```
import {formatTimeForDisplay} from './helper.js';

export class Graph {

    constructor(div, log, startTime, duration) {

        this.div = div;
        this.canvas = div.getElementsByTagName('canvas')[0];
        this.ctx = this.canvas.getContext('2d');
        this.divWidth = div.offsetWidth;
        this.width = this.ctx.canvas.width;
        this.height = this.ctx.canvas.height;
        this.log = log;
        this.startTime = startTime;
        this.duration = duration;
        this.timeScale = new TimeScale(this);

    }

    handleMouseOver(e) {
        var pos = e.offsetX;

        var clickTime = this.getTimeByPos(pos);

        var videoTime = (clickTime.getTime() - this.startTime.getTime()) / 1000;

        video.currentTime = videoTime;
        console.log(videoTime)
    }

    handleMouseMove(e) {
        var offset = e.pageX - this.div.offsetLeft;

        var time = this.getTimeByPos(offset);

        var timeDisplay = formatTimeForDisplay(time);

        this.currentPos.style.left = offset + 'px';
        this.timeDisplay.innerHTML = timeDisplay;

    }

    renderGraph(currentTime) {
        var ctx = this.ctx;

        // Clear the canvas
        ctx.clearRect(0,0,this.width,this.height);

        // Background
        ctx.fillStyle = '#333';
        ctx.fillRect(0, 0, this.width, this.height);

        // y axis
        ctx.strokeStyle = 'white';
        ctx.lineWidth = '1';
        ctx.beginPath();
        ctx.moveTo(0, this.height/2);
        ctx.lineTo(this.width, this.height/2);
        ctx.stroke();

        // Current time
```

```
62          ctx.beginPath();
63          ctx.moveTo(this.width/2,0);
64          ctx.lineTo(this.width/2,this.height);
65          ctx.stroke();
66
67          var startTime = new Date(currentTime.getTime() - (this.duration / 2))
68          // this.startTime = startTime;
69
70          // this.buildTimeScale(startTime);
71          this.timeScale.drawTimeScale(startTime);
72
73          var intervalData = this.log.getDataByTimeInterval(startTime, this.duration);
74
75          intervalData.forEach(function(data) {
76              var dataPos = (data.time.getTime() - startTime.getTime())/this.duration*this.width;
77              ctx.fillStyle = data.count > 0 ? 'white' : 'red';
78              ctx.fillRect(dataPos,(this.height/2),4,-1 * ((this.height/2) * data.count / 5));
79          },this)
80
81      }
82
83      getPosByTime(time, startTime) {
84          return (time.getTime() - startTime.getTime())/this.duration*this.width;
85      }
86
87      getTimeByPos(pos) {
88          return new Date(this.startTime.getTime() + pos/this.divWidth*this.duration);
89      }
90
91      getTimeScaleIncrement() {
92          var increments = [1,5,15,30,60,120,300,600];
93
94          var idealIncrement = this.duration/6/1000;
95
96          for (var i = 0; i < increments.length; i++) {
97              if (increments[i] > idealIncrement)
98                  break;
99          }
100
101         return (increments[i-1] ? increments[i-1] : increments[0]) * 1000;
102     }
103
104 }
105
106
107 class TimeScale {
108
109     constructor(graph) {
110     this.graph = graph;
111     this.incrementLength = this.getTimeScaleIncrement([1,5,10,15,30,60,120,300,600]);
112     }
113
114     getTimeScaleIncrement(increments) {
115
116         var idealIncrement = this.graph.duration/6/1000;
117
118         for (var i = 0; i < increments.length; i++) {
119             if (increments[i] > idealIncrement)
120                 break;
121         }
122
123         return (increments[i-1] ? increments[i-1] : increments[0]) * 1000;
```

```
124     }
125
126     drawTimeScale(startTime) {
127
128         // Setup the drawing paramaters
129         var ctx = this.graph.ctx;
130         ctx.strokeStyle = 'white';
131         ctx.lineWidth = '1';
132         ctx.font = '20px Arial';
133         ctx.fillStyle = "white";
134         ctx.textAlign = 'center'
135
136         // Get the graph's end time
137         var endTime = new Date(startTime.getTime() + this.graph.duration)
138
139         // Determine the first increment
140         var increment = new Date(Math.ceil(startTime.getTime() / 1000) * 1000);
141
142         // Output the increments
143         while (increment <= endTime) {
144             // Get the increment's position
145             var pos = this.graph.getPosByTime(increment, startTime);
146
147             // Output larger hashes and the time for the main increments
148             if (increment.getTime() % this.incrementLength == 0) {
149                 // Output the hash mark
150                 ctx.beginPath();
151                 ctx.moveTo(pos,this.graph.height - 20);
152                 ctx.lineTo(pos,this.graph.height);
153                 ctx.stroke();
154
155                 // Output the time text
156                 var intervalTime = formatTimeForDisplay(increment);
157                 ctx.fillText(intervalTime, pos, this.graph.height - 30);
158             }
159
160             // Otherwise output smaller hashes for each second that isn't a main increment
161             else {
162                 ctx.beginPath();
163                 ctx.moveTo(pos,this.graph.height - 10);
164                 ctx.lineTo(pos,this.graph.height);
165                 ctx.stroke();
166             }
167
168             // Add the interval time to the increment
169             increment = new Date(increment.getTime() + 1000);
170         }
171
172     }
173 }
```