**星空播放器**

——网易云播放器功能的实现和对HTML音视频播放框架的优化

**计算机与网络空间安全学院　 软件工程专业**

**116052019057　刘睿　　指导教师　张大平**

**【摘　要】 使用Vue构建前端页面，使用云服务器部署网易云api服务，实现基础但完善的音乐播放器功能。在实践中掌握前端框架（Vue）的优化方式，探索音视频在不同浏览器上的播放原理并加以优化。最后将整个项目上传至公网使所有人都能体验。**

**【关键词】 播放器；音视频框架；优化；重构；**

目录

[1 是 3](#_Toc123475101)

[1.1 的 3](#_Toc123475102)

[1.1.1 是是 3](#_Toc123475103)

# 是

# 的

# 是是

# 完成前后端云部署

# 完成后端部署

为了保证后端稳定性，也同时为了防止出现跨域等问题，故此选择不用他人已经建好的服务器，而选择自己去租赁并搭建一个新的能在公网上提供网易云api服务的服务器。并在最后将整个播放器页面也部署其上，实现播放器的公网访问。

本人选择了腾讯云服务器，租赁了为期一年的2核(vCPU) 4 GiB配置的服务器作为后端（ip：150.158.49.199），并为其做好相关配置（如开放80和3000端口、打开防火墙等），接着远程为其下载并配置了nodejs服务和相关环境，然后将来自git用户Binaryify的网易云后端源码从github上fork下来并使用pm2代理将其部署在服务器本地上，实现了网易云api服务在公网上的稳定访问。

# 完成前端部署

本人选择从阿里云注册域名“star-player.online”，并将其域名映射至服务器地址150.158.49.199，然后启用pm2代理的express服务作为代理服务器，并通过express实现对80http端口的监听:若有对该端口的请求，即将build后的vue页面作为静态资源提供。依此法即可将前端服务也部署在服务器上且能通过域名对其实现公网访问。

# 项目优化

# HTML canvas优化

使用canvas的drawImage方法在绘制大图时非常消耗系统性能，可以使用div + background-image方法代替以提高性能。

Canvas特性：当画布宽高发生变化时会清空画布。以此法代替颜色填充或clearRect方法可以提高性能。

浏览器上渲染动画 每一秒高达60帧，也就是1秒钟内我们完成60次图像绘制， 也就是每一帧图像的绘制时间其实就是（1000/ 60)。 如果在每一帧动画的时间小于 16.7 ms 辣么就会出现卡顿、丢帧。而canvas 其实是一个指令式绘图系统， 他通过绘图指令来完成绘图操作。

本人在实现canvas时使用js将canvas大小实时设置与屏幕宽高一致（用于绘制鼠标特效和背景），这就导致一个问题，在浏览器缩放比例减小的时候，会导致canvas及其分辨率变大，从而导致画面卡顿的问题。本人经过查阅资料[1]，得出如下结论：

影响canvas两个有很关键的因素：

* 第一个渲染的图形数量多，就是调用绘图指令的次数比较多，
* 第二个渲染的图形大，就是一次绘图渲染的时间比较长

因此针对以上两点，进行如下方法的优化

1. 减少绘图指令的调用。

在绘制中，先将所有路径的勾勒好再一次性进行绘制，而不是分多段多次进行绘制降低效率。简单的说就是每帧只调用一次stroke方法。

1. 分层渲染

为每种特效单独设置一层canvas，并将这些canvas叠在一起，让它们看上去就像在同一层一样。在有特效发生变化时再去重绘其对应的那层canvas，避免了其它无变化的特效元素在canvas上被重复渲染，由此降低了资源消耗。

1. 使用canvas特性

HTML <canvas> 元素有一个非常好的特性，就是你把它的宽高重新赋值一下，它的画布上的东西就被清空了。使用此特性清空大canvas可以极其有效地提升渲染性能。

# CSS、IMG加载顺序、效率优化

我们常用的 jQuery 的 $(document).ready() 方法，就是对 DOMContentLoaded 事件的监听（当然，其内部还会通过模拟 DOMContentLoaded 事件和监听 onload 事件来提供降级方案）。通常推荐在 DOMContentLoaded 事件触发的时候为 DOM 元素注册事件。所以尽快的让 DOMContentLoaded 事件触发，就意味着能够尽快让页面可交互。

而通过测试分析浏览器加载事件图可得，如果页面中静态的写有 script 标签，DOMContentLoaded 事件需要等待 JS 执行完才触发。而 script 标签中的 JS 需要等待位于其前面的 CSS 的加载完成。

综上所述，对于CSS、JS和IMG 的加载有以下几个优化方案：[2]

1. 把CSS写在页面头部，加速cssom的渲染。减小 CSS 文件体积，把单个 CSS 文件分成几个文件以并行加载，减少 CSS 对 JS 的阻塞时间
2. 把js程序写在页面尾部或异步操作中，只在当前页面解析完成之后执行js代码，提高dom和cssom渲染效率。次要的 JS 文件，通过动态插入 script 标签来加载（动态插入的 script 标签不阻塞 DOMContentLoaded 事件的触发）
3. 使用photoshop在保证图片质量的同时尽可能减小资源图片大小。（尤其是static文件夹中的内容）

# 使用gzip降低文件传输大小（尤其是文本文件）

由于前端项目越来越庞大，加之Vue的SPA首屏加载特性，导致系统第一次加载速度越来越缓慢，可能达到几十秒的程度，所以为了优化用户性能体验，我们选择了开启Gzip进行文件压缩，确实达到了显著的效果。

Gzip原本用户UNIX系统的文件压缩，后来逐渐成为Internet最主流的数据压缩格式。当用户访问我们的web站点时，服务器就将我们的网页文件进行压缩，将压缩后的文件传输到客户端，对于纯文本文件我们可以至少压缩到原大小的40%，这样大大提高了传输效率，

这对于首屏加载效率可以有很大的提升。

具体操作：下载并配置compression-webpack-plugin 插件，对webpack打包后的文件进行压缩，并配置nginx gzip转发，达到压缩传输文件大小的效果。

# 使用photoshop尽可能减小资源图片大小

# 优化效果测试方法

使用Fiddler 模拟慢速网络，打开网站，再使用Chrome 开发者工具的 Timeline/ Performance 面板优化前后对比加载的时间差异，以此来体现优化结果。

css，js加载顺序

v-if和v-show

**参考文献**

[1]杨扬，王玉，周洲.油田储层非均质性研究[J].高校地质学报，2000，4（3）：223-230.

<https://blog.csdn.net/cn_514/article/details/125806262>

[2] 转载自AlloyTeam<http://www.alloyteam.com/2014/03/effect-js-css-and-img-event-of-domcontentloaded/>