Documentation for Financial Transactions HTML Page
Jason N.
April 26, 2020

Contents

1 Foreword

Some of the code samples in this document were copied by hand. If there are any discrepencies between code in this document and in the source files, refer to the source files.

This does not apply to the appendix. Code in the appendix was generated directly from the source files.

2 HTML

2.1 Preamble and head

This line declares that the document is an HTML5 document.

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

`<head>` tags are used to contain meta information about the document.

```
2  <head >
3      <meta charset = "UTF -8"/>
4      <link rel="stylesheet" type="text/css" href="./style.css"/>
5      <script src="./script.js"></script >
6  </head >
```

Within the `head` element:

- The first line defines the character set of the document.

- The second line defines the source of an external CSS document.

- The third line defines the source of an external Javascript document.

2.2 Inputs

The input section of this page is contained within `<article>` tags for the purpose of organisation. This can be used to facilitate styling this part of the page with CSS if desired.

```
10  <article id=" inputFields ">
```

The `article` element has been assigned a unique id for the purpose of styling. Specifically, this id is used to define padding and overflow. This is described in further detail in section 4.2 of this document.

All input fields and buttons are contained within `<form>` tags. Althought this is not strictly necessary for the purpose of this project, it is useful for organising data and specifying the fields from which data should be submitted.

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

The attribute `onsubmit` is used to define a Javascript function to be executed when pressed. The form expects that `true` is returned when data is successfully submitted. If so, the default behaviour is to clear

the fields and enter the data in the browser URL bar as arguments. To prevent this behaviour, `onsubmit` is set to `return false`.

The attribute `autocomplete` can be used to specify whether user input from a previous session should be used to populate input fields. This attribute also determines whether or not suggestions are displayed when the user enters data. In this case, `autocomplete` has been set to `off` to prevent these actions from occuring. This does not affect the functionality of the program.

The buttons and input fields within the `form` element are contained within `<section>` tags for organisation. This is primarily done to allow elements to be positioned properly by the CSS file.

2.2.1 Common attributes

All `input` elements in this `form` have been assigned a `name` attribute. The `name` attribute is not strictly relevant in this case, but is often used to identify the data when submitting to a database.

All `input` elements have the `required` attribute. Normally this prevents a `form` from being submitted unless all `required` fields contain data. This does not apply to our case as we have disabled the built-in submit function. However, it does still outline missing fields in red.

2.2.2 Labels

Each of the inputs are given a label to specify to a user the type of information which should be entered in the given field. This is done with the `input` element.

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

The `for` attribute is used to specify an element which corresponds to this label. This is done by setting the attribute to the id of the other element. Labels allow a user to select an input field by clicking the label rather than the field itself. Labels are also used to facilitate the use of assistive technologies.

2.2.3 Date

The date of a transaction is specified through the use of an `input` element with a `type` attribute of `date`. This can be used to effectively restrict the input to a valid date format and provides an intuitive method for inputting data.

```
11   <section>
12       <label for="date">Date:</label><br/>
13       <input id="date" name="date" type="date" required/>
14   </section>
```

This type of input field is also useful for interpreting dates in Javascript, as it provides methods which return the date in various formats to facilitate displaying and comparing dates.

2.2.4 Text

`input` elements with a `type` attribute of `text` can be used to retrieve a string from a user. This is also the field used for numbers, as these can be easily verified and converted in Javascript.

```
16  <section>
17      <label for="account">Account Number:</label><br/>
18      <input id="account" name="account" type="text" placeholder="Account
        ↪ Number" required/>
19  </section>
```

The advantage of taking numbers from an input field is that it allows for characters such as $ to be included. In the case of this project, users are able to submit Dollar Amounts as purely numberic values, or in a currency format. Currently, the program only accepts dollars as a currency, however, it is possible to allow and store any number of currencies. These characters, of course, have to be filtered out before the number is interpretted and re-inserted before displaying the value.

2.2.5 List

Dropdown lists are created using `<select>` tags containing `option` elements. Each `option` element represents a possible value, the first element is selected by default.

```
21  <section>
22      <label for="type">Transaction Type:</label><br/>
23      <select id="type" name="type">
24          <option value=""></option>
25          <option value="BUY">BUY</option>
26          <option value="SELL">SELL</option>
27          <option value="DIVIDEND">DIVIDEND</option>
28          <option value="INTEREST">INTEREST</option>
29          <option value="WITHDRAW">WITHDRAW</option>
30          <option value="DEPOSIT">DEPOSIT</option>
31      </select>
32  </section>
```

The `innerHTML` of an `option` element is the text that will be displayed to the user. The `value` attribute of the element is the value that will be read by Javascript. For this project, the `value` and `innerHTML` were made to be identical so that the text in the table would be the same as the text the user had seen in the list.

2.2.6 Buttons

`button` elements are clickable elements which can execute Javascript code specified by an `onclick` attribute. Text within the `innerHTML` of the `button` will be displayed as text within the button, which is useful for communicating the purpose of the button.

```
49  <section>
50      <button id="add" type="submit" onclick="addTransactionButton();">Add
        ↪ Transaction</button>
51      <button id="save" type="submit" hidden="true" onclick="saveChanges()
        ↪ ;">Save</button>
52      <button id="discard" type="button" hidden="true" onclick="
        ↪ discardChanges();">Discard</button>
53  </section>
```

In this case, three buttons are present, each set to execute a different Javascript function when clicked.

Two of the three buttons have a `type` attribute of `submit`. This causes each function to trigger the `submit` event along with the Javascript function. However, for this project, this event has been disabled by the `form onsubmit="return false"` attribute. Thus, the only difference is that this causes missing fields to be outlined in red when the button is pressed.

The last button is of `type button`. This element functions exactly the same, except it does not trigger the `submit` event. For this project, this means that missing fields will not be highlighted red, as this is not necessary for the 'Discard Changes' button.

Two of the three buttons also have the `hidden="true"` attribute. This causes the page to render as if these elements did not exist, as these elements are only relevant when editing a row. All three buttons are given unique ids so that `hidden` attributes can be added or removed as needed.

2.3 Table

2.3.1 thead

The header of the table is enclosed in `<thead>` tags. This element includes the first row of the table, denoted by `<tr>` tags, which contains headers for each column.

Every cell in the header is denoted by `<td>` tags. These cells differ from normal cells, such as those in the body of the table, in how they format their contents. Using this element for header cells makes them stand out slightly as well as making it easier to differentiate when styling with CSS.

```
61  <th>
62      <section>
63          Transaction ID
64      </section>
65      <section class="sort">
66          <button type="button" onclick="sortTable(0, true)">^</button>
67          <button type="button" onclick="sortTable(0, false)">V</button>
68      </section>
69  </th>
```

The first 8 header cells are split into two separate `section` elements. This was done to allow for the proper positioning of the header text and the sort buttons. For this reason, the latter `section` element is given the class `sort` to differentiate between the two.

Each of the first 8 header cells contain two buttons for sorting. All sorting buttons call the same function `sort(column, ascending)`, however, they pass different arguments to this function. The first argument is the column number, starting from 0, which allows the Javascript function to determine which column to use when comparing rows. The second argument defines whether data should be sorted in ascending or descending order.

The last header cell contains nothing but text. This column is used to contain the delete and edit buttons created for each row.

```
133  <th>Actions</th>
```

2.3.2 tbody

The table body is enclosed in `<tbody>` tags. This element is meant to be the main container of data in a table.

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

The table body is important for this project as it is the parent element of all data which will be manipulated. For this reason, it has been given a unique id to reference in Javascript. This was not strictly necessary, as it is also possible to reference this element by its tag name, being the only `tbody` element. Nevertheless, I consider this to be good practice as it is clear which element is being referred to in Javascript and allows for other tables to be added in the future if necessary without breaking the current functionality.

3 Javascript

3.1 getData()

This function is used to retrieve and format data from the input fields.

```
1  function getData() {
2      var date = document.getElementById("date");
3      var account = document.getElementById("account").value;
4      var type = document.getElementById("type").value;
5      var security = document.getElementById("security").value;
6      var amount = document.getElementById("amount").value;
7      var dAmount = document.getElementById("dAmount").value;
8
9      amount = Number(amount);
10
11     if(dAmount[0] == '$') {
12         dAmount = dAmount.substr(1);
13     }
14     dAmount = Number(dAmount);
15
16     if(validate(date, account, type, security, amount, dAmount)) {
17         var costBasis = calculateCostBasis(amount, dAmount);
18         date = date.value;
19         dAmount = '$' + dAmount.toFixed(2);
20
21         return [ date, account, type, security, amount, dAmount, costBasis
              ↪   ];
22     }
23     else return false;
24 }
```

The function checks whether the data is valid by calling the `validate()` function. If so, data is formatted and sent to the function which called `getData()`. Currently, the caller is either `addTransactionButton()` or `saveChanges()`.

The function first stores the raw values of

3.2 validate()

```
26  function validate(date, account, type, security, amount, dAmount) {
27      if(!validateDate(date)) return false;
28      if(!validateAccount(account)) return false;
29      if(!validateType(type)) return false;
30      if(!validateSecurity(security)) return false;
31      if(!validateAmount(amount)) return false;
32      if(!validateDAmount(dAmount)) return false;
33
34      return true;
35  }
```

3.3 generateId()

```
109  function generateId() {
110      var id = '';
111      var idLength = 6;
112
113      var characters = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789';
114      var charactersLength = characters.length;
115
116      var unique = false;
117
118      while(!unique) {
119          for(var i = 0; i < idLength; i++) {
120              id += characters.charAt(Math.floor(Math.random() *
                     ↪ charactersLength));
121          }
122
123          unique = true;
124          for(var i = 0; i < document.getElementsByClassName('idCell').
                 ↪ length; i++) {
125              if(document.getElementsByClassName('idCell')[i].innerText ==
                     ↪ id) {
126                  unique = false;
127                  break;
128              }
129          }
130      }
131      return id;
132  }
```

3.4 calculateCostBasis()

```
134  function calculateCostBasis(amount, dAmount) {
135      costBasis = '$' + (dAmount / amount).toFixed(2);
136      return costBasis;
137  }
```

3.5 addTransaction()

```
139  function addTransaction(id, date, account, type, security, amount, dAmount
       ↪ , costBasis) {
140      var tableBody = document.getElementById('tableBody');
141      var newRow = tableBody.insertRow(0);
142      newRow.classList += "bodyRow";
143
144      var actionsContent = "<button type='button' onclick='editRow(this)'>
             ↪ Edit</button> <button type='button' onclick='deleteRow(this)'>
             ↪ Delete</button>";
145      var rowContents = [id, date, account, type, security, amount, dAmount,
             ↪  costBasis, actionsContent];
```

```
146
147     for(var i = 0; i < rowContents.length; i++) {
148         var newCell = newRow.insertCell(i);
149         newCell.innerHTML = rowContents[i];
150         if(i == 0) {
151             newCell.classList += "idCell";
152         }
153     }
154 }
```

3.6   deleteRow()

```
173 function deleteRow(button) {
174     var row = button.parentElement.parentElement;
175     document.getElementById("tableBody").removeChild(row);
176
177     if(document.getElementsByClassName('editing').length == 0) {
178         document.getElementById('add').removeAttribute('hidden');
179         document.getElementById('save').setAttribute('hidden', true);
180         document.getElementById('discard').setAttribute('hidden', true);
181     }
182 }
```

3.7   editRow()

```
184 function editRow(button) {
185     if(document.getElementsByClassName('editing').length > 0)
186         document.getElementsByClassName('editing')[0].classList = "bodyRow
              ↪ ";
187
188     var row = button.parentElement.parentElement;
189     var rowContent = row.getElementsByTagName('td');
190     row.classList = "bodyRow editing";
191
192     document.getElementById('date').value = rowContent[1].innerText;
193     document.getElementById('account').value = rowContent[2].innerText;
194     document.getElementById('type').value = rowContent[3].innerText;
195     document.getElementById('security').value = rowContent[4].innerText;
196     document.getElementById('amount').value = rowContent[5].innerText;
197     document.getElementById('dAmount').value = rowContent[6].innerText;
198
199     document.getElementById('add').setAttribute('hidden', true);
200     document.getElementById('save').removeAttribute('hidden');
201     document.getElementById('discard').removeAttribute('hidden');
202 }
```

3.8   saveChanges()

8

```
204 | function saveChanges() {
205 |     data = getData();
206 |     if(data) {
207 |         rowToEdit = document.getElementsByClassName('editing')[0];
208 |         cellsToEdit = rowToEdit.getElementsByTagName('td');
209 |
210 |         for(var i = 0; i < data.length; i++) {
211 |             cellsToEdit[i + 1].innerHTML = data[i];
212 |         }
213 |         rowToEdit.classList = "bodyRow";
214 |     }
215 |
216 |     document.getElementById('add').removeAttribute('hidden');
217 |     document.getElementById('save').setAttribute('hidden', true);
218 |     document.getElementById('discard').setAttribute('hidden', true);
219 | }
```

3.9   discardChanges()

```
221 | function discardChanges() {
222 |     document.getElementsByClassName('editing')[0].classList = "bodyRow";
223 |
224 |     document.getElementById('add').removeAttribute('hidden');
225 |     document.getElementById('save').setAttribute('hidden', true);
226 |     document.getElementById('discard').setAttribute('hidden', true);
227 | }
```

3.10   sortTable()

```
229 | function sortTable(column, ascending) {
230 |     var tableBody = document.getElementById('tableBody');
231 |     var rows = document.getElementsByClassName('bodyRow');
232 |
233 |     var sorting = true;
234 |     while(sorting) {
235 |         sorting = false;
236 |         for(var i = 0; i < (rows.length - 1); i++) {
237 |             rowA = rows[i].getElementsByTagName('td')[column];
238 |             rowB = rows[i + 1].getElementsByTagName('td')[column];
239 |
240 |             var swap = false;
241 |
242 |             if(ascending && rowA.innerHTML.toLowerCase() > rowB.innerHTML.
       |                 ↪ toLowerCase()) swap = true;
243 |             else if(!ascending && rowA.innerHTML.toLowerCase() < rowB.
       |                 ↪ innerHTML.toLowerCase()) swap = true;
244 |
245 |             if(swap) {
246 |                 sorting = true;
```

```
247                        rows[i].parentNode.insertBefore(rows[i + 1], rows[i]);
248                    }
249                }
250            }
251    }
```

4 CSS

4.1 Vertical Scrolling Table

```
29  #table {
30      max-height: 80vh;
31      overflow: auto;
32  }
```

```
39  th {
40      min-width: 200px;
41      width: 10%;
42      position: sticky;
43      background: white;
44      top: 0;
45  }
```

4.2 Horizontal Scrolling on Overflow

```
5   #inputFields {
6       padding: 10px 0;
7       overflow-x: auto;
8   }
9
10  form {
11      min-width: 1900px;
12  }
```

4.3 Miscellaneous

4.3.1 Sort buttons

```
47  th > section {
48      width: 80%;
49      display: inline-block;
50      padding: 0;
51      margin: 0;
52  }
53
54  .sort {
55      width: 10%;
56  }
57
58  .sort > button {
59      padding: 0;
60      border: 0;
61      display: block;
62      width: 100%;
63  }
```

4.3.2 Editing highlight

```
65  .editing {
66      background-color: yellow;
67  }
```

4.3.3 Table borders

```
69  #table,
70  table,
71  td,
72  th {
73      box-shadow: 1px 1px black, inset 1px 1px black;
74  }
```

A HTML Source Code

```
 1  <!DOCTYPE html >
 2  <html >
 3      <head >
 4          <meta charset = "UTF -8"/>
 5          <link rel="stylesheet" type="text/css" href="./style.css"/>
 6          <script src="./script.js"></script >
 7      </head >
 8      <body >
 9          <article id="inputFields">
10              <form onsubmit="return false" autocomplete="off">
11                  <section >
12                      <label for="date">Date:</label ><br/>
13                      <input id="date" name="date" type="date" required/>
14                  </section >
15
16                  <section >
17                      <label for="account">Account Number:</label ><br/>
18                      <input id="account" name="account" type="text"
                          ↪ placeholder="Account Number" required/>
19                  </section >
20
21                  <section >
22                      <label for="type">Transaction Type:</label ><br/>
23                      <select id="type" name="type">
24                          <option value=""></option >
25                          <option value="BUY">BUY</option >
26                          <option value="SELL">SELL </option >
27                          <option value="DIVIDEND">DIVIDEND </option >
28                          <option value="INTEREST">INTEREST </option >
29                          <option value="WITHDRAW">WITHDRAW </option >
30                          <option value="DEPOSIT">DEPOSIT </option >
31                      </select >
32                  </section >
33
34                  <section >
35                      <label for="security">Security:</label ><br/>
36                      <input id="security" name="security" type="text"
                          ↪ placeholder="Security" required/>
37                  </section >
38
39                  <section >
40                      <label for="amount">Amount:</label ><br/>
41                      <input id="amount" name="amount" type="text"
                          ↪ placeholder="Unit Amount" required/>
42                  </section >
43
44                  <section >
45                      <label for="dAmount">$ Amount:</label ><br/>
46                      <input id="dAmount" name="dAmount" type="text"
                          ↪ placeholder="$ Amount" required/>
47                  </section >
48
```

```
49              <section>
50                  <button id="add" type="submit" onclick="
                        ↪ addTransactionButton();">Add Transaction</button
                        ↪ >
51                  <button id="save" type="submit" hidden="true" onclick
                        ↪ ="saveChanges();">Save</button>
52                  <button id="discard" type="button" hidden="true"
                        ↪ onclick="discardChanges();">Discard</button>
53              </section>
54          <form>
55      </article>
56
57      <article id="table">
58          <table>
59              <thead>
60                  <tr>
61                      <th>
62                          <section>
63                              Transaction ID
64                          </section>
65                          <section class="sort">
66                              <button type="button" onclick="sortTable
                                    ↪ (0, true)">^</button>
67                              <button type="button" onclick="sortTable
                                    ↪ (0, false)">v</button>
68                          </section>
69                      </th>
70                      <th>
71                          <section>
72                              Date
73                          </section>
74                          <section class="sort">
75                              <button type="button" onclick="sortTable
                                    ↪ (1, true)">^</button>
76                              <button type="button" onclick="sortTable
                                    ↪ (1, false)">v</button>
77                          </section>
78                      </th>
79                      <th>
80                          <section>
81                              Account Number
82                          </section>
83                          <section class="sort">
84                              <button type="button" onclick="sortTable
                                    ↪ (2, true)">^</button>
85                              <button type="button" onclick="sortTable
                                    ↪ (2, false)">v</button>
86                          </section>
87                      </th>
88                      <th>
89                          <section>
90                              Transaction Type
91                          </section>
92                          <section class="sort">
```

```
 93                                     <button type="button" onclick="sortTable
                                        ↪ (3, true)">^</button>
 94                                     <button type="button" onclick="sortTable
                                        ↪ (3, false)">v</button>
 95                                 </section>
 96                             </th>
 97                             <th>
 98                                 <section>
 99                                     Security
100                                 </section>
101                                 <section class="sort">
102                                     <button type="button" onclick="sortTable
                                        ↪ (4, true)">^</button>
103                                     <button type="button" onclick="sortTable
                                        ↪ (4, false)">v</button>
104                                 </section>
105                             </th>
106                             <th>
107                                 <section>
108                                     Amount
109                                 </section>
110                                 <section class="sort">
111                                     <button type="button" onclick="sortTable
                                        ↪ (5, true)">^</button>
112                                     <button type="button" onclick="sortTable
                                        ↪ (5, false)">v</button>
113                                 </section>
114                             </th>
115                             <th>
116                                 <section>
117                                     $ Amount
118                                 </section>
119                                 <section class="sort">
120                                     <button type="button" onclick="sortTable
                                        ↪ (6, true)">^</button>
121                                     <button type="button" onclick="sortTable
                                        ↪ (6, false)">v</button>
122                                 </section>
123                             </th>
124                             <th>
125                                 <section>
126                                     Cost Basis
127                                 </section>
128                                 <section class="sort">
129                                     <button type="button" onclick="sortTable
                                        ↪ (7, true)">^</button>
130                                     <button type="button" onclick="sortTable
                                        ↪ (7, false)">v</button>
131                                 </section>
132                             </th>
133                             <th>Actions</th>
134                         </tr>
135                     </thead>
136                     <tbody id="tableBody">
```

```
137              </tbody >
138           </table >
139        </article >
140     </body >
141 </html >
```

B Javascript Source Code

```
1   function getData () {
2       var date = document.getElementById("date");
3       var account = document.getElementById("account").value;
4       var type = document.getElementById("type").value;
5       var security = document.getElementById("security").value;
6       var amount = document.getElementById("amount").value;
7       var dAmount = document.getElementById("dAmount").value;
8
9       amount = Number(amount);
10
11      if(dAmount[0] == '$') {
12          dAmount = dAmount.substr(1);
13      }
14      dAmount = Number(dAmount);
15
16      if(validate(date, account, type, security, amount, dAmount)) {
17          var costBasis = calculateCostBasis(amount, dAmount);
18          date = date.value;
19          dAmount = '$' + dAmount.toFixed(2);
20
21          return [ date, account, type, security, amount, dAmount, costBasis
               ↪    ];
22      }
23      else return false;
24  }
25
26  function validate(date, account, type, security, amount, dAmount) {
27      if(!validateDate(date)) return false;
28      if(!validateAccount(account)) return false;
29      if(!validateType(type)) return false;
30      if(!validateSecurity(security)) return false;
31      if(!validateAmount(amount)) return false;
32      if(!validateDAmount(dAmount)) return false;
33
34      return true;
35  }
36
37  function validateDate(date) {
38      realDate = new Date();
39      inputDate = date.valueAsNumber;
40
41      if(isNaN(inputDate)) {
42          alert('Error: No date specified');
43          return false;
44      }
45
46      if(realDate.valueOf() < inputDate) {
47          alert('Error: Date is in the future');
48          return false;
49      }
50
51      return true;
```

```
52  }
53
54  function validateAccount(account) {
55      if(account == '') {
56          alert('Error: Missing Account Number');
57          return false;
58      }
59
60      return true;
61  }
62
63  function validateType(type) {
64      if(type == '') {
65          alert('Error: Missing Transaction Type');
66          return false;
67      }
68
69      return true;
70  }
71
72  function validateSecurity(security) {
73      if(security == '') {
74          alert('Error: Missing Security');
75          return false;
76      }
77
78      return true;
79  }
80
81  function validateAmount(amount) {
82      if(amount == '') {
83          alert('Error: Missing Amount');
84          return false;
85      }
86
87      if(isNaN(amount)) {
88          alert('Error: Invalid Amount');
89          return false;
90      }
91
92      return true;
93  }
94
95  function validateDAmount(dAmount) {
96      if(dAmount == '') {
97          alert('Error: Missing $ Amount');
98          return false;
99      }
100
101      if(isNaN(dAmount)) {
102          alert('Error: Invalid $ Amount');
103          return false;
104      }
105
```

```
106      return true;
107  }
108
109  function generateId() {
110      var id = '';
111      var idLength = 6;
112
113      var characters = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789';
114      var charactersLength = characters.length;
115
116      var unique = false;
117
118      while(!unique) {
119          for(var i = 0; i < idLength; i++) {
120              id += characters.charAt(Math.floor(Math.random() *
                     ↪ charactersLength));
121          }
122
123          unique = true;
124          for(var i = 0; i < document.getElementsByClassName('idCell').
                 ↪ length; i++) {
125              if(document.getElementsByClassName('idCell')[i].innerText ==
                     ↪ id) {
126                  unique = false;
127                  break;
128              }
129          }
130      }
131      return id;
132  }
133
134  function calculateCostBasis(amount, dAmount) {
135      costBasis = '$' + (dAmount / amount).toFixed(2);
136      return costBasis;
137  }
138
139  function addTransaction(id, date, account, type, security, amount, dAmount
     ↪ , costBasis) {
140      var tableBody = document.getElementById('tableBody');
141      var newRow = tableBody.insertRow(0);
142      newRow.classList += "bodyRow";
143
144      var actionsContent = "<button type='button' onclick='editRow(this)'>
             ↪ Edit</button> <button type='button' onclick='deleteRow(this)'>
             ↪ Delete</button>";
145      var rowContents = [id, date, account, type, security, amount, dAmount,
             ↪ costBasis, actionsContent];
146
147      for(var i = 0; i < rowContents.length; i++) {
148          var newCell = newRow.insertCell(i);
149          newCell.innerHTML = rowContents[i];
150          if(i == 0) {
151              newCell.classList += "idCell";
152          }
```

19

```
153        }
154   }
155
156   function addTransactionButton() {
157        var data = getData();
158        if(data) {
159            var date = data[0];
160            var account = data[1];
161            var type = data[2];
162            var security = data[3];
163            var amount = data[4];
164            var dAmount = data[5];
165            var costBasis = data[6];
166
167            var id = generateId();
168
169            addTransaction(id, date, account, type, security, amount, dAmount,
                 ↪    costBasis);
170        }
171   }
172
173   function deleteRow(button) {
174        var row = button.parentElement.parentElement;
175        document.getElementById("tableBody").removeChild(row);
176
177        if(document.getElementsByClassName('editing').length == 0) {
178            document.getElementById('add').removeAttribute('hidden');
179            document.getElementById('save').setAttribute('hidden', true);
180            document.getElementById('discard').setAttribute('hidden', true);
181        }
182   }
183
184   function editRow(button) {
185        if(document.getElementsByClassName('editing').length > 0)
186            document.getElementsByClassName('editing')[0].classList = "bodyRow
                 ↪    ";
187
188        var row = button.parentElement.parentElement;
189        var rowContent = row.getElementsByTagName('td');
190        row.classList = "bodyRow editing";
191
192        document.getElementById('date').value = rowContent[1].innerText;
193        document.getElementById('account').value = rowContent[2].innerText;
194        document.getElementById('type').value = rowContent[3].innerText;
195        document.getElementById('security').value = rowContent[4].innerText;
196        document.getElementById('amount').value = rowContent[5].innerText;
197        document.getElementById('dAmount').value = rowContent[6].innerText;
198
199        document.getElementById('add').setAttribute('hidden', true);
200        document.getElementById('save').removeAttribute('hidden');
201        document.getElementById('discard').removeAttribute('hidden');
202   }
203
204   function saveChanges() {
```

```
205      data = getData();
206      if(data) {
207          rowToEdit = document.getElementsByClassName('editing')[0];
208          cellsToEdit = rowToEdit.getElementsByTagName('td');
209
210          for(var i = 0; i < data.length; i++) {
211              cellsToEdit[i + 1].innerHTML = data[i];
212          }
213          rowToEdit.classList = "bodyRow";
214      }
215
216      document.getElementById('add').removeAttribute('hidden');
217      document.getElementById('save').setAttribute('hidden', true);
218      document.getElementById('discard').setAttribute('hidden', true);
219  }
220
221  function discardChanges() {
222      document.getElementsByClassName('editing')[0].classList = "bodyRow";
223
224      document.getElementById('add').removeAttribute('hidden');
225      document.getElementById('save').setAttribute('hidden', true);
226      document.getElementById('discard').setAttribute('hidden', true);
227  }
228
229  function sortTable(column, ascending) {
230      var tableBody = document.getElementById('tableBody');
231      var rows = document.getElementsByClassName('bodyRow');
232
233      var sorting = true;
234      while(sorting) {
235          sorting = false;
236          for(var i = 0; i < (rows.length - 1); i++) {
237              rowA = rows[i].getElementsByTagName('td')[column];
238              rowB = rows[i + 1].getElementsByTagName('td')[column];
239
240              var swap = false;
241
242              if(ascending && rowA.innerHTML.toLowerCase() > rowB.innerHTML.
                   ↪ toLowerCase()) swap = true;
243              else if(!ascending && rowA.innerHTML.toLowerCase() < rowB.
                   ↪ innerHTML.toLowerCase()) swap = true;
244
245              if(swap) {
246                  sorting = true;
247                  rows[i].parentNode.insertBefore(rows[i + 1], rows[i]);
248              }
249          }
250      }
251  }
```

C   CSS Source Code

```css
 1  body {
 2      font-size: 14px;
 3  }
 4
 5  #inputFields {
 6      padding: 10px 0;
 7      overflow-x: auto;
 8  }
 9
10  form {
11      min-width: 1900px;
12  }
13
14      form > section {
15          width: 14%;
16          display: inline-block;
17      }
18
19      input,
20      select {
21          min-width: 100px;
22          width: 80%;
23      }
24
25      button {
26          width: 40%;
27      }
28
29  #table {
30      max-height: 80vh;
31      overflow: auto;
32  }
33
34  table {
35      width: 100%;
36      margin: auto;
37      border-collapse: collapse;
38  }
39      th {
40          min-width: 200px;
41          width: 10%;
42          position: sticky;
43          background: white;
44          top: 0;
45      }
46
47      th > section {
48          width: 80%;
49          display: inline-block;
50          padding: 0;
51          margin: 0;
52      }
```

```css
53
54     .sort {
55         width: 10%;
56     }
57
58     .sort > button {
59         padding: 0;
60         border: 0;
61         display: block;
62         width: 100%;
63     }
64
65     .editing {
66         background-color: yellow;
67     }
68
69     #table,
70     table,
71     td,
72     th {
73         box-shadow: 1px 1px black, inset 1px 1px black;
74     }
```