代码说明

本项目是基于HTML5 Audio API 实现的。

首先创建一个 `Visualize` 对象。

```
var Visualize = function () {
    this.file = null;  // the file to be dealt with
    this.filename = null; // the name of file mentioned above
    this.audiocontext = null; // the contextx of the audio to be dealt with
    this.source = null; // save the audio
};
```

代码中其实已经注释好，这里再做进一步说明。 `file` 是要处理的文件， `filename` 是文件名， `audiocontext` 是进行音频处理的上下文， `source` 用于保存音频。

统一浏览器API，因为不同浏览器的API不同，会有不兼容的情况，我们要考虑进去。

```
//Unified browser API
Visualize.prototype._prepareAPI = function () {
    //AudioContext
    window.AudioContext = window.AudioContext || window.webkitAudioContext ||
window.mozAudioContext || window.msAudioContext;
    //requestAnimationFrame
    window.requestAnimationFrame = window.requestAnimationFrame ||
window.webkitRequestAnimationFrame || window.mozRequestAnimationFrame ||
window.msRequestAnimationFrame;
    try {
        this.audioContext = new AudioContext();
    } catch (e) {
        console.log('this browser does not support audioContext，please try Chrome or
Firework.');
    }
}
```

`Chrome` 中为 `window.webkitAudioContext` ， `Firefox` 中为 `mozAudioContext` ，采用 `||` ，遇到真值就返回。通过这种方式，我们可以统一浏览器的API。此外， `IE` 浏览器不支持，所以要加一个 `try catch` 语句。

接着需要加载音频文件，有三种方式，这里采用“文件选择”的方式。监听 `input` 的 `onchange` 事件，此事件在 `input` 的值改变时就触发。

```
//add event to the file
Visualize.prototype._addEventListenr = function () {
    var that = this,
        audioInput = document.getElementById('uploadFile'),
        drapContainer = document.getElementsByTagName('canvas')[0],
        selected = document.getElementById('selected');
    //monitor a event
    audioInput.onchange = function () {
        //if 'canceled',then the length will be 0
        if (audioInput.files.length !== 0) {
            that.file = audioInput.files[0];
```

```
            that.fileName = audioInput.files[0].name;
            selected.innerHTML = audioInput.files[0].name;
            that._start();
        }
    }
};
```

这里获取文件之后，就把文件赋值给 `Visualize` 对象的 `file` 属性和 `filename` 属性，方便之后通过 `this.file` 来访问该文件。

得到文件之后，要把获取的文件转化为 `ArrayBuffer` 格式，才能够传给 `audiocontext` 进行下一步编码。

```
Visualize.prototype._start = function () {
    var that = this, //meant Visualize
        file = this.file,
        fr = new FileReader(); //instantiate a FileReader to read the file
    fr.onload = function (e) {
        var fileResult = e.target.result; // get the ArrayBuffer
        var audioContext = that.audiocontext;
        audioContext.decodeAudioData(fileResult, function (buffer) {
//decode successfully
            that._visualize(audioContext, buffer);
        }, function (e) {
            console.log('decode failed!');
        });
    };
    // transfer to FileReader
    fr.readAsArrayBuffer(file);
};
```

在 `audioContext.decodeAudioData` 的回调函数里，解码完成后，把 `audioContext` 和 `buffer` 传递给 `_visualize` 方法进一步处理。

接下来要播放音频。

```
Visualize.prototype._visualize = function (audioContext, buffer) {
    //transfer to BufferSource
    var audioBufferSouceNode = audioContext.createBufferSource(),
        analyser = audioContext.createAnalyser();
    audioBufferSouceNode.connect(analyser);
    analyser.connect(audioContext.destination); //speaker
    audioBufferSouceNode.buffer = buffer;
    audioBufferSouceNode.start(0); //start braodcasting
    this._draw(analyser); //final visualization
};
```

这个函数封装了播放音频和绘制频谱两个功能，具体实现了播放音频，调用了绘制频谱函数。

最后一步是音乐的可视化。

```
Visualize.prototype._draw = function (analyser) {
    //sampling
```

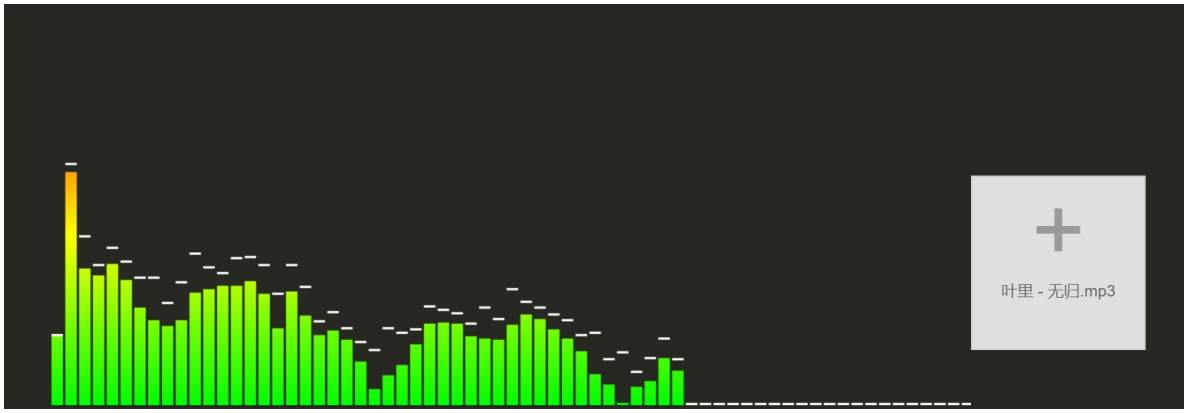
```
    var canvas = document.getElementById('Visualizer'),
        cwidth = canvas.width,
        cheight = canvas.height - 2,
        meterWidth = 10, //spectrum bar width
        gap = 2, //spectrum bar spacing
        capHeight = 2,
        capStyle = '#fff',
        meterNum = 800 / (10 + 2), //number of spectrum bars
        capYPositionArray = [];
    ctx = canvas.getContext('2d');
    gradient = ctx.createLinearGradient(0, 0, 0, 300);
    gradient.addColorStop(1, '#0f0');
    gradient.addColorStop(0.5, '#ff0');
    gradient.addColorStop(0, '#f00');
    var drawMeter = function () {
        var array = new Uint8Array(analyser.frequencyBinCount);
        analyser.getByteFrequencyData(array);
        var step = Math.round(array.length / meterNum); //calculate the sampling step
size
        ctx.clearRect(0, 0, cwidth, cheight);
        for (var i = 0; i < meterNum; i++) {
            var value = array[i * step]; //get the current energy value
            if (capYPositionArray.length < Math.round(meterNum)) {
                capYPositionArray.push(value); //initializes the array that holds the
cap position and pushes the data from the first screen into it
            };
            ctx.fillStyle = capStyle;
            //start drawing the cap
            if (value < capYPositionArray[i]) {
                ctx.fillRect(i * 12, cheight - (--capYPositionArray[i]), meterWidth,
capHeight);
            } else {
                ctx.fillRect(i * 12, cheight - value, meterWidth, capHeight);
                capYPositionArray[i] = value;
            };
            //start drawing spectrum bars
            ctx.fillStyle = gradient;
            ctx.fillRect(i * 12, cheight - value + capHeight, meterWidth, cheight);
        }
        requestAnimationFrame(drawMeter);
    }
    requestAnimationFrame(drawMeter);
};
```

首先要获取某一刻的频谱，最后要让每时每刻的频谱都有所体现，让画面动起来。

最后，再写一些 `css` 样式即可。

结果示例

用 `Chrome` 打开 `index.html` 即可。



参考内容

[1] https://developer.mozilla.org/en-US/docs/Web/API/FileReader

[2] https://developer.mozilla.org/en-US/docs/Web/API/AudioContext

[3] https://developer.mozilla.org/en-US/docs/Web/API/window/requestAnimationFrame