JS 17 MP3 Player Final

In JS 16, we created a pop up banner to display extra information. The user "**asks**" for the extra information by putting the mouse cursor over a song.

Is there a way for us to customize the service? What if someone only wants to listen to songs within the "**EDM**" genre? Is there a way for us to write code that **filters out** all songs and only keeps songs within the "**EDM**" genre?

In this session, we focus on filtering out songs and this gives the user the ability to customize their playlist.

1. Button and onclick event 2. Array manipulation 3. Cloning

==

1. Write The JS Code In Between `<script> </script>`. WRITE ALL OF IT!!! Color code is for explanation.

```
function make_table( tId )
{
      //alert( "inside make a clone()" );
      // -- make table
      var tbDOM = document.createElement("TABLE");
      // tbDOM.setAttribute( "id", "playlistTb" );
      tbDOM.setAttribute( "id", tId );
      console.log( "make new table" );

      // -- put table inside songInfo
      document.getElementById( "songInfo" ).appendChild( tbDOM );
      console.log( "append table" );

}
```

Explanation

There are **two ways** to create multiple music playlists. The **first approach** is to **ONLY delete** what we don't need and then building the playlist by adding on to the remaining songs. The problem with this approach is that we have to keep track of all of the items that we deleted and then fill in the empty slots. This is possible but the code is messy.

The **second approach** is to erase everything and start over. This seems like a lot of work!!! **Actually, it is really efficient to erase everything and just start over.** Remember that we are using functions and so starting over means calling functions to building the playlist again.

This first function definition is building the table by creating a new table element. The code in green uses `document.createElement()` to create a new table element. **This is important because we are creating a new HTML element on the fly by writing Javascript code.**

How important is it? Imagine driving down the road and a flat tire causes you to pull the car over onto the side. In a real work situation, we have to call and then wait for maintenance service to arrive.

Now, imagine being able to create a new wheel on the spot, changing out the flat tire to put in the new tire, and then continuing on. This is what we are doing with `var tbDOM = document.createElement("TABLE");` We are creating a new item on the fly and then using it immediately.

The code in **maroon** gives the new table a name. How do we give an HTML element a name? The answer is to give it an id. Remember that giving an id to an HTML element allows JS to find it. **This is special code because JS is giving the id and this creates dynamic HTML.**

The code in **gold** puts the newly created table into the webpage.

2. Write The JS Code In Between `<script> </script>`. WRITE ALL OF IT!!! Color code is for explanation.

```
function make_row( rId, rNum, tId )
{
	// -- make row
	var trDOM = document.createElement("TR");
	trDOM.setAttribute( "id", rId + rNum );
	console.log( "make new row" );

	// -- put row inside table
	var tbDOM = document.getElementById( tId );
	tbDOM.appendChild( trDOM );
	console.log( "append new row" );

}
```

Explanation

Next, we are making a row. The code in green uses `document.createElement()` to create a new table row, "**TR**". After that, the code in **maroon** gives the newly created row a name by setting its id. Finally, the code in **gold** puts the newly created row inside the table that was created in chapter 1.

3. Write The JS Code In Between `<script> </script>`. **WRITE ALL OF IT!!!** Color code is for explanation.

```
function make_a_clone( id, songObj, rId, rNum )
{

       for( var i = 0; i < tdIdNamesArr.length; i++ )
       {

              var tdDOM = document.createElement( "TD" );
              tdDOM.setAttribute( "id", tdIdNamesArr[i] );

              var trDOM = document.getElementById( rId + rNum );
              trDOM.appendChild( tdDOM );

       }

       var cNodes = trDOM.children;

       cNodes[0].setAttribute( "id", playlist[id].bandId);
       //alert( cNodes[0].innerHTML );
       cNodes[0].innerHTML = songObj.bandName;

       cNodes[1].setAttribute( "id", playlist[id].titleId );
       cNodes[1].innerHTML = songObj.title;

       cNodes[2].setAttribute( "id", playlist[id].captionId );
       cNodes[2].innerHTML = songObj.caption;

       cNodes[3].setAttribute( "id", playlist[id].genreId );
       cNodes[3].innerHTML = songObj.genre;

       //document.getElementById("playlistTb").appendChild( clObj );

       // -- set display to "block"
       //clObj.style.display = "block";

       /* ==================================================
      Insert before

      */
      /*var trRef = document.getElementById( "playlistTb" )
      trRef.insertBefore( clObj, trRef.childNodes[0] );*/

}
```

Explanation

The code in green is code that you have seen before
1. It creates a new data cell "**TD**" by using `document.createElement()`
2. Next it gives the newly created TD an id
3. Then it puts the newly created TD at the end.

The code in **blue** uses a "**for**" loop because we are creating a row for each song in the playlist. So, if there are 2 songs of the genre "**EDM**", the loop runs twice to make 2 rows to match the 2 songs.

4. Write The HTML Code In Between `<body> </body>`. ONLY write the code in BLACK and PURPLE and position it under the code in SKY BLUE.

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

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

<button id = "popSO" onclick = "show_pso()">
      Pop Songs Only
</button>


<button id = "allSongs" onclick = "show_all()">
      Show All
</button>

<button id = "edmSO" onclick = "show_edm()">
      EDM Songs
</button>
```

Explanation

This code will look very familiar to you. We are using event and event handler (ie. action). The code in purple will be focused on. The event is "**onclick**" and this means a mouse click. The action is on the right side of the assignment operator. **The code below will be our example**.

```
<button id = "popSO" onclick = "show_pso()">
      Pop Songs Only
</button>
```

The event is "**onclick**" and the action is "**show_pso()**". This means that when we click on the word "**Pop Songs Only**", the action "**show_pso()**" is called. In our case, "**show_pso()**" is a JS function that will do the work.

Next, we write the JS functions.

5. Write The JS Code In Between `<script> </script>`. WRITE ALL OF IT!!! Color code is for explanation.

```
function show_pso()
{
      alert( "show_pso()" );

      // -- reset currPL and trackNum
      currPL   = [];
      trackNum = 0;

      // -- delete table
      document.getElementById( "playlistTb" ).remove();
      console.log( "deleted table" );

      make_table( "playlistTb" );

      // -- make node
      for( var i = 0; i < playlist.length; i++ )
      {

            if( playlist[i].genre == "Pop" )
            {
                  make_row( "songRow_", i, "playlistTb" );
                  make_a_clone( playlist[i].objId, playlist[i], "songRow_", i );
                  playlist[i].set_listener( i, playlist[i].extraInfo );
                  currPL.push(playlist[i]);

            }
      }

      // -- update highlight
      change_songs( trackNum, currPL );

}
```

Explanation – pso ➔ pop songs only

The code in **orange** was written in **chapter 1** (make a new TABLE). The code in **sky blue** was written in **chapter 2** (make a new row, TR). The code in **green** was written in **chapter 3**, make a new data cell, TD).

The "**if**" statement in **red** is the money maker **because it is our filter**. If we only want pop songs, then all we have to do is check the genre. Remember that genre is a **private attribute of SongProfile**. Since it is a **private attribute**, we must use the **member access operator,** which is the period, to access the private attribute.

We then use the comparison operator, which is the double equal sign (==), to check if the genre is "**Pop**". If **TRUE**, then we keep it. If **FALSE**, then we remove it.

6. Write The JS Code In Between `<script> </script>`. Color code is for explanation.

```
function show_all()
{
       alert( "show_all()" );

        // -- reset currPL and trackNum
        currPL   = [];
        trackNum = 0;

       // -- delete table
        document.getElementById( "playlistTb" ).remove();
       console.log( "deleted table" );

        // -- make table
        make_table( "playlistTb" );

        // -- make node
        for( var i = 0; i < playlist.length; i++ )
        {
              make_row( "songRow_", i, "playlistTb" );
              make_a_clone( playlist[i].objId, playlist[i], "songRow_", i );
              playlist[i].set_listener( i, playlist[i].extraInfo );

              currPL.push(playlist[i]);
       }

        // -- update highlight
        change_songs( trackNum, currPL );

}
```

Explanation

No "**if**" statement this time. Why? Because the function name is "**show_all()**". Since we want all of the songs, this means that no filter is needed and so no "**if**" statement is needed.

The code in **orange** was written in **chapter 1** (make a new TABLE). The code in **sky blue** was written in **chapter 2** (make a new row, TR). The code in **green** was written in **chapter 3**, make a new data cell, TD).

7. Write The JS Code In Between `<script> </script>`.

```
function show_edm()
{
      alert( "show_edm()" );

      // -- reset currPL and trackNum
      currPL   = [];
      trackNum = 0;

      // -- delete table
      document.getElementById( "playlistTb" ).remove();
      console.log( "deleted table" );

      make_table( "playlistTb" );

      // -- make node
      for( var i = 0; i < playlist.length; i++ )
      {

            if( playlist[i].genre == "EDM" )
             {
                  make_row( "songRow_", i, "playlistTb" );
                  make_a_clone( playlist[i].objId, playlist[i], "songRow_", i );
                  playlist[i].set_listener( i, playlist[i].extraInfo );

                  currPL.push(playlist[i]);
            }
      }

      // -- update highlight
      change_songs( trackNum, currPL );

}
```

Explanation – edm ➔ electronic dance music

The code above is a **near copy of Chapter 5** where we used an "**if**" statement as a filter. The only different is the genre. In Chapter 5, we checked if the genre was "**Pop**". **In the code above, we check if the genre is "EDM".**

The code in **orange** was written in **chapter 1** (make a new TABLE). The code in **sky blue** was written in **chapter 2** (make a new row, TR). The code in **green** was written in **chapter 3**, make a new data cell, TD).

8. Write The CSS Code In Between `<style> </style>`

```
body
{
        background-image: url(https://vuongducnguyen.com/images/rain.jpg );


}
```

Explanation

Here we are styling the body, which is the background of the entire website.

CSS Challenge

1. add background images to any element

```
        background-image: url(https://vuongducnguyen.com/images/2150264-1920x1080-
        artsfon.com-77312san-giorgio-maggiore-venice.jpg );

        background-image: url(https://vuongducnguyen.com/images/mHouse.jpg );

        background-image: url(https://vuongducnguyen.com/images/rain.jpg );

        background-image: url(https://vuongducnguyen.com/images/140218140144-dark-sky---big-
        bend-u-s--horizontal-large-gallery.jpeg );

        background-image: url(https://vuongducnguyen.com/images/ziemaRoad.jpg );
```

Continue to next page

2. Pick 3 colors and style your MP3 player using the 3 colors that you selected by changing
    a. background-color
    b. color
    c. font-size
    d. border

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

	width     : 50px;
	height    : 50px;

	/*background-color: #909090;*/
	/*color     : #d0d0d0;*/
	text-shadow: 0px 1px 1px #808080;
	font-weight: bold;

}

#extraInfo
{
	position  : absolute;

	padding   : 0px 0px 0px 0px;
	margin    : 0px 0px 0px 0px;

	width     : 250px;
	height    : 150px;

	background-color: #00ffbb;
	box-shadow: 7px 5px 2px #f0f0f0;
	border    : 1px solid #a0a0a0;

	text-shadow: 0px 1px 1px #a0a0a0;
	font-weight: bold;
	color      : #ffa500;
	font-family: Arial, Tahoma, Serif;
}

button
{
	padding   : 5px;
	margin    : 0px 0px 0px 0px;

	border    : 1px solid #a0a0a0;
	background-color: #00ffbb;
	font-size : 17px;
	color     : #ffa500;

	text-shadow: 0px 1px 1px #b0b0b0;
	font-weight: bold;

}
```

Continue to next page

JS Challenge – you already have the filter …

Change the background color based on the genre that the user selected.

If the genre is “Pop”, then do the following
1. Write JS code to link up with the HTML element whose id is “**bTag**”
2. change the **backgroundColor** to “**#00f6fc**”;

If the genre is “EDM”, then do the following
1. Write JS code to link up with the HTML element whose id is “**bTag**”
2. change the **backgroundColor** to “**#006500**”;

If the user click on the button “Show All”, then do the following
1. write JS code to link up with the HTML element whose id is “**bTag**”
2. change the **backgroundColor** to “**#c0c0c0**”