Report 1：音乐节奏可视化

16307130065 仇均易

一、文件目录

1、code/index.html：主要页面

2、code/music.js：音频解码&频谱图绘制

3、code/style.css：页面样式

4、Report_1.pdf：项目报告

二、开发环境

Javascript + Visual Studio Code

三、算法原理

在该音乐旋律可视化Project中，选择采用AudioContext Web API以及HTML Canvas进行开发。AudioContext是一个专门用于音频处理的接口，简单来说工作原理就是将AudioContext创建出来的各种节点（AudioNode）相互连接，音频数据流经节点并作出相应的处理。

具体原理如下：

1、AudioContext处理音频

- **创建AudioContext对象**

  由于不同浏览器的兼容性问题，需要采用以下方式对不同浏览器进行配置，并创建对象：

  ```
  var AudioContext = window.AudioContext || window.webkitAudioContext;
  var audioContext = new AudioContext();
  ```

- **音频解码**

  当选择音频文件后，读取到的音频文件是二进制类型，因此需要利用AudioContext先进行解码，再进行后续操作，如下方decodeAudioData函数所示（当解码成功则调用传入的回调函数function(buffer)执行后续操作）：

  ```
  audioContext.decodeAudioData(input_file, function(buffer) { ... });
  ```

- **音频处理**

  在该部分中，需要两个音频节点对解码后的数据进行处理：

  1、**BufferSourceNode**：用于播放解码得到的Buffer的节点

  ```
  var BufferSourceNode;
  BufferSourceNode = audioContext.createBufferSource();
  ```

  2、**AnalyserNode**：用于分析音频频谱的节点

```
var AnalyserNode;
AnalyserNode = audioContext.createAnalyser();
AnalyserNode.fftSize = 256;//fftsize为快速傅里叶变换的大小
```

在完成以上部分后，将两个节点顺次连接即可：

```
BufferSourceNode.connect(AnalyserNode);
AnalyserNode.connect(audioContext.destination);
```

其中audioContext.destination是音频最终输出的目标，将最后节点连接到该点才能得到声音

- **播放音频**

  由上一部分知，BufferSourceNode用于播放音频，因此具体实现如下，采用start函数播放：

```
BufferSourceNode.buffer = buffer; //buffer为解码得到的数据
BufferSourceNode.start(0);
```

2、文件读取

HTML5支持文件选择以及读取的功能，因此可以实现不上传即播放的功能。采用HTML5中已有组件fileChooser绘制文件选择器，并利用FileReader异步读取文件，其中读取得到的input_file即为需要解码的二进制文件：

```
var LoadFile = function() {
    var fileReader = new FileReader();
    fileReader.onload = function(e) {
        input_file = e.target.result;
        DecodeMusic();
    }
    fileReader.readAsArrayBuffer(file);
}
```

3、Canvas绘制频谱

以上两个部分完成了音频文件读取以及处理的工作，在该部分中利用canvas&解析得到的数值绘制需要的频谱图：

- **数据解析**

  在第一部分中，通过AnalyserNode分析了音频频谱节点，需要采用以下方式获取分析得到的频谱数据

  （数据均为大小“0-fftsize”的数值，利用此数据即可控制绘制频谱条的高度等状态）：

```
var MusicArray = new Uint8Array(AnalyserNode.frequencyBinCount);
AnalyserNode.getByteFrequencyData(MusicArray);
```

- **Canvas绘制**

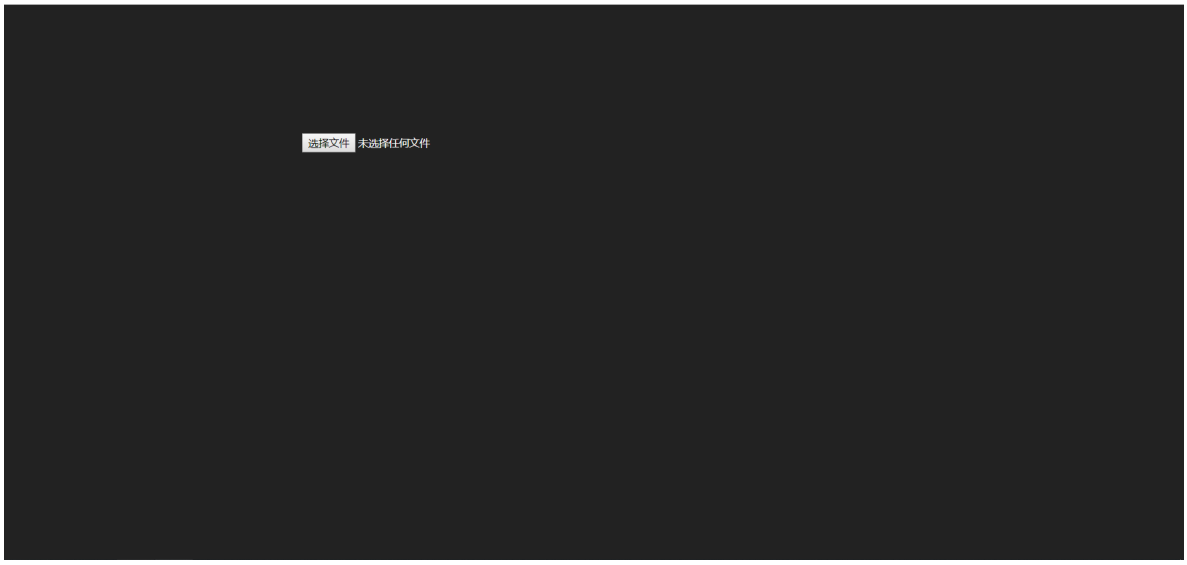
  对于canvas的主要绘制方式，即利用beginPath、moveTo、lineTo、stroke等函数，在该报告中不再赘述。

  由于需要动态显示页面，因此需要利用requestAnimationFrame函数进行动画绘制，主要目的是类似递归调用render函数从而实现对页面的动态渲染：

```
var render = function() {
    window.requestAnimationFrame(render);
}
```

四、程序说明

- 点击**index.html**即可打开如下图所示页面：



- 点击**选择文件**选择本地音频文件，选择完成后提示音频解码中；当音频完成解码，音乐自动播放并在页面中绘制频谱图，如下图所示：