IT5039 Client-side Development
Practical Task 3: Timed Shapes (LO 3, 4)
Jeshua Hertzke - 20210843

Practical Task 3 was carried out in one HTML file called index.html as specified in the assessment. The file is located in the same folder as this document. A breakdown to the tasks is documented below.

Steps:

The Task uses the script from a previous exercise on Canvas as a base, I copied the script and made the changes to the title, the variable and the class name to suit this task. I then created a state object with the states: bgColor, size and perCent with their initial values given in the assessment. This was done using a constructor with “props” inside the parentheses, “context” was not required for this exercise. I also added super to call the constructor of its parent class.

I then added more colours to the colours array using the ones supplied. To give the bgColor a random colour from that array, I created an arrow function called randColor which uses the Math methods to select a random index number of the colors array and return that color. At that time, I also created the other arrow function called randNum which takes one argurment which takes value of the highest random number you want and returns a random number. This function is for the size and perCent states in the timerTick function. For the keyword “this” to work in the tickTimer function, the keyword “this” needed to be binded to the constructor. I used the bind method to do this. I then updated the values of the associated properties in style variable to this.state.”state”, this linked them together.

For the shapes to change every second, I created a componentDidMount() function which contains a setInterval function. A setInterval function repeatedly calls the function inside it with an interval period that you specify. I chose 1000ms so the function repeated every second. The function that was repeated inside is called the timerTick funtion, this function contains a callback method called setState which creates a new version of the state. This function returns the states with the values being the Math functions I described earlier. This changes the value of the states which in turn changes the shapes every second.

I now had one shape that changes colour, size and shape every second. To have sixty shapes on the page, I created an empty array called renderData. I then used a for loop that creates a new shape through JXS and pushes it to the array and then repeats the process sixty times. I also added they property “key” to each shape which contained the number of iterations in the loop for its value, to reduce errors. I then inserted the array into the React.DOM render so that it inserts the array into the div in the HTML with the id of “container”.

I then ran the code through the live server in the Chrome Browser to see it was all working well.

Testing:

I wrote the code through Visual Studio Code and repeatedly tested the code through the Chrome inspector. As I created new functions, I used the console.log() method to print text onto the screen to ensure the function was working. I also had the console open while refreshing the code to check if any errors would arise. To make sure the constructor and states were working I tested them by using a coded colour before changing the value to a math function. At one point the shapes disappeared, this was because the bind method of the keyword “this” was incorrect. Once corrected, I retested, and everything was working well. I last issue I had was that then and again

there would be an empty space where a shape should have been. Upon stopping the setInterval function and checking the empty space on the inspector I found that the value of the bgColor was undefined. I checked the randColor function and found I had made a mistake by using + 1 at the end of the function. I removed it and retested the code, and everything worked perfectly.

I tested the code at different screen sizes and on different on the following browsers:

- **Chrome Browser** - Version 99.0.4844.51 (Official Build) (x86_64) - Latest version
- **Safari** - Version 15.3 (17612.4.9.1.8) – Latest version
- **Firefox** - Version 98.0 - Latest version

All browsers worked well.

Conclusion:

Ideally the React should be done through Create-react-app and the Shape class should be in its own js file but because this is a small exercise it was not necessary. I used arrow functions where possible and only the code required for the task to reduce the lines of code. I also enjoy working with React.

Jeshua Hertzke - 20210843