Sprint 2 Code Summary

Bryson Brandon

11/20/23

CSCI-4250-002

Kinser

Part 1: HTML Structure

```
htmlCopy code
<div class="col">
    <div class="row" id="taskdistance">
        <span>Distance</span>
    </div>
    <div class="row" id="taskbody">
        <span class="distance-info"></span>
    </div>
</div>
```

This is an HTML structure using Bootstrap classes. It creates a column (`col`) with two rows (`row`) inside. The first row has an ID of "taskdistance" and contains the text "Distance." The second row has an ID of "taskbody" and contains an empty `span` element with a class of "distance-info."

```
<script>
//Kept having issues with updateDistances function,
// needs to be defined outside of eventListener
function updateDistances(userCoords) {
taskItems.forEach(function (taskItem) {
var taskLat = parseFloat(taskItem.getAttribute('data-lat'));
var taskLon = parseFloat(taskItem.getAttribute('data-lon'));
var distanceInMeters = distanceToLocation(userCoords, taskLat, taskLon);
var distanceInFeet = metresToFeet(distanceInMeters);
```

```
            // Now can display the current distance from a task in feet next to the task itself
            var distanceElement = taskItem.querySelector('.distance-info');
            distanceElement.textContent = distanceInFeet.toFixed(2) + ' feet';
        });
```

```
    }
document.addEventListener('DOMContentLoaded', function () {
    var taskItems = document.querySelectorAll('#btnCreateTask');


    getLocationAsync(function (userCoords) {
        // Handle the obtained user coordinates
        console.log('User coordinates:', userCoords);

        // Call updateDistances with the obtained user coordinates
        updateDistances(userCoords);
    }, function (error) {
        console.error('Error getting user location:', error);
        // Handle error
    });


    taskItems.forEach(function (taskItem) {
        taskItem.addEventListener('click', function (event) {
            var taskLat = parseFloat(taskItem.getAttribute('data-lat'));
            var taskLon = parseFloat(taskItem.getAttribute('data-lon'));

            getLocationAsync(function (coords) {
                // Call updateDistances with user coordinates when available
                updateDistances(coords);

                var distance = distanceToLocation(coords, taskLat, taskLon);

                if (distance <= 50) {
                    // User is close enough to the task, allow them to answer
                    // Show the modal or perform other actions
                    console.log("User is close enough to the task.");
                    $('#TaskIdInput').val(taskItem.getAttribute('data-id'));
                    $('#HuntIdInput').val(taskItem.getAttribute('data-huntid'));
                    $('#TaskInput').text(taskItem.getAttribute('data-task'));
                    createTaskModal.show();
                } else {
                    // Inform the user that they are not close enough to the task
                    console.log('You are not close enough to access this task.');
                }
            }, function (error) {
                console.error('Error getting user location:', error);
                // Handle error, e.g., inform the user or retry
            });
        });
    });
});
</script>
```

Part 2: JavaScript with Location Handling

This script handles the logic related to task items and user location. It does the following:

- Defines the `updateDistances` function to update the displayed distances for each task item based on the user's coordinates.

- Listens for the DOMContentLoaded event to execute the script.
- Gets user location asynchronously and calls `updateDistances` with the obtained coordinates.
- Adds a click event listener to each task item, updating distances again based on the user's location and checking if the user is close enough to the task.
- If the user is within 50 feet of a task, it logs a message, sets values in a modal, and shows the modal. If not, it logs a message indicating the user is not close enough.
-

```
(function _homeIndexMain() {
    const createTaskModalDOM = document.querySelector("#createTaskModal");
    const createTaskModal = new bootstrap.Modal(createTaskModalDOM);
    const createTaskButton = document.querySelectorAll("#btnCreateTask");

    // Function to update distances in the task list
    function updateDistances(userCoords) {
        createTaskButton.forEach(function (item) {
            var taskLat = parseFloat($(item).data("lat"));
            var taskLon = parseFloat($(item).data("lon"));
            var distanceInMeters = distanceToLocation(userCoords, taskLat, taskLon);
            var distanceInFeet = metresToFeet(distanceInMeters);

            // Now can display the current distance from a task in feet next to the task itself
            var distanceElement = item.querySelector('.distance-info');
            distanceElement.textContent = distanceInFeet.toFixed(2) + ' feet';
        });
    }

    // Attach click event to each task button
    createTaskButton.forEach(item => {
        item.addEventListener("click", event => {
            var taskLat = parseFloat($(item).data("lat"));
            var taskLon = parseFloat($(item).data("lon"));

            getLocationAsync(function (coords) {
                var distance = distanceToLocation(coords, taskLat, taskLon);

                // Check if the user is within 50ft of the task
                if (distance <= 50) {
                    alert('You are within 50 feet of this task!')
                    // Set values in the modal
                    $('#TaskIdInput').val($(item).data("id"));
                    $('#HuntIdInput').val($(item).data("huntid"));
                    $('#TaskInput').text($(item).data("task"));

                    // Show the modal only if the task is incomplete
                    if ($('a[data-id="' + $(item).data("id") + '"] #status').text() == "Incomplete") {
                        createTaskModal.show();
                    }
                } else {
                    // Inform the user that they are not close enough to the task
                    alert('You are not close enough to access this task.');
```

```
                    console.log('You are not close enough to access this task.');
                }
            }, function (error) {
                console.error('Error getting user location:', error);
                // Handle error
            });
        });
    });

    // Call updateDistances with user coordinates here
    getLocationAsync(function (userCoords) {
        updateDistances(userCoords);
    }, function (error) {
        console.error('Error getting user location:', error);
        // Handle error, e.g., inform the user or retry
    });

})();

//event listener
document.addEventListener('DOMContentLoaded', function () {
    var taskItems = document.querySelectorAll('#btnCreateTask');


    taskItems.forEach(function (taskItem) {
        taskItem.addEventListener('click', function (event) {
            var distance = parseFloat(taskItem.getAttribute('data-distance'));

            if (distance <= 50) {
                // Allow user to access the question
                // Show the modal or perform other actions
                createTaskModal.show();
            } else {
                // Inform the user that they are not close enough to the task
                console.log('You are not close enough to access this task.');
            }
        });
    });
});
```

Part 3: JavaScript as an Immediately Invoked Function Expression (IIFE)

This part encapsulates the functionality related to the task creation modal and task buttons. Here's a breakdown:

- Defines an IIFE (Immediately Invoked Function Expression) named `_homeIndexMain`.
- Initializes variables for the modal and task buttons.
- Defines the `updateDistances` function similar to Part 2.
- Attaches a click event to each task button, checking if the user is within 50 feet of the task and showing a modal accordingly.

- Calls `updateDistances` with the user coordinates obtained asynchronously.
- Listens for the DOMContentLoaded event to execute the script.

Part 4: Additional Event Listener

This script adds another event listener for the DOMContentLoaded event. It selects task items and adds a click event listener to each. If the distance attribute of a task item is less than or equal to 50 feet, it shows a modal; otherwise, it logs a message indicating that the user is not close enough.