MUSIC THEME JQUERY APPLICATION

SHAKILA SHRESTHA

PURDUE OF INDIANAPOLIS

CIT 21500: WEB PROGRAMMING

PROF. TJ KANE

11/16/2024

Abstract

Create an interactive music-themed application where users can explore genres, view playlists, and play song previews. The app will use jQuery for dynamic interactions, such as DOM manipulation, event handling, and dynamic content generation.

MUSIC THEME JQUERY APPLICATION

The app will use jQuery for creating the music playlist for the users. It is one webpage that can be used for random song listening. It uses dynamic interactions, such as DOM manipulation, event handling, and dynamic content generation.

Method and Steps

1. Purpose of the Application

The purpose of this application is to create an interactive music platform where users can explore different music genres, view song playlists, and play/pause songs dynamically. The app uses jQuery to allow users to interact with the genre selection buttons, update the playlist based on selected genres, and play songs through an embedded audio player. The application aims to provide a smooth, engaging experience by using jQuery for dynamic content updates, user interaction, and styling.

2. How to Use the Application

To use the application, follow these steps:

- **Select a Genre**:
  On the homepage, you will see a list of genre buttons (e.g., Rock, Pop, Jazz, etc.). Clicking on a genre button dynamically updates the playlist with songs corresponding to that genre. The active genre button is highlighted for better user experience.

- **Interact with Playlist**:

  After selecting a genre, the playlist will display the corresponding songs. Each song in the playlist is shown with its album cover and title. Click on any song in the list to play it.

- **Play/Pause Music**:

  The embedded audio player allows users to play or pause the selected song. Clicking the play button will start the song, and clicking it again will pause the song.

- **Change Genre**:

  You can change genres at any time by clicking on another genre button. The playlist will automatically update to reflect the new genre, and the previous genre's selection will be cleared.

Num	**Description**	**How This Was Implemented in My Code**	**Code Snippet**
1	**Use of jQuery** The application uses jQuery to interact with the DOM, respond to events, and dynamically update content.	The app uses jQuery to access the DOM for interacting with elements such as genre buttons, the playlist, and the audio player. It also dynamically updates the playlist and handles user events like clicks.	$('.genre-button').on('click', function() { ... });
2	**Event Handling**	jQuery event handlers are used to manage genre	'`$('.genre-button').on('click',`

	The application should respond to user interactions, like clicks, to trigger specific actions.	selection (`.genre-button`), song selection (`#playlist li`), and player controls (`#play-button`, `#pause-button`, `#next-button`). These events trigger the appropriate actions like playing the song and updating the song details.	`function () {...}); $('#next-button').on('click', function() {...});`
3	**Dynamic Content Creation** Content like song names and album covers should be added dynamically using jQuery.	When a genre is selected, jQuery is used to generate the corresponding playlist dynamically by appending new song elements to the DOM.	$songNames.append(`<p>${song.title}</p>`);
4	**Styling with jQuery** The app should use jQuery to add or modify styles for user interaction feedback.	When a genre button is clicked, the active genre button is highlighted by dynamically adding/removing the `active` class using jQuery.	$('.genre-button').removeClass('active'); $(this).addClass('active');
5	**Interface is Authentic, Professional, Balanced**	The application is designed with a clean, professional layout. The color scheme is cohesive, and the elements are well spaced. The design is responsive and adapts to	body {background-color: #000000; color: whitesmoke;}

	The application should have a polished, user-friendly design with appropriate visual elements.	different screen sizes.	

6	**DOM Dynamically Built**
	The playlist is dynamically generated using jQuery based on the selected genre. Each song's data (cover, title, etc.) is added to the DOM dynamically.
			`const listItem =`
			`\<li data-index="${index}" data-mp3="${song.mp3}" data-cover="${song.cover}" data-artist="${song.artist}">`
			`` ...` ``

7	**Documentation : Program Code and Project Description**
	
	The code is fully annotated with comments explaining the functionality of each section. The project description explains the goals of the application and its functionality.
		Playlists: The playlists for each genre (Rock, Pop, Jazz) are predefined and dynamically loaded when the user selects a genre button.
		Song Selection: When a user clicks on a song in the playlist, it updates the audio source, album cover, artist name, and song title.
		Player Controls: The "Play", "Pause", and "Next" buttons are interactive, with dynamic feedback (color changes and transformations) using jQuery.
			`// Handle genre selection`
			`// Play selected song`

Play Songs

```
function playSong(song) {
  $('#audio-player').attr('src', song.mp3).get(0).play();
  $('#album-cover').html(`<img src="${song.cover}" alt="Album Cover">`);
  $('#artist-name').text(`Artist: ${song.artist}`);
  $('#song-title').text(`Song: ${song.title}`);
}
```

Additional Considerations

- **Code Structure**: The code follows modern JavaScript best practices, using `const` and `let` for variable declarations, ensuring better scoping and immutability. I avoided using outdated methods like `var`.
- **Testing**: Before final submission, I tested the functionality of the app on different browsers and devices to ensure compatibility and responsiveness.