Bethe OPS Project

Developer Instructions: Sidebar

May 16, 2019

Table of Contents

1 Overview

This document describes how the sidebar works for the Bethe OPS system, and outlines the functionalities related to the sidebar and instructions on how to modify such functionalities and various aspects of the sidebar.

All sidebar-related functionalities are encapsulated within **./views/partials/sidebar.ejs**, **./public/js/sidebar.js**, and **./public/css/style.css**. **sidebar.ejs** contains EJS code that makes up the front-end of the sidebar. **sidebar.js** contains the functions that load the sidebar with the weekly calendar header and event posters, as well as the event profile section when the corresponding event poster in the sidebar is selected; functions that support dynamic updates made on the sidebar (e.g., scrolling through the weekly calendar, clicking on a poster and loading its event profile) are included as well. **./public/css/style.css** contains styling for the sidebar.

2 Required Technologies

The required technologies for the email notification system are:

- EJS
    - This tutorial is a helpful resource for understanding how to use EJS with Node.js applications.
- JavaScript
- jQuery
- CSS
    - Materialize CSS

3 How It Works

The sidebar has two technical parts to it: the front-end and the back-end. The front-end is implemented using EJS, which is a templating language that generates HTML markup with plain JavaScript, and CSS. The back-end encompasses the functionality of the sidebar and handles dynamic changes made related to the sidebar; the back-end is implemented using JavaScript and jQuery.

3.1 Front-End

The EJS code in **./views/partials/sidebar.ejs** that implements the sidebar consists of two parts: the calendar and the space in which event posters are displayed. **sidebar.ejs** mainly serves as a skeleton of the sidebar that is dynamically populated with data on the

back-end in **./public/js/sidebar.js**. That is, the dates displayed in the calendar change with each week, and the posters displayed below the calendar change with each week too and may change when as they are added or removed from the system.

Notice how there are elements that relate to the number of days in a week and appear to be indexed with values [1..7]. As implied, the sidebar displays events by week and displays events for each day of the week. Monday should be displayed first and is treated as day 1, whereas Sunday should be displayed last and is treated as day 7. Likewise, when inserting dates, Monday's date is treated as date 1 and Sunday's as date 7. Thus, when implementing features or functionalities related to the days of the week, keep in mind the days of the week are indexed with values [1..7] as explicitly shown in the ids of the relevant elements pictured below.

```
<div class="dates">
    <div class="date"><a href="#sidebar-event-date-1"><button class="date-button unselectedDate" id="date-1"></button></a></div>
    <div class="date"><a href="#sidebar-event-date-2"><button class="date-button unselectedDate" id="date-2"></button></a></div>
    <div class="date"><a href="#sidebar-event-date-3"><button class="date-button unselectedDate" id="date-3"></button></a></div>
    <div class="date"><a href="#sidebar-event-date-4"><button class="date-button unselectedDate" id="date-4"></button></a></div>
    <div class="date"><a href="#sidebar-event-date-5"><button class="date-button unselectedDate" id="date-5"></button></a></div>
    <div class="date"><a href="#sidebar-event-date-6"><button class="date-button unselectedDate" id="date-6"></button></a></div>
    <div class="date"><a href="#sidebar-event-date-7"><button class="date-button unselectedDate" id="date-7"></button></a></div>
</div>
```

Displaying the dates in the weekly calendar

```
<div class="sidebar-posters-container" id="sidebar-posters-1">
    <div class="sidebar-event-date">
        <a id="sidebar-event-date-1"></a>
    </div>
    <div class="sidebar-event-posters-container" id="sidebar-event-posters-container-1"></div>
</div>
<div class="sidebar-posters-container" id="sidebar-posters-2">
    <div class="sidebar-event-date">
        <a id="sidebar-event-date-2"></a>
    </div>
    <div class="sidebar-event-posters-container" id="sidebar-event-posters-container-2"></div>
</div>
<div class="sidebar-posters-container" id="sidebar-posters-3">
    <div class="sidebar-event-date">
        <a id="sidebar-event-date-3"></a>
    </div>
    <div class="sidebar-event-posters-container" id="sidebar-event-posters-container-3"></div>
</div>
<!-- And so on until 7... -->
```

Displaying the dates and respective event posters in the sidebar

3.2 Back-End

All sidebar-related functionalities in regards to loading information in the sidebar and information in the event profile section are contained in **sidebar.js**. The implementations

in **sidebar.js** support three general categories of functionalities: (i) loading and updating the calendar, (ii) loading event posters, and (iii) selecting an event and loading the event profile and data needed for the sign-up list and forms associated with the selected event.

3.2.1 Calendar

Functionalities for the calendar are defined first in **sidebar.js**. A set of global variables are defined at the top that are referenced by functions supporting the calendar. The calendar is initialized to display the current week's dates as well by calling the function, showCurrentWeek( ) (discussed later in this section).

```
today = new Date();
currentMonth = today.getMonth();
currentYear = today.getFullYear();
currentDate = today.getDate();
currentDay = today.getDay();

// what is currently displayed, not the actual current month/year/date
displayedMonth = currentMonth;
displayedYear = currentYear;
displayedDate = currentDate;

monthAndYear = document.getElementById('month-year');

months = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September', 'October', 'November', 'December'];
days = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday'];

datePos = {}; //Object holding the positions of anchors for sidebar scrolling

// insert dates of actual current week
showCurrentWeek();
```

Date Buttons

Basic jQuery code is used to implement color changes associated with clicking and hovering of the date buttons in the calendar.

```
/**
 * Changes the background color and text color of the currently selected date button to blue and white, respectively.
 */
$('.date-button').on('click', function() {
    // unselect previously selected date
    $('.selectedDate').removeClass('selectedDate').addClass('unselectedDate');
    // select new date
    $(this).addClass('selectedDate').removeClass('unselectedDate');
});

/**
 * Changes the background color and text color of the date button hovered over to blue and white, respectively.
 */
$('.date-button').hover(
    function() { // mousein
        // temporary change color of date button
        $(this).css('background-color', '#053a91');
        $(this).css('color', '#fff');
    },
    function () { // mouseout
        // revert back to normal
        $(this).removeAttr('style');
    }
);
```

In addition, when a user clicks on a date button, the calendar automatically jumps or scrolls to the date and its corresponding events in the event posters section below. A global Object called datePos stores the positions of the anchors associated with the HTML elements whose text content are dates in the sidebar. These anchors have the format “#sidebar-event-date-<i>” where i is some number in 1..7. These elements are depicted in the two figures included in section 3.1. Basic jQuery is used to implement the automatic scrolling given the anchor positions provided by datePos.

```
/**
 * Jumps to the corresponding date's event posters section in the sidebar when a date is clicked on in the calendar
 * header.
 * Sources:
 *  - https://www.w3schools.com/howto/howto_css_smooth_scroll.asp#section1
 */

$('.dates a').on('click', function(event) {
    // Check if positions need to be updated
    updateDatePos(0);

    // Prevent default anchor click behavior
    event.preventDefault();

    // Store hash
    var hash = this.hash;

    // Using jQuery's animate() method to add smooth page scroll
    // The optional number (800) specifies the number of milliseconds it takes to scroll to the specified area
    $('#sidebar').animate({
        scrollTop: (datePos[hash] - 135) + 'px'
    }, 800);
});

/**
 * Stores the positions of each date button in case the user scrolls before clicking a date button.
 */
$('#sidebar').on('scroll', function(event) {
    // Check if positions need to be updated
    updateDatePos($('#sidebar').scrollTop());
});
```

The function, updateDatePos( ), supports this scrolling by updating the positions of the anchors whenever the sidebar loads a new set of events or manual scrolling by the user occurs within the sidebar.

Chevrons

The calendar includes chevrons, one on the left and one on the right, to support scrolling through the calendar which in turn updates the week displayed. Images of chevron icons were used to “implement” the chevrons. The chevrons change color when clicked on. The chevrons are also hoverable and change on hover to indicate they are clickable. The functions used to support chevron functionality are changeChevron( ), blueChevron( ), and whiteChevron( ).

Displaying Weeks

As mentioned before, the function, showCurrentWeek( ), displays the actual current week. getNextDay( ) is used to calculate the date of each day of the current week. Similarly, showPreviousWeek( ) and getPreviousWeekDay( ) are used to calculate and display previous weeks when scrolling left in the calendar, and showNextWeek( ) and getNextWeekDay( ) are used to calculate and display future weeks when scrolling right in the calendar. Not only do these functions update the weeks in the calendar, but they also insert the text labels for the dates of the week in the event posters section.

3.2.2 Loading Event Posters

Functionality for loading the event posters is mainly embodied in the function, loadEventPosters( ). loadEventPosters( ) takes in four parameters: *events*, *attendees*, *userProfile*, and *eventLeaders*. *events*, *attendees*, *userProfile*, and *eventLeaders* are all passed into an event handler function, selectEventPoster( ), which is described later in this section. *events* is also used to load the event posters.

For loading event posters, the general implementation is:

1. Load event posters by day and date.
2. Empty the that contains the event posters for the specified day to load a "fresh" set of events for the specified date. Grab the date already inserted by the week displaying functions (refer to section 3.2.1- Displaying Weeks).

```
for (let i = 1; i <= 7; i++) {
    $('#sidebar-event-posters-container-' + i).empty();

    let date = document.getElementById('sidebar-event-date-' + i).textContent;
    let d = date.substr(date.indexOf(' ') + 1) + ', ' + displayedYear;
```

3. If there are events happening on acquired date, use jQuery to insert HTML code defining the structure of the event posters and corresponding text into the other sidebar code (defined in **sidebar.ejs**). Events are inserted in order of start time.
    a. The HTML element defined for the event poster includes an onclick handler set to the function, selectEventPoster( ), into which *events*, *attendees*, *userProfile*, and *eventLeaders* are passed as arguments.

```
if (d in events) {
    let eventsArr = [];
    for (var title in events[d]) {
        events[d][title].name = title;
        eventsArr.push(events[d][title]);
    }
    // sort events happening on the same date by start time
    eventsArr.sort((a, b) =>
        (timeToSeconds(a.startTime) < timeToSeconds(b.startTime))
        ? -1
        : ((timeToSeconds(a.startTime) > timeToSeconds(b.startTime)) ? 1 : 0)
    );

    let j = 1;
    eventsArr.forEach(function(event) {
        let eventTitle = event.name;
        let eventTime = event.startTime + ' - ' + event.endTime;
        let eventPoster = event.poster;

        $('#sidebar-event-posters-container-' + i).append(
            '<div class="sidebar-event-poster-text-container" tabindex="-1" onclick="selectEventPoster(this, events, attendees
            i +
            ', ' +
            j +
            ');"><div class="sidebar-poster-container"><img class="sidebar-poster" src="' +
            eventPoster +
            '" /></div><div class="sidebar-poster-text-container"><div class="sidebar-poster-event-title" id="sidebar-poster-e
            i +
            '-' +
            j +
            '">' +
```

loadEventPosters( ) also includes basic jQuery code that adds hover functionality to the event posters such that the backgrounds of the event posters darkens. Hovering also indicates to the user that the poster is clickable.

3.2.3 Selecting an Event Poster and Loading an Event Profile

A user selects an event poster by clicking it. Doing so loads the event's profile to the right of the sidebar. In addition to loading the event profile, functionality is included to load event data in the fields of the "Edit Event" form included on the event profile section for event leader users and admin users.

<u>Selecting and Loading an Event Profile</u>

selectEventPoster( ) encompasses the functionality for loading an event's profile and its attendees list (discussed later). It also inserts data into the "Edit Event" form by calling other functions, autoFillEditEvent( ) and fillEditEventLeaders( ) (discussed later).

For loading an event profile, the general implementation is:

1. Grab the date and title of the event selected from the sidebar. The event date and event title are necessary to index *events*, which is an Object containing data on all events currently in the system.

```
let date = document.getElementById('sidebar-event-date-' + i).textContent;
let d = date.substr(date.indexOf(' ') + 1) + ', ' + displayedYear;
document.getElementById('date').innerHTML = d;

let title = document.getElementById('sidebar-poster-event-title-' + i + '-' + j).textContent;
```

2. Replace the featured event initially displayed in the event profile section with the selected event. Insert the information for the event profile into the HTML elements that define the skeleton of the event profile.

```
if (d in events) {
    // load event profile
    document.getElementById('eventdate').innerHTML =
        '<span id="date">' +
        d +
        '</span>' +
        '<span style="color: #afafaf"> from </span>' +
        events[d][title]['startTime'] +
        ' - ' +
        events[d][title]['endTime'];
    document.getElementById('eventtitle').innerHTML = title;
    document.getElementById('eventlocation').innerHTML =
        events[d][title]['location'];
    document.getElementById('eventGRF').innerHTML =
        'GRF ' +
        events[d][title]['eventLeader']['name'] +
        ' (' +
        events[d][title]['eventLeader']['netID'] +
        '@cornell.edu)';
    document.getElementById('eventdescription').innerHTML =
        events[d][title]['description'];
    document.getElementById('sampleposter').src = events[d][title]['poster'];
    document.getElementById('eventdetails').innerHTML =
        '<p>' + events[d][title]['other'] + '</p>';
```

3. Insert the event's ID into hidden <input> elements of the appropriate forms. The event ID is only inserted into the "Edit Event", "Delete Event", and "Add Attendee" forms if the user is not a student, that is, the user is an event leader or admin whom have access to these forms. The event ID is needed for POST requests made when new data is submitted through the forms.

```
// insert the event ID into its appropriate forms
if (userProfile.type !== 0) {
    document.getElementById('editEventId').value = events[d][title]['id'];  // "Edit Event"
    document.getElementById('deleteEventId').value = events[d][title]['id'];  // "Delete Event"
    document.getElementById('addAttendeeEventId').value = events[d][title]['id'];  // "Add Attendee"
}
document.getElementById('eventId').value = events[d][title]['id'];  // "Sign Up"
document.getElementById('removeAttendeeEventId').value = events[d][title]['id'];  // Removal from the attendees list and waitlist
```

4. Disable the "Sign Up" button for the event if the user currently logged in already signed up for the event or the event date has already passed.

```
// disable signup button if student already signed up for selected event or the event date has passed
let eventAttendees = [];
attendees[title].forEach(function(i) {
    eventAttendees.push(i.netID);
})
let signUpButton = document.getElementById('signupbutton');
dateFormat = d + ' ' + convertTimeStringTo24Hours(events[d][title]['startTime']);
let eventDateFormatted = new Date(dateFormat);

if (eventAttendees.includes(userProfile.netID) || (today > eventDateFormatted)) {
    signUpButton.setAttribute('disabled', 'disabled');
} else {
    signUpButton.removeAttribute('disabled');
}
```

5. Load the attendees list and the waitlist for the event below the event profile section by calling loadEventAttendees( ) (discussed later).
6. Insert event data in the fields of the "Edit Event" form included on the event profile section for event leader users and admin users by calling autoFillEditEvent( ) and fillEditEventLeaders( ).

Loading the Attendees List of an Event

Functionality for loading and displaying the attendees list as well as the waitlist of an event is embodied in the function, loadEventAttendees( ). Each time loadEventAttendees( ) is called, as with the event posters and event profile, the section for the attendees list and waitlist is cleared first so a "fresh" list can be loaded. The sign-up list for an event can be made hidden; if that is the case, "The sign-up list for this event is hidden." is displayed.

Otherwise, if the user logged in is a student user, the attendees list is displayed with only the attendees' names and net IDs.

```
$('#dashboard-students').empty();

if (event.isHidden == 1) {
    $('#dashboard-students').append('<p style="color: #777777"><i>The sign-up list is hidden for this event.<i></p>');
```

A red "X" is only displayed by the student's name such that the student can only remove him/herself. An onclick handler called removeName( ) is attached to the red "X"; removeName( ), which is defined in **./public/js/init.js**, inserts the name and id of the student into the remove confirmation modal that pops up.

```
$('#dashboard-students').append(
    '<tr><td id="dRemove"><a style="color: red;" class="waves-effect waves-light modal-trigger" id="' +
    attendeeName +
    '" ' +
    'onclick="removeName(this, ' +
    attendees[i].id +
    ')"' +
    ' href="#removestudent"><i class="material-icons" style="display: inline-flex; vertical-align: top;">clear</i></a></td
    (i + 1) +
    '</td><td id="dName">' +
    attendeeName +
    '</td><td id="dNetid">(' +
    attendeeNetID +
    ')</td></tr>'
);
```

If the user is an event leader user or an admin user, a red “X” is displayed next to ALL attendees’ names. In addition, each attendee’s building or place of residence is displayed (refer to last figure of this subsection for loading the attendees list). The attendees list also always displays the number of spots on the event roster. That is, empty slots are displayed to indicate how many spots are remaining if there are any.

```
// set to empty strings if displaying an empty slot
let attendeeName = (i >= attendees.length) ? '' : attendees[i].name;
let attendeeNetID = (i >= attendees.length) ? '' : attendees[i].netID;
let attendeeBuilding = (i >= attendees.length) ? '' : attendees[i].building;
let html;
```

jQuery is used to insert HTML that defines the attendees list, as indicated in the figures above.

The implementation is the same for the waitlist. However, if the waitlist is empty, “Waitlist currently empty” is displayed.

```
// add waitlist
$('#dashboard-eventleader').append('<tr><td style="background-color: #DDEDFF; margin: 0; color: rgb(8, 61, 145);"><p class="col s10"

// check if there are any people on the waitlist; if so, display people on waitlist
if (attendees.length > event['maxCapacity']) {
    for (let j = event['maxCapacity']; j < attendees.length; j++) {
        $('#dashboard-eventleader').append(
            '<tr><td id="dRemove"><a style="color: red;" class="waves-effect waves-light modal-trigger" href="#removestudent" id="' +
            attendees[j].name +
            '" ' +
            'onclick="removeName(this, ' +
            attendees[j].id +
            ')"' +
            '><i class="material-icons" style="display: inline-flex; vertical-align: top;">clear</i></a></td><td id="dNumber">' +
            (j + 1) +
            '</td><td id="dName">' +
            attendees[j].name +
            '</td><td id="dNetid">(' +
            attendees[j].netID +
            ')</td><td id="dHome"><i class="material-icons" style="display: inline-flex; vertical-align: top;">home</i>' +
            attendees[j].building +
            '</td></tr>'
        );
    }
} else {
    $('#dashboard-eventleader').append('<tr><td style="color: #ccc; padding-left: 50px;"><i>Waitlist currently empty<i></td></tr>');
}
```

Inserting Data Into the “Edit Event” Form

Two functions are used to insert an event’s information into the “Edit Event” form when an event is selected: autoFillEditEvent( ) and fillEditEventLeaders( ). autoFillEditEvent( ) sets the value attribute of all the inputs corresponding to the fields in the “Edit Event” form except for the event leader field.

Since the event leader field’s value is selected from a dropdown, fillEditEventLeaders( ) is a separate function that specifically handles filling the dropdown for the event leader field with the names and net IDs of event leaders currently in the system. fillEditEventLeaders( ) first sets the selected option to the event leader currently chosen and then inserts the other event leaders as options. The dropdown is emptied first and then the HTML consisting of the options is appended.

```
let optionsHtml = '';
optionsHtml += '<option selected value="' + currEventLeader.name + ' (' + currEventLeader.netID + ')' + '">' + currEventLeader
$.each(options, function(key, value) {
    if (key !== currEventLeader.netID) {
        optionsHtml += '<option value="' + value + '">' + value + '</option>';
    }
});
```

Also notice the function call, formSelect( ). formSelect( ) is a jQuery function provided by Materialize CSS. According to the Materialize documentation for <select>, formSelect( ) must be called to initialize any <select> stylized by Materialize. Note that the case may be similar for other HTML elements stylized by Materialize that are used for the Bethe OPS platform.

```
$('.event-leader-select').empty();
$('.event-leader-select').append(optionsHtml);
$('.event-leader-select').formSelect();
```