let tips = [

    "Use the 'Create Performer' button to load the performers in your marching drill!",

    "If you have symmetry in a set you don't need to conceptualize as many movements",

    "You can see what set you are currently on at the top of the page",

    "You can use the 'Next Set', 'Prev Set', and 'Goto Set' buttons to navigate between sets",

    "Moving some performers around other stationary performers allows for them to focus on things like difficult musical sections",

    "Radial symmetry in a set allows for more complex and polished looking sets that are easy to make",

    "If you are writing drill to music, take note of how many counts into the piece repeat signs, key changes, and double bars are",

    "By having a small number of performers moving around other performers that switch periodically makes easy drill that looks complex",

    "If you are writing drill to music, look at who has difficult parts; letting let them stay stationary makes it easier to play music",

    "Below the tips are some sample drills you can load up and look at"

];

// Saunders' Code

// Initial load

let performerData = {};

// PERFORMER FUNCTIONS

$("document").ready(function() {

    // HELPER FUNCTIONS

    let rand = Math.floor(Math.random() \* tips.length);

    $("#nextTipButton").on("click", nextTip);

    function randomizeTip() {

        rand = Math.floor(Math.random() \* tips.length);

        $("#tipDisplay").text("Tip: " + tips[rand]);

    }

    function nextTip() {

        rand = (rand + 1) % tips.length;

        $("#tipDisplay").text("Tip: " + tips[rand]);

    }

    randomizeTip();

    function refreshSetDisplay() {

        $("#setNum").text(curSet + 1);

        $("#setCount").text(numSets);

    }

    // Sets the curSet and numSets variables to be displayed

    document.getElementById('setNum').textContent = curSet + 1;

    document.getElementById('setCount').textContent = numSets;

    // END HELPER FUNCTIONS

    // BUTTON ACTIONS

    $("#createButton").on("click", createPerformer);

    $("#playButton").on("click", playDrill);

    $("#stopButton").on("click", stopDrill);

    $("#saveButton").on("click", saveDrill);

    $("#loadButton").on("click", loadDrill);

    $("#clearButton").on("click", clearDrill);

    $("#createSetButton").on("click", createSet);

    $("#deleteSetButton").on("click", deleteSet);

    $("#prevSetButton").on("click", prevSet);

    $("#nextSetButton").on("click", nextSet);

    $("#gotoSetButton").on("click", gotoSet);

    // END BUTTON ACTIONS

    let curX = 0;

    let curY = 0;

    let pNum = 0; // The counter for redrawing the performers

    function drawPerformer(pNum, isNew) {

        if (isNew) {

            tempSets = [];

            for (i = 0; i < numSets; i++) {

                tempSets.push({ x: 0, y: 0 });

            }

            performerData[pNum] = {

                id: pNum,

                name: "",

                inst: "",

                sets: tempSets

            }

        }

        console.log("Create");

        $("#innerEditor").append("<div id=\"p-" + pNum + "\" class=\"animate\">" + pNum + "</div>")

        $("#p-" + pNum).draggable({

            opacity: 0.7, // Dim the performer when dragging

            cursorAt: { top: 0, left: 0 }, //Default

            appendTo: '#innerEditor',

            containment: '#innerEditor',

            scroll: true,

            // When releasing the performer, we want to record their location

            stop: function() {

                console.log(pNum)

                console.log(curSet);

                // Case: current set is out of bounds

                console.log("make the set");

                console.log(pNum);

                performerData[pNum].sets[curSet] = { x: curX, y: curY };

                console.log(performerData[pNum].sets);

            },

            drag: function(e, ui) {

                curX = ui.position.left;

                curY = ui.position.top;

                $("#xPos").text(curX);

                $("#yPos").text(curY);

                // Future functionality

                // if (e.ctrlKey) {

                //   curX = curX - curX % 5

                // }

                // if (e.altKey) {

                // }

            },

        });

    }

    // Creates one performer on the field. The performer can be dragged to any position on the field, and its position will be recorded for that set.

    function createPerformer() {

        drawPerformer(numPerformers, true);

        numPerformers++;

    }

    // Play function. Still buggy.

    function playDrill() {

        count = 0;

        curSet = 0;

        var elems = [];

        $("#stopButton").html("<i class='fa fa-stop w3-xxlarge'></i> <br><br> Stop");

        // Store number of sets in array.

        while (count < numSets - 1) {

            var elem = count;

            elems.push(elem);

            count++;

        }

        playDrillHelper(elems);

    }

    // Play helper function, using recursion.

    function playDrillHelper(elems) {

        pNum = 0;

        var elem = elems.shift(); // Remove and return first element of array.

        refreshSetDisplay();

        // Do not clear the last set animation.

        if (curSet < numSets - 1) {

            clearDisplay();

            curSet++;

        }

        // Gets the number of performer objects.

        var count = 0;

        for (let id in performerData) {

            count++;

        }

        let iterations = count;

        var deleteVar = []; // change name

        // Reference performerData for the current set

        for (let id in performerData) {

            // Get current performers coordinates and draw them.

            let thisX = performerData[id].sets[elem].x;

            let thisY = performerData[id].sets[elem].y;

            drawPerformer(pNum, false);

            pNum++;

            // Get the next sets coordinates and animate performers traversing to them.

            let thisX2 = performerData[id].sets[elem + 1].x;

            let thisY2 = performerData[id].sets[elem + 1].y;

            var div = $(".animate").last().css({ left: parseInt(thisX), top: parseInt(thisY) });

            //TODO stop button//////////////////////////////////////////////////////////////

            var bool = 0; // 0 = stop, 1 = continue

            deleteVar.push(div);

            $("#stopButton").click(function() {

                if (bool == 0) {

                    for (i = 0; i < iterations; i++) {

                        $(deleteVar[i]).stop();

                    }

                    bool = 1;

                    $("#stopButton").html("<i class='fa fa-stop w3-xxlarge'></i> <br><br> Resume");

                } else {

                    for (i = 0; i < iterations; i++) {

                        let thisX2 = performerData[i].sets[elem + 1].x; // next set

                        let thisY2 = performerData[i].sets[elem + 1].y;

                        /\*\*

                        //Adjusts time/speed of performers. TODO

                        let thisX = performerData[i].sets[elem].x; // current set

                        let thisY = performerData[i].sets[elem].y;

                        let thisX3 = deleteVar[i].position().left;

                        let thisY3 = deleteVar[i].position().top;

                        //distance formula

                        var a = Math.abs(thisX2 - thisX);

                        var b = Math.abs(thisY2 - thisY);

                        var dis1 = Math.sqrt(a \* a + b \* b);

                        var c = Math.abs(thisX2 - thisX3);

                        var d = Math.abs(thisY2 - thisY3);

                        var dis2 = Math.sqrt(c \* c + d \* d);

                        //get new speed

                        var speed = 0;

                        if (dis1 == 0 || dis2 == 0) {

                            speed = 1;

                        } else {

                            var speed = (dis2 / dis1) \* 5000;

                        }

                        \*/

                        deleteVar[i].animate({ top: thisY2, left: thisX2 }, 5000, "linear",

                            function() {

                                // Only perform recursion when last performer of the current set is animated.

                                if (!--iterations && curSet < numSets) {

                                    playDrillHelper(elems);

                                }

                            }

                        );

                    }

                    $("#stopButton").html("<i class='fa fa-stop w3-xxlarge'></i> <br><br> Pause");

                    bool = 0;

                }

            });

            //TODO stop button/////////////////////////////////////////////////////////

            // Recursion used in order to animate performers sequentially.

            div.animate({ top: thisY2, left: thisX2 }, 5000,

                function() {

                    // Only perform recursion when last performer of the current set is animated.

                    if (!--iterations && curSet < numSets) {

                        playDrillHelper(elems);

                    }

                }

            );

        }

    }

    function stopDrill() {

        // Stop function is currently in playDrillHelper method. Will move to stopDrill later.

        //alert("Playback functionality is under development. Check back later.");

    }

    function saveDrill() {

        // Deanna's Code

        // var name = prompt("What would you like to call your drill file", "drill");

        // var dataStr = "data:text/json;charset=utf-8," + encodeURIComponent(JSON.stringify(performerData));

        // var downloadAnchorNode = document.createElement('a');

        // downloadAnchorNode.setAttribute("href", dataStr);

        // downloadAnchorNode.setAttribute("download", name + ".json");

        // document.body.appendChild(downloadAnchorNode);

        // downloadAnchorNode.click();

        // downloadAnchorNode.remove();

        // Saunders' Code

        if (Object.keys(performerData).length == 0) {

            alert("No performer data has been recorded");

        } else {

            const copyToClipboard = str => {

                const el = document.createElement('textarea');

                el.value = JSON.stringify(performerData);

                el.setAttribute('readonly', '');

                el.style.position = 'absolute';

                el.style.left = '-9999px';

                document.body.appendChild(el);

                el.select();

                document.execCommand('copy');

                document.body.removeChild(el);

            };

            copyToClipboard();

            alert("Copied performer data successfully");

        }

    }

    function loadDrill() {

        try {

            let loadThis = prompt("Please enter the saved JSON string.\nNote: This will erase your current drill!", "");

            if (loadThis) {

                clearDrill();

                performerData = JSON.parse(loadThis);

                if (typeof(performerData[0]) != "undefined") {

                    numSets = performerData[0].sets.length;

                    curSet = 0;

                }

                refreshSetDisplay();

            }

        } catch (err) {

            alert(err);

            // alert("The JSON string was invalid");

        }

        redraw(0);

        //Deanna's code

        // performerData = {"0":{"id":0,"name":"Alice","inst":"","sets":[{"x":150,"y":100}]},

        // "1":{"id":1,"name":"Tom","inst":"","sets":[{"x":250,"y":100}]},

        // "2":{"id":2,"name":"Brett","inst":"","sets":[{"x":350,"y":100}]},

        // "3":{"id":3,"name":"Matt","inst":"","sets":[{"x":150,"y":150}]},

        // "4":{"id":4,"name":"Finne","inst":"","sets":[{"x":250,"y":150}]},

        // "5":{"id":5,"name":"George","inst":"","sets":[{"x":350,"y":150}]},

        // "6":{"id":6,"name":"Colin","inst":"","sets":[{"x":150,"y":350}]},

        // "7":{"id":7,"name":"Max","inst":"","sets":[{"x":250,"y":350}]},

        // "8":{"id":8,"name":"Diona","inst":"","sets":[{"x":350,"y":350}]},

        // "9":{"id":9,"name":"Erik","inst":"","sets":[{"x":150,"y":400}]},

        // "10":{"id":10,"name":"Jim","inst":"","sets":[{"x":250,"y":400}]},

        // "11":{"id":11,"name":"Josh","inst":"","sets":[{"x":350,"y":400}]}}

        // redraw();

    }

    function clearDrill() {

        clearDisplay();

        performerData = {};

        numPerformers = 0;

        curSet = 0;

        numSets = 1;

        refreshSetDisplay();

    }

    function clearDisplay() {

        $("#innerEditor").text("");

    }

    // END PERFORMER FUNCTIONS

    // SET FUNCTIONS

    function createSet() {

        const newSet = curSet + 1;

        for (const key of Object.keys(performerData)) {

            performerData[key].sets.splice(newSet, 0, Object.assign(performerData[key].sets[curSet]));

            // performerData[key].sets[newSet] = Object.assign(performerData[key].sets[curSet]);

        }

        numSets++;

        curSet++;

        refreshSetDisplay();

        // nextSet();

    }

    function deleteSet() {

        // Delete the set only if there exists more than 1 set

        if (numSets > 1) {

            // Delete the set "curSet"

            count = 0;

            for (const key of Object.keys(performerData)) {

                $("#p-" + count).remove();

                performerData[key].sets.splice(curSet, 1);

                count++;

            }

            numSets--;

            // And assure we don't go below set 1

            if (curSet > 0) {

                curSet--;

            }

            // Update the performers

            // for (let performer of Object.keys(performerData)) {

            //   console.log(performer);

            // }

            // Redraw the set

            redraw(curSet);

            refreshSetDisplay();

        }

        // Otherwise, clear the drill

        else {

            clearDrill();

        }

    }

    function prevSet() {

        // Ensure we can go to the previous set

        if (curSet > 0) {

            curSet--;

            refreshSetDisplay();

            clearDisplay();

            redraw(curSet);

        }

    }

    function nextSet() {

        // Ensure we can go to the next set

        if (curSet < numSets - 1) {

            curSet++;

            refreshSetDisplay();

            clearDisplay();

            redraw(curSet);

        }

    }

    function gotoSet() {

        var set = prompt("Please enter the set number you'd like to navigate to:", "");

        if (set == null) {

            return;

        } else if (set > numSets || set <= 0 || isNaN(set)) {

            window.alert("The set number you entered does not exist!");

        } else {

            curSet = set - 1;

            refreshSetDisplay();

            clearDisplay();

            redraw(curSet);

        }

    }

    // Redraw should be called when:

    // - a set is changed (next set, prev set, goto set)

    // - a drill is loaded

    function redraw(redrawnSet) {

        // clearDisplay() should be called OUTSIDE OF REDRAW

        pNum = 0;

        // Reference performerData for the current set

        for (let id in performerData) {

            let thisX = performerData[id].sets[redrawnSet].x;

            let thisY = performerData[id].sets[redrawnSet].y;

            drawPerformer(pNum, false);

            pNum++;

            $(".animate").last().css("left", parseInt(thisX));

            $(".animate").last().css("top", parseInt(thisY));

        }

        // Build all

    }

})