# Web Programming Course Project

**Project Topic：黄鑫璞’s Gameland**

School: Information Engineering

Major: Software Engineering

Name: Raymond Wangsa Putra / 黄鑫璞

Student ID: L20253008

2020-12-28

1. **Project Application Description**

This web is a web to play some mini game based on Javascript technology. This web work well-tested on Google Chrome, Mozilla Firefox, and Microsoft Edge. You need to login before you play the games. Beside the games, this web provides beautiful background color transititon to satify your eye, and the colors are customizeable up to four colors. This web will have login function, logout function, drag game function, math game function and settings function.

1. **Project Function Description**

* What technologies are used?

This web uses simple web technologies such as HTML, CSS, and Javascript. This web relies much on Javascript because of the animation, transition, and the games are Javascript based.

* What functions are included?

This web has several functions of commong website such as login and logout. This web also has settings function. The main interest of this web is the games. This web has two playable games. The first game is called “drag game” and the second game is called “math game”.

* What dose each function do?
* Login function will record your username to the storage.
* Logout function will clear your username from the storage and log you out from the website.
* The first game challenges you to help the bird to reach the nest by hovering your cursor/pointer around the bird.
* The second game challenge you to choose the right answer based on the given question, this game is good for kid to learn elementary math. Settings function to set a new username and change the background colors based on what the user prefers.

1. **Description of Key Functions**

**Login function:**

This is the most basic function in this web to get data from the user.

Below is the structure of the login form itself:

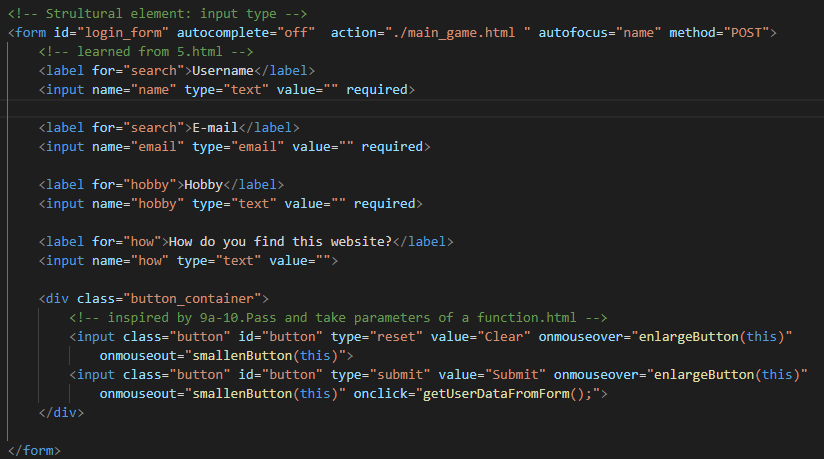


Image 1.1: Login form structure

The form is contained by <form> tag. Inside of the <form> there are tags for the input form using <input> tag. The form will retrieve username, e-mail, hobby, and how the user found this website by using the <input> tag. Every input is required except how the user found the website.

The <div class=”button\_container”> contains two input button. The first <input> is to clear the input by the user from the form. The first <input> has type=”reset” attribute. The second <input> tag is to submit the user input to store it in localStorage as the type=”submit” and also the second input has attribute onClick=”getUserDataFromForm();” this method is the method that will store the user input in localStorage, the method is located in “user\_settings.js” as the code is shown below:

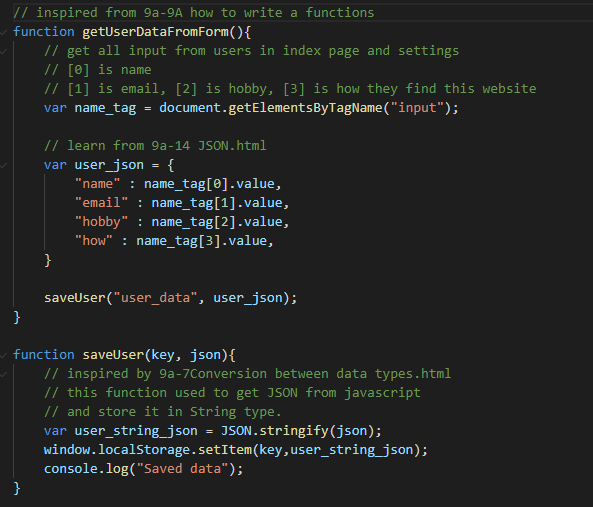


Image 1.2: saving user input from form for login and settings from JS.

getUserDataFromForm() will retrieve user input and convert it into a JSON object and saveUser() will store it in localStorage. Before storing it in localStorage, saveUser() will store convert the JSON object passed by getUserDataFromForm()into a string by using stringtify(), and then store the user data. After submitting the form, the user will be redirected to the game main hall.

As you can see in image 1.1 in page 2, The <div class=”button\_container”> contains two button. Both two buttons have same attributes: onmouseover="enlargeButton(this)" and onmouseout="smallenButton(this)". These methods will make animation for the button when user cursor/pointer hover around the button or leaving the button as you can compare it in image 1.3 and image 1.4 in page 4.

Text, letter

Description automatically generated

Image 1.3: Login form without user hovering

Image 1.4: Login form with user hovering

Code below (image 1.5) is the JS code to make the code animation when the user hover around the button. As the name said, enlargeButton(item) is to enlarge the button by making the height into 100% and make the button background color turn into blue, and smallenButton(item) is to make the button smaller by reducing the height 60% and thhen turn the background color to white. These methods are in “/Scripts/button\_animation.js”, the code is shown below in image 1.5

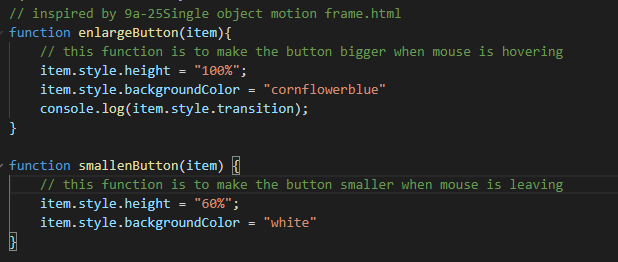


Image 1.5: methods for making the button bigger and smaller when the user hover around.

The styling of button inside <div class=”button\_container”> in image 1.1 for its original state (without user hovering) is shown below:

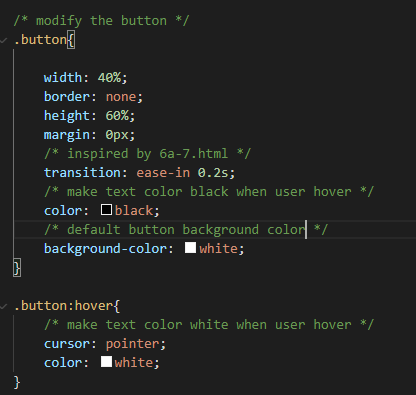


Image 1.6: CSS for normal state of the buttons.

Now we come into the styling of the login form function, I want to highlight about the form entrance animation where it will come from center and enlarge and then turn into its normal size as you can see below in image 1.8 and 1.9 in the next page.

Image 1.8: Login form entrance (2)

Image 1.7: Login form entrance (1)

This is the effect of <div class="center\_container"> where it will make the form larger and smaller for the entrace animation effect. We can see the code for the animation in the below code:

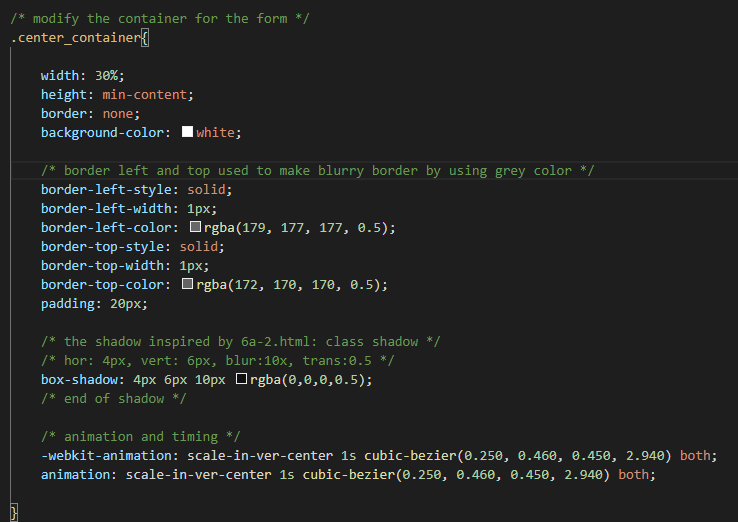


Image 1.9: CSS for the container for the form.

The CSS in image 1.9 has border property to make blurry border using gray color and box-shadow to make shadow effect for the login form. The animation for the enlarge and then normal state is using the scale-in-ver-center animation, with delay 1 seconds and cubic-bezier(). The keyframe is shown below:

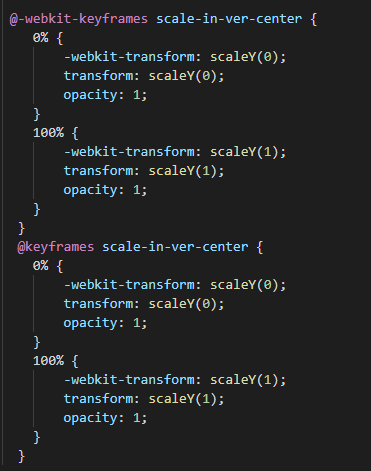


Image 1.10: Keyframes for scale-in-ver-center.

The keyframe will scale the height of the form using scaleY(0) at 0%, and then it will grow up to its normal state by using scaleY(1) at 100%. The bouncing effect is made due to my cubic-bezier() methods in the CSS shown in image 1.9 in page 7. The cubic-bezier() should only contain value from 0 to 1 (inclusively). But I use 2.940, to make the form larger than its normal size and then it will return to its normal size after the animation end.

**Game hall and Background color**

We cannot talk about the game with out the game hall, where the user selects the game to play. The game hall will have header for the user to navigate to settings or logout. The background colors also have ability to change like an animation, the user can change the colors in the settings. The background will show up to 4 colors.

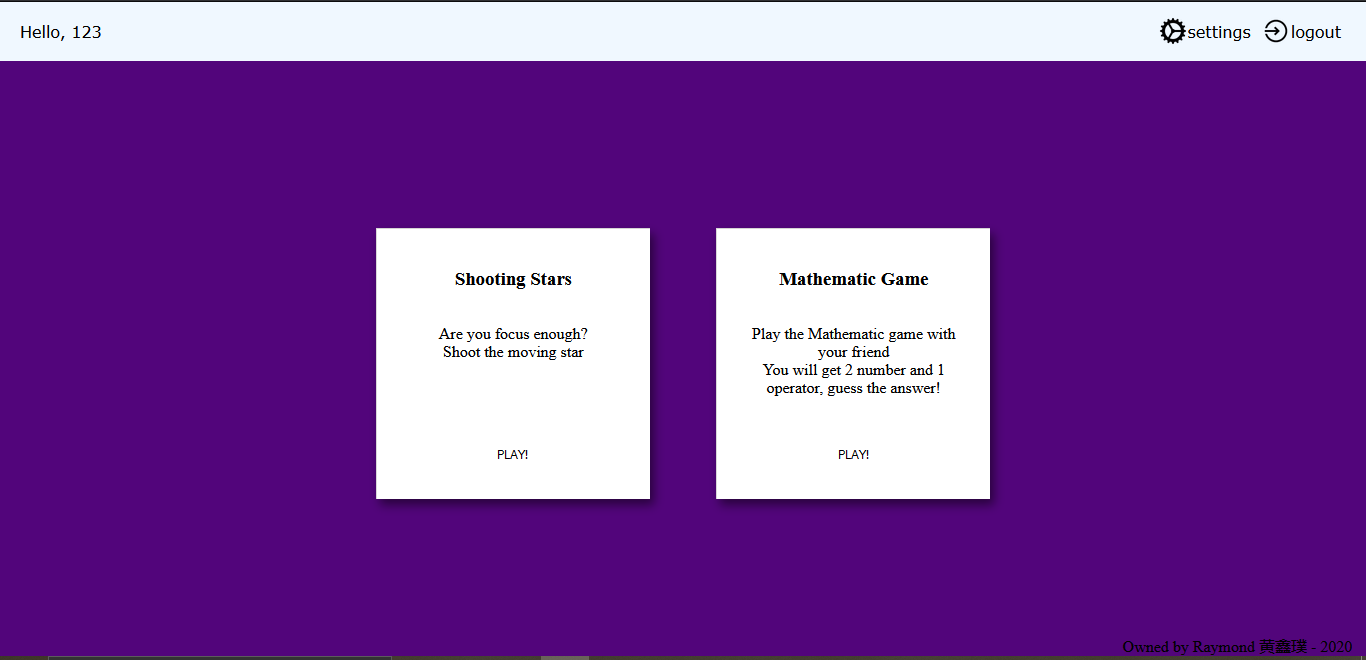


Image 2.1: the game hall appearance/view.

At the left side of the header, there is a greeting to the user and the right side of the header there is navbar. The code below will show the structure of the header:



Image 2.2: the header’s structure for game hall

As you can see, the <body> tag has 2 <nav> tags. The first tag is to contain the greetings and the second tag is to contain the navigation bar. The <body> tag has attribute onload="setFirstColorForBody(); setUserName(); alert('Data saved!');". The first method (setUserName()) is used to make the greeting call the user’s name. The second method (setFirstColorForBody()) is used to make the background color matches the color saved by the user. The third method (alert(‘Data saved!’)) is used to tell the user that their data is saved. The code for the first and second method is shown below:

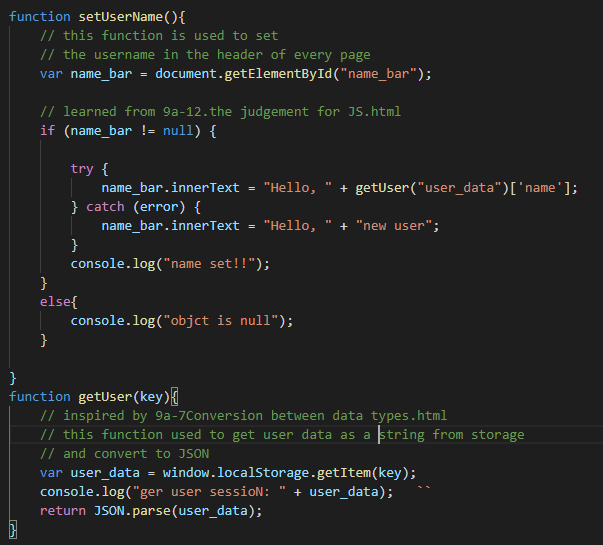


Image 2.3: the code to set the user’s name for the greetings.

This setUserName() is located in “/Scripts/user\_settings.js” and this method will get the element from the HTML with id “name\_bar” and then it will call getUser(“user\_data”)[‘name’] to get the user’s name from the storage and then apply it to the name\_bar.

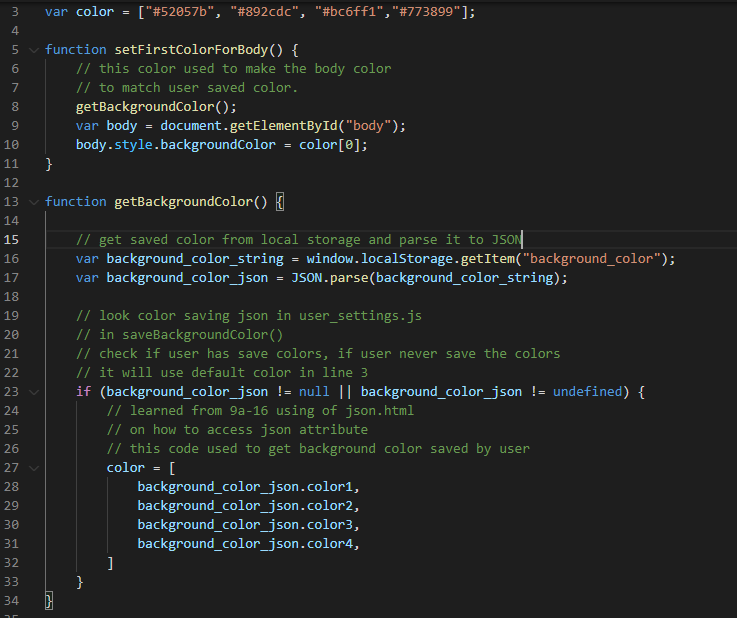
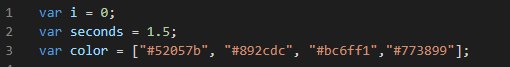


Image 2.4: the code to set the first color for background color animation.

Image 2.4 show the code to set the first color for the background by using setFirstColorForBody(). Inside of setFirstColorForBody(), we will call getBackgroundColor(), this method will get the color saved by the user from the storage. If the colors do not exist, it will not execute line 27 and will use default color from line 3.

Now we probably wondering how the background change color from time to time. First thing we need to include

<script src="../Scripts/background\_color.js"></script> in the HTML page inside the <head> tag. Inside the "../Scripts/background\_color.js" there are code as shown below:



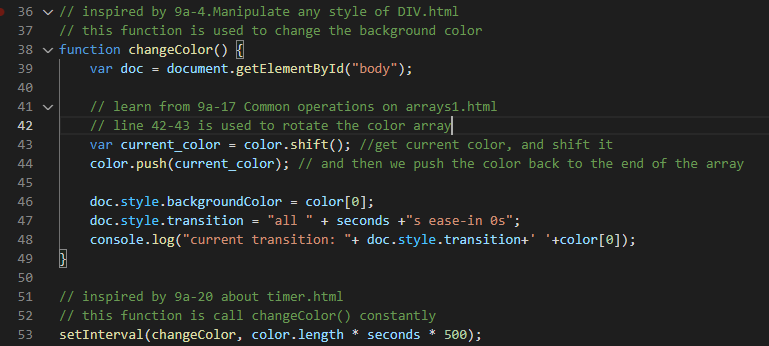


Image 2.5: the code to make the background color change constantly

changeColor() will get the body tag, and then get the current color in line 43, after they get it, the method will push the color the back to the end of the array. So the color will rotate all the time and the next color will always be at index 0 and we can access it by using color[0] like line 46. After that we should call it constantly by using setInterval() in line 53. With that, the background color will change all the time. Below is the example of color changing:

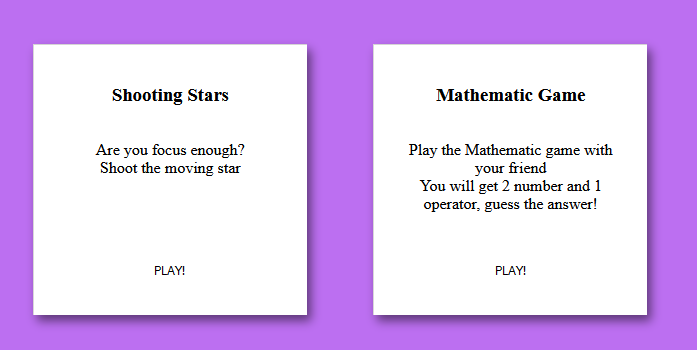
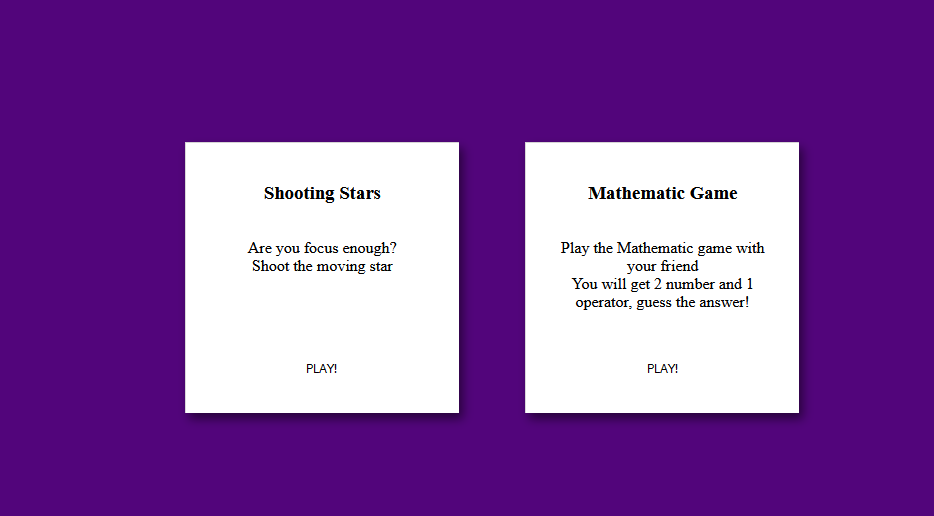


Image 2.7: Background color 2

Image 2.6: Background color 1

If the user has their own background colors saved, the onload in the <body> tag will load the user’s colors as I already explained in the previous page.

The user can go to play the game by using the “PLAY!” button at the center of the card. When the user hovers their pointer/cursor on the “PLAY!” button, the button will turn to blue color. This button uses the same code and concept as login form button which I already explained in image 1.3 and image 1.4 in page 4.

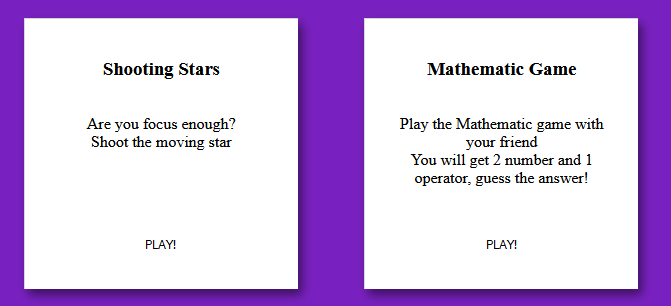


Image 2.8: game hall option

**Drag game function**

“Drag game” is one of two playable games in my website. Below is the preview for drag game:

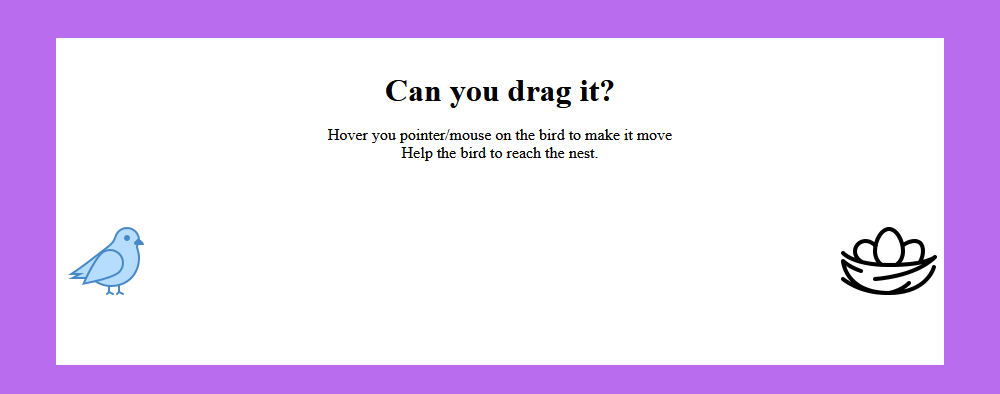


Image 3.1: drag game preview

The bird will move when the user hover around the bird and will back to normal when user’s cursor/pointer leave the bird. To make that happen, I will use <script src="../Scripts/drag\_game\_script.js"></script> in this page to include function needed to make the bird move, back, and winning condition. Also, I will add <link rel="stylesheet" type="text/css" href="../Style/drag\_game.css"> and <link rel="stylesheet" type="text/css" href="../Style/game.css"> to make normal state and styling for the game.

The structure of of the bird and the nest are quite simple as shown in the screenshot below:

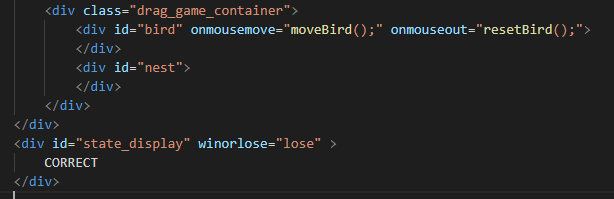


Image 3.2: drag game main strucutre for bird and nest

In the <div id=”bird> there are onmousemove=”moveBird();” and onmouseout=”resetBird();”. Those methods are used to make the bird move and back. Below it, there is a <div id=”state\_display> with attribute winorlose=”lose”, this div is used to show banner when someone win the game.

Now we will see the code and concept of the bird and nest. We will take a look at the bird and nest styling in CSS as shown below:

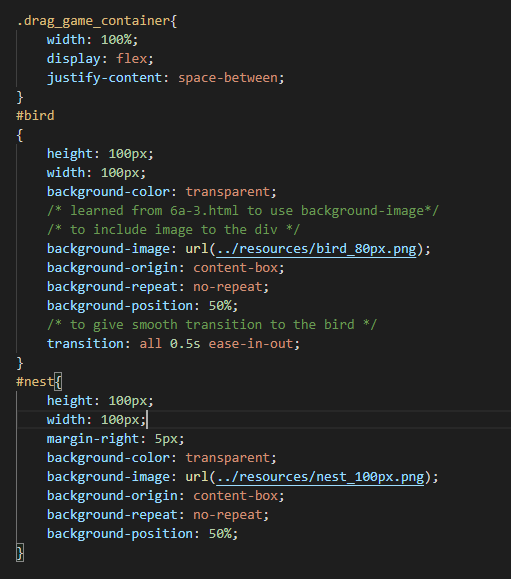


Image 3.3: drag game main strucutre for bird and nest

The drag\_game\_container that contain the bird and the nest will display in flex and will make the content justify with space-between, this will make the bird and the nest stay far apart. The bird and the nest will have size 100x100 px, both will use background-image to make the div have the image of bird and nest.

Now we jump into the code, to make the bird move we will use moveBird() as shown below:

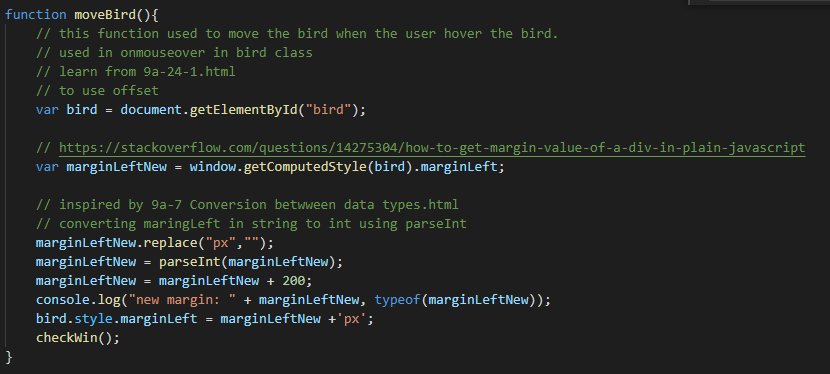


Image 3.4: moveBird() to make the bird move when the user hover

This method will get the element with id=”bird” and then get its margin-left, after it, it will add more 200px to the margin-left, causing the bird will move to the right like below:



Image 3.5: bird is moving toward the nest

After it, the method will call another method called checkWin() that looks like this below:

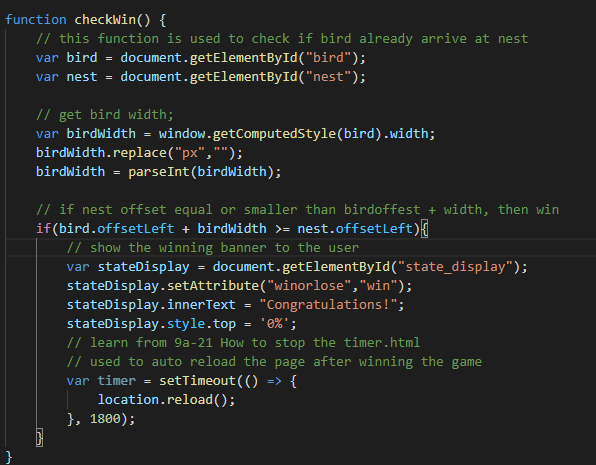


Image 3.6: code to check if user already win

This method will get the element with id=”bird” and id=”nest” and then check if they are already close enough or not. In the if(), we can see the method will get the id=”state\_display” and set its condition to “win”, adding “Congratulations!” to the div and show it to the user to tell the users they are already win. Below is the example when user win:

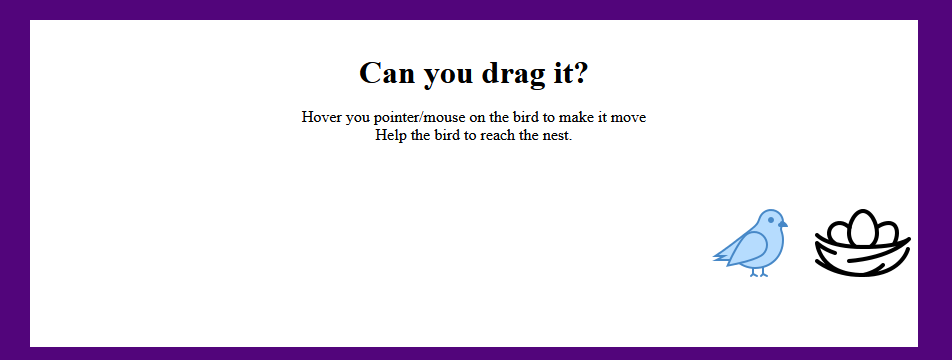


Image 3.7: The user almost win



Image 3.8: the web show the user they are winning

The next method is onmouseout=”resetBird();”, this method will make bird fly back to normal state when user cursor/pointer leave the bird, the code will looks like this:

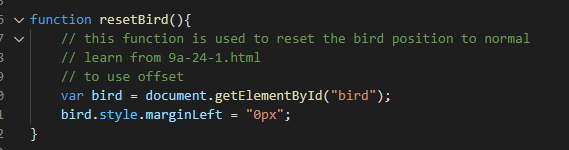


Image 3.9: method to reset the bird back.

This method work by getting the bird element, and then set the margin to 0px; causing the bird will fly back like in image 3.1 page 13.

This feature also has a header that looks like and use the same code and concept like game hall in page 8-9. The difference is, this header only has “back” button in its navbar to go to back to game hall

**Math game function**

For the math game, it will show 1 question with 3 options. The question and options will have structure by using table strucutre as shown below:

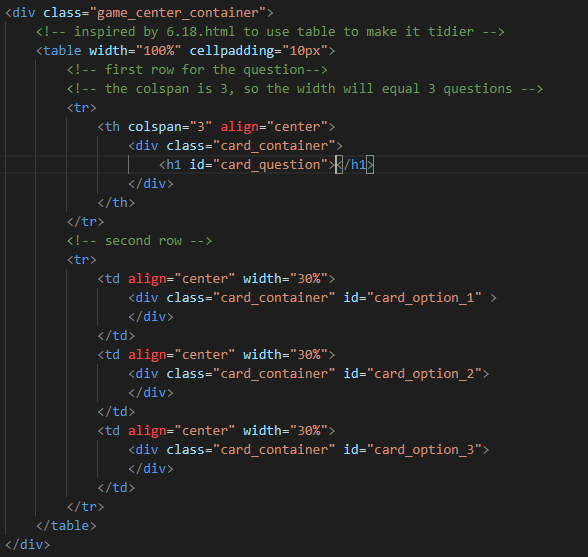


Image 4.1: math game structure using table

The first <tr> will contain the the question, with colspan= “3” so the width of the question will equal to 3 options which each has 1 colspan. The second <tr> will contain three <td> tags for 3 options. Each <td> will contain a div with class=”card\_container” and id for each option.

The questions will be loaded when the body onload is triggered in the code below:



Image 4.2: body for math game

The onload will call several methods I already explained before and than it will call makeQuestion(). makeQuestion() is responsible for generating question to the table. It will work based on the code below:

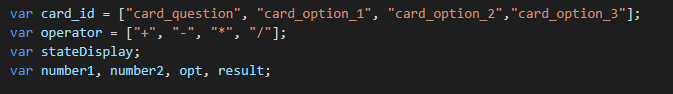


Image 4.3: initial variable for makeQuestion()

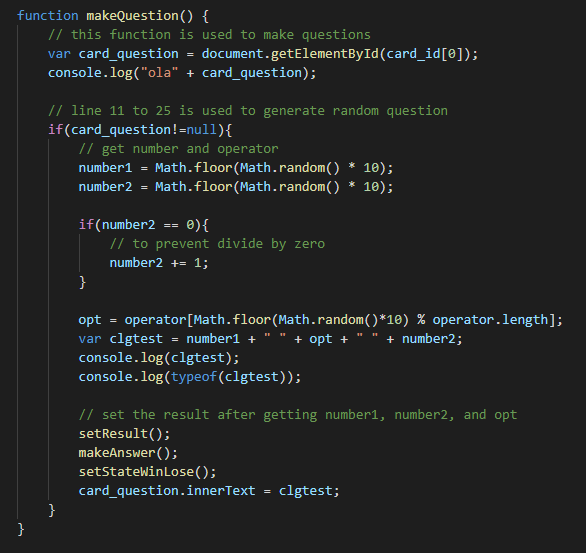


Image 4.4: makeQuestion()

In the image 4.3, it shows variable that contain the id for question and options elements, also operator that can be used by makeQuestions() to make question. In the makeQuestion(), it will generate a random question by using Math.random(). The setResult() is used to do calculation to get the correct result of the questions. The makeAnswer() is used to set the options generated by JS to the options in the <td> tags in the second <tr> in image 4.1. setStateWinLose() has the same purpose with drag game’s set checkWin() in image 3.6 in page 16, to show banner if the user selects the correct or wrong options. The code can be seen below:

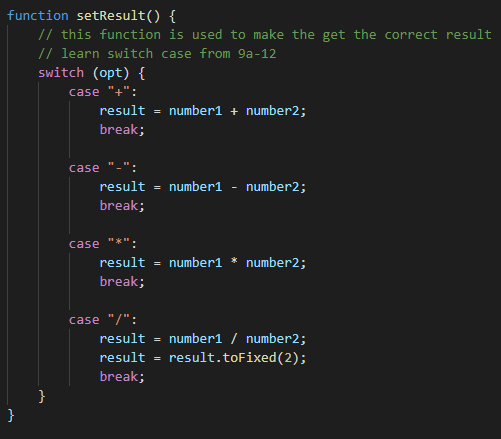


Image 4.5: setResult()

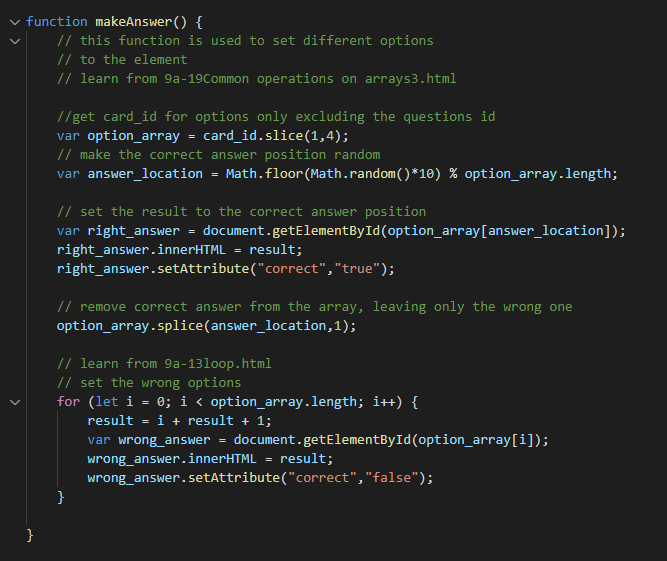


Image 4.6: makeAnswer()



Image 4.7: setStateWinLose()

When the user choose the wrong answer it will show a banner written “wrong answer” like this:

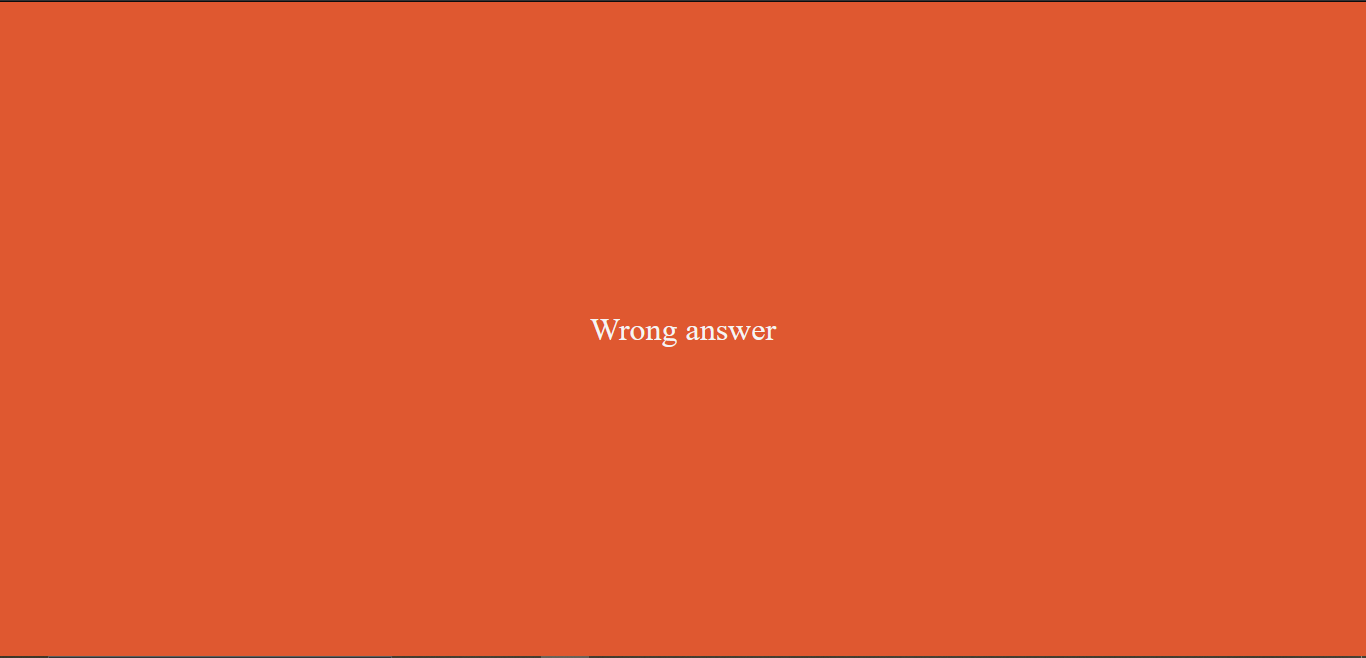


Image 4.8: Wrong answer banner

But if the user select the correct answer, it will show something like this:

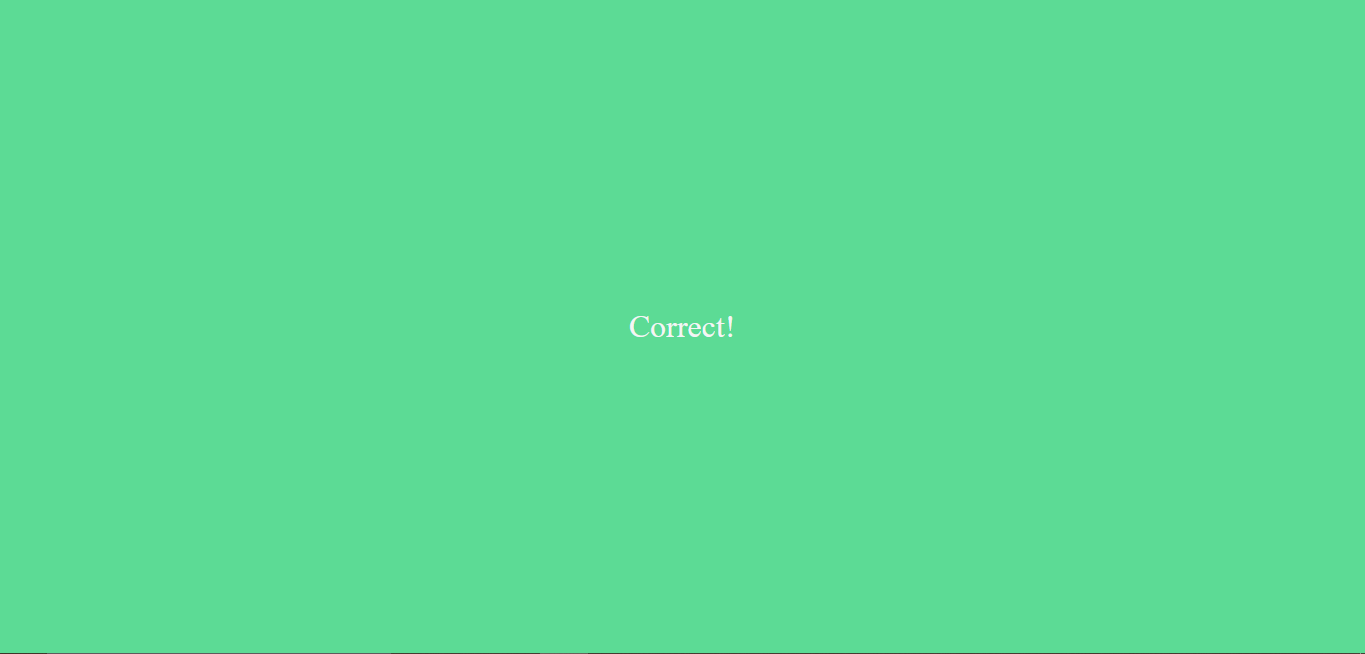


Image 4.9: Correct! Banner

**Settings function**

Why I make settings function? User can be bored with my default settings, so in this function I will let the user to choose the background colors.

Appearance:

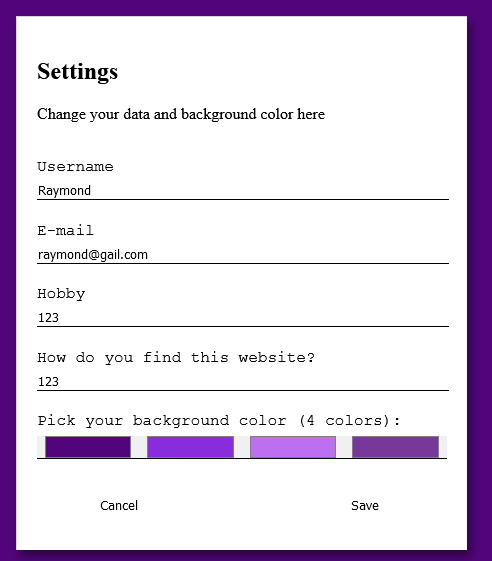


Image 5.1: settings form

The settings form almost have the same structure, code, and concept with login function I already explained in page 2 and 3. But this time, we have 4 colors input.

In the code below:



Image 5.2: settings form structure

As we can see, we added 4 input with type=”color”, each input has its own id. Also we add more method to the onclick in type=”submit” button. saveBackgroundColorFromForm(); will save the colors selected by the user with mechanism like this:

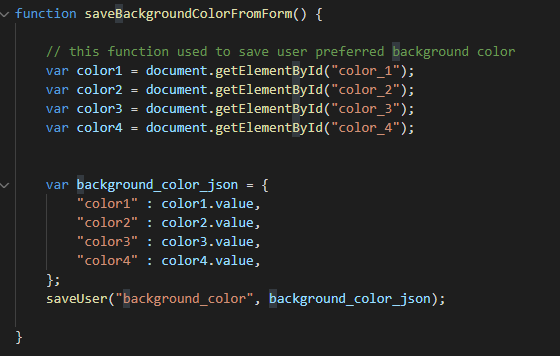


Image 5.3: method to save colors.

The method will get the color from the input, and save it in JSON, and using the saveUser() to store it in the storage. These colors will be used by getBackgroundColor() in page 10.

1. **Course Harvest**

From this course I have learned how to make form and input and applied it in login function and settings function.

From this course I have learned how to make header, main, footer structure like in 1.html and applied it to all of my webpage.

From this course I have learned how to make paragraph, big fonts by using <p> dan <h1> family tags 6.1-6.3.html and applied it to some parts of my page.

From this course I have learned how to modify <p> tag and <h1> tag and use it to modify it according to my need.

From this course I have learned how to use CSS attribute selector like 6.4.html and 6.4expansion.html and applied it to my need in /Style/game.css

From this course I have learned how to use hover like 6.5.html and applied it to several button in my project.

From this course I have learned how to use table tag like 6.18.html and used it in my game.

From this course I have learned how to use radius and box-shadow from example 6a-2.html

From this course I have learned how to use background image like 6a-3.html and applied it to drag game for the bird and the nest.

From this course I have learned to make transition and animation with keyframes like 6a-7.html and 6a-8.html and applied it to make my webpage animation more smooth.

From this course I have learned how to use basic javascript, like making variable, making if-else, using loops, making and modifying array, and making functions.

From this course I have learned how to create and modify JSON object like 9a-14.html, 9a-15.html, and 9a-16.html. I used it to store my user data in the local storage.

From this course I have learned how to set timer to call the function/method constantly by using setInterval, setTimeout and also clear the timer by using clear(). By learning this I can make animation color for the background.

From this course I have learned how to modify CSS style from Javascript like 9a-4.html and 9a.html

From this course I have learned how to make endless rolling display, and it inspired me to make idea to make infinite background color animation.

**课程设计评审意见**

（1）**Web Page Appearance**（20 marks）：Excellent ( ) , Good ( ), Satisfactory ( ), Unsatisfactory ( ), Poor ( )；

（2）**Web Page Function**（30 marks）：Excellent ( ) , Good ( ), Satisfactory ( ), Unsatisfactory ( ), Poor ( )；

（3）**Description of Key Function**（40 marks）：Excellent ( ) , Good ( ), Satisfactory ( ), Unsatisfactory ( ), Poor ( )；

（4）**Course Harvest**（10 marks）：Excellent ( ) , Good ( ), Satisfactory ( ), Unsatisfactory ( ), Poor ( )；

Teacher Signature：