Report

In the pong game that I coded, I created a total of 6 objects which I append to the canvas, the ball, scores and paddle for both the player and comp are created using the createElements() function instead of creating it in the html. This is because it will be easier to make changes to their attributes in the function. Also, I made all the objects which attributes can differ the same way to make it consistent. By creating the objects the same way, I would not get confuse when I am calling the objects. There are a few interesting parts that I discovered throughout the process of making this pong game.

One of it being the function to move the ball. This is because among the other codes, it took me the longest to make this function working well. The function can be categorise as a high order function as it takes a function as an input. It is not difficult to code it however the amount of code required due to the various number of possibilities made the code long and messy. Making the ball move is very time consuming as I need to try and error numerous times to fix an error. To make the code look more appealing and organise, I broke the moving ball function into 2 parts which is the x and y axis. By splitting the function, understanding the code got a lot easier. After understanding what is needed to do and know the possibilities which are needed, it is a lot easier to code it even though it is quite lengthy.

To make the ball bounce the other direction when it collide with something, I add the codes into the moving ball function using if statements. It helps to prevent unnecessary calling to set the attribute of the ball. To change the x attribute of the ball when It collides with either the wall or the paddles, all I did was just setting the ball direction times a negative 1. By that it will got from a constant increment of the speed value or a decrement of the speed value. It goes the same for the y attribute of the ball. That makes the ball bounce in the opposite direction when collides with something. If the ball were to collide with either the left or right wall, it will then reset the balls location. There will be two location the ball can reallocate which will depend on whether the ball hit the left or right wall. By hitting the walls, the scorer would then gain a point of it. The x and y axis function can be categorised as a lazy evaluation as the possibilities are coded in different if statements, the coded will only be perform if needed else it will just pass along.

Winning the game against the computer will be a bit of a challenge as the computer paddle is set to follow the ball's direction in a fixed speed. Due to the fact that it y value is set based on the ball's direction in a fixed speed, there are chances that it may go out of bounds. Therefore, I prevent it from happening by setting the y value of the comp paddle constant if it touches the boundaries of the canvas. As the computer hardly misses the ball, to let the player to stand a chance to win, I had created a sweet spot in the middle of the paddle which when the ball bounds of from the center of the paddle it will increase the ball speed by 3 but if the ball were to bounce of some other parts of the paddle, it will revert the ball speed back to its original. As the computer paddle is moving constantly following the ball in a fixed speed which is immutable, it might have a higher chance of missing the ball if the speed is increased. To make it more challenging, a sweet spot is also create in the computer paddle even though the probability of it hitting the spot is low. The sweet spot of the paddle's code are coded in the moving ball x axis function to minimise redundancy.

Next interesting part would be the keyboard inputs to control the player paddle. There are various ways to code but I choose to code the way I did because after a few try and error, my current codes make the paddle move smoothly and it also follow the Functional Relative Programming. This is because in my code, I include rxjs operators in it. By following the Functional Relative Programming, it makes it easier to understand even though it looked more complicated.

To end the game, either one of the players must reach a score of 7, I created a function to check if any player has the score of 7. The function is then called in moving ball function so that a separate checking method is not needed to carry out the ending game method. If either player or computer score reaches 7, the canvas will be emptied and a text displaying where display where player win or loses. A restart

function is created if player would like to play a new round. Player could only restart the game once it ends by pressing a keyboard input. The game would not restart mid game as there is a checker when a score reaches 7, it will indicate that the game has ended and the option to restart will be available. As it is a keyboard input, I use a similar way by using observables to code it. If the keyboard input is successfully inputted, the game will recall the entire game which would create the objects, initialise the scores back to 0 and remove the text which indicate whether player lose or win.

Overall, I think that my codes can be more efficient than it to be compared to now. I could apply a better code which follows the Functional Relative Programming. There could be other ways that doesn't make the codes as lengthy as it is currently.

As shown below is a sample picture of how my pong game would look like. I prepared instructions for players to move and control the ball so they do not need to figure out how to move it themselves. I also added a statement to tell players that they can only restart when the game ends end a useful tip to win against the computer.