CSCE 120: Learning To Code

Presenting Data I
Hacktivity 5.2

Introduction

Prior to engaging in this hacktivity, you should have completed all of the pre-class activities as outlined in this module. At the start of class, you will be randomly assigned a partner to work with on the entirety of this hacktivity as a *peer programming* activity. Your instructor will inform you of your partner for this class.

One of you will be the driver and the other will take on the navigator role. Recall that a driver is in charge of the keyboard and computer while the navigator is in charge of the handout and directing the activity. However, you are *both* responsible for contributing and discussing solutions. If you were a driver/navigator in the prior activity, switch roles for this activity.

You will use the same project from the previous Hacktivity. If you need to, you can re-download it from GitHub using the URL, `https://github.com/cbourke/LedgerApp`.

1 Building an App

Consider the ledger application depicted in Figure 1.

Recreate the webpage in Figure 1 so that it is fully functional. That is:

- The user should be able to enter data into the ledger
- The ledger should support five items and have a totals row
- When the user clicks the "Clear Form" button it should clear all the entries in all of the inputs
- When a user clicks the "Calculate" button it should sum the figures in the Debit and Credit columns and display a result at the bottom

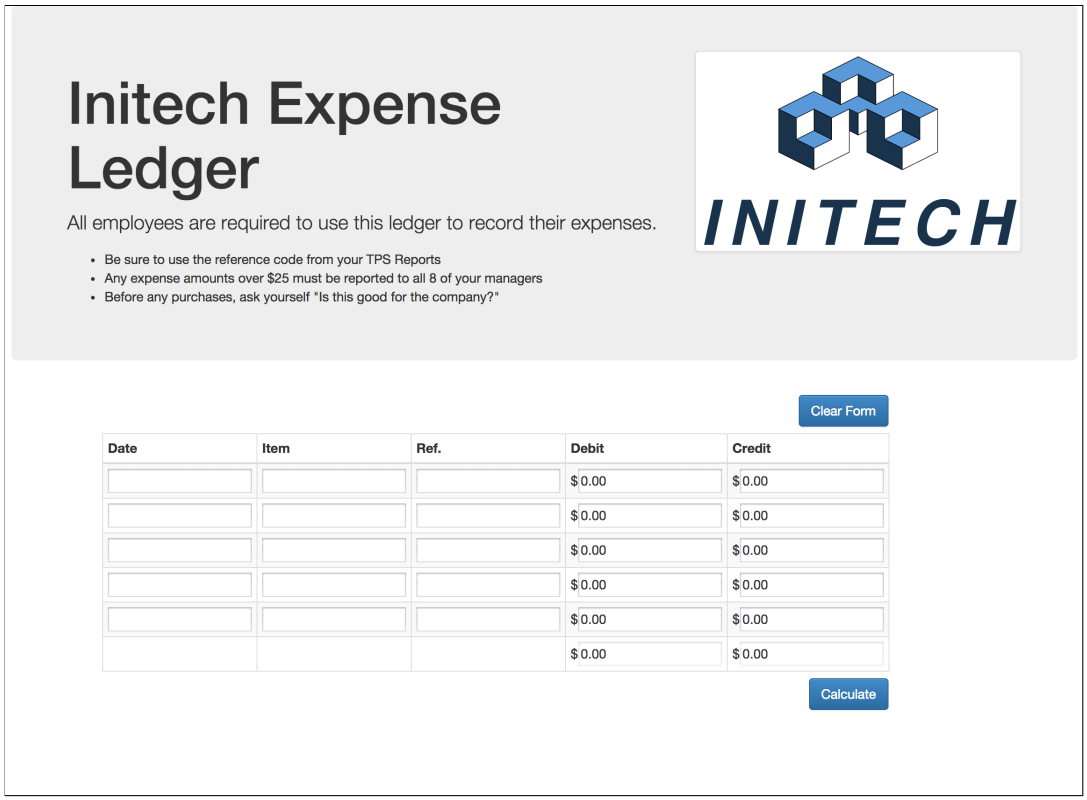

Figure 1: Initech Ledger App

We have provided an HTML file (`index.html`) to get you started. It includes code that brings in the jQuery and Bootstrap libraries. You should make your page look as close as possible to the page in the screen shot but it doesn't not need to look *exactly* like the example.

We have also provided the JavaScript code necessary to provide the functionality described above (see `ledger.js`). To make it work you need to do the following.

- Include the `ledger.js` script in your page using the `script` tag
- The credit and debit input boxes should have ID attributes: `debit01` through `debit05` and `credit01` through `credit05` respectively
- The totals input boxes should have ID attributes `totalCredit` and `totalDebit` respectively
- The two buttons should have an `onClick` attribute that references the relevant function in the JavaScript file (examine the source file to determine which functions should be called).

If you need help, try examining previous project code (such as the tax application or currency conversion application).

2 Error Handling

Negative values in our ledger should be considered invalid. If a credit is negative it is, in fact, a debit and vice versa. Furthermore, a user could enter a non-numeric value, leading to incorrect results. In this exercise, you will add functionality to check for these errors.

We have provided a function in the `ledger.js` source file to allow you to display an error message in the page. However, to make it work you will need to add a `<div>` element in your page with the ID `errMsgArea`.

2.1 Checking For Negative Values

Examine the code in the `computeResult()` function. The code provided checks for negative values for the first credit and debit fields. Discuss how the code works with your partner. Write additional code that raises an error for negative values entered in the other fields.

2.2 Checking For Non-Numeric Values

If a user enters non numeric values into one of the forms, the variables will have the value `NaN` (not a number). You can test if a variable is not a number by using the function `isNaN(a)` which returns true if the variable `a` is not a number and false otherwise.

Write additional code that raises an error for this type of input with a different, appropriate error message. Hint: instead of checking every individual value, you may try checking the total value since anything added to `NaN` value is still not a number. You may need to restructure some of the code so that it *does not* compute a total that would display "NaN" to the user.

3 Adding a Feature

In addition to totals for debits and credits we can compute a *grand total* which is the total credits less the total debits, which may be positive (surplus) or negative (deficit).

Add code to formulate a message indicating the grand total and whether or not it is a surplus or deficit (but report it as a positive number in either case). You can use the `displayGrandTotal()` function we've provided. You will need to add another `<div>` element with the ID `grandTotalDiv`.