**Nanyang Technological University**

Name: Ang Hao Jie

Matriculation No.: U1721621G

Lab Group: TS6

Course: CZ3006 Net-Centric Computing

Assignment 2 Report

**Implementation**

**1. Client side (index.html)**

Figure 1 shows the design of the webpage I have created. Client is able to access this webpage through index.html to order fruits from this website. The top textbox requests user to input their username. Below the username textbox is a table which categorize into 3 different components. They are fruits that user can purchase from the website, the price of each fruit, and the quantity of fruits that the user would like to purchase. Below the table form are payment methods. VISA payment is selected by default. Users may choose one of the payment methods they prefer. Below the payment method section is the submit button. Initially, it will be disable until user orders a fruit by keying a value into one of the *Quantity* textbox.

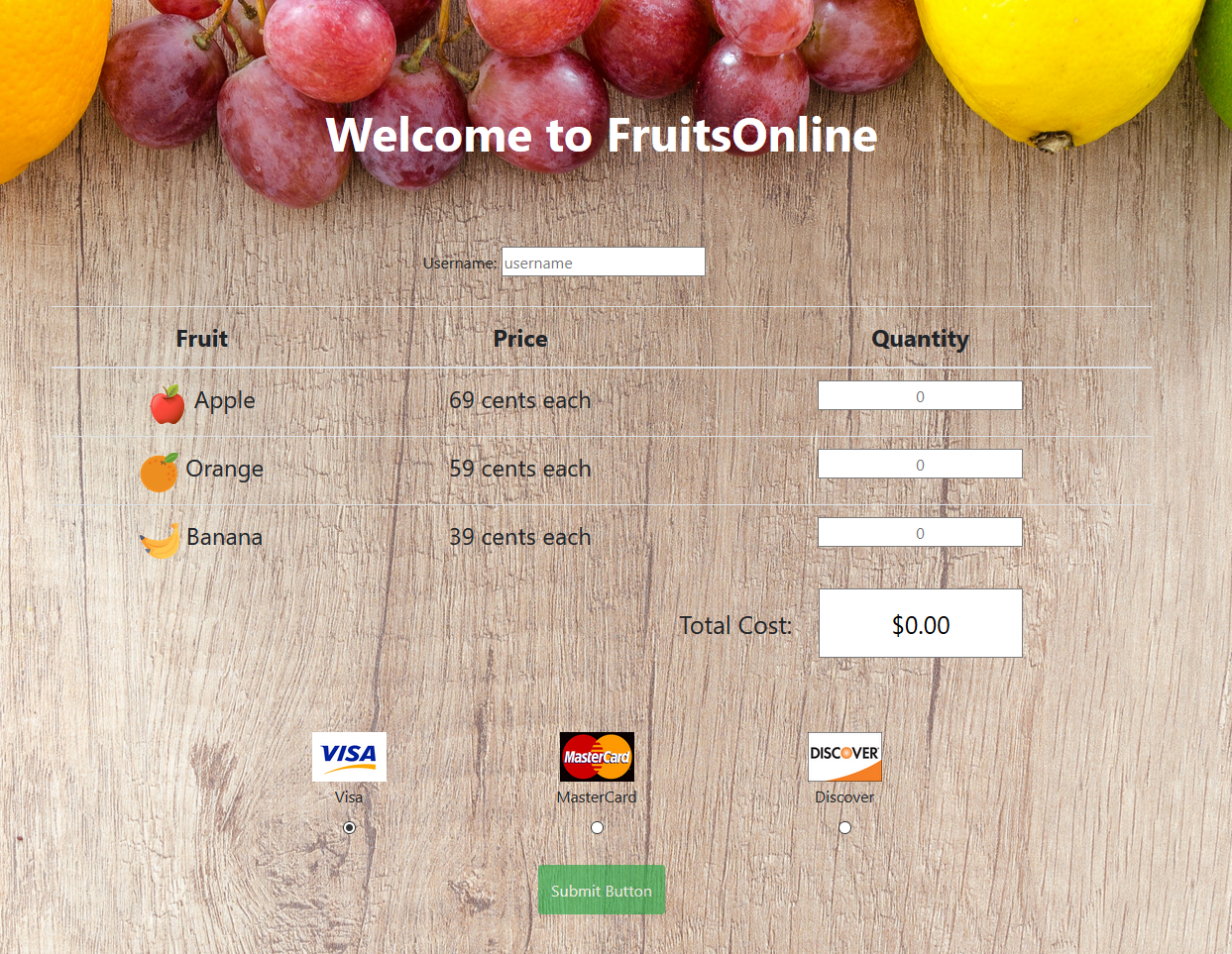


Figure 1: Client side webpage display

**1.1 Username – Textbox**

The top of the webpage contains a textbox for user to input. This can be created by using the *input* element in HTML (in Figure 1.1a). The *name* attribute, userName, in *input* element allows php file in the server to obtain the value keyed in by the user. The *required* attribute will highlight the textbook and notify the user to input a value into the textbook if there is no value input in the username textbook upon clicking the submit button (in Figure 1.1b).

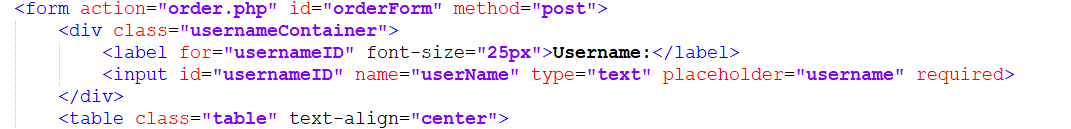


Figure 1.1a: Input Element for Username

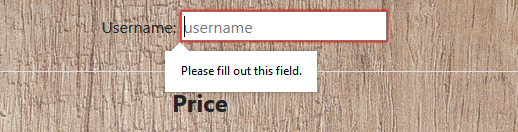


Figure 1.1b: Notify user field required to fill-up

**1.2 Order table**

Below the username’s textbox contains a table with fruits to purchase and quality of fruits to order. The table can be created by using the *table* element (in Figure 1.1c). In the *table* element, the class attribute reference the design of the table to css in “css/ bootstrap.css”. In the third column of the table contain rows of textboxes created by *input* element for user to key in the amount of individual fruits they would like to purchase.

The *onChange* attribute is an event that reference to *changesToValue(this.id)* which is a JavaScript method(in Figure 1.1c). This event is triggered when there is a change to the value in the quantity textbox. When calling *changesToValue()*, *id* of this *input* element will be passed to the method. In the *changesToValue()* method, the value in the quantity textbox will be check if it is a valid input through *validations()* method (in Figure 1.1d). If the *validations()* method returns true, *calculateTotalCost()* will be called to change the total cost displayed. Finally, *enableSubmitBtn()* method will be called enable or disable the submit button.

In the *validations()* method, if the value in the quantity textbox is a positive integer or empty, a Boolean of true will be returned back to *changesToValue()* method (in Figure 1.1e). Else, alert messages will be displayed to the user, that quantity textbox will be set empty, textbox that display the total cost of the ordered fruits will be set to “NaN”, and lastly, Boolean of false will be returned back to *changesToValue()*.



Figure 1.1c: Table Element to display fruit, price, and fill-up quality

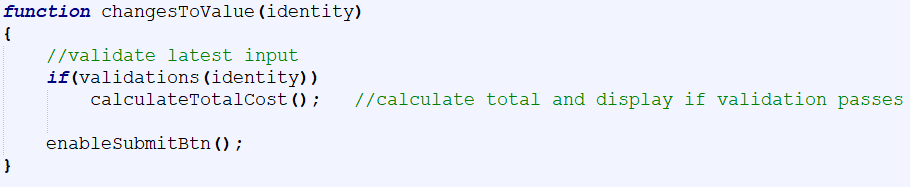


Figure 1.1d: Method body for changesToValue()

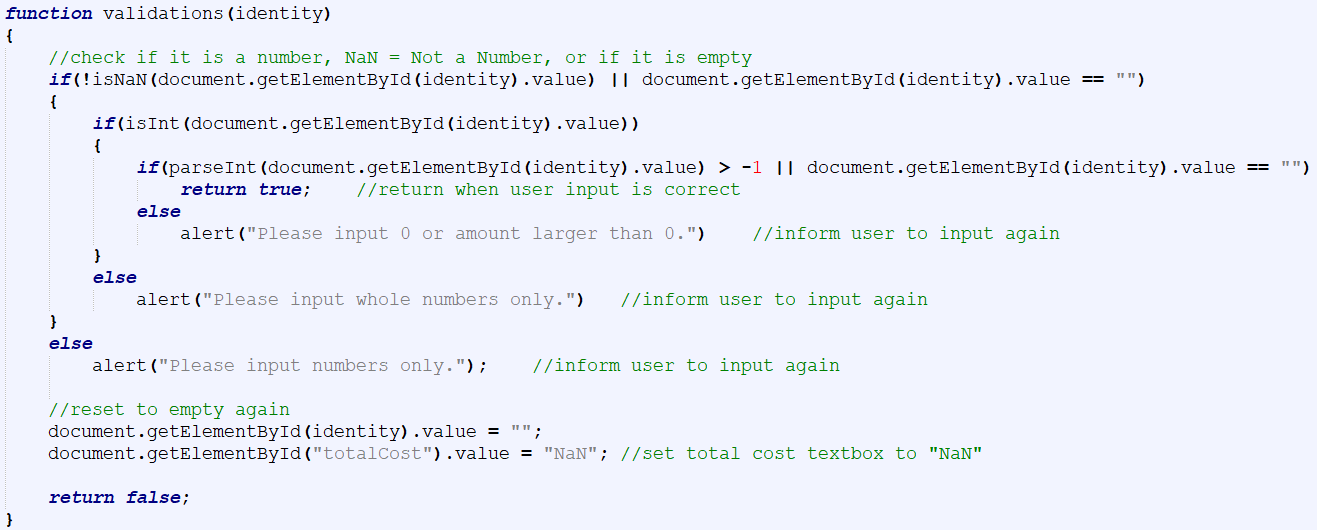


Figure 1.1e: Method body for validations()

**1.3 Total Cost – Textbox**

Below the table of orders contains a textbox which displays the total cost of the fruits ordering by the user using *input* element in HTML (in Figure 1.1f). By default, the textbox will display “$0.00” if no fruits are being added to the list of fruits using is ordering. Based on the *onChange* event in Figure 1.1c, the *changesToValue()* method will call the *calculateTotalCost()* method that will change the value of the total cost displayed (in Figure 1.1g). The total cost textbox will change its display to “NaN” if numbers are not input into any of the quantity textbox (in Figure 1.1e and 1.1h).

The *onfocus* event will calls the *blur()* method which will prevent user from editing the content in the total cost textbox and triggers *onblur* event which calls *focus()* method if the mouse is not on top of the total cost textbox (in Figure 1.1f).

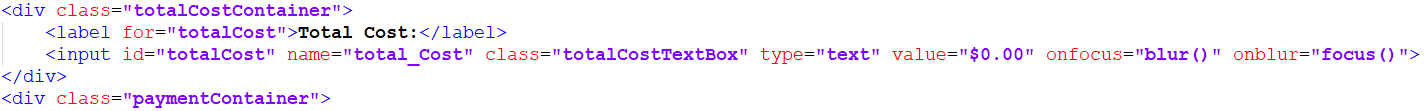


Figure 1.1f: input element textbox to display total cost

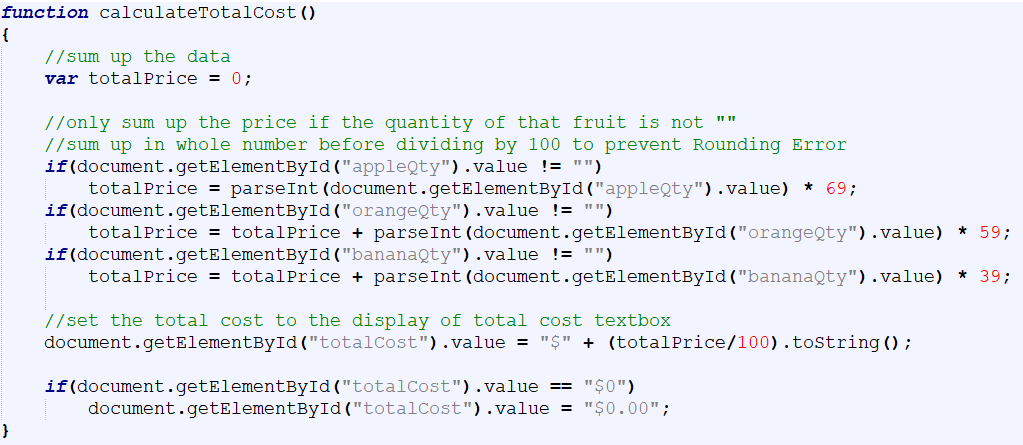


Figure 1.1g: method body for calculateTotalCost()



Figure 1.1h: Total Cost textbox displays “NaN”

**1.4 Payment methods**

In the section below the table of orders contains a table of different payment methods user can choose to pay for their order. *Input* element of *radio* type is used to allow user to click the choice of payment method they prefer (in Figure 1.1i). The checked attribute in input element for VISA results in VISA to be selected by default. Users may click other payment methods such as MasterCard or Discover if they prefer to pay through those payment methods. The name attribute has to be the same so that only one radio open will be selected at one time. With the same name, only the selected payment option will be sent to the server.



Figure 1.1i: Table for Payment Methods

**1.5 Submit button**

The submit button using the *input* element with *submit* type in HTML which will submit the value of the elements in the form to “order.php” in the server (in Figure 1.1j). *disabled* attribute is included so that at page start up, the button will be disabled. The button will only be enabled or disabled when the onChange event is triggered (in Figure 1.1c). enableSubmitBtn() method in JavaScript will determine if the submit button is enabled or disabled based on the value displayed on the total cost textbox (in Figure 1.1k). If the value of the textbox is “$0.00”, the submit button will be disabled to prevent user from submitting empty order request. If the value is “NaN”, the submit button will be disabled as non-integer value was input into the quantity textboxes. Else, the submit button will be enabled.



Figure 1.1j: input Element for submit button to submit the form

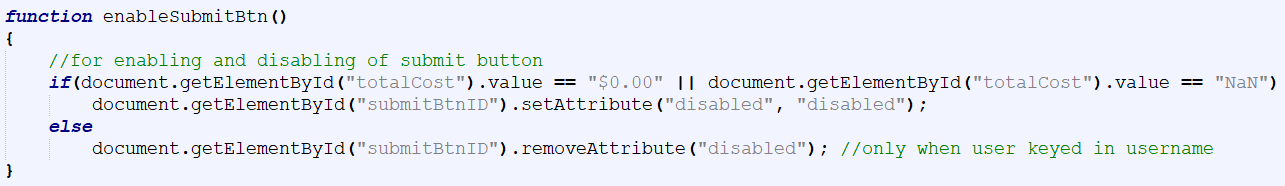


Figure 1.1k: Method body of enableSubmitBtn()

**2. Server side (order.php)**

Figure 2 shows the receipt that will be display to the user from order.php in the server after successful order from the user. The receipt received by the user is in the form of a table where the username of the customer, quantity or apple, orange, and banana will be displayed. Total price of all the fruits ordered by the user and the preferred payment method by the user will be displayed. Below the table of receipt is a Home button which will bring the user back to the order home page as shown in Figure 1.

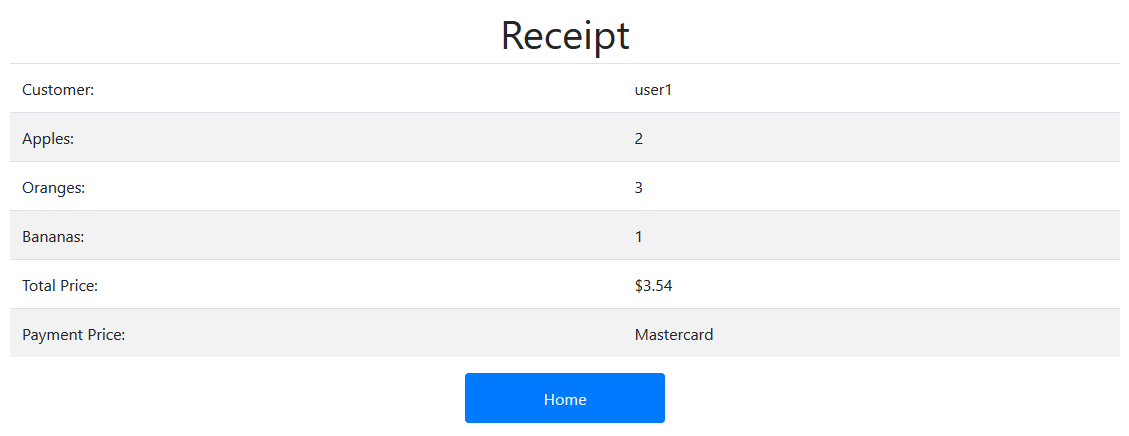


Figure 2

**2.1 Receive order from client**

After user clicks on the submit button in index.html, values of the elements will be sent to the server through POST method and handles by order.php (in Figure 2.1a). The value of the element can be obtain by using the name attribute in the element in HTML in index.html (in Figure 2.1b). The values are set in the local variables of order.php.

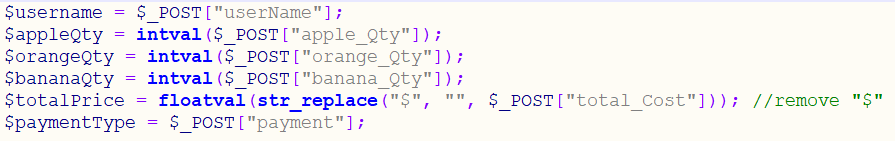


Figure 2.1a: Receive order value from client



Figure 2.1b: *input* element in index.html with name attribute

**2.2 Check the existence of order.txt**

The existence of “order.txt” is being checked (in Figure 2.1c). If it does not exist, it will be created. “order.txt” is important as it is acting as a local database then stores the total order.

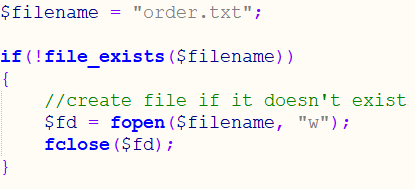


Figure 2.1c: Create “order.txt” if does not exist

**2.3 Writes order to “order.txt”**

After receiving a new order, the amount of each individual will be recorded in “order.txt”. Firstly, “order.txt” will be checked if it is empty before writing new data of the new order of each individual fruits into it. This is to prevent existing data from being overwritten if “order.txt” is not empty. If it is not empty, the current data in “order.txt” will be extracted and the quantity of each individual fruits in the new order will be added into the total order in “order.txt” that was just extracted before writing the new total value data into “order.txt” and closing it (in Figure 2.1d).



Figure 2.1d: Reading and writing of data into “order.txt”

**2.4 Display of receipt to user**

After storing the data into “order.txt”, the receipt of the new order that was just made by the user will be send to the user in the form of HTML (in Figure 2.1e). Figure 2 shows an example of the receipt that the user will view after successful order.



Figure 2.1e: Echo HTML to client to display receipt

**Documentation**

Below are the list of source files required and their details:

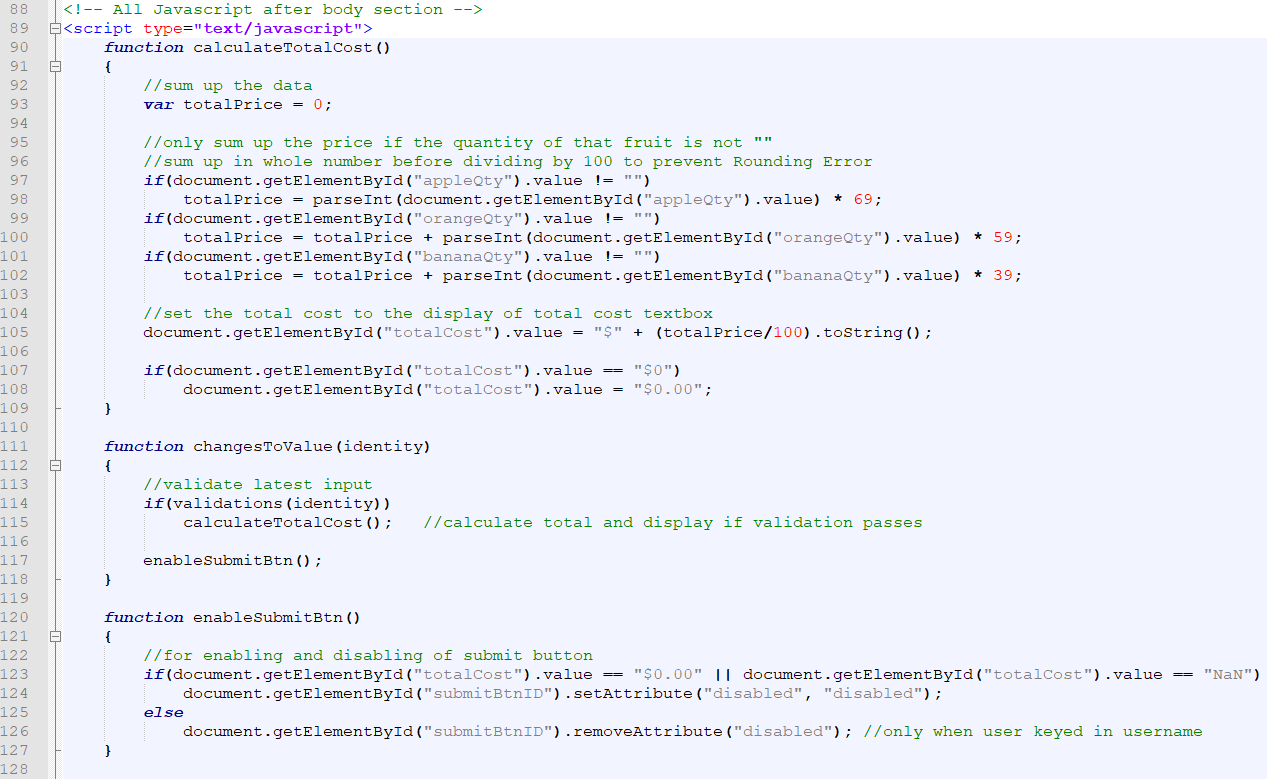
* index.html : Main webpage send to the client to order fruits
* style.css : A CSS file to change the layout and appearance of each element in index.html
* bootstrap.min.css : A CSS file by bootstrap to change the layout and appearance of certain elements in index.html
* order.php : Implements the server side logic for this application
* order.txt : A local database to save the total number of each individual fruits ordered
* apple.png : An image used to display image of an apple in index.html
* banana.png : An image used to display image of a banana in index.html
* orange.png : An image used to display image of an orange in index.html
* fruits.jpg : An image used as the background of the webpage display to the client in index.html
* discover.jpg : An image used to display image of Discover in the payment option section
* mastercard.jpg : An image used to display image of MasterCard in the payment option section
* visa.jpg : An image used to display image of VISA in the payment option section

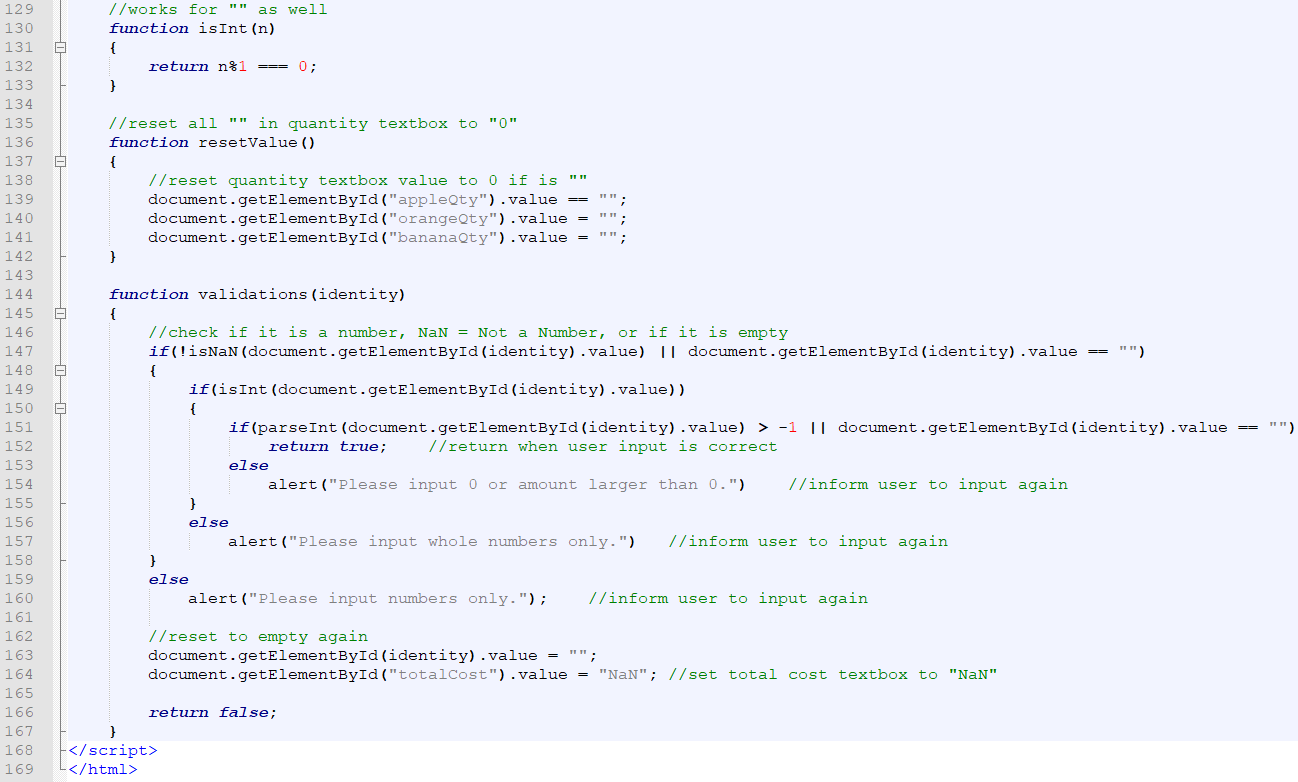
**Source code**

**1. index.html**









**2. order.php**

