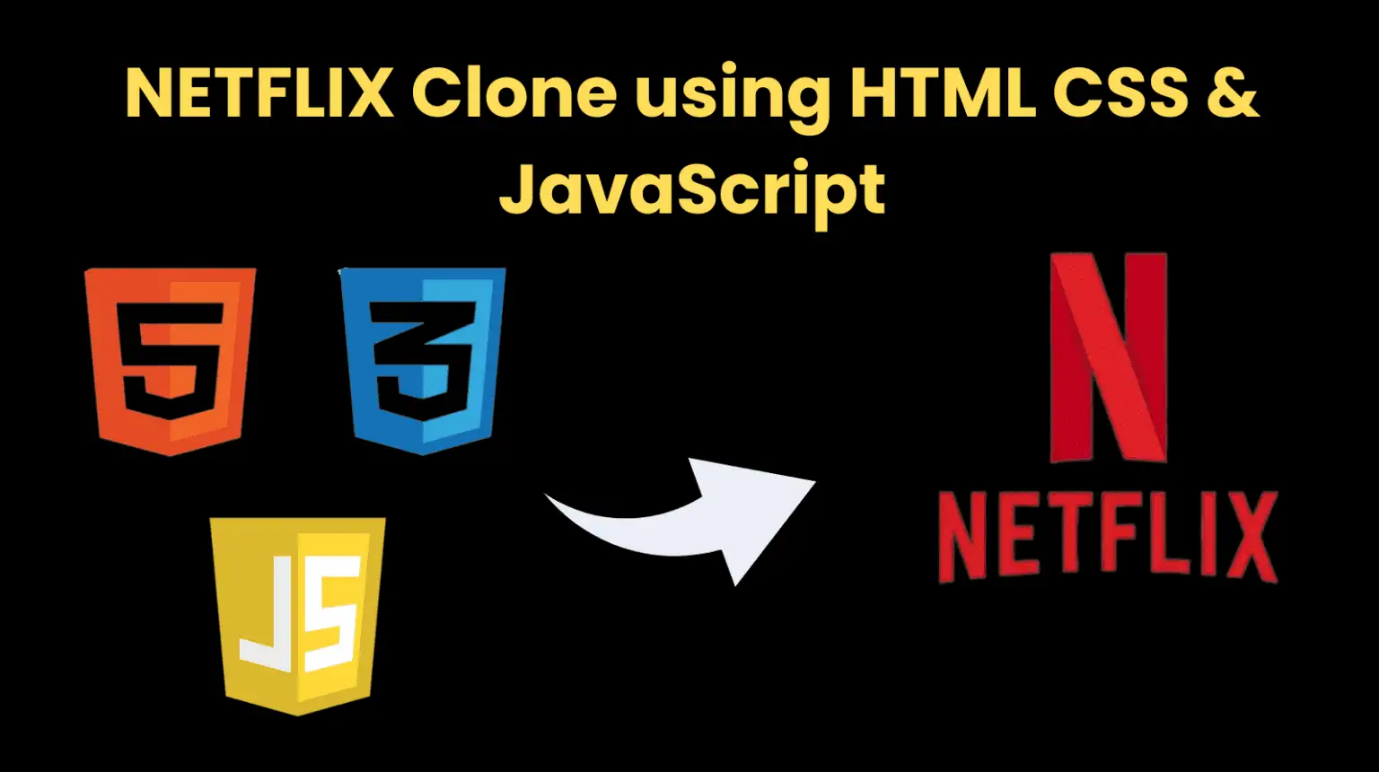
**NETFLIX Clone using HTML CSS & JavaScript**

**Introduction :**

The **Netflix clone** project is a web application developed using**HTML, CSS, and JavaScript**, aiming to replicate the user interface and some features of the popular streaming service, **Netflix**. Let’s break down the project into its key components and functionalities. This project is implemented using a combination of three fundamental web technologies: HTML (Hypertext Markup Language), CSS (Cascading Style Sheets), and JavaScript. Each of these technologies plays a crucial role in defining the structure, styling, and interactivity of the web application. To enhance user interaction, the JavaScript file introduces dynamic elements to the project. A key feature is the implementation of accordions within the FAQ section. Each FAQ title functions as a toggle switch, expanding or collapsing the corresponding answer upon user click. Furthermore, a visual indicator in the form of an icon dynamically changes between a plus and a times symbol to signify the current state of each accordion. This JavaScript functionality adds a layer of user engagement and interactivity to the project. In essence, the Netflix Clone project encapsulates the convergence of **HTML, CSS, and JavaScript** to create a compelling and responsive user interface. It successfully emulates the design philosophy and some interactive elements of Netflix, providing users with a familiar yet distinct experience. This endeavor showcases the capabilities of web technologies in recreating sophisticated interfaces and underscores the importance of a well-structured, styled, and interactive web presence.

**Introduction**

As part of my journey to learn web development, I undertook a project to create a basic Netflix clone using HTML, CSS, and JavaScript. This project demonstrates my skills in front-end development, specifically in building responsive web applications that fetch data from external sources using APIs. By integrating the TMDB (The Movie Database) API and the YouTube Search API, I built a simple yet functional clone that displays a collection of movies and TV shows, allowing users to search for specific titles and view trailers.

**Tech Stack**

* **HTML5**: Used for creating the structure of the web pages.
* **CSS3**: Applied for styling the user interface, ensuring a visually appealing design.
* **JavaScript**: Implemented to handle interactivity, API requests, and dynamic content rendering.

**Features of the Netflix Clone**

1. **Movie and TV Show Data**:  
   The project uses the TMDB API to fetch movie and TV show data. It displays a collection of popular movies and TV shows on the homepage, and users can search for specific titles using the search bar. Additionally, the YouTube Search API is used to fetch and display trailers for the selected movies or TV shows.
2. **User-Friendly Interface**:  
   The interface is designed to be simple, clean, and easy to navigate. The layout mimics the look and feel of Netflix, making it intuitive for users to browse through content. The use of hover effects, smooth transitions, and minimalistic design elements enhances the overall user experience.
3. **Responsive Design**:  
   The Netflix clone is built with a responsive design approach, ensuring it works seamlessly across all devices, including desktops, laptops, tablets, and smartphones. CSS media queries are used to adapt the layout, font sizes, and image scales to different screen sizes.
4. **API Integration**:  
   The project showcases skills in API integration by using the TMDB API to fetch movie and TV show data, such as titles, descriptions, posters, and ratings. It also integrates the YouTube Search API to fetch trailers based on the selected titles, demonstrating how to work with external data sources.
5. **Front-End Development**:  
   The clone is built using core front-end technologies (HTML, CSS, and JavaScript). HTML forms the backbone of the structure, CSS adds style and layout, and JavaScript brings interactivity by dynamically fetching data from the APIs and rendering it on the web page.
6. **Project Management**:  
   The project is managed using version control (Git) to keep track of changes, ensure proper organization, and maintain a clean codebase. Best practices for file organization, code formatting, and documentation are followed to ensure the project is easy to understand and extend.

**How the Netflix Clone Works**

**1. HTML Structure**

The foundation of the project is built using HTML. The main components include:

* **Header Section**: Contains the Netflix-like logo, search bar, and navigation menu.
* **Main Content**: Displays rows of movies and TV shows fetched from the TMDB API. Each row is dynamically created using JavaScript based on the genre or category.
* **Footer**: Includes additional information or links.

**Explanation :**

**HTML Structure:**

The HTML file defines the structure of the web page, including the header, main content sections, and footer. Notable elements include:

* **Header:** Contains the Netflix logo, language dropdown, and a “Sign in” button.
* **Main Sections:** Divided into a hero section, features section, frequently asked questions (FAQ) section, and a “Get Started” email section.
* **Footer:** Comprising multiple rows with links and information, including a language dropdown and Netflix India details.

**CSS Styling:**

The CSS file is responsible for styling the HTML elements, providing a visually appealing and responsive layout. Some key styling aspects include:

* **Global Styling:** Resets default margin and padding, sets the background color, and defines the font family.
* **Navbar Styling:** Ensures the navigation bar is positioned at the top, styled the logo, language dropdown, and “Sign in” button.
* **Hero Section Styling:** Creates an overlay on the hero background image, styles the hero card with the title, subtitle, and email form.
* **Feature Section Styling:** Styles individual feature cards with details and images, including the background videos and GIFs.
* **FAQ Section Styling:** Styles FAQ accordions, making them visually appealing and ensuring smooth transitions.

**JavaScript Interactivity:**

The JavaScript file adds interactivity to the project, particularly to the FAQ section. Key JavaScript functionalities include:

* **Accordions:** Enables the toggle functionality for FAQ accordions. Clicking on a question expands or collapses the corresponding answer, and the icon changes between a plus and a times symbol accordingly.
* **Event Listeners:** Attaches click event listeners to each FAQ title, triggering the accordion functionality.

**Project Responsiveness:**

The project incorporates media queries to ensure responsiveness across different screen sizes. Notable breakpoints include adjustments for larger screens, such as tablets and desktops, to enhance the user experience.

**Source Code 1: Netflix Working Clone**

**HTML (index.html)**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Netflix clone</title>

<link rel="icon" type="image/x-icon" href="favicon.ico">

<link rel="stylesheet" href="index.css">

<script src="index.js" defer></script>

</head>

<body>

<!-- header start -->

<header id="header" class="">

<div class="header-cont container">

<div class="left-cont">

<img decoding="async" class="header-brand" src="https://images.ctfassets.net/4cd45et68cgf/7LrExJ6PAj6MSIPkDyCO86/542b1dfabbf3959908f69be546879952/Netflix-Brand-Logo.png?w=684&h=456" alt="">

<ul class="main-nav">

<li class="nav-item active">Home</li>

<li class="nav-item">Tv Shows</li>

<li class="nav-item">Movies</li>

<li class="nav-item">News & Popular</li>

<li class="nav-item">My List</li>

<li class="nav-item">Browse by Languages</li>

</ul>

</div>

<div class="right-cont">

<svg width="24" height="24" viewBox="0 0 24 24" fill="none" xmlns="http://www.w3.org/2000/svg" class="search-icon"><path fill-rule="evenodd" clip-rule="evenodd" d="M13 11C13 13.7614 10.7614 16 8 16C5.23858 16 3 13.7614 3 11C3 8.23858 5.23858 6 8 6C10.7614 6 13 8.23858 13 11ZM14.0425 16.2431C12.5758 17.932 10.4126 19 8 19C3.58172 19 0 15.4183 0 11C0 6.58172 3.58172 3 8 3C12.4183 3 16 6.58172 16 11C16 11.9287 15.8417 12.8205 15.5507 13.6497L24.2533 18.7028L22.7468 21.2972L14.0425 16.2431Z" fill="currentColor"></path></svg>

<svg width="24" height="24" viewBox="0 0 24 24" fill="none" xmlns="http://www.w3.org/2000/svg" class="Hawkins-Icon Hawkins-Icon-Standard"><path fill-rule="evenodd" clip-rule="evenodd" d="M13 4.57092C16.3922 5.05624 18.9998 7.9736 18.9998 11.5V15.2538C20.0486 15.3307 21.0848 15.4245 22.107 15.5347L21.8926 17.5232C18.7219 17.1813 15.409 17 11.9998 17C8.59056 17 5.27764 17.1813 2.10699 17.5232L1.89258 15.5347C2.91473 15.4245 3.95095 15.3307 4.99978 15.2538V11.5C4.99978 7.97345 7.6076 5.05599 11 4.57086V2H13V4.57092ZM8.62568 19.3712C8.6621 20.5173 10.1509 22 11.9993 22C13.8477 22 15.3365 20.5173 15.373 19.3712C15.38 19.1489 15.1756 19 14.9531 19H9.04555C8.82308 19 8.61862 19.1489 8.62568 19.3712Z" fill="currentColor"></path></svg>

<img decoding="async" src="data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAOEAAADhCAMAAAAJbSJIAAAAjVBMVEUNfoD///8Ae30AeHoAdXcAdnl6q6z2+/vo8vLx+Ph1sbIAdHfu9PQHfH8AgYPl8vI7kJKRvr+hx8jO4+PJ3d4eioxppqja6eni7OxdoqRSnqC42dqXxsaw09Ntra8ph4l/uLmGtrdapqeozc02lZdMmZuuzM3A2NnU4+SRvL2fy8xSmZpEkZO+1NV1qaqNxlfBAAAEIElEQVR4nO3bXXeiOhQGYNgJg6AiiIiggtbP0U7//8+beJzjoCYtSAQ7630uetVmvZuEEOnWMAAAA " alt="">

</div>

</div>

</header>

<!-- header end -->

<!-- banner section -->

<div class="banner-section" id="banner-section">

<div class="banner\_fadeBottom"></div>

</div>

<!-- banner section end -->

<!-- movies sections -->

<div class="movies-cont container" id="movies-cont">

</div>

<!-- movies sections end -->

</body>

</html>

**CSS (styles.css)**

/\* Normalise css \*/

html,body{

background-color: #141414;

padding:0;

margin: 0;

font-family: Helvetica,Arial ,sans-serif;

}

.container{

width: 100%;

max-width: 1250px;

margin: 0 auto;

}

\*{

box-sizing: border-box;

}

\*::-webkit-scrollbar{

display: none;

}

/\* Header Css \*/

header{

background-color: transparent;

position: fixed;

z-index: 100;

left:0;

top: 0;

width: 100%;

transition: all ease 600ms;

}

header.black-bg{

background-color: rgb(20, 20, 20);

}

.header-cont{

display: flex;

flex-direction: row;

justify-content: space-between;

}

.left-cont, .right-cont{

display: flex;

flex-direction: row;

align-items: center;

}

.header-brand{

max-width: 110px;

}

.main-nav{

list-style: none;

display: inline-flex;

flex-direction: row;

}

.nav-item{

margin-right: 20px;

color: #e5e5e5;

opacity: .83;

}

.nav-item.active{

color: white;

opacity: 1;

font-weight: 500;

}

.right-cont svg, .right-cont img{

margin-right: 22px;

cursor: pointer;

}

.right-cont svg{

color: white;

}

.right-cont img{

max-width:40px;

}

/\* Movies Section CSS \*/

.movies-section{

margin: 41px 0;

}

.movie-section-heading{

font-size: 20px;

font-weight: 700;

line-height: 1.2;

color: white;

cursor: pointer;

margin-bottom: 10px;

}

.explore-nudge{

color: #54b9c5;

font-size: 12px;

display: none;

}

.movie-section-heading:hover .explore-nudge{

display: inline-block;

}

.movies-row{

display: flex;

flex-direction: row;

align-items: center;

flex-wrap: nowrap;

overflow-x:auto;

overflow-y: hidden;

}

.movie-item{

width: 245px;

object-fit: contain;

margin-right: 6px;

border-radius: 4px;

position: relative;

}

.move-item-img{

width: inherit;

}

.movie-item .iframe-wrap{

position: absolute;

top: 0;

left: 0;

width: 100%;

height: 100%;

z-index: 10;

display: none;

border: none;

outline: none;

}

.movie-item:hover .iframe-wrap{

display: block;

}

.movie-item:hover{

transform: scale(1.3);

transition: transform linear 300ms;

}

/\* Banner Section \*/

.banner-section{

background-repeat: no-repeat;

background-size: cover;

min-height: 98vh;

padding-top: 80px;

position: relative;

}

.banner-content{

display: flex;

flex-direction: column;

padding-top: 70px;

}

.banner\_\_title{

color: white;

font-size: 84px;

line-height: 1.1;

margin: 0;

margin-bottom: 16px;

max-width: 40%;

}

.banner\_\_info{

margin-bottom: 14px;

font-size: 22px;

font-weight: 700;

line-height: 1.5;

color: white;

}

.banner\_\_overview{

font-size: 16.4px;

color: white;

line-height: 1.3;

max-width: 40%;

}

.action-buttons-cont{

display: flex;

flex-direction: row;

align-items: center;

}

.action-button{

border:none;

background-color: white;

padding: 8px 23px;

margin-right: 11px;

border-radius: 4px;

display: flex;

align-items: center;

flex-direction: row;

font-size: 16px;

font-weight: 700;

cursor:pointer;

}

.action-button:last-child{

background-color: rgba(109, 109, 110, 0.7);

color:white;

}

.banner\_fadeBottom{

height: 120px;

background:linear-gradient(180deg,

transparent,

rgba(37,37,37,.61),

#141414

);

position: absolute;

left:0;

bottom: 0;

width: 100%;

}

/\* RESPONSIVE CSS \*/

@media (max-width:767px) {

.main-nav{

display: none;

}

.banner\_\_overview, .banner\_\_title{

max-width: 100%;

}

}

**JavaScript (index.js)**

// Consts

const apikey = "e950e51d5d49e85f7c2f17f01eb23ba3";

const apiEndpoint = "https://api.themoviedb.org/3"

const imgPath = "https://image.tmdb.org/t/p/original";

const apiPaths = {

fetchAllCategories: `${apiEndpoint}/genre/movie/list?api\_key=${apikey}`,

fetchMoviesList: (id) => `${apiEndpoint}/discover/movie?api\_key=${apikey}&with\_genres=${id}`,

fetchTrending:`${apiEndpoint}/trending/all/day?api\_key=${apikey}&language=en-US`,

searchOnYoutube: (query) => `https://www.googleapis.com/youtube/v3/search?part=snippet&q=${query}&key=AIzaSyA\_eZ5WJhmYhRQOM8-jAyVIzzdfWUlp\_P0`

}

// Boots up the app

function init() {

fetchTrendingMovies();

fetchAndBuildAllSections();

}

function fetchTrendingMovies(){

fetchAndbuildMovieSection(apiPaths.fetchTrending, 'Trending Now')

.then(list => {

const randomIndex = parseInt(Math.random() \* list.length);

buildBannerSection(list[randomIndex]);

}).catch(err=>{

console.error(err);

});

}

function buildBannerSection(movie){

const bannerCont = document.getElementById('banner-section');

bannerCont.style.backgroundImage = `url('${imgPath}${movie.backdrop\_path}')`;

const div = document.createElement('div');

div.innerHTML = `

<h2 class="banner\_\_title">${movie.title}</h2>

<p class="banner\_\_info">Trending in movies | Released - ${movie.release\_date} </p>

<p class="banner\_\_overview">${movie.overview && movie.overview.length > 200 ? movie.overview.slice(0,200).trim()+ '...':movie.overview}</p>

<div class="action-buttons-cont">

<button class="action-button"><svg width="24" height="24" viewBox="0 0 24 24" fill="none" xmlns="http://www.w3.org/2000/svg" class="Hawkins-Icon Hawkins-Icon-Standard">< fill="currentColor"></path></svg> &nbsp;&nbsp; Play</button>

<button class="action-button"><svg width="24" height="24" viewBox="0 0 24 24" fill="none" xmlns="http://www.w3.org/2000/svg" class="Hawkins-Icon Hawkins-Icon-Standard"><path fill-rule="evenodd" clip-rule="evenodd" fill="currentColor"></path></svg> &nbsp;&nbsp; More Info</button>

</div>

`;

div.className = "banner-content container";

bannerCont.append(div);

}

function fetchAndBuildAllSections(){

fetch(apiPaths.fetchAllCategories)

.then(res => res.json())

.then(res => {

const categories = res.genres;

if (Array.isArray(categories) && categories.length) {

categories.forEach(category => {

fetchAndbuildMovieSection(

apiPaths.fetchMoviesList(category.id),

category.name

);

});

}

// console.table(movies);

})

.catch(err=>console.error(err));

}

function fetchAndbuildMovieSection(fetchUrl, categoryName){

console.log(fetchUrl,categoryName);

return fetch(fetchUrl)

.then(res => res.json())

.then(res => {

// console.table(res.results);

const movies = res.results;

if (Array.isArray(movies) && movies.length) {

buildMoviesSection(movies.slice(0,6), categoryName);

}

return movies;

})

.catch(err=>console.error(err))

}

function buildMoviesSection(list, categoryName){

console.log(list, categoryName);

const moviesCont = document.getElementById('movies-cont');

const moviesListHTML = list.map(item => {

return `

<div class="movie-item" onmouseenter="searchMovieTrailer('${item.title}', 'yt${item.id}')">

<img decoding="async" class="move-item-img" src="${imgPath}${item.backdrop\_path}" alt="${item.title}" />

<div class="iframe-wrap" id="yt${item.id}"></div>

</div>`;

}).join('');

const moviesSectionHTML = `

<h2 class="movie-section-heading">${categoryName} <span class="explore-nudge">Explore All</span></span></h2>

<div class="movies-row">

${moviesListHTML}

</div>

`

const div = document.createElement('div');

div.className = "movies-section"

div.innerHTML = moviesSectionHTML;

// append html into movies container

moviesCont.append(div);

}

function searchMovieTrailer(movieName, iframId) {

if (!movieName) return;

fetch(apiPaths.searchOnYoutube(movieName))

.then(res => res.json())

.then(res => {

const bestResult = res.items[0];

const elements = document.getElementById(iframId);

console.log(elements, iframId);

const div = document.createElement('div');

div.innerHTML = `<iframe width="245px" height="150px" src="https://www.youtube.com/embed/${bestResult.id.videoId}?autoplay=1&controls=0"></iframe>`

// window.open(youtubeUrl,'\_blank');

elements.append(div);

})

.catch(err=>console.log(err));

}

window.addEventListener('load',function() {

init();

window.addEventListener('scroll', function(){

// header ui update

const header = document.getElementById('header');

if (window.scrollY > 5) header.classList.add('black-bg')

else header.classList.remove('black-bg');

})

})

**Output:**

