The purpose of this application is to be able to read/view ticket listings and descriptions on the ticketing website in real time. There is also a create button that you can click that will open fields for you to input information and then once you press the save button it will create the artist name, add a picture, and information about the concert. You can also click the update button in order to update the artist name and information about the concert. Finally if you wish to delete the listing you can click the delete button and it will remove the corresponding listing.

Create the Concert Set

The code below is used to iterate through each object in the array and create a h2 tag, img, and p tag. Then I appended the concert line to the wrapper so that it would addd the content. Finally, I hid the paragraph so that it would not show up until it was activated by a button.

const concertSet = () =>{

concerts.forEach((concert)=>{

let concertLine = $(`<div class = "card"><div class = "cardContent">

<h2> ${concert.artist}</h2>

<img src = "${concert.image}">

<p id= "about"> ${concert.about}</p>

</div>

</div>`);

console.log('task complete')

$(".wrapper").append(concertLine);

$('p').hide();

});

};

Read Operation

Once the user clicks the read button it will call the following function. This function starts with jQuery selecting the showInfo div that corresponds to the specified concert based on the index. The eq function is used to select element and a matched set of elements at the corresponding elements. The info.html displays the about information about the concert based on the index that is passed into the function.

const viewHandler = (index) => {

    let info = $(".showInfo:eq(" + index + ")");

    console.log("INFO", info);

    info.html(`${concerts[index].about}!!<button class = "close">X</button>`);

    console.log("sucess")

  // Add the details to the display area

    info.find(".close").on("click", () => {

      setTimeout(function () {

        // !!! Use .html to reset the content

        info.html("");

      }, 1000);

    });

  };

Create Operation

The create function activates when the button is pressed. The newArtist, newAbout, and newImage fields will pop up where the user can enter information into the specified input fields. After the user fills the input fields they can press the save button which will save the information inputted into a new object which will be added to the concerts array. Finally the code will display the information in the corresponding tags based on the new object that is created along with the crud buttons.

  const createConcert = (index) => {

    $("button.add").prop("disabled", true);

    let newArtist = $(`<input type = "text" id = "newArtist" size = "30" value = "Artist" />`)

    let newAbout = $(`<input type = "text" id = "newAbout" size = "60" value = "About" />`)

    let newImage = $(`<input type = "text" id = "newImage" size = "40" value = "Image" />

      .<button class="saveNew">Save</button>`)

    $(".add").after(newImage);

    $(".add").after(newAbout);

    $(".add").after(newArtist);

    $(".saveNew").on("click", function () {

    let newConcert = {

      artist: $("input#newArtist").val(),

      about: $("input#newAbout").val(),

      image: $("input#newImage").val()

    }

    concerts.push(newConcert);

    console.log(concerts)

    let area = $(`.concerts .${panel\_class}`);

    let concertLine = $(`<div class="card">

      <div class="cardContent">

         <h2>${newConcert.artist}</h2>

         <p class="about">${newConcert.about}</p>

         <img src="${newConcert.image}">

         <div class = "showInfo"></div>

         <div class = "crud\_buttons"><button class="view">Read</button><button class="update">Update</button><button class="delete">Delete</button></div>`);

         $(".wrapper").append(concertLine);

         $('.about').hide();

    })

  }

Remove Operation

When the user clicks the delete button the following function is executed. This function selects the .card class using jQuery and deletes the corresponding card base on the index. The remove() actually deletes it from the webpage where it is not visible. Using splice removes the current object from the array.

const removeHandler = (index) => {

    console.log("removing")

    let itemRemove = $(".card").eq(index);

    itemRemove.remove();

    concerts.splice(index,1);

    console.log(concerts);

  };

Update Operation

The update button is similar to the create button but does add two input fields in order to change the artist name and information about the concert. The value has the previous text inputted so that you do not have to retype everything if you just want to change certain words. When the save button is pressed it creates a object with the new inputs as ids and uses .val() to receive the input that was entered. The .find finds the tag(h2) and the .text method is used to set the content of the selected h2 tag at the specified index. The remove() removed the updateLine until the button is pressed again

  const updateHandler = (index) => {

    let concertUpdate = $(".showInfo").eq(index);

    console.log("update sucess")

    let updateLine = $(`

      <div class = "update">

      <input type = "text" id = "updateArtist" value= '${concerts[index].artist}' size = "25">

      <input type = "text" id = "updateAbout" value= '${concerts[index].about}' size = "40">

      <button class="saveUpdate">save</button></div>`);

   concertUpdate.html(updateLine);

      $(".saveUpdate").on("click", function (){

          concerts[index] = {

            artist: $("#updateArtist").val(),

            about: $("#updateAbout").val()

          };

          let updateCard = $(".card:eq(" + index + ")");

          updateCard.find("h2").text(concerts[index].artist);

          updateCard.find("p.about").text(concerts[index].about);

          updateLine.remove();

      });