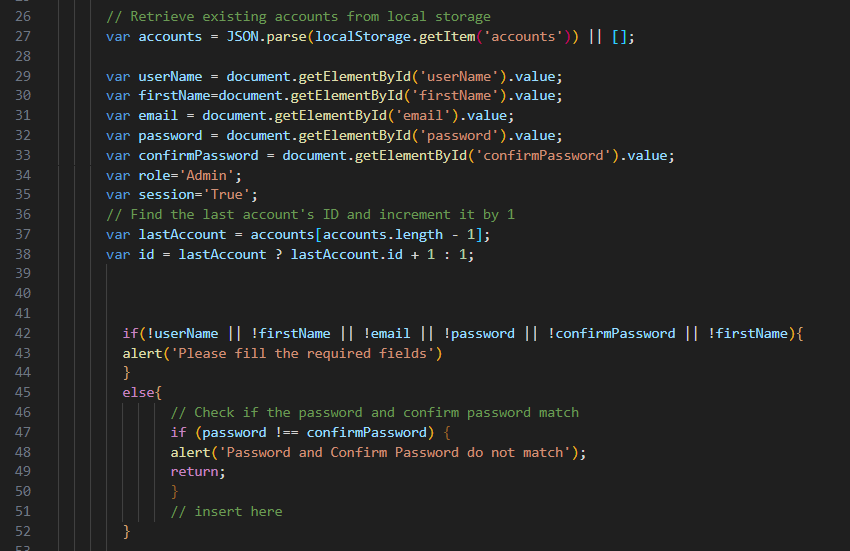
**Admin Module**

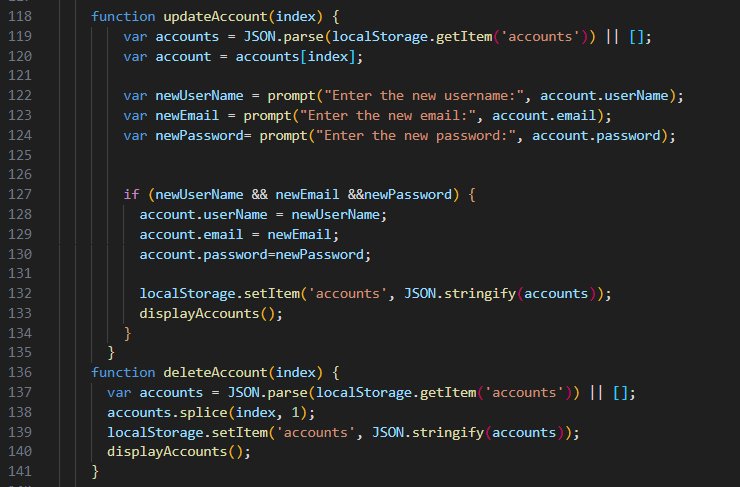
1. create-admin.html
   1. Add button will not work if any of the fields are empty. It will ask you to fill up the textbox
   2. If password and Confirm password didn’t match. It will show an error message saying they didn’t match

**Example code for creating an account**

Reference file: **create-admin.html**

****

* Line 27 is where we call the records from localStorage from browser
* “accounts” is the key name from our local storage
* We created a variable called accounts to handle the get item process
* Line 29-38 is where we set value with our variable (ex: var username), we are calling the value of ‘userName’ inside the getElementById. This is the id of our textbox from html
* role=’Admin’ to set automatically the role once we create an admin account
* Line 42-44 is to validate our inputs (!userName) means if there is no value on textbox then ask the user to put value
* Line 45- 52 is the validation for password and confirm password matching
  1. If the form has valid values. It will generate account with the Role value of “Admin”.
  2. User can also update and delete record of account from the list  
     Reference file: **create-admin.html**



* Line 118-135 is an example for update function. We call again the data from local storage using get item and used prompt to update the record. Instead of using textbox. We use prompt to simply update record
* Line 136-141 is for delete record. Using the index value, we can select the row record we want to delete using splice function in javascript
* After we don the splice, using the localStorage.setItem we are returning the updated objects which is the accounts

1. index.html
   1. Used bootstrap for designing to be presentable to user
      * Most of the html page uses bootstrap library to lessen the time when it comes on layout and design

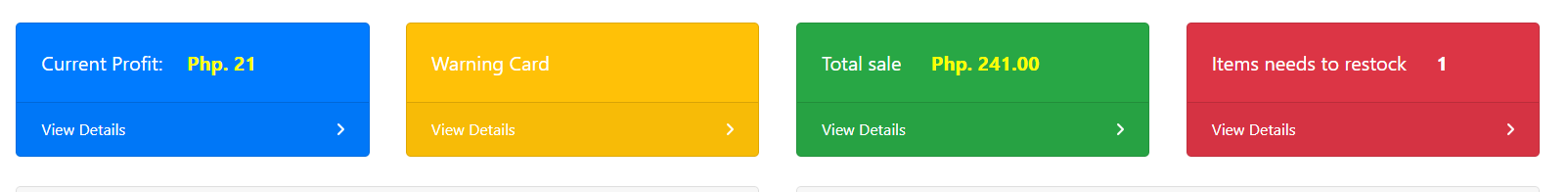
Reference file: **index.html**



* + - The image above show how we call the reference library for Jquery,bootstrap and fontsAwesome
    - We used class ids for our html elements so it will be more responsive

Example of class id:  


* 1. Used combination of JQuery,Javascript for functionalities
* Index.html is the main dashboard for admin where we can see the Running profit, Total sale and also the items that we need to restock.  
  Reference file: **index.html**



* If we click the details on the red card, it will display all the items that has the stock value of less than 10. It helps the admin to monitor the available items.
* The logic behind this is connected to POS view. Once the order from order list has been submitted/transaction complete, it will automatically filer that order and match to the item to subtract the order quantity from item stock.

**Example:**

**{items}**

|  |  |  |
| --- | --- | --- |
| **Title** | **Stock** | **Id** |
| **Hotdog** | 15 | **4** |
| **Shampoo** | 10 | **5** |

**{orders}**

|  |  |  |  |
| --- | --- | --- | --- |
| **Title** | **Stock** | **Id** | **fKeyId** |
| **Hotdog** | 15 | **1** | **4** |
| **Shampoo** | 10 | **2** | **5** |

* Once the staff clicked the complete transaction button from POS, there is a logic there that will filter {orders} using their fKeyId. If the value of this fKeyId matches the Id from {items}, it will automatically subtracts them

1. register.html
   1. We use a bootstrap form for the inputs.
   2. Same code process from admin create account, it will create new account but this time the role sets to “Staff” who can only access the POS.
   3. Has field validaitions
2. staff-list.html
   1. Can view the list of all accounts (Admin and Staff)

Sample code:

Reference file: **staff-list.js**

****

* line 1 is to wrap the code for display. We use $(document).ready(function) to load the accounts data during the page load
* Line 3 is to call the localStorage key name “accounts”
* Line 6 to line 25 is the process to display records in table using the “Data Tables”.

Documentation reference:

<https://www.jsdelivr.com/documentation>

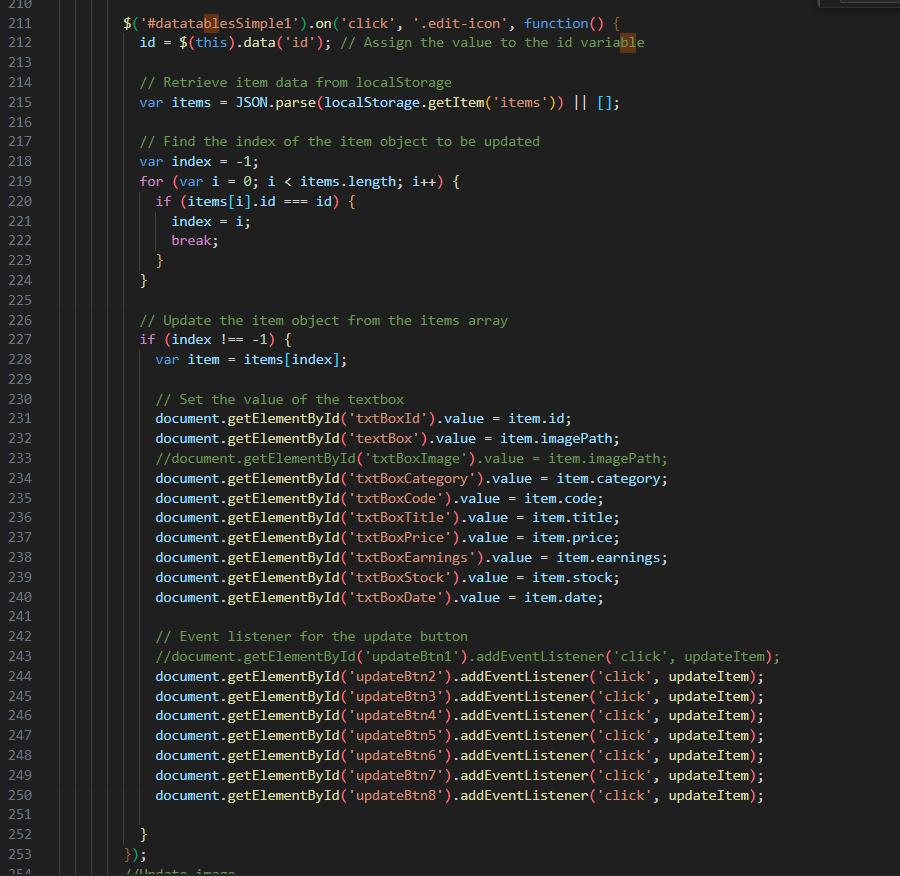
<https://datatables.net/>

We used dataTable to maximize the functionality of a table such as search filtering, automatic bind of data to the table, enable the pagination and can be sortable based on column

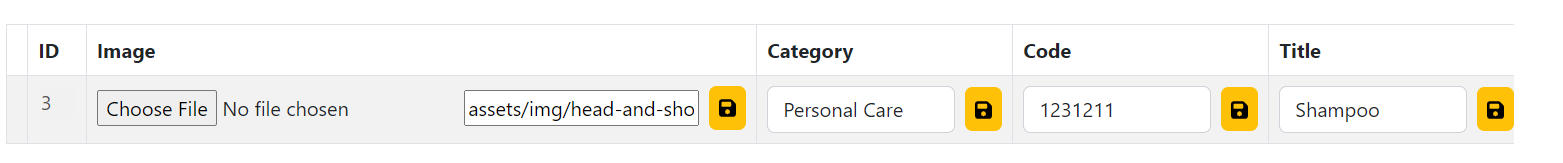
1. manage-items.html
   1. Admin can view all the list for Items it also uses data Tables functionality
   2. This page has Create,Edit/Update, and delete functionalities
   3. Create is same process on how to create accounts.
   4. In contrast to updating via prompts or alerts, we have implemented a textbox on our HTML page that automatically fills in with the corresponding value when an item is clicked. The concept behind this approach is that when we click on a specific row, the index of that item is retrieved, and the values associated with that index are displayed in the textboxes. Consequently, we can easily make updates to the displayed values.

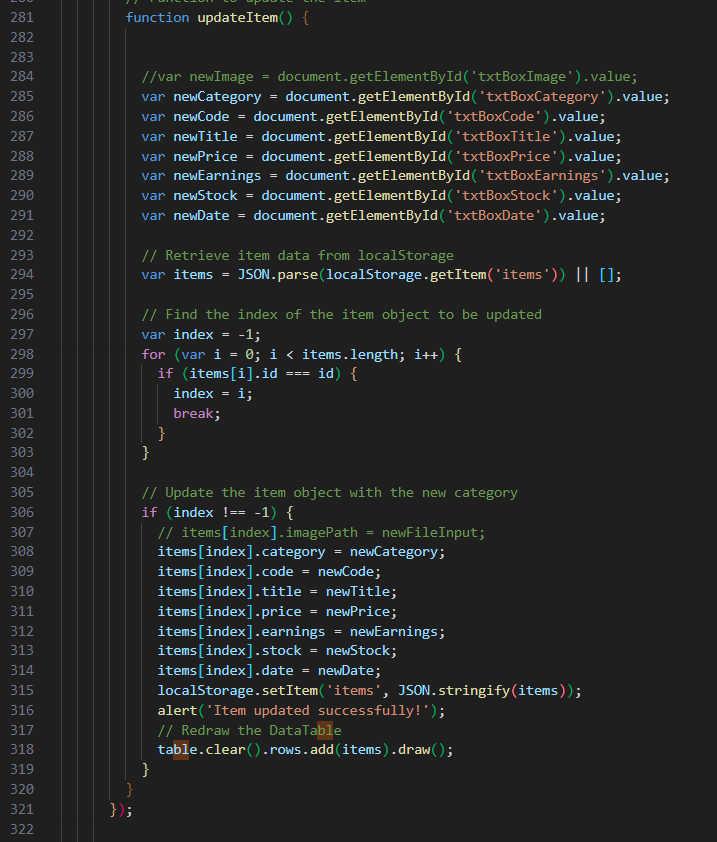
Example code:

Reference file: **manage-item.js**

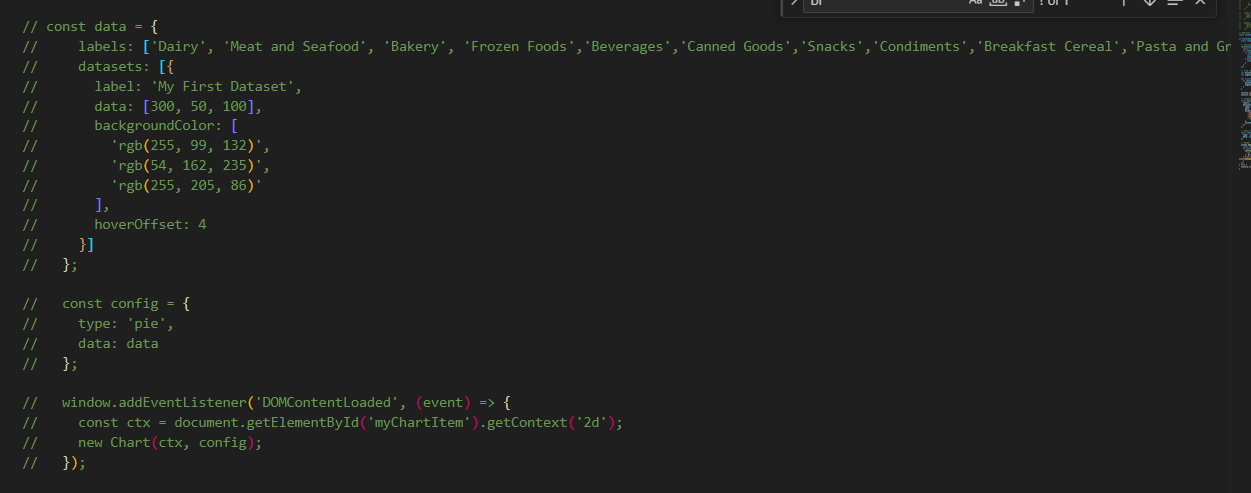
****

* We will skip the code part for the insert of items and delete of items because it is same from the previous screenshot. It is almost the same
* The code for inserting and deleting items remains unchanged from the previous screenshot, so we will skip discussing it again as it is essentially the same.
* Line 217 to line 224 is to filter the index of what we need to update
* LINE 231 TO LINE 240 is where we fetch the value and sets to the textbox

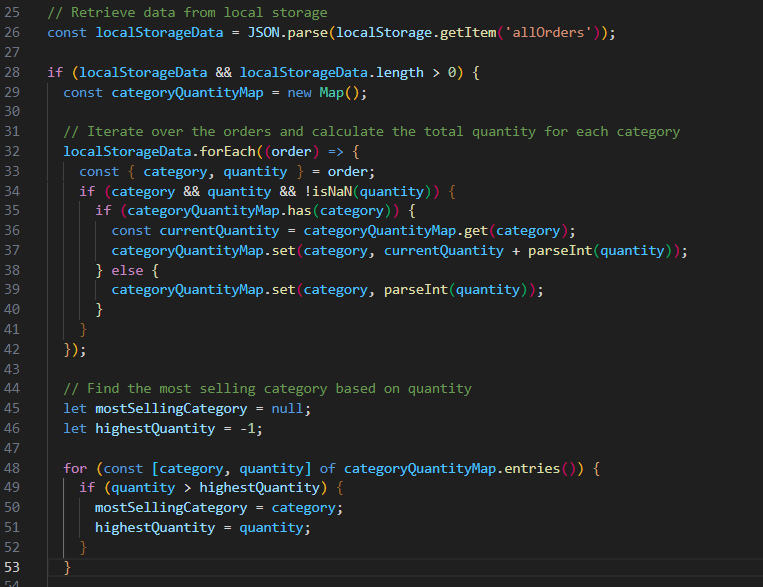


* Line 244-250 is where we call the Id of the designated button of textbox that we need to update and we add eventListener to call the function updateItem. This updateItem function code is:  
  
  + Line 285 to 291 is where we create new variables that will handle the new value for our field
  + Line 316 to 318 is to update the item record with our new value based on the fetch index

1. chart-report.html
   1. We use javascript to load the records from local storage.
   2. We use a javascript example for creating pie chart
      * Reference file: **pie-chart.js**



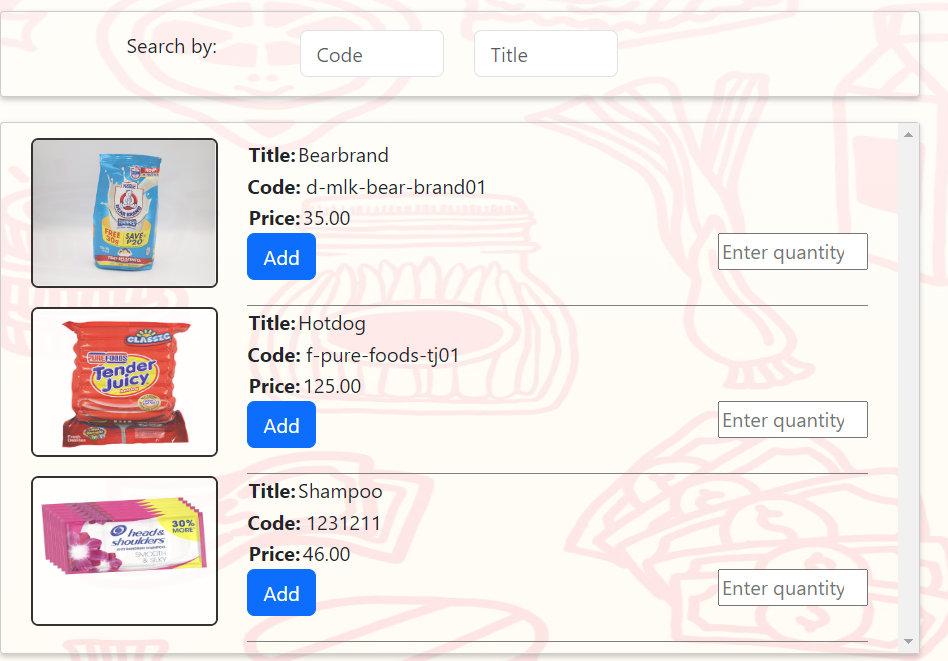
This is a pie chart that has a static data. We use this pattern in our custom chart

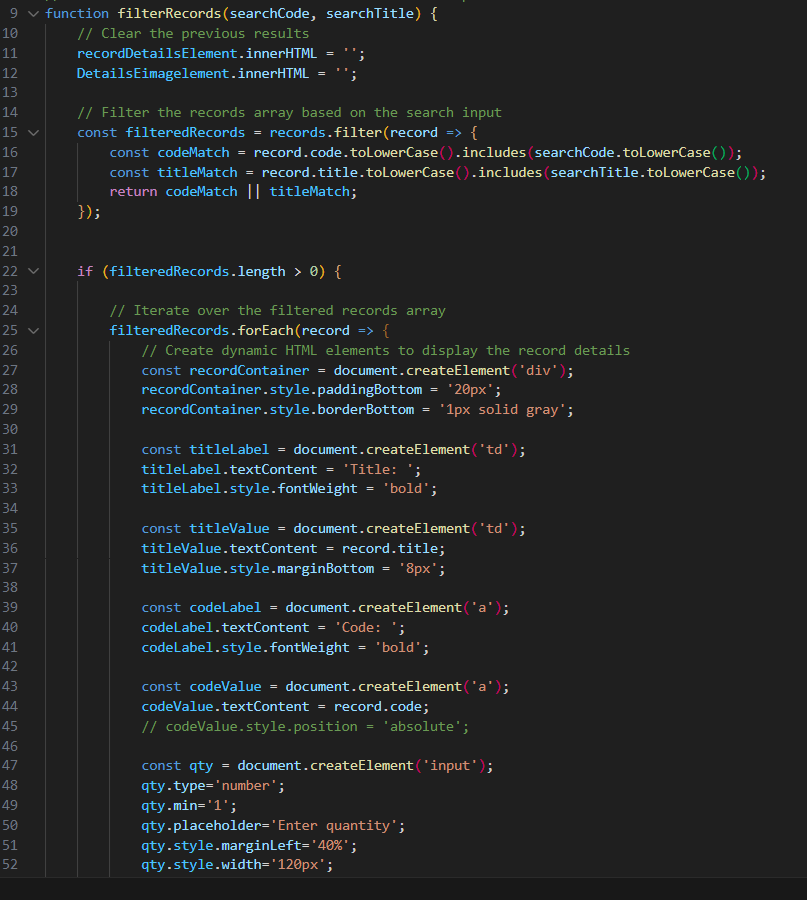


The image above is our custom pie chart. We call again the record from local storage.   
Line 28 to 42- we used for each and Map function to get the total quantity per category of items to see what is the most selling category

1. bar-chart-report.html
   1. Same on Pie Chart process, we used javascript documentation for creating a graph using static data then customized it using our own data records

**Staff Module**

1. POS-view.html
   1. To display the view for Items. We used foreach function to iterate rows of item  
      

The code behind this logic:  
Reference file: **POS-view.js**

* + 1. Line 15- 18 is the process to filter the records based on Title or Code
    2. Starting line 22, it is the function for repeating the rows of record. It will filter each data from “record” variable then it will display on the page using the create elements

Reference file: **POS-view.js**



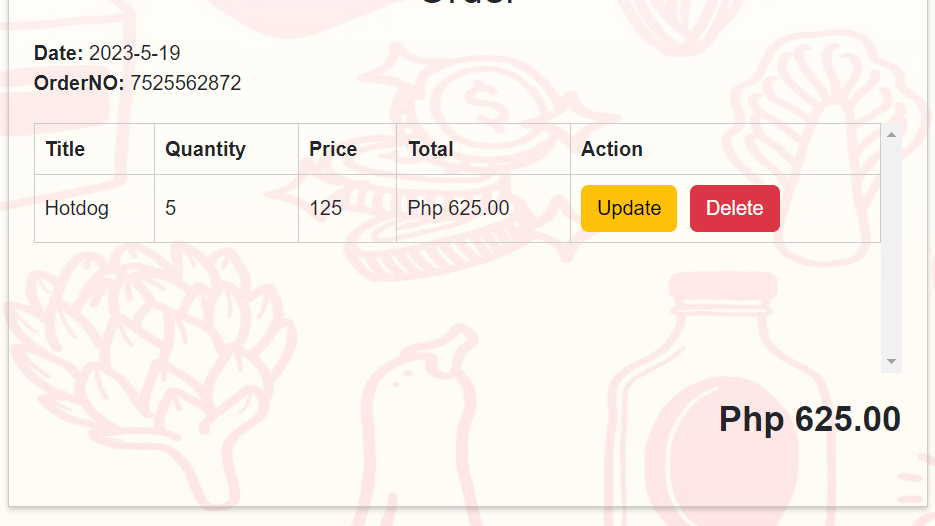
* This is the button click function for addBtn. It will first validate if the textbox field for quantity has a value. If there is a value then it will proceed on the process where it will first check if the order that you need to add to the order list on the rightside is already existing using the soredOrders.find(function(order)){}. If it detects that there is a already existing order item from the list, it will ask you to just update the quantity instead of duplicating the rows of item
* Line 80 -Line 102 is where we get the value of the clicked item and stored in a variable then we create a new object named newItem. Refer to the below image



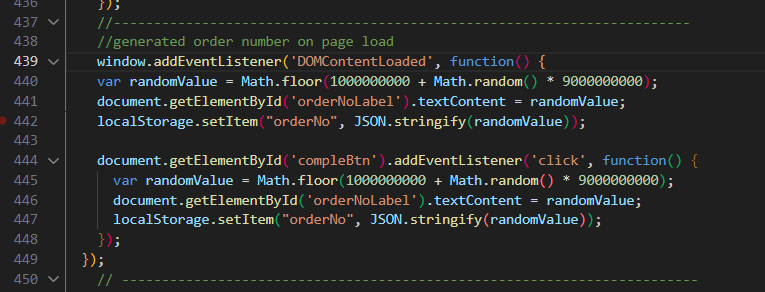
* We need this newItem because this is the object that we need to push on our localStorage namen “orders”



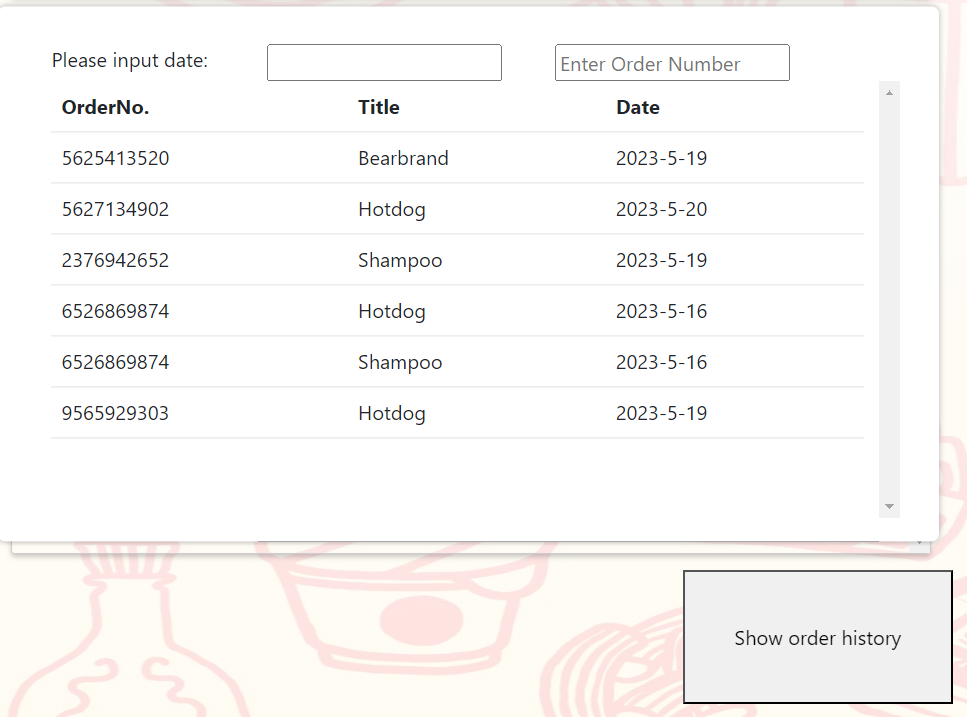
* This is how we display the “orders ”, the list on the right side
* Same concept on how we display the item



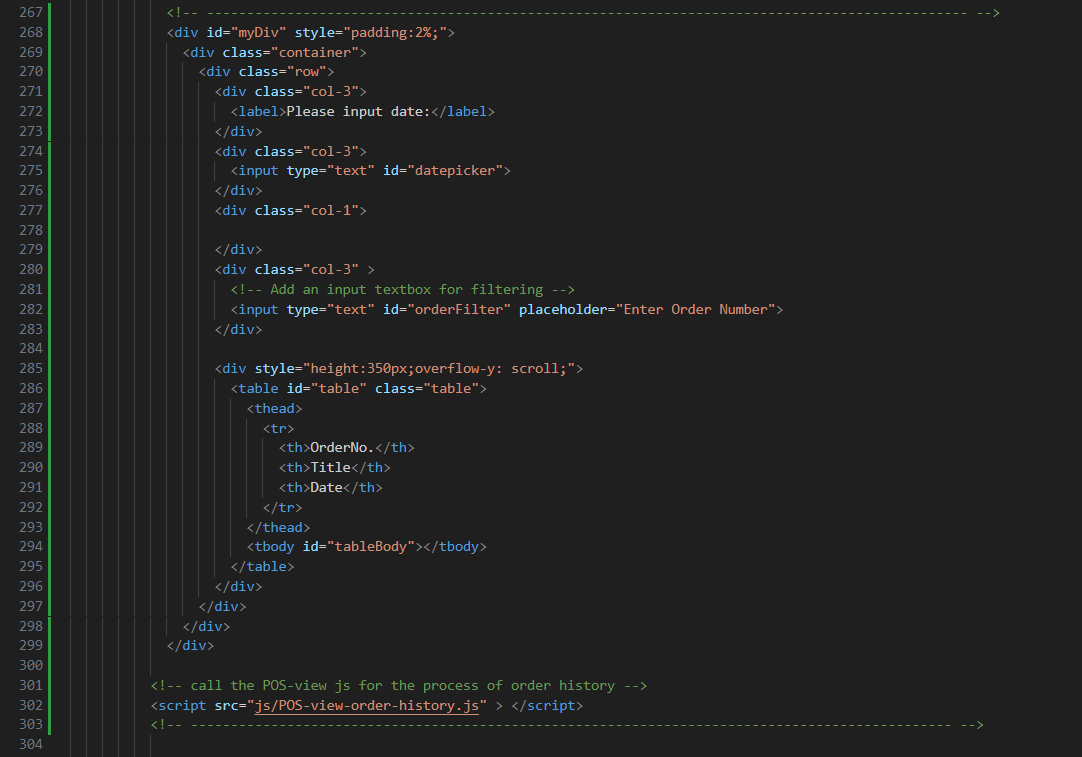
* The order number is auto generated every time we click the button “complete transaction”
* This is the logic behind that:



* Line439 to442 is where we generate random 10 digit during a page load
* Line 444 to 449 will generate random 10 digit every time we click on button for complete transaction
  1. We also create a slider using Jquery. If we click the button order history, there is a container slides from left and display all the records filterable by date and orderNo



* The code behind this is:

Reference File: **POS-view.html**  


* Line 268 – 299 is just the display for the table and line 302 is where we call the javascript function for slider and the display
* If we visit the file POS-view-order-history.js, we can see all the logic for filtering using calendar and orderNo and also the display of records. We separate this script to the POS-view.js to avoid override of javascript functionality.