Project 1 - 音乐可视化

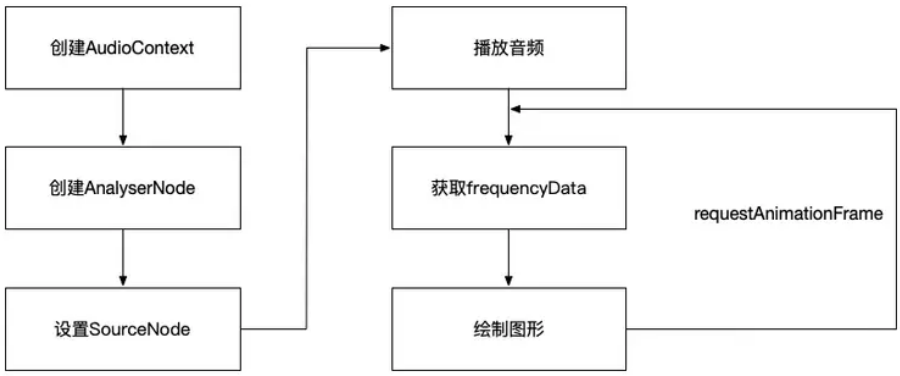
陈登仕

16307130153

1 实现原理

本项目选用 Web Audio API 进行音乐可视化开发。具体实现原理如下：

Web Audio 在可视化中进行的工作如下：



简单来说，就是（1）获取数据；（2）映射数据两个过程。

1 取数据

Web Audio API 在 **audio context（音频上下文）** 内处理音频 。因此，在对音频的任何操作之前，都必须要创建 AudioContext。

```
const context = new AudioContext();
```

而之后，为了获取音频的频率数据，进而实现音频的可视化，需要创建一个 AnalyserNode。

```
const analyser = context.createAnalyser();
analyser.fftSize = 8192; // 控制方块多少
```

其中，fftSize 是一个参数，用以决定 frequencyData 的长度。

之后，需要将音频结点，关联到 AudioContext 上，作为整个音频分析过程的输入。

```
let src = context.createMediaElementSource(audio);
src.connect(analyser);
analyser.connect(context.destination);
```

最后，获取 frequencyData，也就是获取频率数据，为可视化做好准备。

```
const bufferLength = analyser.frequencyBinCount;
const dataArray = new Uint8Array(bufferLength);
analyser.getByteFrequencyData(dataArray);
```

2 可视化

本项目利用 Canvas 2D 进行可视化。将 dataArray 映射为图形数据。不需要再考虑 Web Audio 的操作。

Canvas 是一个序列帧的播放。它在每一帧中，都要先清空 Canvas，再重新绘制。

它的代码框架为：

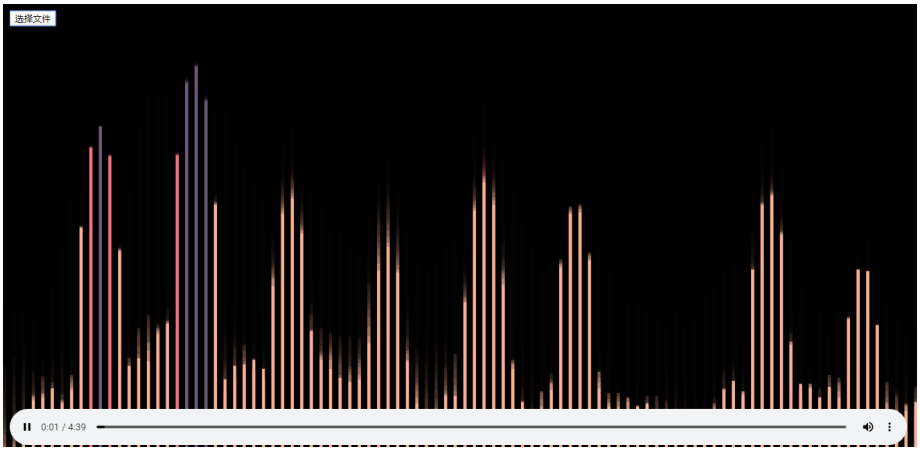
```
// 下述代码参考，掘金-网易云团队代码
function renderFrame() {
    requestAnimationFrame(renderFrame);

    // 更新频率数据
    analyser.getByteFrequencyData(dataArray);

    // bufferLength表示柱形图中矩形的个数
    for (var i = 0, x = 0; i < bufferLength; i++) {
        // 根据频率映射一个矩形高度
        barHeight = dataArray[i];

        // 根据每个矩形高度映射一个背景色
        var r = ...
        var g = 250 * (i / bufferLength);
        var b = 50;

        // 绘制一个矩形，并填充背景色
        ctx.fillStyle = "rgb(" + r + "," + g + "," + b +
")";
        ctx.fillRect(x, HEIGHT - barHeight, barWidth,
barHeight);

        x += barWidth + 1;
    }
}

renderFrame();
```

因此，只需要根据 dataArray 的数值，按照一定规则进行可视化即可。

2 程序说明

具体内容见代码注释。

效果展示



3 参考材料

Web Audio在音频可视化中的应用

Web Audio API的运用