# CPSC1520 – JavaScript 1 Assignment 3: Financial Transactions [10%]

## Introduction

This assignment builds on what you have already learned and demonstrated in class through the exercises and assessments to this point (i.e. up to and including DOM APIs). You are responsible for adding the JavaScript that will bring this page to life.

## Overview

For this assignment, you are required to implement the necessary JavaScript that will enable a user to enter a series of transactions (debits and credits). The page will validate form input (e.g. type must be either debit or credit) and will add the entered transaction to the transactions table. The page will also track the totals for each type of transaction and allow the user to delete transactions.

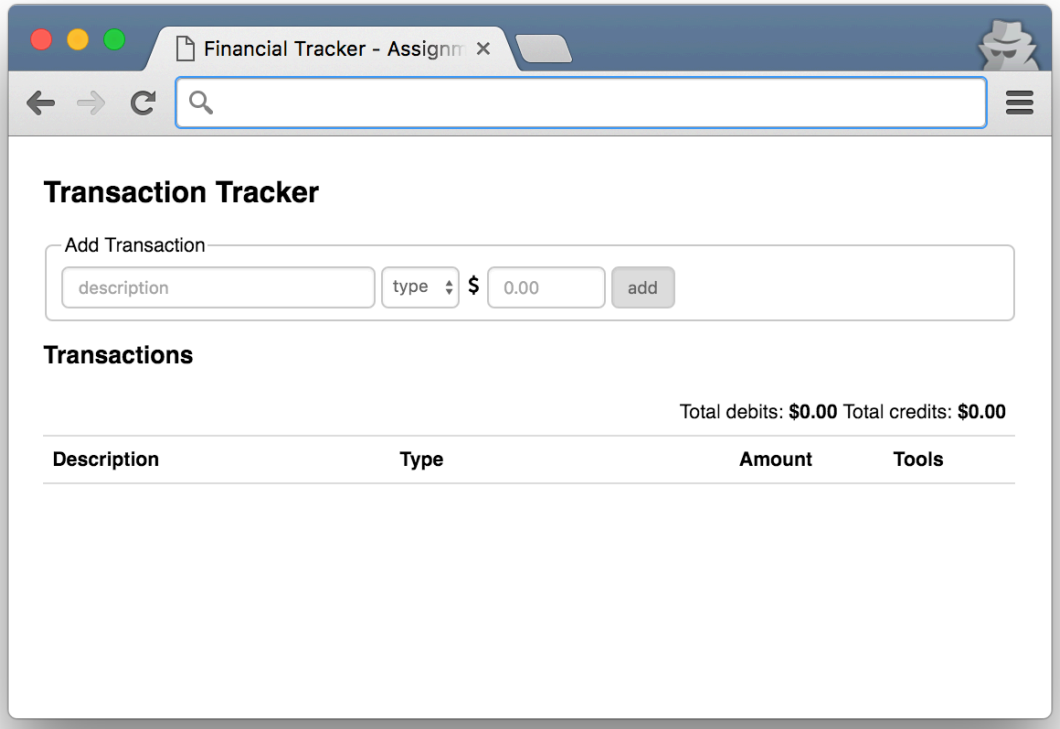


*Figure 1. Initial user interface for the assignment*

The form should only allow transactions that have had a valid type chosen (i.e. not type) and a positive numeric value in the amount field to be added (use the provided div.error element to provide error messages). Currency values should be displayed with a dollar sign and to two decimal places.

You will need to make use of the number type's **toFixed()** method, which you can read more about here[1].

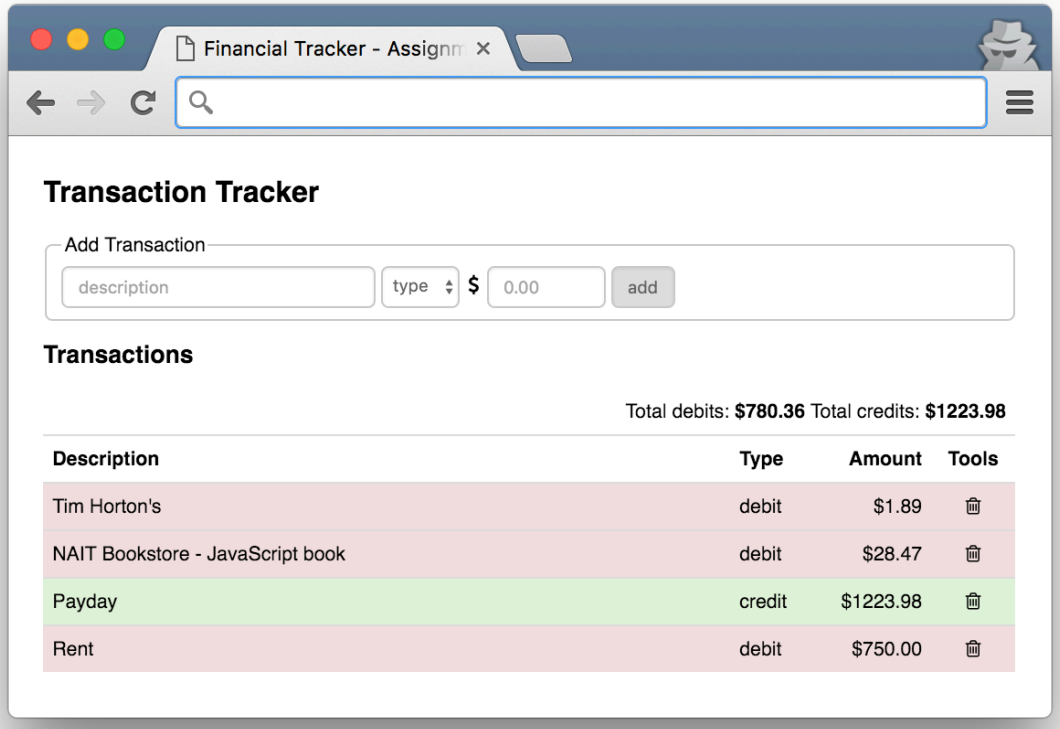


*Figure 2. Several transactions added*

The markup for the individual transactions (both credit and debit) is as shown:

```
▼<tr class="debit">
    <td>NAIT Bookstore - JavaScript book</td>
    <td>debit</td>
    <td class="amount">$28.47</td>
  ▼<td class="tools">
    ▶<i class="delete fa fa-trash-o">…</i>
    </td>
  </tr>
▼<tr class="credit">
    <td>Payday</td>
    <td>credit</td>
    <td class="amount">$1223.98</td>
  ▼<td class="tools">
    ▶<i class="delete fa fa-trash-o">…</i>
    </td>
  </tr>
```

*Figure 3. Markup for individual transactions*

## Required Tasks

The following is a list of requirements for this assignment:

- An event listener is required for handling submission of the transaction form
    - The page should not refresh
    - The user must choose a proper type from the select dropdown and the amount must be a positive number
        - If either of these conditions are not met, display an error message in the div.error element and do not add the transaction
    - Upon meeting the previous conditions:
        - DOM APIs must be used to construct the necessary document fragment to add to the DOM
            - See figure 3 above
        - With each addition, the totals must be correctly tracked and displayed
        - The form should be reset after each successful addition
- An event listener is required for handling clicking on the trash icon
    - The user should be presented with a confirm dialog prompt allowing them to either continue with or abort the deletion
    - If the user chooses to continue, remove the transaction from the table and correctly update and display the appropriate total
- A timer should be set so that if the user is inactive for over 2 minutes (think about how this can be determined) an alert dialog will be shown informing them of an immanent page refresh
    - When the user closes the alert window, refresh the page
        - Look into **window.location**[2] for how this may be accomplished
- ***Only DOM APIs are to be used to interact with the DOM in this assignment***
    - **do not use innerHTML**
- If you are unsure of any of the requirements, **be sure to ask your instructor for clarification**.

## References


1. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Number/toFixed
2. https://developer.mozilla.org/en-US/docs/Web/API/Window/location

# Marking Key

| Tasks | Grade | Marks | Total |
|---|---|---|---|
| **Form Event Listener**<br>• Event listener added and page does not refresh<br>• Fields are validated<br>• Transactions are added correctly<br>• Totals correctly updated and displayed | | <br>**1**<br>**1**<br>**5**<br>**3** | |
| **Trash Icon Event Listener**<br>• Event listener added<br>• Transaction row is removed from the DOM<br>• Totals correctly updated and displayed | | <br>**1**<br>**3**<br>**1** | |
| **Refresh Timer**<br>• Timer alerts after 2 minutes of inactivity<br>  ○ The page refreshes after the alert window is closed<br>• Timer is reset every time a transaction is added<br>  ○ Think of a more fluid method of accomplishing this | | **3**<br>**1**<br>**1**<br>**[1]** | |
| **Code Formatting and Style** | | **-3** | |

| Comments | SCORE | GRADE |
|---|---|---|
| | **20** | |

# Marking Rubric

| Marks | 5 Marks Criteria |
|---|---|
| **5** | **Task was completed with the highest of proficiency adhering to best practices and followed subject matter guidelines all tasks were completed to a professional standard.** |
| **4** | **Task was completed well some minor mistakes. Well above average work shows good understanding of the task and high degree of competence** |
| **3** | **Satisfactory work some features missing or incorrectly implemented. Show a moderate level of understanding in the task with room for improvement.** |
| **2** | **Below average work. Task was poorly complete. Show understanding of the task and the requirements to implement but implementation was poorly executed.** |
| **1** | **Some of the task was completed. Showed a lack of understanding in the subject matter and very poorly executed** |
| **0** | **Not completed.** |

| **Marks** | **3 Marks Criteria** |
|---|---|
| **3** | **Proficient shows a high degree of competence in completing task.** |
| **2** | **Capable above average degree of competence in completing task** |
| **1** | **Satisfactory shows a satisfactory degree of competence in completing task.** |
| **0** | **Shows a limited degree of competence in completing task.** |

| **Marks** | **1 Marks Criteria** |
|---|---|
| **1** | **Task Completed satisfactorily** |
| **0** | **Task was not** |