**Step:1**

First create a Timer Class which will have a constructor with 3 variables (durationInput, startButton, pauseButton). Then inside the constructor we are going to assign **‘this’** to the above parameters passed in the constructor. Below the constructor, we are going to create our first method **‘start()’.**

Then in our HTML file, we are going to create an input field with two buttons (Start {id=start} and Pause{id=pause}). Then outside our class, we are going to use the **document.querySelector** to get those id’s and assign them to the desired variables (durationInput, startButton, pauseButton) and pass them to the constructor by creating an object of the Timer class.

**Step:2**

First, we are creating another method called **‘tick()’**, then inside the start method, we are calling the tick method at the start and after that we are using the built-in function **setInterval(calling *function, time*)** to start the event after every one second. **setInterval(calling *function, time*)** function accepts two parameters **(this.tick{function}, 1000 {ms})**. Below the start method, we are creating our **‘pause()’** method in which we are using built-in function **clearInterval()** which will clear the timer (interval) of the start method.

**Step:3**

Then in our html file, we are setting a default value of the input field. In our **tick()** method, we are setting the value of **this.durationInput** to a new variable **timeRemaining**. This variable will be used for the count down as it will decrease by 1 after each second. We are then updating the **this.durationInput** to the new value by using the **value** tag at the end of **this.durationInput (this.durationInput.value)**.

**Step:4**

For a better approach, we are using setters and getters for step:3. In the getter, we are getting the time that is in the input field **(this.durationInput)**. We are converting it into float and then returning it to the setter. In the setter, we are updating the **this.durationInput (this.durationInput.value)** to **time** that was returned from the getter. In the **tick()** method, we are also updating the condition for checking if the count down gets below 0 or not. So, if the count down is less than or equal to 0, then we are going to call the **this.pause()** method and else we are going to decrease the value by 1.

**Step:5**

Now we are setting some optional callbacks that is passed in the constructor to display some messages in the console log. The callbacks are **onStart()** which displays a message when the timer stars, **onTick()** which displays message each time the timer is decreased by 1 and **onComplete()** which displays a message when the timer is completed. In the constructor, we are checking if any callbacks are passed or not. If present, then we will set it in the **‘this’** variable. Then each method is used in the appropriate position.

**Step:6**

Now we are going to modify our 3 methods **[onStart(), onTick() and onComplete()]**. Now, in our **onTick()** method, we have to pass the **timeRemaining** otherwise the offset will not be accurate. Inside the method, we are setting the attribute [ **circle.setAttribute('stroke-dashoffset', perimeter \* timeRemaining / duration - perimeter); {** element.setAttribute(name, value) **}]** and in our timer.js file, we are passing the **this.timeRemaining** to the **onTick()** method. To make the circle smooth, we have decreased the interval value to 20 and for that we also have to update the this.timeRemaining with 0.2.