JS 15 MP3 Player OOP

In JS 14, we were introduced to public versus private, class versus objects. Remember that a class is a generic model while an object is a copy of a class that is can be customized.

We have a single Honda Accord car model (**class**) but we can have multiple copies of the Accord of different color (**objects**).

We created a class called `SongProfile` and learned how to access private variables and functions using the **member access operator**, which is the period ➔ song_0.BandName = "Jackson 5";

In this session, we will add new attributes to the class `SongProfile` and create multiple objects. Next, we will allow the user to customize their experience by adding buttons to choose songs to be played.

1. Private attribute 2. Making objects 3. Onclick event to customize playlist

===

JS Challenge 1. Look at the code in between <script> </script>

Go to the **class definition of SongProfile** and do the following (first person view)

a. Create a private variable named "`title`" and load the variable `sTitle` into it.
b. Create a private variable called "`titleId`" and load the data "`songTitle`" into it

c. Create a private variable called "`caption`" and load the variable `sCaption` into it
d. Create a private variable called "`captionId`" and load the data "`caption`" into it

JS Challenge 2. Look at the code in between <script> </script>

Create 3 more objects and **initialize** the objects using the data below. Write the code under the code in **PURPLE**. The code in **BLUE** is your example.

Object name: Song 1
Band name: “Code Ninjas Curly Braces”
Song Title: “Where’s My Curly Braces”
Genre: “Syntax”
Caption: “open/close”

Object name: Song 2
Band name: “The Delimiters”
Song Title: “Just My Functions ”
Genre: “Actions”
Caption: “parameters”

Object name: Song 3
Band name: “Memory Element”
Song Title: “What’s A Variable?”
Genre: “Var”
Caption: “Don’t forget to load me with data”

```
function init_system()
{
      alert( "inside init_system()" );

      /* ===================
      bName, sTitle, sCaption, sGenre, sFN


      */
       var song_0 = new SongProfile( "Magic!", "Rude", "#1", "Pop", "blah.mp3");

       playlist.push( song_0 );


      // -- put code here for the JS Challenge 2




}
```

1. Write The HTML Code In Between `<body> </body>`. Write ALL of it!!! Put the code below the `<body>` tag.

```
<button id = "minusOne" onclick = "dec()">
      Back
</button>


<button id = "plusOne" onclick = "incr()">
      Next
</button>
```

Explanation

This will look familiar to you guys. We are creating two buttons called "**Back**" and "**Next**". We then attach the `onclick` event to each button. The action is either "**dec()**" or "**incr()**".

As you guys can see, a playlist and a media player work together to create a music service. The playlist is an array and the index is the current song being played (**remember that the index indicates our current position in the array**). We decrement the index to go back to the previous song and increment the index to go forward one song.

2. Write The CSS Code In Between <style> </style>

```
#songInfo
{
      padding   : 0px 0px 0px 0px;
      margin    : 0px 0px 0px 0px;

      width     : 550px;
      height    : 250px;

      /*border    : 2px solid #606060;*/

}

#songInfo table#playlistTb
{
      padding   : 0px 0px 0px 0px;
      margin    : 0px 0px 0px 0px;

      width     : 100%;
      height    : 100%;

      text-align: center;
      font-family: Arial, Tahoma, Serif;

      border-collapse: collapse;
}
```

```
#songInfo table#playlistTb tr
{
      padding    : 0px 0px 0px 0px;
      margin     : 0px 0px 0px 0px;

      border-bottom    : 2px solid #c0c0c0;
      background-color: #909090;

}

#songInfo table#playlistTb tr th
{
       padding    : 0px 0px 0px 0px;
      margin     : 0px 0px 0px 0px;

      width      : 50px;
      height     : 50px;

      border     : 2px solid #606060;
      background-color: #606060;
      color      : #e0e0e0;

}

#songInfo table#playlistTb tr td
{
       padding    : 0px 0px 0px 0px;
      margin     : 0px 0px 0px 0px;

       width      : 50px;
      height     : 50px;

      /*background-color: #909090;*/
}
```

Explanation

We are styling the playlist. In this case, the playlist is a vertical list. We can create a vertical list by using a table structure.

When using a `<table>`, remember that
1. A table is compose of **rows** and this is why we have `<tr>`.
2. Each row has a **table header** and this is why we have `<th>`.
3. We also have **table data** and this is why we have `<td>`.

You can be creative and change the style to be anything you like.

HTML Challenge 1

1. Make a button and give it an id of "**ran**". In between the open and closing tag of the button, put the word "**Random**".
2. Next, attach a click event of **onclick**. When the user clicks on this paragraph tag, the JS function "**ran_sel()**" is called.

JS Challenge 1

1. Underneath `<script>`, create a variable called **ranBit** and load the data **0** into it.
2. Write a function definition called **ran_sel()**
    a. Inside the function definition of **ran_sel()**, load the data 1 into the variable **ranBit**.

3. Go to the function definition of **incr()** and **do 3a. underneath the purple comment** .
    3a. Check if **ranBit** is equal to 1.
        i. If true, then put the code below inside the open and closing curly braces of the "**if**" statement
        ii. `Math.floor(Math.random() * playlist.length-1);`

```
if( trackNum > playlist.length-1 )
{
	trackNum = 0;
}

// -- random sel



// -- update highlight
change_songs( trackNum, playlist );
```

HTML Challenge 1

1. Make a button and give it an id of "**popSO**", which stands for "**pop songs only**".
2. In between the open and closing tag of the button, write "**pop songs only**"
3. Next, attach a click event of **onclick**. When the user clicks on this paragraph tag, the JS function "**count_pso()**" is called.

HTML Challenge 2

1. Make a paragraph tag and give it an id of "**numPS**", which stands for "**number of pop songs**".
2. In between the open and closing paragraph tag, write "**number of pop songs**".

JS Challenge

1. Write a function definition called **count_pso()**
4. Inside the function definition, do the following
   a. Declare a variable and name it "**genreCount**" and set it equal to 0
   b. Declare a variable named n and load the value of 0 into it
   c. Write a "for" loop using the variable n as the digital key
   d. The array is named "playlist". Next, loop until **n < playlist.length**
   e. **INSIDE THE body** of the loop
      i. Access the private attribute "**genre**" and check if it is equal to "**pop**"
      ii. If so, increment the variable "**genreCount**" by 1

```
for( ; ; )
{
     Playlist[ ]
}
```

   f. Outside the "for" loop, link JS with HTML by using `document.getElementById()` and the id of "**numPS**"

   g. User **.innerHTML** to display the number of pop songs found