

**NANYANG
TECHNOLOGICAL
UNIVERSITY**

SINGAPORE

CZ3006 Net Centric Computing

Assignment 2: Web Applications using JavaScript and PHP

See Jie Xun

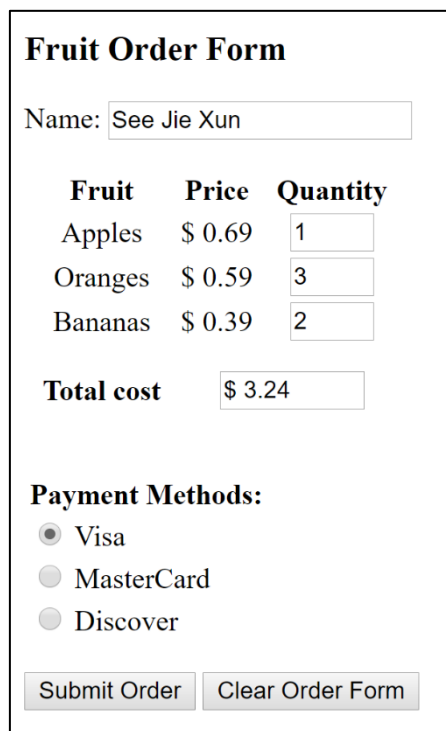
U1522059A

Declaration: This assignment was completed individually, and no parts were left unfinished.

1 Introduction

HTML, JavaScript, and PHP form the backbone of Web applications that we know and use all the time. The assignment involves the creation of a simple Web application, demonstrating understanding of Web client-side programming techniques using JavaScript and Web server-side programming techniques using PHP. This report provides a description of the application design, and addresses the learning objectives of the assignment, with close reference to the actual code used in the application.

2 Description of design



Fruit Order Form

Name:

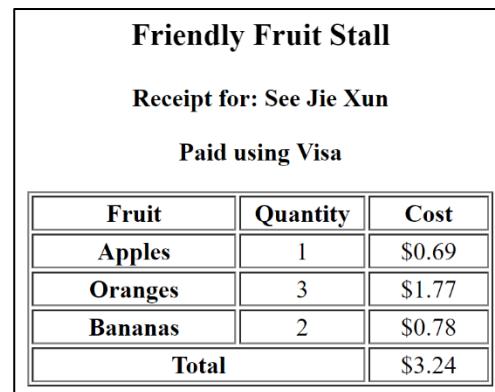
Fruit	Price	Quantity
Apples	\$ 0.69	<input type="text" value="1"/>
Oranges	\$ 0.59	<input type="text" value="3"/>
Bananas	\$ 0.39	<input type="text" value="2"/>

Total cost

Payment Methods:

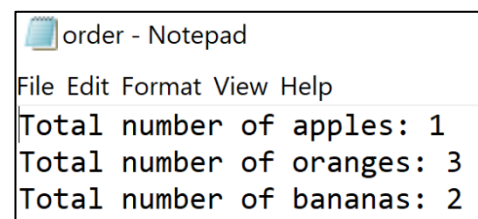
☒ Visa
☐ MasterCard
☐ Discover

Figure 1: Completed fruit order form



Friendly Fruit Stall		
Receipt for: See Jie Xun		
Paid using Visa		
Fruit	Quantity	Cost
Apples	1	\$0.69
Oranges	3	\$1.77
Bananas	2	\$0.78
Total		\$3.24

Figure 2: Fruit order receipt



order - Notepad

File Edit Format View Help

Total number of apples: 1
Total number of oranges: 3
Total number of bananas: 2

Figure 3: Text file of cumulative orders

The fruit order form is implemented in HTML, using a form that is composed of a table to list the three fruits and another table to list the payment methods.

The “Submit Order” and “Clear Order Form” buttons are in a <p> object after the two <table> objects.

2.1 Checking for valid input quantities

To check that the input for fruit quantities is valid, a JavaScript function “setTotal(element)” is used. This function is invoked whenever the input for any of the three fruit quantities is changed. It checks that all the quantities are integers, and if so, it displays the total cost accordingly. If one or more of the quantities are not positive integers, the “NaN” is displayed as the total cost. Also, if the quantity that the user just entered is not a positive integer, an alert is shown requesting for a valid quantity. This function can be seen in line 8 of Appendix 1.

A helper function “processNumber(number)” (line 30 in Appendix 1) is used to perform regex and check if the input is valid. If valid, it returns the input as an integer, if not, it returns “NaN”.

2.2 Checking that form is filled

The function “checkForm()” (line 59 in Appendix 1) is called when the form is submitted. This function checks that the name entry has been filled, and that the order is valid and non-zero, by calling the

functions “nameValidate()” and “totalValidate()”. If both pass, then “checkForm()” returns true, else it returns false. As “return checkForm()” is invoked on the “onsubmit” property of the form, the form will only proceed to be submitted if “checkForm()” returns true.

2.3 Receiving the form

When the form is submitted successfully, “receipt.php” (Appendix 2) receives the form information through a post request from the client side. It extracts the order quantities from the form and populates the necessary values in a receipt that it returns to the client side. Thus, the client is shown a receipt of the order that was just made.

2.4 Updating the cumulative order amounts

The server side also maintains an “order.txt” which records the cumulative order quantities for each fruit. This is done using built-in PHP functions for reading and writing into files. Also, regular expressions are used to read the previous number of orders, so that they can be incremented with the new order. The exact implementation can be seen in Appendix 2 starting from line 22.

3 Learning points (with code references)

3.1 How are HTML elements and attributes represented in the JavaScript binding to DOM (Document Object Model)?

The HTML DOM is “a standard object model and programming interface for HTML”, which allows a program to access and make changes to the document structure, style, and content. The elements and attributes are represented in a tree structure, as nodes and objects.

3.2 How can an event handler be associated with an event generated by a specific HTML element in the DOM event model?

One effective way of doing this is to use the **this** keyword. When **this** is used in an event handler, it refers to the specific HTML element that generated the event. This is implemented in the input HTML elements for the number of fruits that the user wants to buy, which interfaces with the JavaScript function “setTotal(element)”.

```
85.         <tr>
86.             <td align="center"> Apples </td>
87.             <td> $ 0.69 </td>
88.             <td align="center"> <input type="text" id="apples" name="apples" si
ze="2" onchange="setTotal(this)" /> </td>
89.         </tr>
90.     </tr>
```

Figure 4: HTML code defining input for “apples”

As seen in the snippet in Figure 4, the input element has an “onchange” event handler which calls the JavaScript function “setTotal(this)”, thus passing the element itself as input to the function.

```
6.     <script type="text/javascript">
7.         <!--
8.         function setTotal(element) {
9.             apples = document.getElementById("apples"); // using element ID to get
reference to the element
10.            oranges = document.getElementById("oranges");
11.            bananas = document.getElementById("bananas");
12.            numApples = processNumber(apples.value);
13.            numOranges = processNumber(oranges.value);
14.            numBananas = processNumber(bananas.value);
```

```

15.
16.         if (!isNaN(numApples) && !isNaN(numOranges) && !isNaN(numBananas)) {
17.             price = numApples*0.69 + numOranges*0.59 + numBananas*0.39;
18.             total = document.getElementById("total");
19.             total.value = "$ " + parseFloat(price).toFixed(2); // setting the t
otal price to have two decimal places
20.         }
21.         else {
22.             total = document.getElementById("total");
23.             total.value = "NaN";
24.         }
25.
26.             newValue = element.value;
27.             if (isNaN(newValue) || (newValue < 0)) alert("Please enter a valid quan
tity.") // Generating alert if input is NaN or negative
28.         }

```

Figure 5: Code snippet showing setTotal(element) function

The “setTotal(element)” JavaScript function receives the HTML element that called it, and is able to unpack the information in the input element in line 26 in Figure 5. This is the number of apples that the customer wants to order.

3.3 How to access and manipulate HTML document elements from JavaScript?

As alluded to in section 3.2, HTML document elements can be accessed from JavaScript when we have a reference to the HTML element object. For form elements, we can use [element_reference].value to access or set its value. This is done in line 26 in Figure 2. Alternatively, [element_reference].innerHTML can be used to access or set the inner body of any other type of element. [element_reference].innerText can also be used to set text content.

We can use the names of elements to get a reference to an element we are interested in. For example, we can use “document.appleRow.price”, assuming we want to access an element with the name “price” that is nested in an element named “appleRow” in the document. This approach, however, is error prone and inefficient for deeply nested elements.

Alternatively, we can use “getElementById” method, which requires the element to have a unique ID. This approach works regardless of how the element is nested in other elements in the document, making it more convenient. This is the method used in lines 9 to 11 in Figure 5.

3.4 How do HTML documents at the client side send information to web servers? How do server-side programs receive the information?

HTML documents at the client side send information to web servers using the Hypertext Transfer Protocol (HTTP), which is an application layer protocol for transmitting information between the client side and web servers. Under HTTP, there are several methods that a request can be performed; GET and POST are the most commonly used.

```

67. <form onsubmit="return checkForm()" method="post" action="/receipt.php">

```

Figure 6: HTML code snippet defining a form that submits data to web server

The line of code in Figure 6 illustrates an example of using a HTTP request to send information to a web server. The “action” attribute determines where the data is sent, where “/receipt.php” is a relative url. The “method” attribute denotes that the POST method is to be used, which causes the form data to be included in the request body that is sent to the server.

```
9.      <?php
10.      $name = $_POST["name"]; // extract the form fields from the POST data
11.      $apple = $_POST["apples"];
12.      $orange = $_POST["oranges"];
13.      $banana = $_POST["bananas"];
14.      $payment = $_POST["payment"];
```

Figure 7: PHP code snippet for receiving HTTP request

The server side can then retrieve the data sent in a HTTP request. PHP code used in the application for doing so is seen in Figure 7.

3.5 How do server-side programs access files?

Server-side programs typically access information stored in databases. However, in this assignment, a simple text file is used to store information, and this text file is read and re-written as a user interacts with the client-side program.

```
21.      // read file to get current total orders
22.      $filename = 'order.txt';
23.      $file = @fopen($filename, 'r');
24.      $fileContents = @file($filename);
```

Figure 8: PHP code snippet for reading text file

Figure 8 shows the PHP code used by the server side to read the file “order.txt”, which stores the cumulative number of fruit orders by customers using the application. The fopen function in line 23 returns a file pointer, and the file function in line 24 reads all the lines of the file into an array.

Note: the @ operator is used to suppress error messages which will arise when trying to read the file if it does not yet exist. Line 29 of “receipt.php” (Appendix 2) then checks if the file has successfully been opened and performs the requisite updates.

3.6 How do server-side programs generate HTML documents and send them to the client side?

These are dynamic web pages, which are created as requested and sent from the server side to the client side.

Appendix 1 (index.html)

```
1. <!DOCTYPE html>
2. <html>
3.
4. <head>
5.   <title>Friendly Fruit Stall</title>
6.   <script type="text/javascript">
7.     <!--
8.       function setTotal(element) {
9.         apples = document.getElementById("apples"); // using element ID to get
           reference to the element
10.        oranges = document.getElementById("oranges");
11.        bananas = document.getElementById("bananas");
12.        numApples = processNumber(apples.value);
13.        numOranges = processNumber(oranges.value);
14.        numBananas = processNumber(bananas.value);
15.
16.        if (!isNaN(numApples) && !isNaN(numOranges) && !isNaN(numBananas)) {
17.          price = numApples*0.69 + numOranges*0.59 + numBananas*0.39;
18.          total = document.getElementById("total");
19.          total.value = "$ " + parseFloat(price).toFixed(2); // setting the t
           otal price to have two decimal places
20.        }
21.        else {
22.          total = document.getElementById("total");
23.          total.value = "NaN";
24.        }
25.
26.        newValue = element.value;
27.        if (isNaN(newValue) || (newValue < 0)) alert("Please enter a valid quan
           tity.") // Generating alert if input is NaN or negative
28.      }
29.
30.      function processNumber(number) { // returns the input as an integer, or els
           e returns "NaN" if input isn't an integer
31.        if (number == "") return 0
32.        else if (/^\d+$/ .test(number)) return parseInt(number)
33.        else return 'NaN'
34.      }
35.
36.      function nameValidate() {
37.        nameDoc = document.getElementById("name");
38.        if (nameDoc.value == "") {
39.          alert("Please enter your name.");
40.          return false;
41.        }
42.        return true;
43.      }
44.
45.      function totalValidate() {
46.        total = document.getElementById("total");
47.        price = total.value;
48.        if (isNaN(price.substring(2))) {
49.          alert("Please enter a valid order.");
50.          return false;
51.        }
52.        else if (price == 0) {
53.          alert("Please choose at least one item."); // require the user to o
           rder at least one item
54.          return false;
55.        }
56.        return true;
57.      }
```

```

58.
59.     function checkForm() {
60.         if (nameValidate() && totalValidate()) return true;
61.         else return false;
62.     }
63.     -->
64. </script>
65. </head>
66. <body>
67.     <form onsubmit="return checkForm()" method="post" action="receipt.php">
68.         <h3> Fruit Order Form </h3>
69.         <p>
70.             Name:
71.             <input type="text" placeholder="e.g. John Calvin" id="name" name="name"
/>
72.         </p>
73.
74.         <!-- A table for item orders -->
75.         <table border = "0">
76.
77.             <!-- Column headings -->
78.             <tr>
79.                 <th width="80"> Fruit </th>
80.                 <th> Price </th>
81.                 <th width="80"> Quantity </th>
82.             </tr>
83.
84.             <!-- Now, the table data entries -->
85.             <tr>
86.                 <td align="center"> Apples </td>
87.                 <td> $ 0.69 </td>
88.                 <td align="center"> <input type="text" id="apples" name="apples" si
ze="2" onchange="setTotal(this)" /> </td>
89.             </tr>
90.             <tr>
91.                 <td align="center"> Oranges </td>
92.                 <td> $ 0.59 </td>
93.                 <td align="center"> <input type="text" id="oranges" name="oranges"
size="2" onchange="setTotal(this)" /> </td>
94.             </tr>
95.             <tr>
96.                 <td align="center"> Bananas </td>
97.                 <td> $ 0.39 </td>
98.                 <td align="center"> <input type="text" id="bananas" name="banana
s" size="2" onchange="setTotal(this)" /></td>
99.             </tr>
100.            <tr height="50">
101.                <th> Total cost </th>
102.                <th colspan="2">
103.                    <input type="text" size="7" onfocus="this.blur()" id=
"total" name="total" placeholder="0"/> <!--
- blurred because it isn't for user input -->
104.                </th>
105.            </tr>
106.        </table>
107.        <br/>
108.
109.        <!-- A table for payment methods -->
110.        <table>
111.            <tr>
112.                <th>Payment Methods:</th>
113.            </tr>
114.            <tr>
115.                <td>
116.                    <input type="radio" name="payment" id="payment" value="V
isa" checked/> Visa

```

```
117.                 </td>
118.             </tr>
119.         <tr>
120.             <td>
121.                 <input type="radio" name="payment" id="payment" value="M
asterCard" /> MasterCard
122.             </td>
123.         </tr>
124.         <tr>
125.             <td>
126.                 <input type="radio" name="payment" id="payment" value="D
iscover" /> Discover
127.             </td>
128.         </tr>
129.     </table>
130.
131.     <!-- The submit and reset buttons -->
132.     <p>
133.         <input type="submit" value="Submit Order" />
134.         <input type="reset" value="Clear Order Form" />
135.     </p>
136. </form>
137. </body>
138. </html>
```


Appendix 2 (receipt.php)

```
1. <!DOCTYPE html>
2. <html>
3.
4. <head>
5.     <title>Receipt from Friendly Fruit Stall</title>
6. </head>
7.
8. <body>
9.     <?php
10.         $name = $_POST["name"]; // extract the form fields from the POST data
11.         $apple = $_POST["apples"];
12.         $orange = $_POST["oranges"];
13.         $banana = $_POST["bananas"];
14.         $payment = $_POST["payment"];
15.         if ($apple == "") $apple = 0;
16.         if ($orange == "") $orange = 0;
17.         if ($banana == "") $banana = 0;
18.
19.         $total = ($apple * 0.69 + $orange * 0.59 + $banana * 0.39);
20.
21.         // read file to get current total orders
22.         $filename = 'order.txt';
23.         $file = @fopen($filename, 'r');
24.         $fileContents = @file($filename);
25.
26.         $prevApples = 0;
27.         $prevOranges = 0;
28.         $prevBananas = 0;
29.         if ($fileContents !== FALSE){ // then the file already exists and we manage
d to read it
30.             // find previous counts using regular expressions
31.             preg_match("/Total number of apples: (\d+)/", $fileContents[0], $prevAp
ples);
32.             preg_match("/Total number of oranges: (\d+)/", $fileContents[1], $prevO
ranges);
33.             preg_match("/Total number of bananas: (\d+)/", $fileContents[2], $prevB
ananas);
34.
35.             // first array value
36.             $prevApples = intval($prevApples[1]);
37.             $prevOranges = intval($prevOranges[1]);
38.             $prevBananas = intval($prevBananas[1]);
39.
40.             fclose($file);
41.         }
42.
43.         // computing new fruit counts
44.         $newApples = $prevApples + $apple;
45.         $newOranges = $prevOranges + $orange;
46.         $newBananas = $prevBananas + $banana;
47.
48.
49.         // write new fruit counts to file
50.         $file = fopen($filename, 'w');
51.         fwrite($file, "Total number of apples: $newApples \r\n");
52.         fwrite($file, "Total number of oranges: $newOranges \r\n");
53.         fwrite($file, "Total number of bananas: $newBananas \r\n");
54.
55.         // closing the file
56.         fclose($file);
57.     ?>
58.     <!-- Table for receipt -->
```

```

59.         <table border="border" width="300">
60.             <caption>
61.                 <b>
62.                     <p style="font-size:20px">Friendly Fruit Stall</p>
63.                     <p>Receipt for: <?php print $name; ?></p>
64.                     <p>Paid using <?php print $payment; ?></p>
65.                 </b>
66.             </caption>
67.             <br>
68.             <tr>
69.                 <th width="140">Fruit</th>
70.                 <th width="80">Quantity</th>
71.                 <th width="80">Cost</th>
72.             </tr>
73.             <tr>
74.                 <th>Apples</th>
75.                 <td align="center">
76.                     <?php print ("apple"); ?>
77.                 </td>
78.                 <td align="center">
79.                     <?php print ("$.number_format($apple * 0.69, 2)); ?> <!--
- Setting the price to have two decimal places -->
80.                 </td>
81.             </tr>
82.             <tr>
83.                 <th>Oranges</th>
84.                 <td align="center">
85.                     <?php print ("orange"); ?>
86.                 </td>
87.                 <td align="center">
88.                     <?php print ("$.number_format($orange * 0.59, 2)); ?>
89.                 </td>
90.             </tr>
91.             <tr>
92.                 <th>Bananas</th>
93.                 <td align="center">
94.                     <?php print ("banana"); ?>
95.                 </td>
96.                 <td align="center">
97.                     <?php print ("$.number_format($banana * 0.39, 2)); ?>
98.                 </td>
99.             </tr>
100.            <tr>
101.                <th colspan=2>Total</th>
102.                <td align="center">
103.                    <?php print ("$.number_format($total, 2)); ?>
104.                </td>
105.            </tr>
106.        </table>
107.
108.    </body>
109.
110. </html>

```