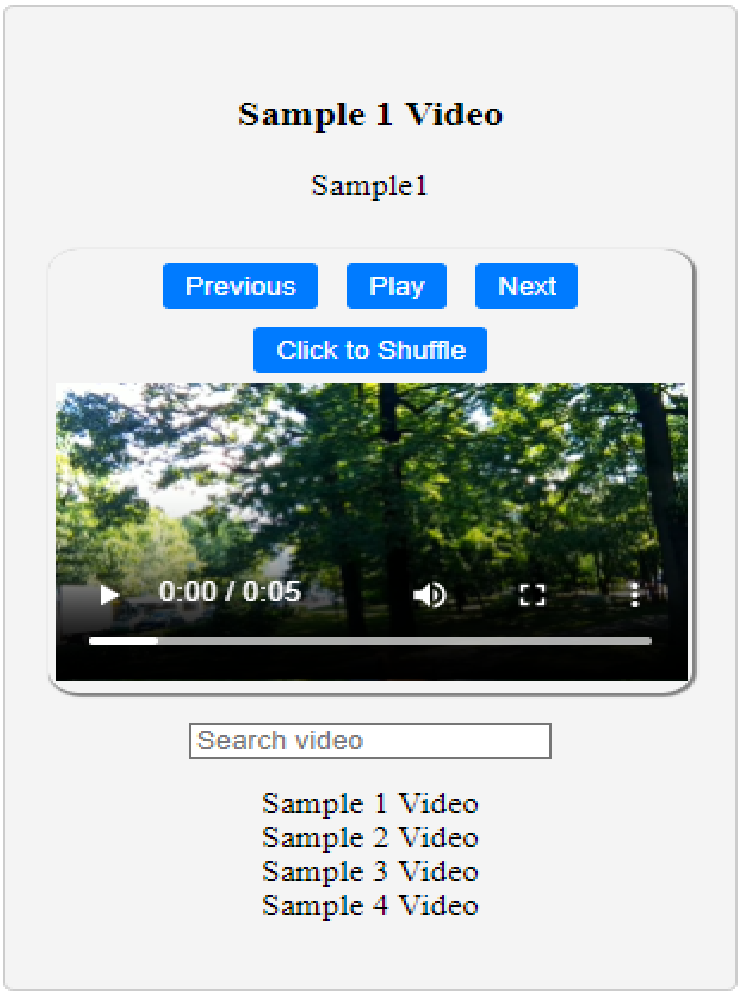
Built-in Functions in JavaScript - Assessment

Introduction

This week, you will be working on a project titled **Video Player App**. This project will include concepts of Array such as searching an item in an array, shuffling items of the array, finding previous and next items of the array etc. You will be using array methods such as **indexOf**, **length** and Math module's methods such as **Math.floor** and **Math.random**.

The details of the Video Player App are mentioned below:

- The App contains a Video Player single-page website.
- The webpage contains a Video Player element which can store mp4 videos in a play list.
- It can then play those videos in the order given in the playlist.
- The playlist is displayed at the bottom of the video player below the search bar.



- The following are the functionalities of this app:

This file is meant for personal use by jwn1534@gmail.com only.
Sharing or publishing the contents in part or full is liable for legal action.

a. **Playing the Video:** When the user clicks on the **Play** button once the website is loaded, the Video Player plays the first video in the playlist. The user can select a different video by clicking on the videos in the playlist given at the bottom. Once a new video is selected, clicking on the **Play** button will now play the selected video. (This is already implemented.)

b. **Playing the Next Video:** When the user clicks on the **Next** button, the next video in the playlist is played.(This is to be implemented according to the tasks given in the Problem Statement.)

c. **Playing the Previous Video:** When the user clicks on the **Previous** button, the previous video in the playlist is played.(This is to be implemented according to the tasks given in the Problem Statement.)

d. **Shuffling the Playlist:** When the user clicks the **Click to Shuffle** button, the playlist gets shuffled. The shuffled playlist is displayed. The songs will now be played according to the shuffled playlist. Also, as the user clicks the **Click to Shuffle** button, the button's text content is changed to **Click to Unshuffle**. Clicking on the button again should play the original playlist as the list is now unshuffled.(This is to be implemented according to the tasks given in the Problem Statement.)

e. **Searching Songs:** There's a search box below the Video player. The user can search a video based on its title. The playlist should get updated.(This is already implemented.)

- You will find the following files in the folder:
  - **index.html:** This file contains the html for the web page.
  - **style.css:** This file contains the styling for index.html.
  - **script.js:** In this file, you have to add code to complete the tasks which are mentioned in the Problem Statement.
  - **videos.**js: This file contains an array videos consisting of video objects with properties such as **title, artist-name** and **url**.
  - **videos sub-folder:** This sub-folder contains four sample mp4 video files.

Housekeeping points

- This is a minimal example and may not follow some standard practices.
- We focus on the main flow, and not much error handling.

Problem Statement

1. **Update Play List**: You have to write code for the **handleClubClick()** function.
   - As and when the playlist is changed (on the page load and thereafter), this function is called.
   - The function takes a playlist as an argument.
   - A new <li> element is created for each video in the playlist.

This file is meant for personal use by jwn1534@gmail.com only.
Sharing or publishing the contents in part or full is liable for legal action.

- Then each <li> element is added to the videoList <ul> element.

2. **Video Selection from Playlist:** You have to write code for clicking and selecting a video from the playlist.
    - Add **click** event listener to the videoList HTML element.
    - In the callback function of the **click** event:
        i. Find the index of the clicked video from the playlist.
        ii. Call the **playvideo()** function.
        iii. Pass argument to **playvideo()** function based on the following condition:
            1. If the **isShuffle** flag is true(which means the video playlist is shuffled) then shuffled videos should be passed as arguments.
            2. If the **isShuffle** flag is false(which means the video playlist is not shuffled), the originalList should be passed as argument.

3. **Next Button Click:** You have to write code for clicking the **Next** button.
    - Add click event listener to the Next button.
    - In the callback function of the click event:
        i. Increase the current video index by 1.
        ii. If the current video index becomes equal to the length of the playlist, then set the current index to 0.
        iii. Play the video (for playing the video, refer to the ii and iii parts of the second bullet point of Task2).

4. **Previous Button Click:** You have to write code for clicking the **Previous** button.
    - Add click event listener to the Previous button.
    - In the callback function of the click event:
        i. Decrease the current video index by 1.
        ii. If the current index becomes less than 0, then set the current index to the (length of playlist -1).
        iii. Play the video (for playing the video, refer to the ii and iii parts of the second bullet point of Task2).

5. **Shuffling the Playlist:** You have to write code for the **shuffleArray()** function.
    - This function takes the original playlist (which needs to be shuffled) as an argument.
    - Traverse the array.
    - For each item's index generate a random index.
    - Interchange the items of the original index and random index with each other.
    - For example, if the original index is 1 and the generated random index is 3, then replace the element at index 1 with 3 and 3 with 1.
    - This way the whole list will be shuffled.

Program Organization

- You will be getting a zip folder containing a folder named **Built-in Functions in JavaScript Assessment_For Coders.**

This file is meant for personal use by jwn1534@gmail.com only.
Sharing or publishing the contents in part or full is liable for legal action.

- The **Built-in Functions in JavaScript Assessment_For Coders** folder has 4 files namely **index.html, style.css** and **script.js, videos.js** and a **videos** subfolder consisting of 4 sample videos.
- You are required to add functionalities to complete the tasks (stated above in the problem statement) in **script.js** file.

Evaluation Rubric

Total Project Points: 60

- Correctness:
  Correctness of implementation
    - Problem statement - point 1 **(20%)** : **12 Points**
    - Problem statement - point 2 **(20%)** : **12 Points**
    - Problem statement - point 3 **(20%)** : **12 Points**
    - Problem statement - point 4 **(20%)** : **12 Points**
    - Problem statement - point 5 **(20%)** : **12 Points**

Program Instructions

● The **Built-in Functions in JavaScript Assessment_For Coders** folder should have **index.html, style.css** and **script.js, videos.js** and a **videos** sub-folder.
● The **Built-in Functions in JavaScript Assessment_For Coders** folder should be compressed as zip/rar.
● Project will not be evaluated if the submitted project is not in the zip/rar format.

This file is meant for personal use by jwn1534@gmail.com only.
Sharing or publishing the contents in part or full is liable for legal action.