CS2003 Practical-Web 01

Tutor: Ozgur Akgun

Overview

In this practical I was asked to create a webpage that allows users to create a playlist from a selection of The Beatles songs. Songs can be added and removed from the playlist and the duration of the playlist is shown.

I have completed all parts of this practical fully. I have achieved a solution where the user can add and remove songs to the playlist from a collection of The Beatles songs and the duration of the playlist is shown correctly.

Design

To add and remove songs to and from the playlist I designed this so the user would click a button ‘ADD’ or ‘REMOVE’ and the song would be added or removed from the playlist. To make sure the correct songs were added or removed when calling the methods for adding and removing songs I would pass the id of the song into the method. This meant I was easily able to determine which song was to be added or removed and carry out the different functionalities needed. For example, using this to determine whether the song was already present within the playlist and making sure that if the song was to be removed it didn’t appear in the playlist.

When designing the HTML code, I decided to create the basic skeleton of the table so that if for whatever reason the JavaScript didn’t work correctly in the browser then it would still show a basic outline of the tables just with no data rather than a completely empty webpage.

To find the correct part of the DOM to get or import HTML code into I used document.querySelector(#id_name). The # character prefix shows that this is an id. To place HTML code within the section that has the id of wherever I want the code to go I used variable.innerHTML = `html_code_string`.

When adding a new song to the playlist to retrieve the HTML code already within the playlist section I used document.getElementById(id).innerHTML.toString(). The code to add the new song would then be concatenated on the end of this then be sent back to the HTML page. This meant that songs that were already present in the playlist would stay and the new song would be added at the end of the playlist.

When I tried adding the duration_ms field of each song to the JSON file this caused some formatting errors that I didn’t expect. To overcome this I changed the time into the mm:ss format beforehand and then placed the data in the JSON. As the time was already converted to the mm:ss format this meant I didn’t have to convert it when adding all the songs to the list of songs or each time a song was added to the playlist.

This did mean however that when calculating the duration of the playlist it wasn't as simple or efficient as it would be just using the milliseconds measurement. I designed this part so that it would read the first two characters of the string as the minutes, and the last two characters to be the seconds. I totalled each, divided the total of the seconds to get the total minutes then found the remainder to get the number of extra seconds. As shown in my testing and solution this technique of achieving the correct result works, but it isn't as efficient as simply just using the milliseconds.

When reading the data from the JSON file I took advantage of using the promises functionality. I called fetch() on the JSON file and used .then() to gather the response, then an additional .then() to write all the songs to the HTML that has all the songs. As JavaScript has an asynchronous runtime the other methods would have run so I made use of the promises to counteract this.

To store the songs added to the playlist I used an array. In JavaScript the size of an array doesn't have to be specified when initialising this proved useful due to the changing of size of a playlist. To perform the operation of removing the songs from the playlist I used a predefined function of JavaScript arrays array.splice(start, deleteCount). This allowed me to remove the song from the playlist and not have to create a whole new array for the new playlist like you would have to using an array in Java.

I designed my playlist to not allow duplicate songs, I made this decision as I believe this made more sense generally for only one version of each song to be allowed on the playlist. An alert is shown to the user if they attempt to add a song that has already been added to eliminate any confusion the user might experience if they attempt to add a song and it doesn't add to the playlist as they would expect if they forgot they added it previously.

Testing

The tests shown below showcase testing on the different functionalities of the webpage.

What is being tested?	**Pre-conditions**	**Expected outcome**	**Actual outcome**
All the songs appear in the songs list on the window load.	Load the webpage.	All songs present in songs table.	All songs present in songs table.
Adding a song to the playlist.	Click "ADD" button on a song.	The song added appears in the playlist table.	The song added appears in the playlist table.
Removing a song to the playlist.	Add a song to the playlist. Click "REMOVE" on a song added to the playlist.	The song removed from the playlist is no longer shown in the playlist.	The song removed from the playlist is no longer shown in the playlist.
The songs are shown in order in which they were added.	Add three songs to the playlist. Remove the second song.	The song which was shown second should now be shown last.	The song which was shown second should now be shown last.

	Add the second song again.		
The duration of the playlist is correct.	Add three songs.	The duration shown should be the total of the duration of the three songs.	The duration shown should be the total of the duration of the three songs.
The number of songs is correct.	Add four songs.	The number of songs shown is equal to the number of songs added (in this case 4).	The number of songs shown is equal to the number of songs added (in this case 4).
The duration changes correctly after song is added/removed.	Add four songs. Remove a song.	The duration of the playlist accounts for the song being removed.	The duration of the playlist accounts for the song being removed.
The number of songs changes correctly after song is added/removed.	Add four songs. Remove a song.	The number of songs of the playlist should go down to account for song being removed.	The number of songs of the playlist should go down to account for song being removed.

Evaluation

From the tests above all the expected outcomes match the actual outcomes, this outlines that my solution provided fully meets the requirements given in the specification. I believe what I have produced matches what was expected from this practical.

Conclusion

In this practical I have achieved a fully functioning webpage that utilises client-side JavaScript functionality. My solution allows users to create a playlist by adding and removing songs.

Initially I found writing the JavaScript portion of the practical difficult as it was a new language to me, however after researching and getting used to the syntax it became to utilise the full functionality JavaScript offers.

Given more time I would have liked to add a play button so the user could listen to the music they added to their playlist. Additionally, I would have liked to allow the user to create multiple playlists instead of just the one. Finally, I would have liked to make the webpage remembered the user's playlist even if the page was refreshed.