Date:06.05.2022  **PRACTICAL No.7** Roll No.CO20363

**AIM – Music Box using Javascript**

**1)HTML CODE**

<!DOCTYPE *html*>

<html>

<head>

  <meta *charset*="utf-8">

  <meta *name*="viewport" *content*="width=device-width, initial-scale=1, user-scalable=no">

  <title>CS193X Music Box</title>

  <link *href*="https://fonts.googleapis.com/css?family=Open+Sans" *rel*="stylesheet">

  <link *rel*="stylesheet" *href*="style.css" />

  <link *rel*="stylesheet" *href*="musicbox-style.css" />

  <link *rel*="icon" *type*="image/png" *href*="images/favicon-32x32.png" *sizes*="32x32" />

  <link *rel*="icon" *type*="image/png" *href*="images/favicon-16x16.png" *sizes*="16x16" />

  <script *src*="./lib/dancer.min.js"></script>

  <script *src*="./lib/fetch.js"></script>

  <script *src*="audio-player.js" *defer*></script>

  <!--*TODO(you): Add any additional scripts here!*-->

  <script *src*="menu-screen.js" *defer*></script>

  <script *src*="play-button.js" *defer*></script>

  <script *src*="gif-display.js" *defer*></script>

  <script *src*="music-screen.js" *defer*></script>

  <script *src*="app.js" *defer*></script>

  <script *src*="script.js" *defer*></script>

</head>

<body>

  <div *id*="menu">

    <h1>Music Box</h1>

    <fieldset>

      <form>

        <h2>1. Choose a song:</h2>

        <select *id*="song-selector"></select>

        <h2>2. Choose a theme:</h2>

        <input *type*="text" *id*="query-input" *placeholder*="Enter a word or phrase" />

        <div *id*="error" *class*="inactive">Not enough gifs for this theme. Please try another.</div>

        <input *type*="submit" *value*="Go" />

      </form>

    </fieldset>

  </div>

  <div *id*="audio-player" *class*="inactive">

    <div *class*="container">

      <div *class*="gif first show"></div>

      <div *class*="gif second"></div>

    </div>

    <footer *class*="footer">

      <div *class*="button pause"></div>

    </footer>

  </div>

</body>

</html>

**2)CSS Code**

#*audio-player* {

    display: flex;

    flex-direction: column;

    height: 100*vh*;

}

.*gif* {

    flex-grow: 1;

    background-size: cover;

    background-position: center center;

    background-repeat: no-repeat;

    font-size: 50*px*;

    color: red;

}

.*show* {

    z-index: 2;

}

.*container* {

    height: inherit;

    position: relative;

}

.*first* {

    position: absolute;

    top: 0;

    right: 0;

    bottom: 0;

    left: 0;

}

.*second* {

    position: absolute;

    top: 0;

    right: 0;

    bottom: 0;

    left: 0;

}

.*footer* {

    background-color: purple;

    color: white;

    height: 70*px*;

    display: flex;

    justify-content: center;

    align-items: center;

}

.*button* {

    width: 60*px*;

    height: 60*px*;

}

.*pause* {

    background-image: url(images/pause.png);

    background-size: cover;

}

.*play* {

    background-image: url(images/play.png);

    background-size: cover;

}

/\**\**

*\* This style is provided by the course staff. Please do not modify this file!*

*\* Please add your additional stying to  `musicbox-style.css` instead.*

*\**\*/

body {

  font-family: 'Open Sans', sans-serif;

  margin: 0;

}

#*menu* {

  display: flex;

  flex-direction: column;

  justify-content: center;

  align-items: center;

  height: 100*vh*;

  width: 100*vw*;

}

#*menu* {

  font-size: 1.5*em*;

  text-align: center;

}

#*menu* fieldset {

  border: 0;

  margin: 0;

  padding: 0;

  width: 90*%*;

  max-width: 700*px*;

}

#*menu* h1 {

  margin: 0;

}

#*menu* h2 {

  font-size: 1.25*em*;

  font-weight: normal;

  margin: 0.5*em* 0;

}

#*menu* input[*type*=submit] {

  margin-top: 1*em*;

}

#*menu* input, #*menu* select {

  border: 0;

  box-sizing: border-box;

  font-size: 24*px*;

  height: 2*em*;

}

#*menu* input[*type*=text], #*menu* select {

  width: 100*%*;

}

#*menu* input:*focus*, #*menu* select:*focus* {

  outline: none;

}

#*menu* input[*type*=submit] {

  background-color: #e0e0e0;

  border: 0;

  width: 100*%*;

  max-width: 180*px*;

}

#*menu* input[*type*=text]:*focus* {

  border-bottom: 2*px* solid black;

}

#*menu* input[*type*=text] {

  background-color: white;

  border-bottom: 2*px* solid #e2e2e2;

}

#*menu* select {

  background-color: #ddfff8;

}

#*menu* .*submit-button* {

  text-align: center;

}

.*inactive* {

  display: none !important;

}

**3)JS**

//*This class will represent the music visualizer as a whole, similar to the*

//*role that the `App` class played in HW3.*

//

//*See HW4 writeup for more hints and details.*

*class* App {

  constructor() {

    this.menu = new MenuScreen();

    this.music = new MusicScreen();

    this.showMusicScreen = this.showMusicScreen.bind(this);

    document.addEventListener('open-music-screen', this.showMusicScreen);

  }

    showMusicScreen(event) {

      this.music.show(event.detail);

  }

}

//*NOTE: We are expecting you to \*create\* an AudioPlayer, but we are \*not\**

//*expecting you to modify the contents of this file.*

*class* AudioPlayer {

  constructor() {

    this.\_onKickCallback = this.\_onKickCallback.bind(this);

    this.lastKickTime = -1;

    this.dancer = new Dancer();

    this.kick = this.dancer.createKick({

      onKick: this.\_onKickCallback

    });

    this.kick.on();

  }

  setSong(songUrl) {

    let audio = new Audio();

    audio.crossOrigin = 'anonymous';

    audio.loop = 'true';

    audio.src = songUrl;

    this.dancer.pause();

    this.dancer.load(audio);

  }

  play() {

    this.dancer.play();

    const nowTime = Date.now();

*if* (this.lastKickTime === -1) {

      this.lastKickTime = nowTime;

    }

  }

  pause() {

    this.dancer.pause();

  }

  setKickCallback(kickCallback) {

    this.kickCallback = kickCallback;

  }

  \_onKickCallback() {

*if* (!this.kickCallback) {

*return*;

    }

    const KICK\_THRESHOLD = 0.2;

    const nowTime = Date.now();

    const diff = (nowTime - this.lastKickTime) / 1000;

*if* (diff > KICK\_THRESHOLD) {

      this.lastKickTime = nowTime;

      this.kickCallback();

    }

  }

}

*class* GifDisplay {

  constructor() {

    this.onJsonReady = this.onJsonReady.bind(this);

    this.loadGIF = this.loadGIF.bind(this);

    this.arrUrls = [];

    this.arrGIF = [];

    this.topic;

    this.i = 0;

  }

  viewGif(topic) {

    this.topic = topic;

    this.query = encodeURIComponent(this.topic);

    this.gifPath = `https://api.giphy.com/v1/gifs/search?api\_key=yCb3a2vGu9hxzySF74WnYQ9PgHS1ufbV&q=${this.query}&limit=25&offset=0&rating=G&lang=en`;

*return* fetch(this.gifPath)

      .then(this.onResponse)

      .then(this.onJsonReady);

  }

  onResponse(response) {

*return* response.json();

  }

  onJsonReady(json) {

*if* (!json.data) {

*return*;

    }

*for* (let i = 0; i < json.data.length; i++) {

      const url = json.data[i].images.original.url;

      this.arrUrls.push(url);

    }

    this.renderGif();

  }

  renderGif() {

    const gifContainerFirst = document.querySelector('#audio-player .first');

    const gifContainerSecond = document.querySelector('#audio-player .second');

    gifContainerFirst.style.backgroundImage = `url(${this.arrUrls[0]})`;

    gifContainerSecond.style.backgroundImage = `url(${this.arrUrls[1]})`;

  }

  bufferGif() {

    const container = document.getElementsByClassName('gif');

    const gifShowed = document.querySelector('#audio-player .show');

    const randomGif = Math.floor(Math.random() \* this.arrGIF.length);

*for* (const gif of container) {

*if* (!gif.classList.contains('show')) {

        gif.classList.add('show');

      }

    }

    gifShowed.classList.remove('show');

    gifShowed.style.backgroundImage = `url(${this.arrGIF[randomGif].src})`;

  }

  loadGIF() {

*if* (this.i > 24) {

*return*

    }

    let gif = new Image();

    gif.src = this.arrUrls[this.i];

    this.arrGIF.push(gif);

    this.i++;

    gif.addEventListener("load", this.loadGIF);

  }

}

//*This class will represent the menu screen that you see when you first load*

//*the music visualizer.*

//

//*See HW4 writeup for more hints and details.*

*class* MenuScreen {

  constructor() {

    this.song = document.getElementById('song-selector');

    this.gif = document.getElementById('query-input');

    this.containerElement = document.getElementById('menu');

    this.nameSong = document.createElement('option');

    this.getListSong();

    this.getListTopics();

    this.onSubmit = this.onSubmit.bind(this);

    const form = document.querySelector('form');

    form.addEventListener('submit', this.onSubmit);

  }

  getListSong() {

    fetch('https://yayinternet.github.io/hw4-music/songs.json')

      .then((response) => {

*return* response.json();

      })

      .then((data) => {

*for* (const song in data) {

          this.nameSong.text = data[song].title;

          this.nameSong.value = data[song].songUrl;

          this.song.appendChild(this.nameSong.cloneNode(true));

        }

      });

  }

  getListTopics() {

    const topics = ['candy', 'charlie brown', 'computers', 'dance', 'donuts', 'hello kitty', 'flowers', 'nature', 'turtles', 'space'];

    const topic = topics[Math.floor(Math.random() \* 10)];

    this.gif.value = topic;

  }

  hide() {

    this.containerElement.classList.add('inactive');

  }

  onSubmit(event) {

    event.preventDefault();

    const gifAndSong = {

      gif: this.gif.value,

      song: this.song.value

    }

    document.dispatchEvent(new CustomEvent('open-music-screen', {detail: gifAndSong}));

  }

}

*class* MusicScreen {

  constructor() {

    this.gifDisplay = new GifDisplay();

    this.playButton = new PlayButton();

    this.audioPlayer;

    this.containerElement = document.getElementById('audio-player');

    this.showNewGif = this.showNewGif.bind(this);

    this.audioPlay = this.audioPlay.bind(this);

    this.audioPause = this.audioPause.bind(this);

    document.addEventListener('show-new-gif', this.showNewGif);

    document.addEventListener('audio-pause', this.audioPause);

    document.addEventListener('audio-play', this.audioPlay);

  }

  async show(gifAndSong) {

*try* {

*await* this.gifDisplay.viewGif(gifAndSong.gif);

*if* (this.gifDisplay.arrUrls.length < 3) {

*throw* new Error('Not enough gifs for this theme. Please try another')

      }

      this.gifDisplay.loadGIF();

      document.getElementById('menu').classList.add('inactive');

      this.startPlayer(gifAndSong.song);

    } *catch* (error) {

      console.error(error);

      document.getElementById('error').classList.remove('inactive');

    }

  }

  startPlayer(song) {

    this.audioPlayer = new AudioPlayer();

    this.containerElement.classList.remove('inactive');

    this.audioPlayer.setSong(song);

    this.audioPlayer.play();

    this.audioPlayer.setKickCallback(this.kick);

  }

  kick() {

    document.dispatchEvent(new CustomEvent('show-new-gif'));

  }

  showNewGif() {

    console.log(this.gifDisplay.arrGIF.length);

    this.gifDisplay.bufferGif();

  }

  audioPlay() {

    this.audioPlayer.play();

  }

  audioPause() {

    this.audioPlayer.pause();

  }

}

*class* PlayButton {

  constructor() {

    this.button = document.querySelector('.button');

    this.playStop = this.playStop.bind(this);

    this.button.addEventListener('click', this.playStop);

  }

  playStop() {

*if* (this.button.classList[1] === 'pause') {

      this.button.classList.remove('pause');

      this.button.classList.add('play');

      document.dispatchEvent(new CustomEvent('audio-pause'));

    } *else* *if* (this.button.classList[1] === 'play') {

      this.button.classList.remove('play');

      this.button.classList.add('pause');

      document.dispatchEvent(new CustomEvent('audio-play'));

    }

  }

}

//*NOTE: You may, but do not have to, modify this file, such as to create other*

//*classes or to call methods on `App`. You may not add any global state*

//*variables.*

const app = new App();