Project Work: Music Player

Lehtosaari Maija 001196337

Maija.lehtosaari@student.lut.fi

Declaration of AI usage:

For this project I used AI (ChatGPT) a few times to proofread my code or find an error that I spent a long time with. For example, my music volume didn't go up in increments of 1 when using key controls, so I asked AI to help me find the error. After getting some suggestions on what the error might be, I tried different ways to fix it.

I used various internet sources for new functions (draggable, events) that I have added below. I also used the source code from the course as a base for my music player.

Key Controls
"Space" = STOP / PLAY Music

"1, 2, 3, 4" = select the track for given number

"Arrow left" = volume down for selected track

"Arrow right" = volume up for selected track

I picked the music player as my course project so that I could use the source code provided and have more time for experimenting and trying out different things with CSS, HTML and java script.

I started by finding the source codes for the music player and importing them to my visual studio code. I first read through the code and tried different buttons to see what features the player already had. I couldn't find a stop button or a way to stop the music, so I decided to pick this as my first task.

Stop Button

I first made the stop button in html and got the element by id in java script. After this I searched how to pause the music in java script. I read source [1] and tried to implement it. Initially I had some issues since it wouldn't stop all tracks, but I found a way to fix this by looping through a list of audios to ensure all of them are paused.

Volume Sliders for Each Track

For this, I searched for slider tutorials online to see how they can be created. I read through source [2] and started by making one html slider and giving it values. Then I edited java script code to find the slider with id and change the volume according to what the input is. Once I saw that it worked for track 1 I created 3 more volume slides. Finally, I changed the code so that it checks what number the audioRange is so that the right audio is changed.

After this step I also created my css file and added styles to the sliders as well as some of the body. For the sliders I took inspiration from the w3schools article but tried also some of my own ideas.

Drag and Drop

I tried to find if the course had any examples of how drag and drops are done but I couldn't find any easily. After this I read some articles and saw that there are certain apis and tools that can help you do this function. I decided against this since I was worried that it could mess with responsiveness or would make the music maker not work on certain search engines. I ended up finding a tutorial that used only html and java script for the drag and drop and this is what I used.

I made sure that my drag feature worked on one instrument button first and then I added it to the rest of the buttons:

I edited the source code to give the audio buttons draggable attribute and created the drag start event. The tracks were the destination, so I gave them drag over listener and data transfer like the article / tutorial did. This took me some time since the article code was for one drag and drop but my audios work through variables and were more complicated.

Once I got all my drags to work and go on right tracks, I modified the CSS for these. I added some new classLists for the buttons and tracks. For these I added certain modifiers such as colors, padding, margins and heights. I tried different values to see what looks good. I also added a background through HTML to make the music player look more interesting.

Instrument length visualized

I started by googling functions on how to extract audio length as a number. I found a source that showed how to extract metadata from videos and audio. After getting the duration I didn't know how to use it. This took me the most amount of time from the task. I tried to add it to class list or straight to CSS.

I found a source that showed how to style width straight in java script and this ended up working. The difference between lengths was so small that I tried different multipliers to see which would make it look ok on my browser. I noticed that when I made my screen smaller (mobile size) the sample style would go over my track. I tried different CSS settings but ended up setting my tracks max-width to 97% and instrument max-width to 92%, so that it would be more responsive.

At this point I tried the application on different browsers to see if it was responsive and worked. I couldn't try the application on mobile, but I did resize my window to see if it would scale well, if all the buttons can be pressed and nothing gets hidden.

Key controls

I searched key codes for the buttons I wanted to make events for. I wrote a small function for space bar which would check if music was playing or not and stop or play it based on that.

After this I made four keybinds that would select tracks from 1-4. This was so that the user could press "1" and change volume with arrows. Creating the volume change for the arrows was a bit more difficult since I initially tried to create new input listeners, and this didn't do anything (I an existing listener for sliders). I searched up "how to start event after eventlisterner js" and read articles regarding eventListeners. I found a function for dispatchEvent() and tried few variations until I got one of them to work.

For some reason my volume worked in increments of 1 but volume up would go 0->1->11->100. I struggled with finding a cause for this: I checked the button range, how I handled the value in the input and could not determine what caused this. I managed to fix the issue by adding brackets and ensuring that my volume is handled as number.

Categories

In short, I found out how to add categories for objects and added them to the existing instruments. After this I made a band category button so I could test out one category first. After I was able to get only band items to appear when pressing this button, I copied the codes for the other two categories. Then I styled the buttons and edited the addButtons code so that no instruments would be visible when loading the page (category must be picked first).

Features I suggest getting points from:

Feature	**Max points**
Well written PDF report	3
Application is responsive and can be used on both desktop and mobile environment	4
Application works on Firefox, Safari, Edge and Chrome	3
The application has clear directory structure and everything is organized well	2
Stop button that stops music	1
Adjustable volume per track	2
Drag'n'drop new instruments to the tracks (with mouse or touch screen)	4
Instrument length is visualized by length/width	4
Keyboard shortcuts for: "space" play or stop music, "1,2,3,4" for selecting track and arrow left and right to change volume up or down.	2
Instruments are categorized	2
CSS styling (hover for category buttons, background image, individual styles for buttons/tracks)	2
Total	29

Sources:

1. https://codetogo.io/how-to-stop-audio-in-javascript/ (stop button)
2. https://www.w3schools.com/howto/howto_js_rangeslider.asp (range sliders)
3. https://medium.com/@future_fanatic/how-to-create-drag-and-drop-functionality-in-javascript-a-step-by-step-tutorial-8ea236ef9416 (for drag start / drag function)

4. https://developer.mozilla.org/en-US/docs/Web/API/HTMLMediaElement/loadedmetadata_event

   https://stackoverflow.com/questions/53692749/javascript-change-css-width

   https://stackoverflow.com/questions/34647536/how-to-get-audio-duration-value-by-a-function
   Audio length and visual

5. https://medium.com/analytics-vidhya/implementing-keyboard-controls-or-shortcuts-in-javascript-82e11fccbf0c
   https://www.toptal.com/developers/keycode key binds