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.

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

```
<section>
    <label for="type">Transaction Type:</label><br/>
    <select id="type" name="type">
        <option value=""></option>
        <option value="BUY">BUY</option>
        <option value="SELL">SELL</option>
        <option value="DIVIDEND">DIVIDEND</option>
        <option value="INTEREST">INTEREST</option>
        <option value="WITHDRAW">WITHDRAW</option>
        <option value="DEPOSIT">DEPOSIT</option>
    </select>
</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.

```
<section>
    <button id="add" type="submit" onclick="addTransactionButton();">Add
        ↪ Transaction</button>
    <button id="save" type="submit" hidden="true" onclick="saveChanges()
        ↪ ;">Save</button>
    <button id="discard" type="button" hidden="true" onclick="
        ↪ discardChanges();">Discard</button>
</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. 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 calling function is either `addTransactionButton()` or `saveChanges()`.

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

3.2 validate()

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

3.3 generateId()

```
108 function generateId() {
```

```
    var id = '';
    var idLength = 6;

    var characters = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789';
    var charactersLength = characters.length;

    var unique = false;

    while(!unique) {
        for(var i = 0; i < idLength; i++) {
            id += characters.charAt(Math.floor(Math.random() *
                ↪ charactersLength));
        }

        unique = true;
        for(var i = 0; i < document.getElementsByClassName('idCell').
            ↪ length; i++) {
            if(document.getElementsByClassName('idCell')[i].innerText ==
                ↪ id) {
                unique = false;
                break;
            }
        }
    }
    return id;
}
```

3.4 calculateCostBasis()

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

3.5 addTransaction()

```
function addTransaction(id, date, account, type, security, amount, dAmount
    ↪ , costBasis) {
    var tableBody = document.getElementById('tableBody');
    var newRow = tableBody.insertRow(0);
    newRow.classList += "bodyRow";

    var actionsContent = "<button type='button' onclick='editRow(this)'>
        ↪ Edit</button> <button type='button' onclick='deleteRow(this)'>
        ↪ Delete</button>";
    var rowContents = [id, date, account, type, security, amount, dAmount,
        ↪  costBasis, actionsContent];

    for(var i = 0; i < rowContents.length; i++) {
        var newCell = newRow.insertCell(i);
```

```
148         newCell.innerHTML = rowContents[i];
149         if(i == 0) {
150             newCell.classList += "idCell";
151         }
152     }
153 }
```

3.6 deleteRow()

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

3.7 editRow()

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

3.8 saveChanges()

```
203 function saveChanges() {
204     data = getData();
205     if(data) {
```

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

3.9 discardChanges()

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

3.10 sortTable()

```
228 function sortTable(column, ascending) {
229     var tableBody = document.getElementById('tableBody');
230     var rows = document.getElementsByClassName('bodyRow');
231 
232     var sorting = true;
233     while(sorting) {
234         sorting = false;
235         for(var i = 0; i < (rows.length - 1); i++) {
236             rowA = rows[i].getElementsByTagName('td')[column];
237             rowB = rows[i + 1].getElementsByTagName('td')[column];
238 
239             var swap = false;
240 
241             if(ascending && rowA.innerHTML.toLowerCase() > rowB.innerHTML.
                    ↪ toLowerCase()) swap = true;
242             else if(!ascending && rowA.innerHTML.toLowerCase() < rowB.
                    ↪ innerHTML.toLowerCase()) swap = true;
243 
244             if(swap) {
245                 sorting = true;
246                 rows[i].parentNode.insertBefore(rows[i + 1], rows[i]);
247             }
248         }
249     }
```

```
250 }
```

4 CSS

4.1 Vertical Scrolling Table

```
#table {
    max-height: 80vh;
    overflow: auto;
}
```

```
th {
    min-width: 200px;
    width: 10%;
    position: sticky;
    background: white;
    top: 0;
}
```

4.2 Horizontal Scrolling on Overflow

```
#inputFields {
    padding: 10px 0;
    overflow-x: auto;
}

form {
    min-width: 1900px;
}
```

4.3 Miscellaneous

4.3.1 Sort buttons

```
th > section {
    width: 80%;
    display: inline-block;
    padding: 0;
    margin: 0;
}

.sort {
    width: 10%;
}

.sort > button {
    padding: 0;
    border: 0;
    display: block;
    width: 100%;
}
```

4.3.2 Editing highlight

```
.editing {
    background-color: yellow;
}
```

4.3.3 Table borders

```
#table,
table,
td,
th {
    box-shadow: 1px 1px black, inset 1px 1px black;
}
```

A HTML Source Code

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset = "UTF-8"/>
        <link rel="stylesheet" type="text/css" href="./style.css"/>
        <script src="./script.js"></script>
    </head>
    <body>
        <article id="inputFields">
            <form onsubmit="return false" autocomplete="off">
                <section>
                    <label for="date">Date:</label><br/>
                    <input id="date" name="date" type="date" required/>
                </section>

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

                <section>
                    <label for="type">Transaction Type:</label><br/>
                    <select id="type" name="type">
                        <option value=""></option>
                        <option value="BUY">BUY</option>
                        <option value="SELL">SELL</option>
                        <option value="DIVIDEND">DIVIDEND</option>
                        <option value="INTEREST">INTEREST</option>
                        <option value="WITHDRAW">WITHDRAW</option>
                        <option value="DEPOSIT">DEPOSIT</option>
                    </select>
                </section>

                <section>
                    <label for="security">Security:</label><br/>
                    <input id="security" name="security" type="text"
                        ↪ placeholder="Security" required/>
                </section>

                <section>
                    <label for="amount">Amount:</label><br/>
                    <input id="amount" name="amount" type="text"
                        ↪ placeholder="Unit Amount" required/>
                </section>

                <section>
                    <label for="dAmount">$ Amount:</label><br/>
                    <input id="dAmount" name="dAmount" type="text"
                        ↪ placeholder="$ Amount" required/>
                </section>

```

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

        <article id="table">
            <table>
                <thead>
                    <tr>
                        <th>
                            <section>
                                Transaction ID
                            </section>
                            <section class="sort">
                                <button type="button" onclick="sortTable
                                    ↪ (0, true)">^</button>
                                <button type="button" onclick="sortTable
                                    ↪ (0, false)">v</button>
                            </section>
                        </th>
                        <th>
                            <section>
                                Date
                            </section>
                            <section class="sort">
                                <button type="button" onclick="sortTable
                                    ↪ (1, true)">^</button>
                                <button type="button" onclick="sortTable
                                    ↪ (1, false)">v</button>
                            </section>
                        </th>
                        <th>
                            <section>
                                Account Number
                            </section>
                            <section class="sort">
                                <button type="button" onclick="sortTable
                                    ↪ (2, true)">^</button>
                                <button type="button" onclick="sortTable
                                    ↪ (2, false)">v</button>
                            </section>
                        </th>
                        <th>
                            <section>
                                Transaction Type
                            </section>
                            <section class="sort">
```

```
                                <button type="button" onclick="sortTable
                                    ↪ (3, true)">^</button>
                                <button type="button" onclick="sortTable
                                    ↪ (3, false)">v</button>
                            </section>
                        </th>
                        <th>
                            <section>
                                Security
                            </section>
                            <section class="sort">
                                <button type="button" onclick="sortTable
                                    ↪ (4, true)">^</button>
                                <button type="button" onclick="sortTable
                                    ↪ (4, false)">v</button>
                            </section>
                        </th>
                        <th>
                            <section>
                                Amount
                            </section>
                            <section class="sort">
                                <button type="button" onclick="sortTable
                                    ↪ (5, true)">^</button>
                                <button type="button" onclick="sortTable
                                    ↪ (5, false)">v</button>
                            </section>
                        </th>
                        <th>
                            <section>
                                $ Amount
                            </section>
                            <section class="sort">
                                <button type="button" onclick="sortTable
                                    ↪ (6, true)">^</button>
                                <button type="button" onclick="sortTable
                                    ↪ (6, false)">v</button>
                            </section>
                        </th>
                        <th>
                            <section>
                                Cost Basis
                            </section>
                            <section class="sort">
                                <button type="button" onclick="sortTable
                                    ↪ (7, true)">^</button>
                                <button type="button" onclick="sortTable
                                    ↪ (7, false)">v</button>
                            </section>
                        </th>
                        <th>Actions</th>
                    </tr>
                </thead>
                <tbody id="tableBody">
```

```
                </tbody>
            </table>
        </article>
    </body>
</html>
```

B Javascript Source Code

```
function getData() {
    var date = document.getElementById("date").value;
    var account = document.getElementById("account").value;
    var type = document.getElementById("type").value;
    var security = document.getElementById("security").value;
    var amount = document.getElementById("amount").value;
    var dAmount = document.getElementById("dAmount").value;

    amount = Number(amount);

    if(dAmount[0] == '$') {
        dAmount = dAmount.substr(1);
    }
    dAmount = Number(dAmount);

    if(validate(date, account, type, security, amount, dAmount)) {
        var costBasis = calculateCostBasis(amount, dAmount);
        dAmount = '$' + dAmount.toFixed(2);

        return [ date, account, type, security, amount, dAmount, costBasis
            ↪ ];
    }
    else return false;
}

function validate(date, account, type, security, amount, dAmount) {
    if(!validateDate(date)) return false;
    if(!validateAccount(account)) return false;
    if(!validateType(type)) return false;
    if(!validateSecurity(security)) return false;
    if(!validateAmount(amount)) return false;
    if(!validateDAmount(dAmount)) return false;

    return true;
}

function validateDate(date) {
    realDate = new Date();
    inputDate = document.getElementById('date').valueAsNumber;

    if(isNaN(inputDate)) {
        alert('Error: No date specified');
        return false;
    }

    if(realDate.valueOf() < inputDate) {
        alert('Error: Date is in the future');
        return false;
    }

    return true;
}
```

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

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

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

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

C CSS Source Code

```
body {
    font-size: 14px;
}

#inputFields {
    padding: 10px 0;
    overflow-x: auto;
}

form {
    min-width: 1900px;
}

    form > section {
        width: 14%;
        display: inline-block;
    }

    input,
    select {
        min-width: 100px;
        width: 80%;
    }

    button {
        width: 40%;
    }

#table {
    max-height: 80vh;
    overflow: auto;
}

table {
    width: 100%;
    margin: auto;
    border-collapse: collapse;
}
    th {
        min-width: 200px;
        width: 10%;
        position: sticky;
        background: white;
        top: 0;
    }

    th > section {
        width: 80%;
        display: inline-block;
        padding: 0;
        margin: 0;
    }
```

```
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      }
```