**MUSIC PLAYER PROJECT USING HTML, CSS AND JAVASCRIPT**

**By Intern Kai**

**HTML code:**

<!DOCTYPE html>

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Music Player</title>

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

</head>

<body>

    <div class="container">

        <div class="details">

            <div class="track-number">

                PLAYING x OF y

            </div>

            <div class="track-album">

                <img id="track-art" src="https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQcOftyvbwOrbDL8Z2po2CA-0VoAyFoXB6nqQ&usqp=CAU" alt="">

            </div>

            <div class="track-name">

                Track Name

            </div>

            <div class="track-artist">

                Track Artist

            </div>

        </div>

        <div class="buttons">

            <div class="previous" onclick="prevTrack()">

                <i class="fa-solid fa-backward-step"></i>

            </div>

            <div class="play-pause" onclick="playPause()">

                <i class="fa-solid fa-circle-play"></i>

            </div>

            <div class="next" onclick="nextTrack()">

                <i class="fa-solid fa-forward-step"></i>

            </div>

        </div>

        <div class="sliders">

            <div class="seek-slider">

                <div class="current">00.00</div>

                <input type="range" max="100" min="0" value="0" class="seek" onchange="seekChange()">

                <div class="duration"></div>

            </div>

            <div class="volume-slider">

                <i class="fa-solid fa-volume-low"></i>

                <input type="range" max="100" min="1" value="50" class="volume" onchange="updateVolume()">

                <i class="fa-solid fa-volume-high"></i>

            </div>

        </div>

    </div>

    <script src="https://kit.fontawesome.com/d7706770d3.js" crossorigin="anonymous"></script>

    <script src="script.js"></script>

</body>

</html>

**First section: details class**

Display the details of the track such as track number, track album (track art), name of the track and the artist of the track.

**Second section: buttons class**

Display all the buttons used in the music player such as “previous button”, “play and pause button” and “next button”. An icon for each button is used and found from the Font Awesome website. Implemented a Javascript function for each of the buttons.

**Third section: sliders class**

Display all the sliders used in the music player which are the “seek slider” for the progression of the track and “volume slider” for controlling the volume of the track. For each slider, a “range” type input is created. A few icons are used to specify the usage of each slider. Javascript functions are included in each slider as well.

**CSS code:**

@import url('https://fonts.googleapis.com/css2?family=Poppins:wght@600&family=Roboto&display=swap');

\*{

    margin: 0;

    padding: 0;

    box-sizing: border-box;

    font-family: "Poppins";

}

body{

    width: 100%;

    height: 100vh;

    background-color: #46FE97;

    display: flex;

    align-items: center;

    justify-content: center;

}

.container{

    border: 3px solid black;

    border-radius: 10px;

    padding: 10px;

    width: 60%;

}

.container .details{

    text-align: center;

}

.container .details .track-number{

    margin: 10px;

}

.container .details .track-album{

    margin: 10px;

}

.container .details .track-album img{

    width: 250px;

    height: 250px;

    border-radius: 20px;

}

.container .details .track-name{

    margin: 10px;

    margin-bottom: 0px;

    font-weight: bold;

    font-size: 40px;

}

.container .details .track-artist{

    font-size: 20px;

}

.container .buttons{

    display: flex;

    align-items: center;

    justify-content: center;

    margin: 10px;

}

.container .buttons div{

    margin: 0px 20px;

}

.container .buttons .previous , .next{

    font-size: 30px;

    cursor: pointer;

}

.container .buttons .play-pause{

    font-size: 50px;

    cursor: pointer;

}

.container .sliders .seek-slider{

    display: flex;

    align-items: center;

    justify-content: center;

}

.container .sliders .seek-slider input{

    width: 200px;

    margin: 5px;

}

.container .sliders .volume-slider{

    display: flex;

    align-items: center;

    justify-content: center;

}

.container .sliders .volume-slider input{

    width: 120px;

    margin: 5px;

}

**Font-family:**

For this music player, I used Google Fonts – Poppins as my font family. To do this, I imported the google font into the stylesheet and then specify it using font-family property.

**Alignment of content:**

I aligned all the elements in the center of the webpage. This is often done using display:flex, align-items:center and justify-content:center on the parent containers.

**Javascript code:**

let trackNumber  = document.querySelector(".track-number");

let trackAlbum  = document.querySelector(".track-album");

let trackName  = document.querySelector(".track-name");

let trackArtist = document.querySelector(".track-artist");

let playPauseBtn = document.querySelector(".play-pause");

let previousBtn = document.querySelector(".previous");

let nextBtn = document.querySelector(".next");

let seekSlider = document.querySelector(".seek");

let volumeSlider = document.querySelector(".volume");

let current = document.querySelector(".current");

let duration = document.querySelector(".duration");

let currentTrack = document.createElement('audio');

let trackIndex=0;

let timer;

let trackPlaying = false;

let totalDuration;

let currentProgress;

let trackList = [

    {

        name: "2002",

        artist: "Anne Marie",

        image: "2002.jpg",

        source: "2002.mp3"

    },

    {

        name: "Perfect",

        artist: "Ed Sheeran",

        image: "perfect.jpg",

        source: "Perfect.mp3"

    },

    {

        name: "Until I Found You",

        artist: "Stephen Sanchez",

        image: "until i found you.jpg",

        source: "Until I Found You.mp3"

    },

]

// Reset to default value

function resetValues(){

    current.innerHTML = "00.00";

    duration.innerHTML = "00.00";

    seekSlider.innerHTML = "00.00";

    seekSlider.value = 0;

    trackPlaying = false;

    playPauseBtn.innerHTML = '<i class="fa-solid fa-circle-play"></i>';

}

// Load the track

function loadTrack(){

    // Reset the track when new track is loaded

    resetValues();

    // Load new track

    currentTrack.src = trackList[trackIndex].source;

    currentTrack.load();

    // Update the duration

    currentTrack.addEventListener("loadedmetadata", displayDuration);

    // Update the track number

    trackNumber.innerHTML = "PLAYING " + (trackIndex+1) + " OF " + (trackList.length);

    // Update track art

    newArt = document.getElementById("track-art");

    newArt.src = trackList[trackIndex].image;

    // Update track name

    trackName.innerHTML = trackList[trackIndex].name;

    // Update track artist

    trackArtist.innerHTML = trackList[trackIndex].artist;

    timer = setInterval(seekRunning,1000);

    currentTrack.addEventListener("ended", nextTrack);

}

function random\_bg\_color(){

    let red = Math.floor(Math.random()\*(256-64+1))+64;

    let green = Math.floor(Math.random()\*(256-64+1))+64;

    let blue = Math.floor(Math.random()\*(256-64+1))+64;

    let bgColor = "rgb(" + red + ", " + green + ", " + blue + ")";

    document.body.style.backgroundColor = bgColor;

}

function playTrack(){

    trackPlaying = true;

    currentTrack.play();

    playPauseBtn.innerHTML = '<i class="fa-solid fa-pause"></i>';

}

function pauseTrack(){

    trackPlaying = false;

    currentTrack.pause();

    playPauseBtn.innerHTML = '<i class="fa-solid fa-circle-play"></i>';

}

function playPause(){

    if(trackPlaying==true){

        pauseTrack();

    }

    else{

        playTrack();

    }

}

function prevTrack(){

    if(trackIndex==0){

        trackIndex = trackList.length - 1;

    }

    else{

        trackIndex -= 1;

    }

    loadTrack();

    playPause();

    random\_bg\_color();

}

function nextTrack(){

    if(trackIndex==trackList.length-1){

        trackIndex = 0;

    }

    else{

        trackIndex += 1;

    }

    loadTrack();

    playPause();

    random\_bg\_color();

}

function seekChange(){

    clearInterval(timer);

    // Changing the progress of the song through the seek slider

    currentProgress = (seekSlider.value/100)\*totalDuration;

    currentTrack.currentTime = currentProgress;

    timer = setInterval(seekRunning, 1000);

    // Update the current time

    displayCurrent();

}

function displayDuration(){

    totalDuration = currentTrack.duration;

    let durationMinutes = Math.floor(totalDuration / 60);

    let durationSeconds = Math.floor(totalDuration - durationMinutes \* 60);

    if (durationSeconds < 10){

        durationSeconds = "0" + durationSeconds;

    }

    if (durationMinutes < 10){

        durationMinutes = "0" + durationMinutes;

    }

    duration.innerHTML = durationMinutes + ":" + durationSeconds;

}

function displayCurrent(){

    // Calculate the time left and the total duration

    let currentMinutes = Math.floor(currentTrack.currentTime / 60);

    let currentSeconds = Math.floor(currentTrack.currentTime - currentMinutes \* 60);

    // Add a zero to the single digit time values

    if (currentSeconds < 10){

        currentSeconds = "0" + currentSeconds;

    }

    if (currentMinutes < 10){

        currentMinutes = "0" + currentMinutes;

    }

    // Display the updated duration

    current.innerHTML = currentMinutes + ":" + currentSeconds;

}

function seekRunning(){

    let seekPosition = 0;

    // Check if the current track duration is a legible number

    seekPosition = currentTrack.currentTime \* (100 / currentTrack.duration);

    seekSlider.value = seekPosition;

    displayCurrent();

}

function updateVolume(){

    currentTrack.volume = volumeSlider.value/100;

}

loadTrack();

**Creating variables and initializing the variables:**

Create the variables for each of the elements to be controlled in the music player which are the track number, track art, track name, play and pause button, previous button, next button, current time, duration of the track, seek slider and volume slider. This is done by using the querySelector to select the classes of the elements.

**Creating a list of objects for the tracks:**

Create a list for the tracks. In the list, make sure there are objects and attributes for each track. The attributes are the track name, track artist, image source of the track art and the source of the track in mp3 format. I also created an audio element using document.createElement(‘audio’)

**Creating a function to reset the values:**

A function resetValues() is created to reset all the values to initial values once the track ends or when user skip the track.

**Creating a function to load the track:**

Load the track when the music player is open, when the users skip the track or once the current track ends. In this function, I updated the details of the track as well as inserting the source of the mp3 track. These are done by accessing the attributes from the list of track.

**Creating a function to create random background color:**

I created a function that randomly create bright background color in RGB format. A random background color should be displayed once a new track is loaded.

**Creating functions for the buttons:**

Functions such as prevTrack(), nextTrack(), playTrack(), pauseTrack() and playPause() are created to control the flow of the track. The prevTrack() and nextTrack are controlled using the index of the list of track. In this case, index 0, 1, 2. The pauseTrack(), playTrack() and playPause() are controlled using the Boolean variable trackPlaying.

**Creating a function for the user to alter the seek slider:**

A function seekChange() is created to jump the track to the desired timeframe when the user change the value of the seek slider (range input).

**Creating a function for the user to alter the volume of the track:**

A function updateVolume() is created to allow the user to lower and higher the volume of the track when he/she change the value of the volume slider (range input)

**Creating functions to display the current time as well as the duration of the track:**

Function displayCurrent() is used to display the current timeframe of the track. This function should work when the track is playing or when the user alter the progress of the track. Function displayDuration() is used to display the duration of the track playing.

**Screenshot of interface of the music player**

