

# 数据可视化 音乐可视化

李春芳

计算机与网络空间安全学院

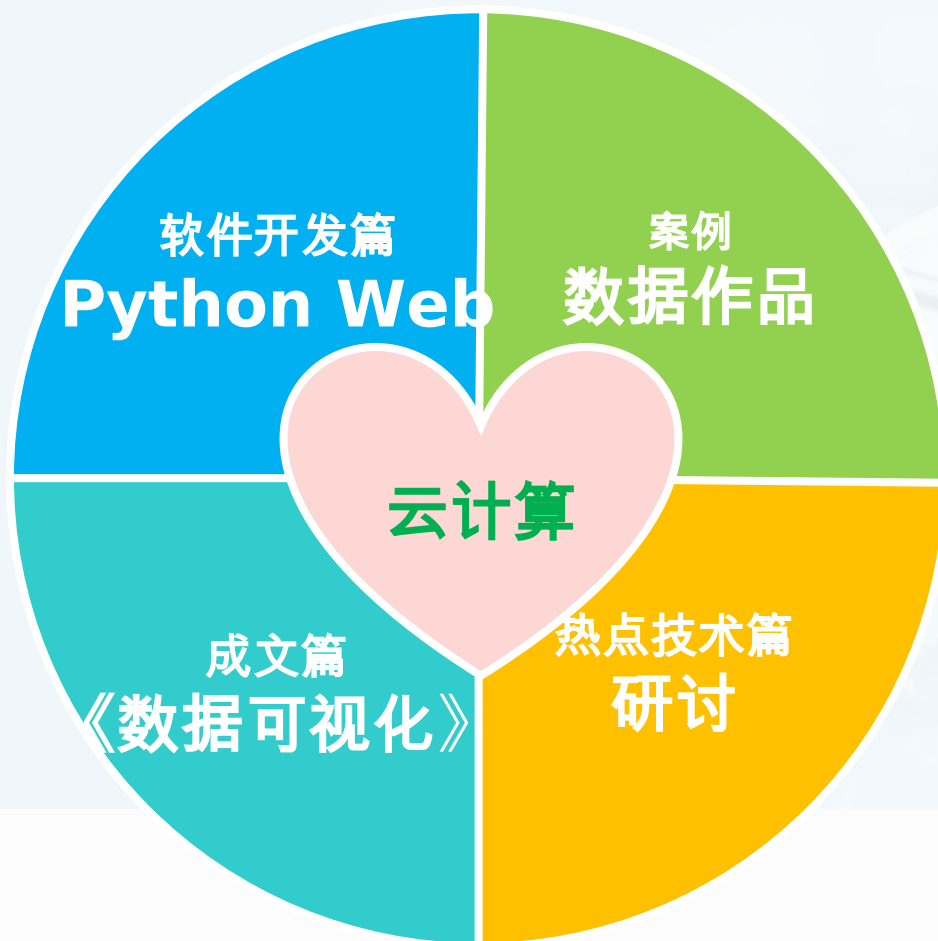
2020-6-10



中国传媒大学  
COMMUNICATION UNIVERSITY OF CHINA

# 实践课程概述

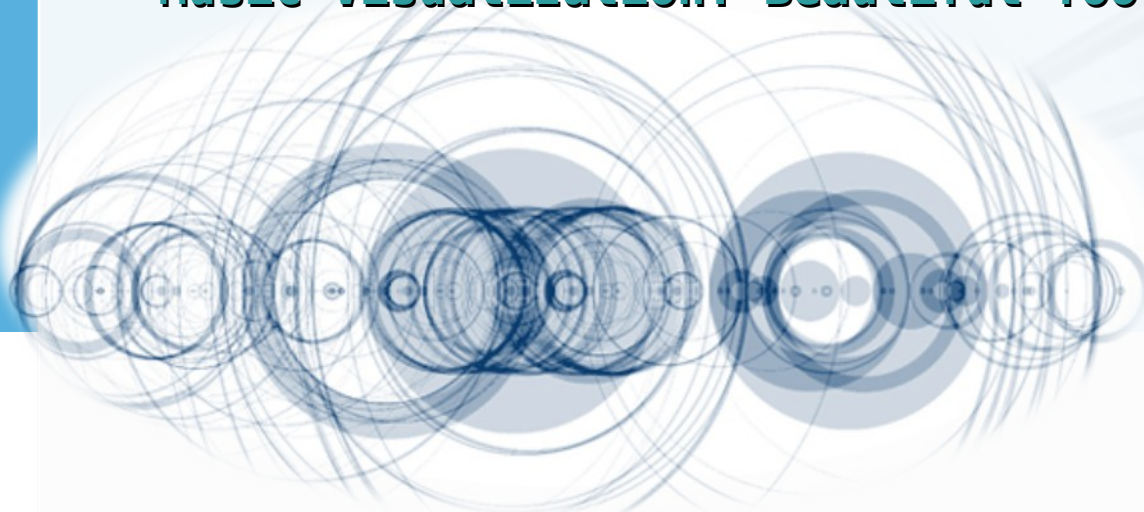
课程目标：软件开发中的数据可视化



# 第九章

# 音乐可视化

Music Visualization: Beautiful Tools to 'See' Sound





WIKIPEDIA  
The Free Encyclopedia

[Main page](#)  
[Contents](#)  
[Featured content](#)  
[Current events](#)  
[Random article](#)  
[Donate to Wikipedia](#)  
[Wikipedia store](#)

[Interaction](#)

[Help](#)  
[About Wikipedia](#)  
[Community portal](#)  
[Recent changes](#)  
[Contact page](#)

[Tools](#)

[What links here](#)  
[Related changes](#)  
[Upload file](#)

[Article](#) [Talk](#)

[Read](#) [Edit](#) [View history](#)

## Music visualization

From Wikipedia, the free encyclopedia



It has been suggested that *Music visualization (simulation)* be merged into this article. ([Discuss](#)) *Proposed since July 2017.*



This article **needs additional citations for verification**. Please help [improve this article](#) by [adding citations to reliable sources](#). Unsourced material may be challenged and removed. *(February 2009)* ([Learn how and when to remove this template message](#))

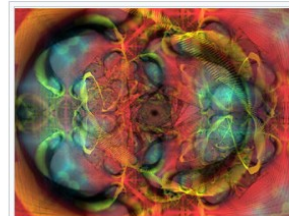
**Music visualization** or **music visualisation**, a feature found in electronic music visualizers and [media player software](#), generates animated [imagery](#) based on a piece of music. The imagery is usually generated and rendered in real time and in a way synchronized with the music as it is played.

Visualization techniques range from simple ones (e.g., a simulation of an [oscilloscope](#) display) to elaborate ones, which often include a plurality of [composited](#) effects. The changes in the music's [loudness](#) and [frequency spectrum](#) are among the properties used as input to the visualization.

Effective music visualization aims to attain a high degree of visual [correlation](#) between a musical track's spectral characteristics such as frequency and [amplitude](#) and the objects or components of the visual image being rendered and displayed.

### Contents [\[hide\]](#)

- [Definition](#)
- [History](#)



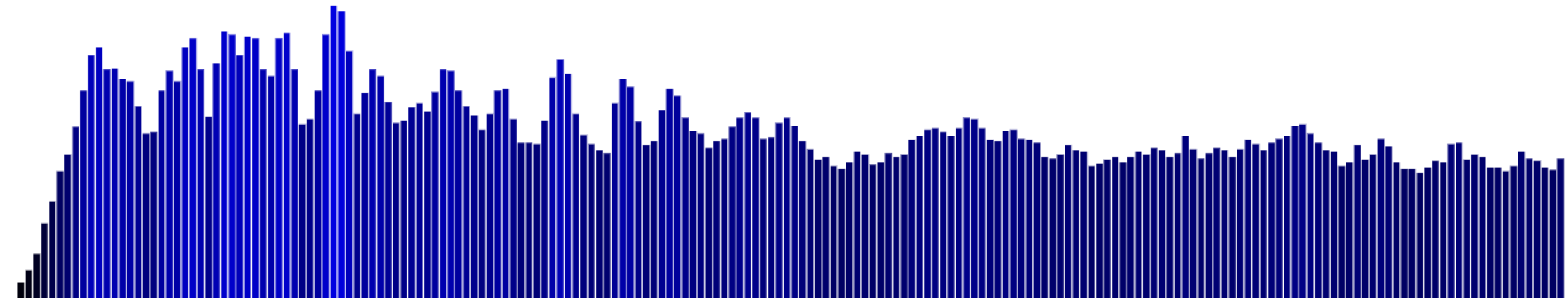
Screenshot of preset included in [MilkDrop](#), a PC based music visualization software (version 1.04d, 2001)

# 基于 D3 的音乐可视化

## Music Visualization with D3.js

- ◆ <http://bignerdranch.github.io/music-frequency-d3/>

Play the Audio Pause the Audio Increase Volume Decrease Volume



# 基于 JS 的音乐可视化

## Music Visualization with D3.js

```
<audio id="audioElement" src="./audio/Odesza - Above The Middle.mp3"></audio>
<div>
  <button onclick="document.getElementById('audioElement').play()">Play the Audio
</button>
  <button onclick="document.getElementById('audioElement').pause()">Pause the Audio
</button>
  <button onclick="document.getElementById('audioElement').volume+=0.1">Increase Volume</button>
  <button onclick="document.getElementById('audioElement').volume-=0.1">Decrease Volume</button>
</div>
```

← → ↻ ⓘ 不安全 | bignerdranch.github.io/music-frequency-d3/

Play the Audio

Pause the Audio

Increase Volume

Decrease Volume





# 基于 JS 的音乐可视化

## Music Visualization with D3.js

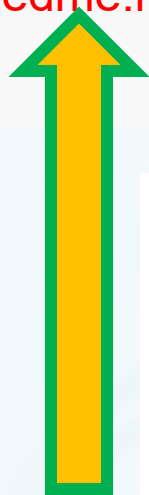
|| 0:23 / 4:13 ————— 🔊 ⋮

### 音乐控件

```
<div align="right">
```

```
  <audio id="audioElement" src="needme.mp3" controls="controls"></audio>
```

```
</div>
```



# 基于 JS 的音乐可视化 Music Visualization with D3.



```
<script>
```

```
// 取音乐的频率
```

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

```
var audioElement =  
document.getElementById('audioElement');
```

```
var audioSrc =  
audioCtx.createMediaElementSource(audioElement);
```

```
var analyser = audioCtx.createAnalyser();  
audioSrc.connect(analyser);  
audioSrc.connect(audioCtx.destination);
```

```
//var frequencyData = new  
Uint8Array(analyser.frequencyBinCount);
```

```
var myhist = document.getElementsByTagName("rect");
```

```
var frequencyData = new Uint8Array(200);
```





# 基于 JS 的音乐可视化

## Music Visualization with D3.js

```
◆ function every20 () {  
◆     analyser.getBytesFrequencyData (frequencyData) ;  
  
◆     for (var idx in myhist) {  
◆         if (myhist[idx].getAttribute && frequencyData[idx])  
◆         {  
◆             myhist[idx].setAttribute ("y", y-frequencyData[idx]) ;  
◆             myhist[idx].setAttribute ("height", frequencyData[idx]) ;  
◆         }  
◆         //console.log (frequencyData) ;  
◆     }  
◆ }  
◆ window.setInterval (every20, 50) ;  
◆ </script>
```

1000 毫秒 / 25  
帧 = 40 毫秒

# <http://www.imooc.com/learn/299>

## ◆ 在线教程

The screenshot shows a web browser at the URL [www.imooc.com/learn/299](http://www.imooc.com/learn/299). The page is for a course titled 'HTML5音乐可视化' (HTML5 Music Visualization) on the Imooc platform. The course is categorized under '课程 \ 前端开发 \ Html5 \ HTML5音乐可视化'. It has a '开始学习' (Start Learning) button, 62588 learners, an '高级' (Advanced) difficulty level, a duration of 2 hours and 0 minutes, and a 9.9 overall rating. The course description states: '简介：单纯的音乐播放是否过于单调，在听音乐的同时如果也能看见音乐是否更加带感。本课程将带nvas将你的音乐以你喜欢的形式可视化出来，让你的音乐动起来。' (Introduction: Is simple music playback too monotonous? If you can also see the music while listening, it will be more interesting. This course will bring nvas to visualize your music in the form you like, making your music move.) The page also features tabs for '章节' (Chapters), '评论' (Comments), '问答' (Q&A), and '笔记' (Notes). The chapter list includes '第1章 课程整体介绍' (Chapter 1: Course Overview) and '第2章 音乐获取及播放' (Chapter 2: Music Acquisition and Playback), with specific lesson titles and durations listed below each.

课程 \ 前端开发 \ Html5 \ HTML5音乐可视化

## HTML5音乐可视化

[开始学习](#) | ☆

学习人数	难度级别	课程时长	综合评分
62588	高级	2小时 0分	9.9

简介：单纯的音乐播放是否过于单调，在听音乐的同时如果也能看见音乐是否更加带感。本课程将带nvas将你的音乐以你喜欢的形式可视化出来，让你的音乐动起来。

**章节**    评论    问答    笔记

- 第1章 课程整体介绍
  - 1-1 HTML5音乐可视化课程介绍 (07:23)
- 第2章 音乐获取及播放
  - 2-1 [HTML5音乐可视化]构建应用前后端 (10:37)
  - 2-2 ajax请求服务端音频资源数据 (06:26)

# 在线文档

## ◆ <https://www.w3.org/TR/webaudio/>

### TABLE OF CONTENTS

#### Introduction

##### Features

Modular Routing

##### API Overview

- 1 The Audio API**
  - 1.1 The BaseAudioContext Interface
    - 1.1.1 Attributes
    - 1.1.2 Methods
    - 1.1.3 Callback DecodeSuccessCallback()  
Parameters
    - 1.1.4 Callback DecodeErrorCallback()  
Parameters
    - 1.1.5 Lifetime
    - 1.1.6 Lack of Introspection or Serialization  
Primitives
    - 1.1.7 System Resources Associated with  
BaseAudioContext Subclasses
  - 1.2 The AudioContext Interface
    - 1.2.1 Constructors
    - 1.2.2 Attributes
    - 1.2.3 Methods
    - 1.2.4 AudioContextOptions
      - 1.2.4.1 Dictionary AudioContextOptions  
Members

ading [MathJax]/jax/output/HTML-CSS/fonts/TeX/fontdata.js

## Web Audio API

W3C Candidate Recommendation, 18 September 2018

### This version:

<https://www.w3.org/TR/2018/CR-webaudio-20180918/>

### Latest published version:

<https://www.w3.org/TR/webaudio/>

### Editor's Draft:

<https://webaudio.github.io/web-audio-api/>

### Previous Versions:

<https://www.w3.org/TR/2018/WD-webaudio-20180619/>

<https://www.w3.org/TR/2015/WD-webaudio-20151208/>

<https://www.w3.org/TR/2013/WD-webaudio-20131010/>

<https://www.w3.org/TR/2012/WD-webaudio-20121213/>

<https://www.w3.org/TR/2012/WD-webaudio-20120802/>

<https://www.w3.org/TR/2012/WD-webaudio-20120315/>

<https://www.w3.org/TR/2011/WD-webaudio-20111215/>

### Feedback:

[public-audio@w3.org](mailto:public-audio@w3.org) with subject line "[webaudio] ... message topic ..." ([archives](#))

### Test Suite:

<https://github.com/web-platform-tests/wpt/tree/master/webaudio>

### Issue Tracking:

[GitHub](#)

### Editors:

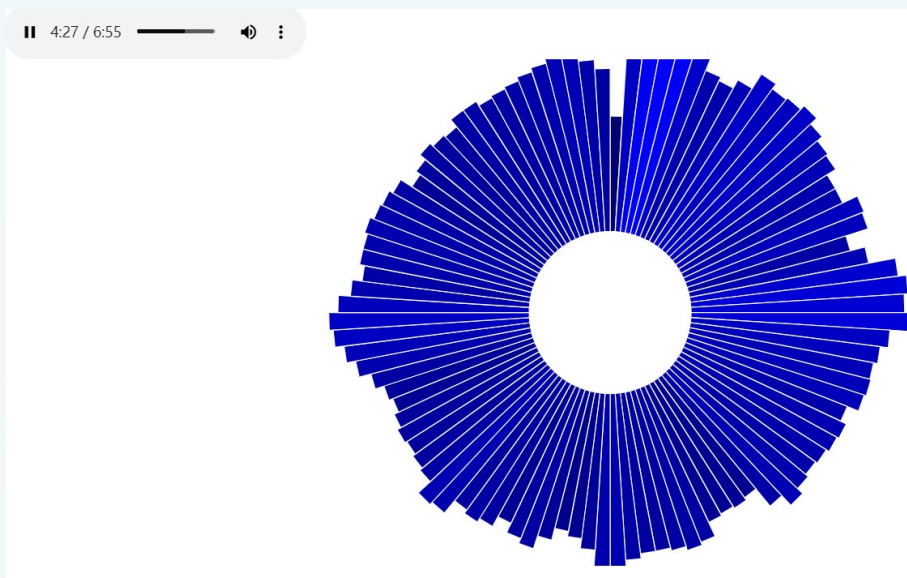
[Paul Adenot](#) (Mozilla (<https://www.mozilla.org/>))

[Raymond Toy](#) (Google (<https://www.google.com/>))

Former Editors:

# 音乐可以驱动任何的可视化图形

- ◆ 音乐分析器不变
- ◆ 图形随意换



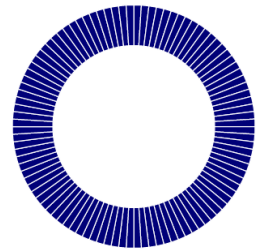
```
var audioCtx = new (window.AudioContext || window.webkitAudioContext)();  
var audioElement = document.getElementById('audioElement');  
console.log(audioElement);  
var audioSrc = audioCtx.createMediaElementSource(audioElement);  
var analyser = audioCtx.createAnalyser();
```

```
//绑定分析器到音频媒体元素  
audioSrc.connect(analyser);  
audioSrc.connect(audioCtx.destination);
```

```
var frequencyData = new Uint8Array(100);
```

# 原生数据环图数据准备

› 用 D3.V5 做一个不动的玫瑰图



```
var dataset=new Array(100);
for(var i=0;i<dataset.length;i++){
    dataset[i]=new Array();
    dataset[i][0]=120;
    dataset[i][1]=100+Math.floor(Math.random()*(255-100));
}
```

```
var pie = d3.pie()
    .value(function(d){return d[0];});

var arcPath=d3.arc();//内外半径
    .innerRadius(innerR);

var svg=d3.select("body")
    .append("svg")
    .attr("width",width)
    .attr("height",height);
```

# 用 Path 绘制原生数据玫瑰图

```
var arcs=svg.selectAll("path")
    .data(pie(dataset))//原生数据-->起止角度
    .enter()
    .append("path")
    .attr("transform", "translate(" + width/2 + "," + height/2 + ")")
    .attr("fill",function(d,i){//填充颜色
        return 'rgb(0,0,'+dataset[i][1]+'')';
    })
    .attr("stroke","#FFF")
    .attr("d",function(d,i){
        arcPath.outerRadius(dataset[i][1]);
        return arcPath(d);    //起止角度(内外半径)-->路径的参数
    });
```

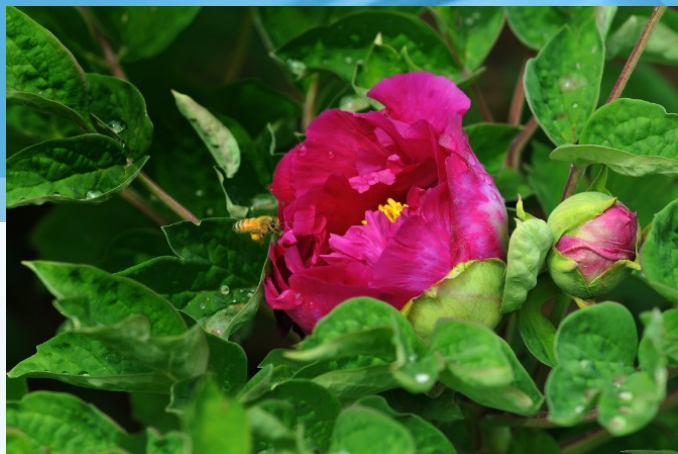


## D3 请求动画帧

### ◆ requestAnimationFrame(renderChart);

```
// 连续循环更新
function renderChart() {
  requestAnimationFrame(renderChart);
  analyser.getBytesFrequencyData(frequencyData);
  //console.log(frequencyData);
  /*
  for(var i=0;i<100;i++){
    frequencyData[i]=Math.floor(255*Math.random());
  }
  */
  svg.selectAll('path')
    .data(pie(dataset))
    .attr("fill",function(d,i){//填充颜色
      return 'rgb(0,0,'+frequencyData[i]+'')';
    })
    .attr("d",function(d,i){
      arcPath.outerRadius(frequencyData[i]+ innerR);
      return arcPath(d);
    });
}
renderChart();
```

# 谢谢



## 期待各位的佳作