```
import {TimeSlider} from './timeslider.js';
import {Log} from './log.js';
import {Graph} from './loggraph.js';

export class VideoPlayer {

    constructor(div) {
        this.div = div;
        this.videoLoaded = false;
        this.displayRequested = false;
        this.initialized = false;
        this.lastUpdate = new Date();
    }

    loadVideo(src) {
        // Get the video object
        this.video = this.div.getElementsByClassName('video')[0];

        // Load the video
        this.video.src = src;

        // Add an event listener to track when the video finishes loading since it can take some time
        this.video.addEventListener('loadeddata',() => {
            this.videoLoaded = true;
            this.initializePlayer();
        })

    }

    displayPlayer(log) {
        console.log("log",log)
        this.startTime = log.startTime;
        this.log = log;
        this.displayRequested = true;
        this.initializePlayer();
    }

    initializePlayer() {

        if (this.videoLoaded && this.displayRequested && !this.initialized) {

            // This variable prevents the video from inititalizing again if user backs away from the video and
returns
            this.initialized = true;

            // Display the video
            this.div.getElementsByClassName('videoLoading')[0].classList.add('hide');
            this.video.classList.remove('hide');

            // Set up the graph and render the first frame
            this.graph = new Graph(document.getElementById('graph'),this.log,this.startTime,30000);
            this.graph.renderGraph(this.startTime);

            // Set up the slider
            this.slider = new
TimeSlider(this.video,document.getElementById('playbackControl'),this.startTime,this.video.duration*1000);

            // When the slider updates the start time update startTime and rerender the graph
            this.slider.onNewStartTime = (newTime) => {
                const delta = newTime.getTime() - this.startTime.getTime();
                this.startTime = newTime;
```

```
                this.log.incrementTime(delta);
                this.graph.renderGraph(this.slider.currentTime);
            }

            // When the slider's time changes (this happens when seeking) rerender the graph so it matches
            this.slider.onTimeUpdate = (newTime) => {
                this.graph.renderGraph(newTime)
            }

            // Update the date at the top of the player
            document.getElementById('date').innerHTML = `${this.startTime.getMonth() +
1}/${this.startTime.getDate()}/${this.startTime.getFullYear()}`;

            // Listen for keybaord commands
            addEventListener('keyup', this.handleKeyPress.bind(this))

            // Start everything animating
            window.requestAnimationFrame(this.animate.bind(this));

        }

    }

    animate() {
        if (!this.video.paused) {
            this.slider.setTime(this.video.currentTime);
            this.graph.renderGraph(new Date(this.startTime.getTime() + Math.round(this.video.currentTime * 1000)))
        }
        window.requestAnimationFrame(this.animate.bind(this));
    }

    handleKeyPress(e) {
        if (e.keyCode == 38 || e.keyCode == 40) {
            const count = e.keyCode == 38 ? 1 : -1;
            this.log.addData(count, this.slider.currentTime);
            this.graph.renderGraph(this.slider.currentTime)

            if (this.onLogChange) this.onLogChange(this.log);
        }
    }

}
```