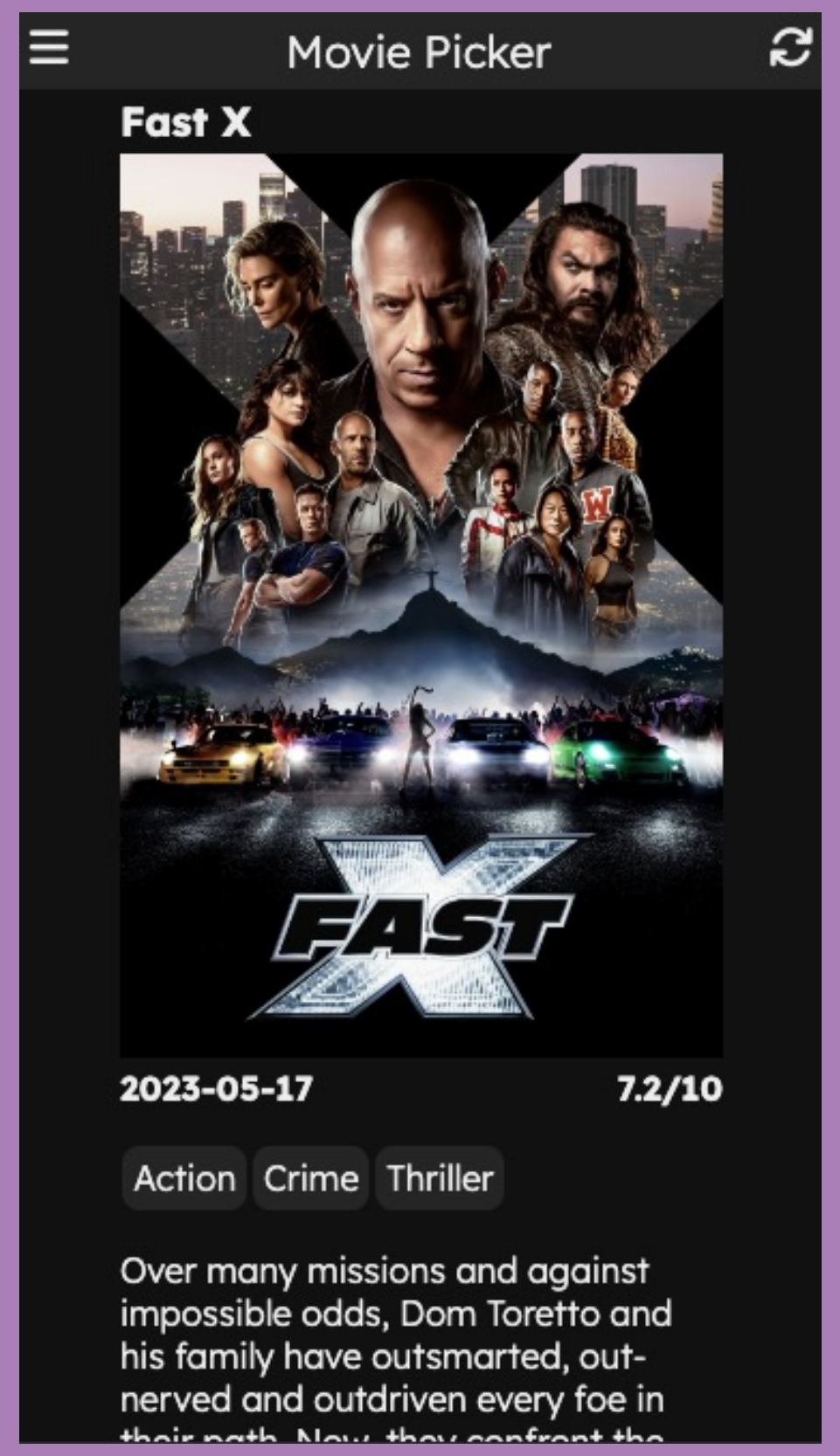
Blog
https://shane.cikeys.com/blog/

Screenshots

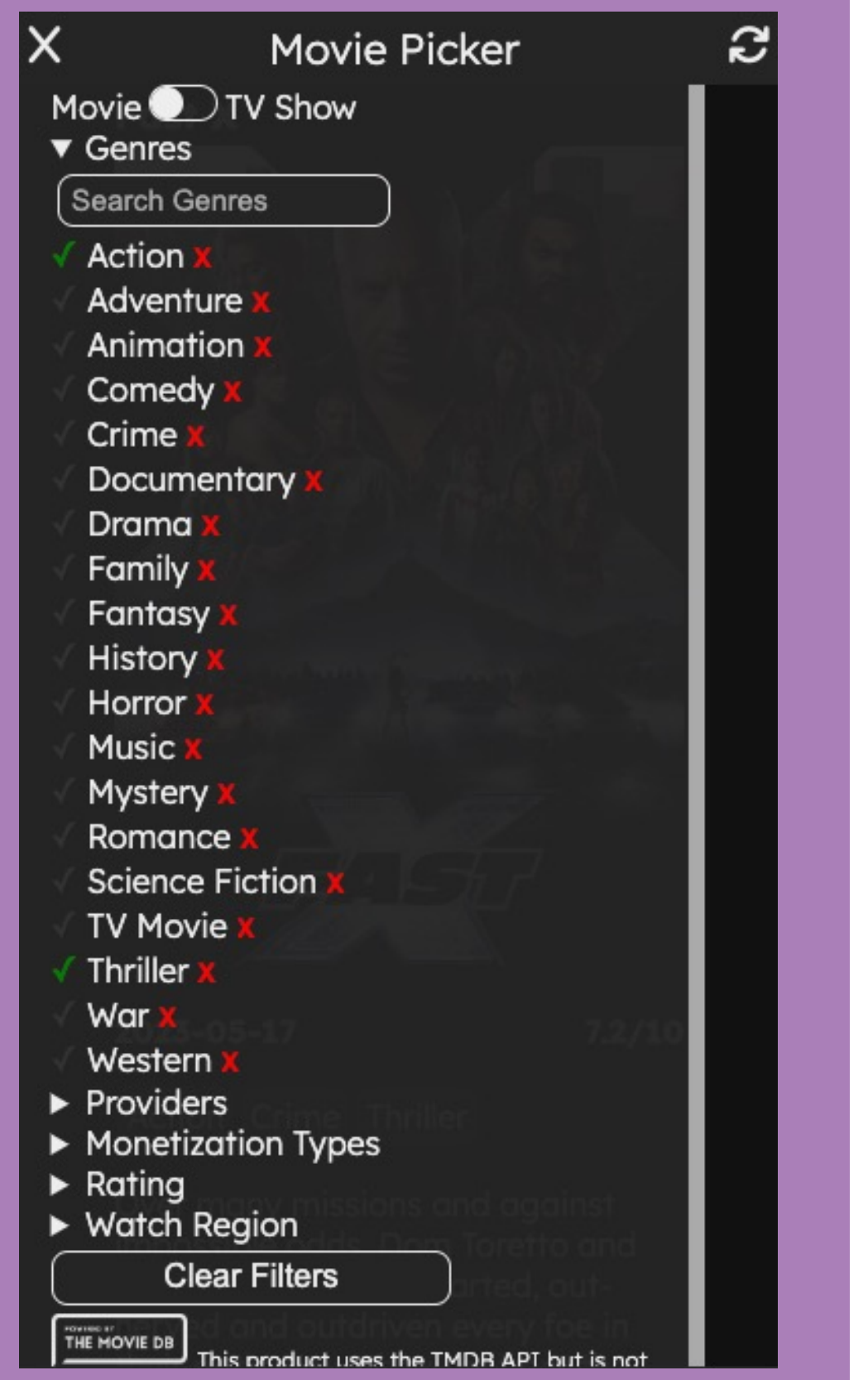


Fast X
2023-05-17
7.2/10
Action Crime Thriller

Over many missions and against impossible odds, Dom Toretto and his family have outsmarted, out-nerved and outdriven every foe in their path. Now, they confront the most lethal opponent they've ever faced: A terrifying threat emerging from the shadows of the past who's fueled by blood revenge, and who is determined to shatter this family and destroy everything—and everyone—that Dom loves, forever.



Over many missions and against impossible odds, Dom Toretto and his family have outsmarted, out-nerved and outdriven every foe in their path. Now, they confront the



Code

```javascript
$('.selector-search').on('input', function() {
    const el = $(this);
    const value = el.val().toLowerCase();
    const lis = $('li', el.parent());
    if(value === '') {
        lis.show();
        return;
    }
    lis.each(function() {
        const name = $('label', this).html().toLowerCase();
        if(name.includes(value)) {
            $(this).show();
        } else {
            $(this).hide();
        }
    });
});
```

This code handles the filtering search for genres and providers.

```javascript
genres.forEach(item => {
    $('#genre-selector').append(
        `<li>
            <input type="checkbox" value="${item.id}" id="genre-${item.id}" ${selectedGenres.has(item.id.toString()) ? 'checked' : ''}>
            <label for="genre-${item.id}" ${excludedGenres.has(item.id.toString()) ? 'style="text-decoration: line-through;"' : ''}>${item.name}</label>
            <input
                type="button"
                value="X"
                data-genre-id='${item.id}'
                title="Exclude ${item.name} Genre"
                class="borderless-button exclude-button"
            >
        </li>`
    );
});
```

This code dynamically adds list items to the unordered list of genres that the user can filter by. jQuery is used for the DOM manipulation but is possible without it as well

```javascript
async function getMedia(args, baseURL) {
    let url = new URL(baseURL);
    for(const arg in args) {
        if(!args[arg]) continue;
        url.searchParams.set(arg, args[arg]);
    }
    url = url.toString();
    try {
        const res = await fetch(url, tmdbOptions);
        return await res.json();
    } catch (e) {
        return null;
    }
}
```

This handles creating a URL and using the fetch function to query the TMDB api. The baseURL passed in is either the movie endpoint or the TV endpoint from the TMDB api.

```javascript
const args = {
    'watch_region': $('#regions').val(),
    'certification_country': 'US',
    'certification.gte': 'G',
    'certification.lte': 'PG-13',
    'with_genres': [...selectedGenres].join(','),
    'without_genres': [...excludedGenres].join(','),
    'with_watch_providers': [...selectedProviders].join('|'),
    'with_watch_monetization_types': [...selectedMonetizationTypes].join('|'),
    'vote_average.gte': $('#min-slider').val(),
    'vote_average.lte': $('#max-slider').val(),
    'page': currentPage
};
```

These are the arguments passed into the TMDB REST API. Some values are constant but most values are pulled directly from the DOM or from variables with values changed from DOM event listeners.

```javascript
const selectedGenres    = mediaTypeIsMovie ? selectedMovieGenres    : selectedTvGenres;
const excludedGenres    = mediaTypeIsMovie ? excludedMovieGenres    : excludedTvGenres;
const selectedProviders = mediaTypeIsMovie ? selectedMovieProviders : selectedTvProviders;
const previousFilters   = mediaTypeIsMovie ? previousMovieFilters   : previousTvFilters;
const seenMedia         = mediaTypeIsMovie ? seenMovies             : seenTvs
const getMediaProviders = mediaTypeIsMovie ? getMovieProviders      : getTvProviders;
const getGenre          = mediaTypeIsMovie ? getMovieGenre          : getTvGenre;
$('#error-info').hide();
$('#movie-info').show();
seenMedia.add(media.id);
$('#movie-poster').attr('src', makeImageURL(media.poster_path));
$('.movie-title').text(media[mediaTypeIsMovie ? 'title' : 'name']);
$('#movie-date').text(media[mediaTypeIsMovie ? 'release_date' : 'first_air_date']);
$('#movie-rating').text(`${parseFloat(media.vote_average).toFixed(1)}/${defaults.rating.max}`);
$('#movie-desc').text(media.overview);
```

This code snippet shows the ternaries for the different functions used for the two different types of media, movies and TV shows. It also shows how values are added int existing elements in order to display the information for the movie or TV show.