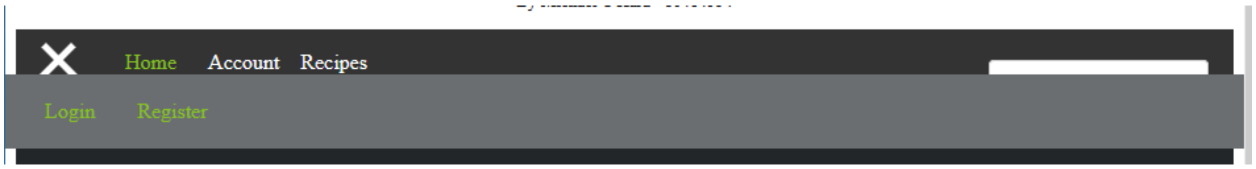
EE417 Assignment 2 – Report

Michael O'Hara - 16414554

Q1. Dynamic Menu Bar

There are 2 functions linked to the menu bar. The first method related to this is called `myfunction()` it begins on line 216 of my JavaScript file. This method has a simple function, it displays a button associated with the hamburger menu of my web app, and when the button is click this method is responsible for changing the icon from hamburger menu to an X, this function also calls the second function.

The second function is used to hide and reveal the links, it sets the style element called display to either block or none depending on what state its in when the button is clicked. Once the links are displayed the user can hover over account or recipes to access links to the subpages as shown below



Q2. Creating a Form with validation checks

For this task, the register and login pages are used.

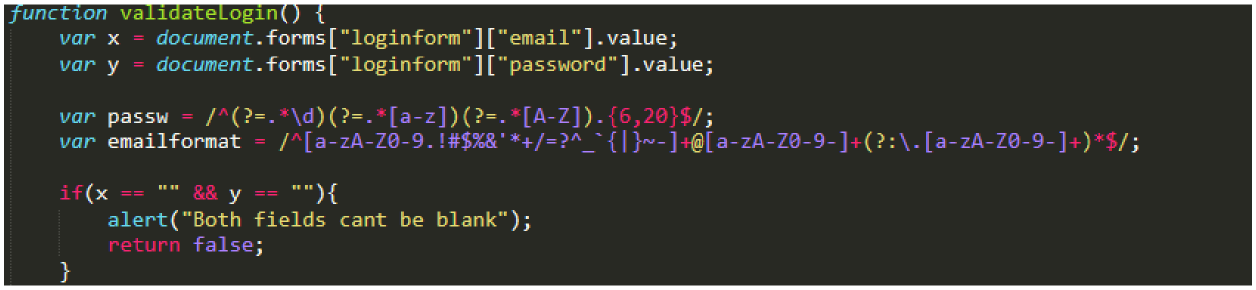
Firstly, the login page has JavaScript functionality starting on line 2 of "*myScript.js*" with the function `ValidateLogin`. This function is comprised of 3 basic validation checks. The first check makes sure that both fields are not blank. (In later assignments checks for credentials may be added). The next validation check is to make sure the email is of the correct format. This is done by comparing the entered value to a template regular expression, which can be seen on line 17. That expression being:

```
var emailformat = /^[a-zA-Z0-9.!#$%&'*+/=?^_`{|}~-]+@[a-zA-Z0-9-]+(?:\.[a-zA-Z0-9-]+)*$/;
```

The final check of this function is to check the password is of the correct format, this being at least one uppercase letter, at least one lowercase letter and at least one number. It also must be between 6 and 20 character in length. This is also done using a regular expression.

```
var passw = /^(?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{6,20}$/;
```

For the register page several other validation checks were implemented. These include one for an Eircode that must be of the format of 4 digits. These checks are broken down into 2 smaller functions encapsulate the business logic. These are called `ValidateAccount` and `ValidatePersonal` and begin on lines 33 and 65, respectively. These two functions are then called by another smaller function called `Validate` which is called once the form is submitted and then calls the check functions and if everything is correct will alert the user that the user was created successfully.



```
function validateLogin() {
    var x = document.forms["loginform"]["email"].value;
    var y = document.forms["loginform"]["password"].value;

    var passw = /^(?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{6,20}$/;
    var emailformat = /^[a-zA-Z0-9.!#$%&'*+/=?^_`{|}~-]+@[a-zA-Z0-9-]+(?:\.[a-zA-Z0-9-]+)*$/;

    if(x == "" && y == ""){
        alert("Both fields cant be blank");
        return false;
    }
```

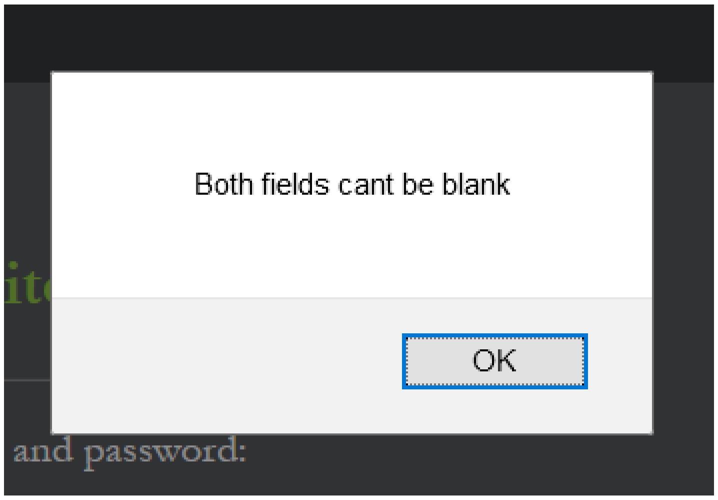
Q3 Storing form data and displaying to table

For this task I split it into 2 functions again, one for saving the data and one for displaying the data. The saving function is called `collect()`, it starts on line 100 of myscript.js. This function is linked to the submit button on the register page. This means once the user presses the submit button the previous sections function will be called to check the inputs and if they are all correct then collect will be called. The implementation of this function consists of finding the elements of the of form with id “register”, then it will iterate over these elements and get the name and value entered for each of them. It will then add the name and value to array and then convert the array to a JSON object and finally it will add that to localStorage with the key being the first name the user entered.

The second function is called `Display()` and it begins on line 128, this function finds an empty table element at the bottom of the register and then it uses a for loop to iterate through all the elements stored in local storage. This for loop will create a new row for the table for each element in local storage, it then creates “td” elements for each answer and then adds the answer to them and then appends these cells to the row and finally appends the row to the table on the page. This function is called when the Display button at the bottom of the page.

Q4 JavaScript Functionality

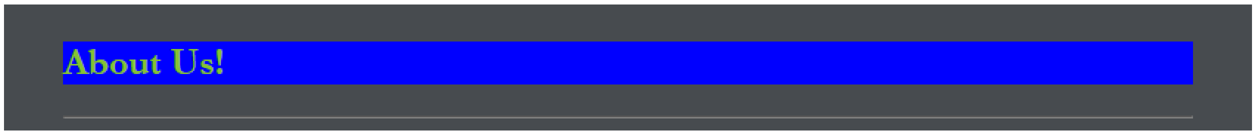
I have used several java script feature throughout the web app. Once example of this is the alert, this is used to show a message to user if they have completed a task or if they have encountered an error. An example of this can be seen below:



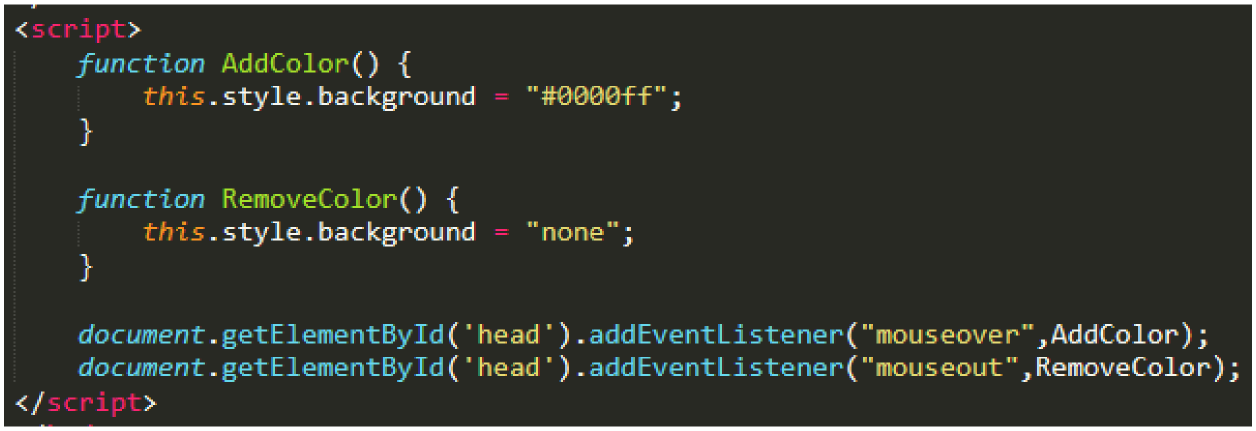
Both fields cant be blank

This alert occurs when the user has left both fields blank on the login screen. These also will appear if the format of user entered information such as email address is in the incorrect format. These are a great way of getting the users attention to display key info.

Another function I added was a mouse over, when a user hovers there mouse over a certain area on the page it will change the background color of the element to blue



This functionality was achieved with the following script, it adds the color on mouse over and removes the color after the mouse moves away from the area.

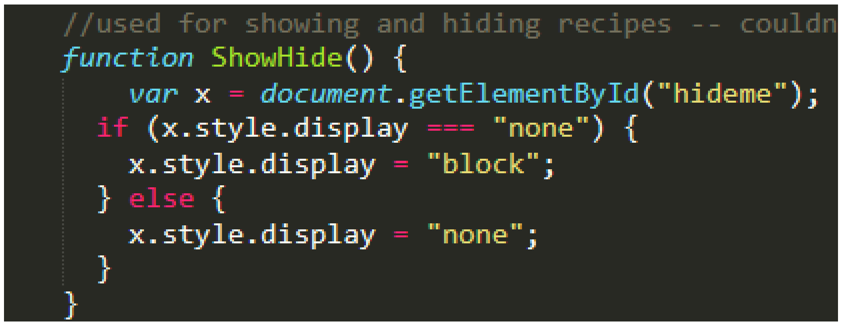


```
<script>
    function AddColor() {
        this.style.background = "#0000ff";
    }

    function RemoveColor() {
        this.style.background = "none";
    }

    document.getElementById('head').addEventListener("mouseover",AddColor);
    document.getElementById('head').addEventListener("mouseout",RemoveColor);
</script>
```

I used an on click function on my recipes pages, there is a button which can be click by the user it will hide the ingredients and steps and then once clicked again it will be shown again, this was done using the following snippet of code.



```
   //used for showing and hiding recipes -- couldn
   function ShowHide() {
       var x = document.getElementById("hideme");
     if (x.style.display === "none") {
       x.style.display = "block";
     } else {
       x.style.display = "none";
     }
   }
```

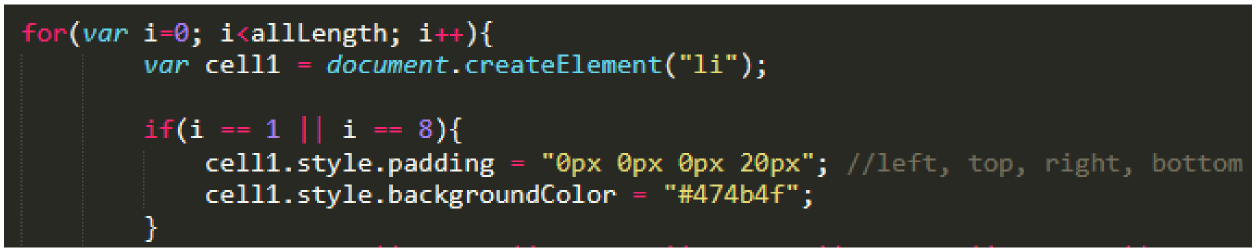
It sets the display style option to none or to block depending on the state that style option is in.

Q5 Capture and Display DOM of a page

For this task I used a function called `Display_DOM()` which begins on line 195 of myscript.js. This function collects all the html tags of the index page of my web app. It does this by calling `document.getElementById(“*”);`

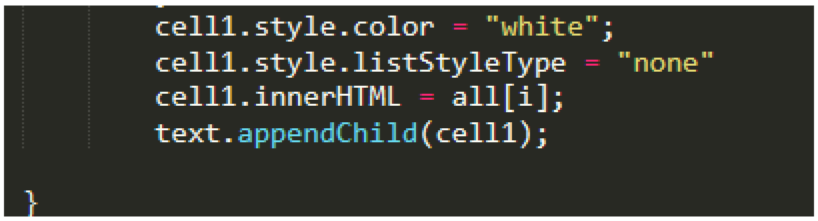
This method iterates through the webpage and collects all the html tags on the page, I then used a for loop in order to display these on the bottom of the page. This only occurs once the user clicks on the Get DOM button.

The implementation of the second half of this method is bad in my opinion but due to time constraints I was not able to find another way to complete this task. The call of `document.getElementById(“*”);` returns a HTML collection which iterating over caused a number of issues such as the elements being undefined when being returned. But the way it works is it will create a “li” element and then compare the index of which the for loop is at to a number of if statements and based on the condition it will apply styles to the “li” and then it will populate it with the content of the HTMLCollection at the index and then append it to the list.



```
for(var i=0; i<allLength; i++){
        var cell1 = document.createElement("li");

        if(i == 1 || i == 8){
            cell1.style.padding = "0px 0px 0px 20px"; //left, top, right, bottom
            cell1.style.backgroundColor = "#474b4f";
        }
```



```
        cell1.style.color = "white";
        cell1.style.listStyleType = "none"
        cell1.innerHTML = all[i];
        text.appendChild(cell1);

}
```