Documentation for Financial Transactions HTML Page

Jason N.

June 1, 2020

Contents

1 Setup **1**
- 1.1 Google API . . . 1
- 1.2 Firebase . . . 1
- 1.3 MySQL . . . 1

2 HTML **2**
- 2.1 Preamble and head . . . 2
  - 2.1.1 Preamble . . . 2
  - 2.1.2 meta charset . . . 2
  - 2.1.3 link rel=”stylesheet” . . . 2
  - 2.1.4 Scripts . . . 2
- 2.2 Inputs . . . 2
  - 2.2.1 Labels . . . 3
  - 2.2.2 Date . . . 3
  - 2.2.3 Text . . . 3
  - 2.2.4 List . . . 3
  - 2.2.5 File . . . 3
  - 2.2.6 Buttons . . . 4
- 2.3 Filters . . . 4
  - 2.3.1 Tooltip . . . 4
  - 2.3.2 Checkbox . . . 4
- 2.4 Options . . . 4
- 2.5 Table . . . 4
- 2.6 frozenColumns . . . 4
- 2.7 sort buttons . . . 5
  - 2.7.1 tbody . . . 5
- 2.8 Firebase scripts . . . 5

3 Main Javascript **8**
- 3.1 formattedStringToNumber() . . . 8
- 3.2 numberToFormattedString() . . . 8
- 3.3 getData() . . . 8
- 3.4 validate() . . . 8
  - 3.4.1 Check empty . . . 8
  - 3.4.2 Check NaN . . . 8
  - 3.4.3 Check date . . . 8
- 3.5 generateId() . . . 8
- 3.6 calculateCostBasis() . . . 8
- 3.7 addTransaction() . . . 8
- 3.8 fileIdGenerator() . . . 8
- 3.9 addTransactionButton() . . . 8
- 3.10 addTransactionWithFileName() . . . 8
- 3.11 deleteRow() . . . 8
- 3.12 editRow() . . . 8
- 3.13 saveChanges() . . . 8
- 3.14 discardChanges() . . . 8
- 3.15 sortTable() . . . 8
- 3.16 resetDate() . . . 8
- 3.17 validateFilters() . . . 8
  - 3.17.1 always true . . . 8
  - 3.17.2 date range . . . 8
  - 3.17.3 amount range . . . 8

3.17.4 generic ranges . . . 8
3.18 stringFilter() . . . 8
3.19 applyFilter() . . . 8
3.20 clearFilter() . . . 8
3.21 unfilterAll() . . . 8
3.22 toggleID() . . . 8
3.23 readFile() . . . 8
3.24 saveFile() . . . 8
3.25 applyTypes() . . . 8
3.26 editTypes() . . . 8
3.27 toggleSection() . . . 8
3.28 loadDataLists() . . . 8
3.29 window.onload = function() . . . 8

4 firebaseScript.js **9**
4.1 clearFirebase() . . . 9
4.2 writeToFirebase() . . . 9
4.3 readFromFirebase() . . . 9
4.4 clearFirestore() . . . 9
4.5 writeToFirestore() . . . 9
4.6 readFromFirestore() . . . 9

5 googleApiScript.js **10**
5.1 Global Variables . . . 11
5.2 loadSheetData() . . . 11
5.3 getNewSheetData() . . . 11
5.4 populateSheetSelector() . . . 11
5.5 getNewTabData() . . . 11
5.6 populateTabSelector(arrayOfTabs) . . . 11
5.7 getAllUserSheets() . . . 11
5.8 getTabsOfSheet() . . . 11
5.9 tableToArrays() . . . 11
5.10 arraysToTable() . . . 11
5.11 authenticate() . . . 11
5.12 loadClientSheets() . . . 11
5.13 loadClient() . . . 11
5.14 readGoogleSheetDB() . . . 11
5.15 readGoogleTypes() . . . 11
5.16 readCurrentTypes() . . . 11
5.17 writeGoogleSheetDB() . . . 11
5.18 setGoogleRows() . . . 11
5.19 clearGoogleRow() . . . 11
5.20 writeGoogleDB() . . . 11
5.21 gapi.load() . . . 11

6 imageFirestore.js **12**

7 localStorageScript.js **13**

8 mysqlScript.js **14**

9 CSS **15**
9.1 Vertical Scrolling Table . . . 15
9.2 Horizontal Scrolling on Overflow . . . 15
9.3 Miscellaneous . . . 15
9.3.1 Sort buttons . . . 15
9.3.2 Editing highlight . . . 15
9.3.3 Table borders . . . 15

1 Setup

1.1 Google API

1.2 Firebase

1.3 MySQL

2 HTML

2.1 Preamble and head

2.1.1 Preamble

Declares the document as HTML5.

```
1 <!DOCTYPE html>
```

2.1.2 meta charset

Specifies that characters in the file are encoded in UTF-8.

```
4 <meta charset = "UTF-8"/>
```

2.1.3 link rel=”stylesheet”

Imports the CSS file.

```
5 <link rel="stylesheet" type="text/css" href="./style.css"/>
```

2.1.4 Scripts

Imports the main Javascript file, responsible for the table and UI.

```
7 <script src="./script.js"></script>
```

Imports the Google API library.

```
8 <script src="https://apis.google.com/js/api.js"></script>
```

Imports other Javascript files, responsible for database management.

```
9  <script src="./googleApiScript.js"></script>
10 <script src="./mysqlScript.js"></script>
11 <script src="./localStorageScript.js"></script>
12 <script src="./imageFirestore.js"></script>
```

2.2 Inputs

Disables autocomplete which remembers past user input by default. `return false` specifies that no `POST` request should be made to the server.

```
20 <form onsubmit="return false" autocomplete="off">
```

2.2.1 Labels

Identifies the purpose of the field to the user, allows the user to select the field by clicking the label. This element is also used by accessibility tools to identify the field.

```
22 <label for="date">Date:</label><br/>
```

2.2.2 Date

The `date` input type is supported by most modern browsers and provides an intuitive UI for selecting dates. It also includes methods for converting or verifying the `Date` object.

```
13 <input id="date" name="date" type="date" placeholder="yyyy-mm-dd"/>
```

2.2.3 Text

The `text` input type allows the user to input a string. For numbers, this string has to be parsed in Javascript.

```
28 <input id="account" name="account" list="accountsList" type="text"
   ↪ placeholder="Account Number"/>
```

2.2.4 List

Lists are created using the `select` element, containing `option` elements. Each option has a `value` which is used in Javascript, and `innerText` which is seen by the user.

```
34 <label for="type">Transaction Type:</label><br/>
35 <select id="type" name="type">
36     <option value=""></option>
37     <option value="BUY">BUY</option>
38     <option value="SELL">SELL</option>
39     <option value="!DIVIDEND">DIVIDEND</option>
40     <option value="!INTEREST">INTEREST</option>
41     <option value="!WITHDRAW">WITHDRAW</option>
42     <option value="!DEPOSIT">DEPOSIT</option>
43 </select>
```

2.2.5 File

Files are uploaded using the `file` input type. The `multiple` attribute allows the user to upload multiple files, which are interpreted as an array of files in Javascript.

```
64     <label id="fileUploadLabel" for="fileUpload">Upload file</label>
65     <input id="fileUpload" name="fileUpload" type="file" onchange="
          ↪ fileUploadChanged();" multiple/>
```

2.2.6 Buttons

Buttons with the `submit` type can be used to check that all required sections are complete and highlight them in red. These buttons can also be used to send a `POST` request to a server if desired. The `onclick` attribute specified the function and parameters that should be executed when pressed.

```
70 <button id="add" type="submit" onclick="addTransactionButton();">Add
   ↪ Transaction</button>
```

2.3 Filters

Filter HTML elements are handled exactly the same as their counterparts in the input section. Some fields have two elements to handle a lower and upper bound, but these are handled solely in Javascript.

2.3.1 Tooltip

The `span` element is a generic container. The `title` attribute will display its value as a tool tip when the element is hovered.

```
118 <span title="Enter search terms here. Separate search terms with && or ||
    ↪ for AND and OR statements, respectively. Exclusive filters are
    ↪ marked by a leading !. Use || to filter by multiple securities (e.g.
    ↪  SPY || TLT) and && to exclude multiple securities (e.g. !SPY && !
    ↪ TLT).">?</span><br/>
```

2.3.2 Checkbox

The input type `checkbox` provides a toggleable input field which can be evaluated as `true` or `false` with Javascript.

```
153 <label for="filterNa">Filter N/A:</label>
154 <input id="filterNA" name="filterNA" type="checkbox"/>
```

2.4 Options

The options section uses buttons, text inputs, a file input, and drop down menus, which are decribed in the inputs section. The special handling of these elements is done in Javascript.

2.5 Table

2.6 frozenColumns

Cells in columns that are meant to be always visible are marked with a `frozenColumnx` class, where `x` is the column number. CSS is used to keep the column in place when scrolling.

```
<th class="frozenColumn1">
    <section>
        Transaction ID
    </section>
```

2.7 sort buttons

Sorting is done using buttons with an `onclick` attribute that calls a function `sortTable()`. The parameters passed are the column index and a boolean value indicating whether the column should be sorted in ascending or descending order.

```
<section class="sort">
    <button type="button" onclick="sortTable(0, true)">^</button>
    <button type="button" onclick="sortTable(0, false)">v</button>
</section>
```

2.7.1 tbody

```
<tbody id="tableBody">
</tbody>
```

2.8 Firebase scripts

These scripts are taken directly from the firebase documentation. They are required for firebase and its components to function. The `firebase-app.js` script is the main script and is required for all firebase features. The next three scripts are required for collecting analytics data, the realtime database, and firestore, respectively.

The configuration contains API keys and project information required to identify the app. The key is not secret, though it is unique to the project. As it is easily obtained by users of the app, it is strongly recommended to whitelist your domain in the project settings.

Unlike the other scripts, the firebase script is declared at the bottom, as it requires that the SDKs have loaded first.

```
<!-- The core Firebase JS SDK is always required and must be listed first
    ↪ -->
<script src="https://www.gstatic.com/firebasejs/7.14.2/firebase-app.js"></
    ↪ script>

<script src="https://www.gstatic.com/firebasejs/7.14.2/firebase-analytics.
    ↪ js"></script>
<script src="https://www.gstatic.com/firebasejs/7.14.2/firebase-database.
    ↪ js"></script>
<script src="https://www.gstatic.com/firebasejs/7.14.3/firebase-firestore.
    ↪ js"></script>

<script>
```

```
// Your web app's Firebase configuration
var firebaseConfig = {
    apiKey: "AIzaSyAmZLFZHDAB9evhvNunxOe5GxXRd_OizmU",
    authDomain: "financial-transactions-6f065.firebaseapp.com",
    databaseURL: "https://financial-transactions-6f065.firebaseio.com",
    projectId: "financial-transactions-6f065",
    storageBucket: "financial-transactions-6f065.appspot.com",
    messagingSenderId: "82206982479",
    appId: "1:82206982479:web:8937bbd1bd4fb6022b053a",
    measurementId: "G-0564DT8RNQ"
};
// Initialize Firebase
firebase.initializeApp(firebaseConfig);
firebase.analytics();

var database = firebase.database();
var firestore = firebase.firestore();
</script>

<script src="./firebaseScript.js"></script>
```

3 Main Javascript

3.1 formattedStringToNumber()

3.2 numberToFormattedString()

3.3 getData()

3.4 validate()

3.4.1 Check empty

3.4.2 Check NaN

3.4.3 Check date

3.5 generateId()

3.6 calculateCostBasis()

3.7 addTransaction()

3.8 fileIdGenerator()

3.9 addTransactionButton()

3.10 addTransactionWithFileName()

3.11 deleteRow()

3.12 editRow()

3.13 saveChanges()

3.14 discardChanges()

3.15 sortTable()

3.16 resetDate()

3.17 validateFilters()

3.17.1 always true

3.17.2 date range

3.17.3 amount range

4 firebaseScript.js

4.1 clearFirebase()

4.2 writeToFirebase()

4.3 readFromFirebase()

4.4 clearFirestore()

4.5 writeToFirestore()

4.6 readFromFirestore()

5 googleApiScript.js

IMPORTANT: googleApiScript.js is required for all databases to function, including firebase realtime database, firestore, Google sheets, and MySQL. This is due to the fact that it contains the `tableToArrays()`, `arraysToTable()`, and `readCurrentTypes()` functions, which are used by other database scripts. If you wish to use another database but have the Google sheets database removed, move these functions a different script or add a new script and include it in the HTML file.

5.1 Global Variables

5.2 loadSheetData()

5.3 getNewSheetData()

5.4 populateSheetSelector()

5.5 getNewTabData()

5.6 populateTabSelector(arrayOfTabs)

5.7 getAllUserSheets()

5.8 getTabsOfSheet()

5.9 tableToArrays()

5.10 arraysToTable()

5.11 authenticate()

5.12 loadClientSheets()

5.13 loadClient()

5.14 readGoogleSheetDB()

5.15 readGoogleTypes()

5.16 readCurrentTypes()

5.17 writeGoogleSheetDB()

5.18 setGoogleRows()

5.19 clearGoogleRow()

5.20 writeGoogleDB()

5.21 gapi.load()

6 imageFirestore.js

7 localStorageScript.js

8 mysqlScript.js

9 CSS

9.1 Vertical Scrolling Table

9.2 Horizontal Scrolling on Overflow

9.3 Miscellaneous

9.3.1 Sort buttons

9.3.2 Editing highlight

9.3.3 Table borders