

- 一、点播场景
- 二、直播场景
  - 1、在VUE中使用video.js
    - npm安装video.js
    - main.ts中统一引入对应的css文件
    - roomLiving.vue中基于http-flv协议进行拉流
  - 2、处理视频跨域问题
  - 3、规划串流密钥
- 三、完整的直播平台音视频业务大概是什么样的

## 2、仿抖音项目集成直播客户端

阶段目标：在仿抖音项目中集成点播和直播功能

# 一、点播场景

---

核心代码：

```
<video controls :poster="anchorInfo.avatar" width="100%" style="background-color: rgb(18, 9, 37);">
  <source src="@/1.mp4">
</video>
```

补充：基于Nginx部署服务以及添加视频

# 二、直播场景

---

## 1、在VUE中使用video.js

---

### npm安装video.js

```
npm install video.js
```

### main.ts中统一引入对应的css文件

```
import 'video.js/dist/video-js.min.css'
```

### roomLiving.vue中基于http-flv协议进行拉流

```

<template>
  ....
  <!-- class需要指定videojs提供的video-js -->
  <video class="video-js" ref="videoplayer"
    width="100%" style="background-color: rgb(18, 9, 37);width:100%;height: 100%;">
</video>
  ....
</template>
<script setup>
import videojs from 'video.js'

// http-flv协议拉流
const url = 'http://192.168.65.210/hls/test.m3u8'

const videoplayer = ref(null)
const myPlayer = ref(null)

onMounted(() => {
  myPlayer.value = videojs(videoplayer.value,
    {
      autoplay: false, //false:不自动播放,true:自动播放, muted:静音自动播放,play:自动播
      放,any:自动播放。
      poster: "/img/2.jpeg", //默认封面
      controls: true, //显示控制组件
      controlBar: true, //true, 显示控制栏。另外也可以设置一个Object, 设置详细属性
      bigPlayButton: true, //在视频上显示大播放按钮。
      sources: [{
        src: url
      }]
    },
    () => {
      console.info("播放器加载完成")
    }
  )
})
</script>

```

核心是给Video标签指定一个videojs函数创建的播放器。videojs函数功能非常强大，基本上完成了一个完整播放器所需要的全部功能。

videojs函数需要传入三个核心参数。

- 第一个参数是对应的播放器元素。
- 第二个参数是options，即播放器的相关设置。这里设置项非常多，具体配置可以参考官网 <https://videojs.com/guides/options/>
- 第三个参数是一个可选的回调函数。当播放器初始化完成后，就会触发这个回调函数。

另外，videojs函数创建出来的对象，还支持一些播放器控制功能。常见的如，play():播放；pause():暂停；paused():获取是否暂停；currentTime():获取当前播放位置；currentTime(number):设置播放位置。

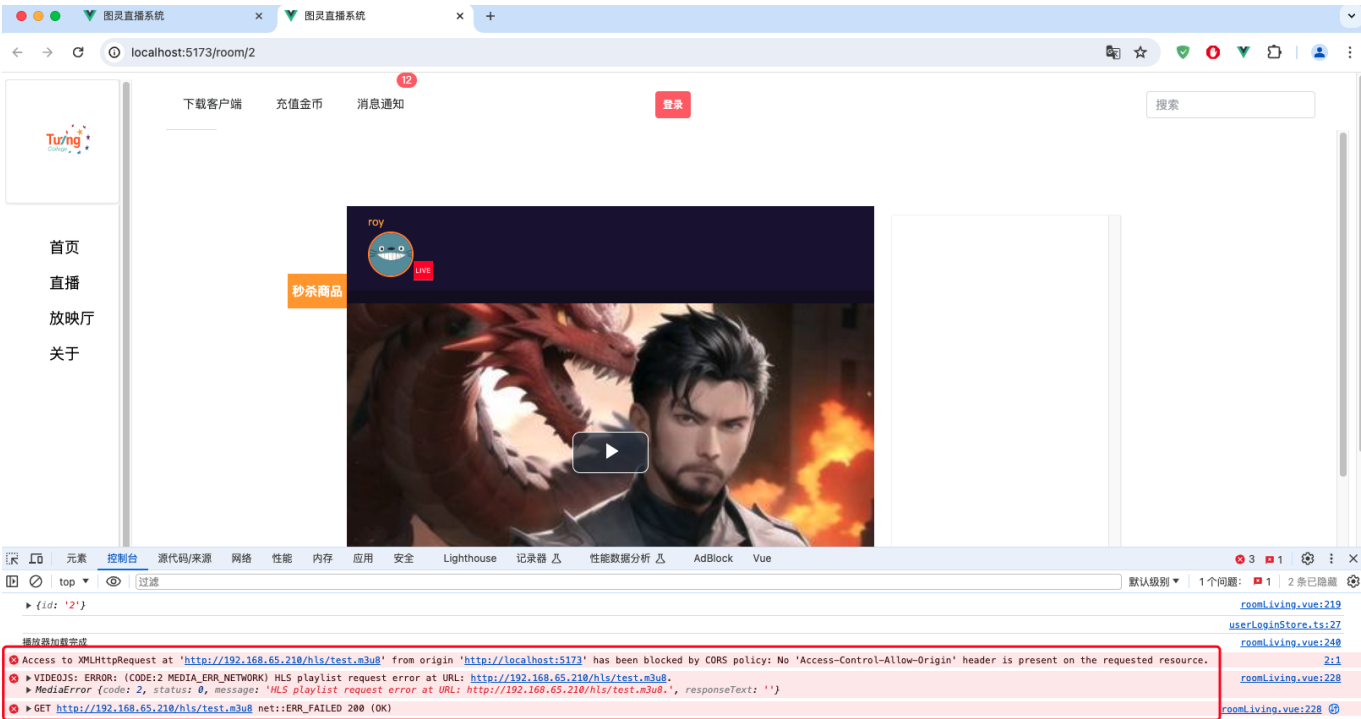
除此之外，videojs对象还支持非常多的监听事件。例如开始请求数据(loadstart)，正在请求数据(progress)，开始播放(play)，播放中(playing)，暂停(pause)，视频结束(ended)，加载错误(error)，视频跳转结束(seeked)，播放速率改变(ratechange)，播放时长改变(timeupdate)，音量改变(volumechange)，网速异常(stalled)等。甚至还支持自定义事件。具体参见官网

例如我们可以给视频加载错误增加一个提示

```
myPlayer.value.on('error',(error)=>{
  console.info(error)
})
```

## 2、处理视频跨域问题

如果视频推流地址和项目部署地址不在同一个域名下，就会产生跨域的问题。



跨域问题在客户端是无法解决的，只能在服务端解决。对于http-flv协议来说，可以在nginx中给对应的http端口配置可跨域。

例如在nginx的配置文件nginx.conf中，对/hls端口，增加以下配置

```
http{
  server {
    listen 80;
    server_name localhost;
    .....
    location /hls {
      add_header 'Access-Control-Allow-Origin' '*' always;
      add_header 'Access-Control-Allow-Credentials' 'true';
    }
    .....
  }
}
```

然后重启nginx服务，就可以正常进行拉流播放了。

```
nginx -t #检查nginx配置文件
nginx -s reload #重新加载nginx配置文件
```

### 3、规划串流密钥

之前案例当中，拉流地址都是指定的同一个地址：<http://192.168.65.210/hls/test.m3u8>，这当然无法满足真正的直播间需求。每个直播间应该有不同的直播地址。

在进入直播间时，给每个直播间访问地址配置了动态的路由。

例如在 @/router/index.ts中，对于直播间的访问路由是这样配置的

```
{
  path: '/room/:id',
  name: 'livingRoom',
  component: () => import('@/views/roomlist/roomLiving.vue'),
  meta: {title: '直播间'}
},
```

在roomliving.vue中，就可以通过路由组件获取到动态设置的id属性。

```
<script setup>
import { useRoute } from 'vue-router'
let route = useRoute()
//房间ID json格式。 {id,1}
console.info(route.params)
</script>
```

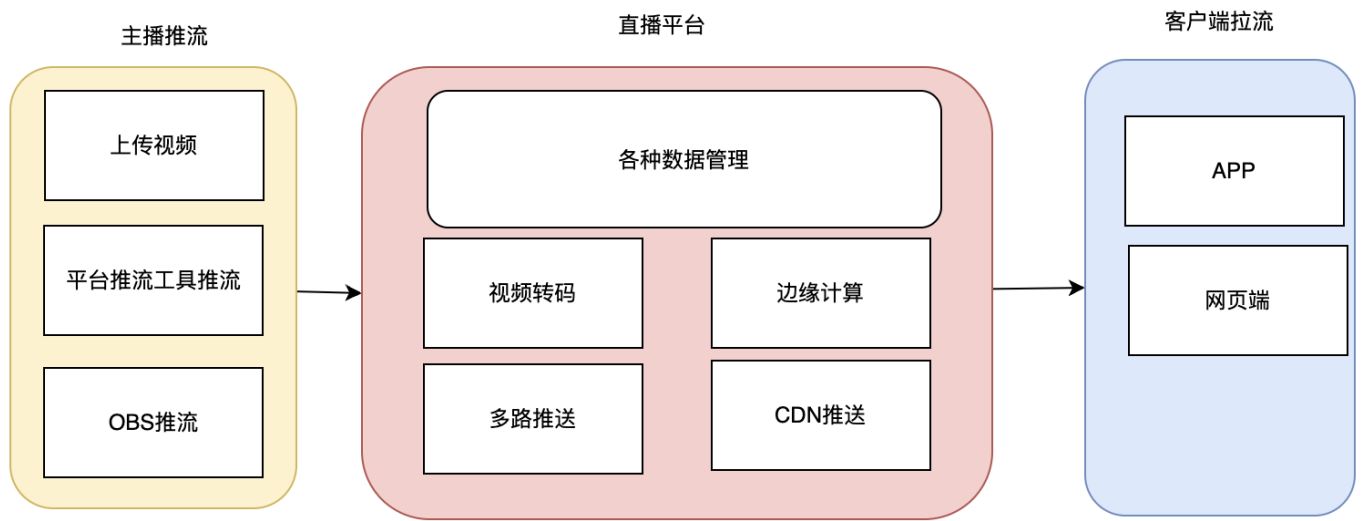
通过路由组件就可以获取到当前直播间的ID，这就是直播间的唯一标识。只要用这个直播间ID找到一个单独的直播密钥，就可以实现每个直播间的内容区分。例如：

```
http://192.168.65.210/hls/#{id}.m3u8
```

这样，只要主播往房间的对应地址进行推流，那么在直播间页面就可以拉取到对应的内容。

最后，目前我们只是实现了直播最核心的音视频业务，围绕音视频业务，实际上还有大量的周边业务。例如主播管理、直播间管理等等。在做前端设计时，都抽取成了单独的变量。这样后续只要通过双向绑定，将这些变量改为从后端获取，就可以实现完整的管理功能。这些管理功能从技术上来看，都是一些基础的CRUD操作，没有什么难度，大家酌情补充。

## 三、完整的直播平台音视频业务大概是什么样的



参考网页: [https://blog.csdn.net/weixin\\_48201324/article/details/123810653](https://blog.csdn.net/weixin_48201324/article/details/123810653)

<https://videojs.com/guides>