# **2025年夏季《移动软件开发》实验报告**

​

姓名：聂宇航 学号：23170001072

​

| 姓名和学号？ | 聂宇航，23170001072 |
| --- | --- |
| 本实验属于哪门课程？ | 中国海洋大学25夏《移动软件开发》 |
| 实验名称？ | 实验4：媒体API之口述校史 |
| 博客地址？ | [《移动软件开发》实验四实验报告-CSDN博客](https://blog.csdn.net/2301_80144482/article/details/151121342?spm=1001.2014.3001.5502) |
| Github仓库地址？ | [这个是《移动软件开发》这门课的实验代码与报告](https://github.com/amieon/MobileSoftwareDevelopmentLab) |

（备注：将实验报告发布在博客、代码公开至 github 是 加分项，不是必须做的）

## 一、实验目标

1、学习使用快速启动模板创建小程序的方法；2、学习不使用模板手动创建小程序的方法。

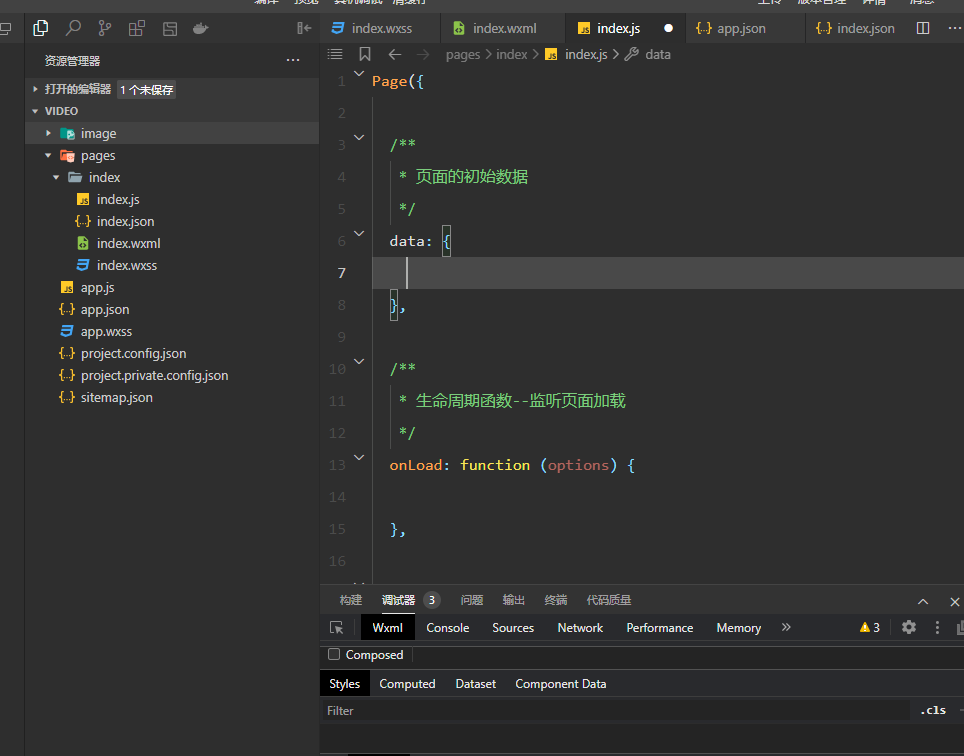
## **二、实验步骤**

### **1.初始准备**

和实验1一样对创造的项目进行页面配置

项目创建完毕后,在根目录中会生成文件夹pages用于存放页面文件。一般来说首页默认命名为 index,表示小程序运行的第一个页面;其他页面名称可以自定义。本项目只需要保留首页： (1)将 app.json 文件内 pages 属性中的“pages/logs/logs”删除,并删除上一行末尾的逗号 (2)按快捷键Ctrl+S保存当前修改

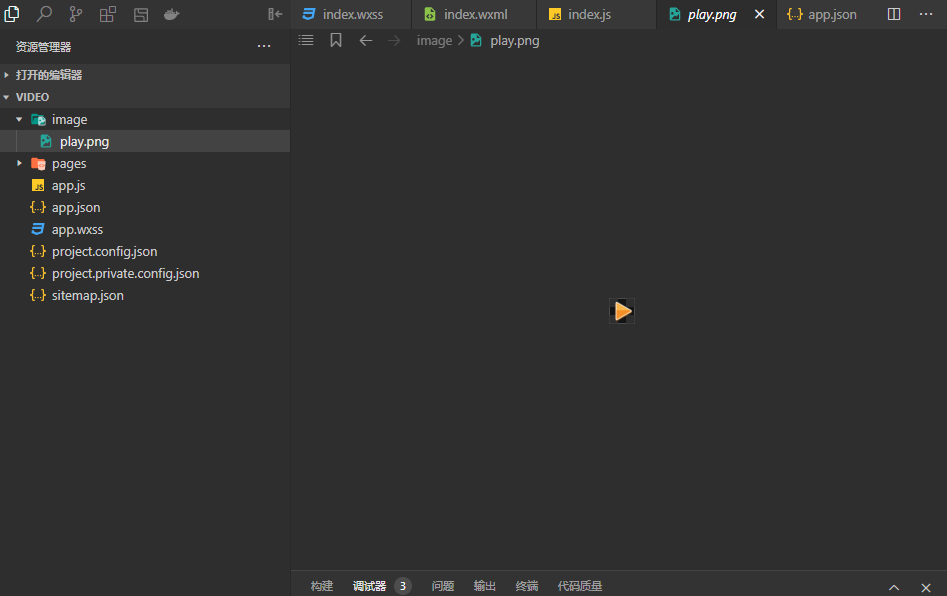
然后我们要删除和修改一些文件： (1)删除utils 文件夹及其内部所有内容 (2)删除 pages 文件夹下的 logs 目录及其内部所有内容。 (3)删除index.wxml和index.wxss 中的全部代码。 (4)删除index.js中的全部代码,并且输入关键词 page找到第二个选项按回车键让其自动补全函数。 (5)删除 app.wxss 中的全部代码, (6)删除 app.js中的全部代码,并且输入关键词 app 找到第一个选项按回车键让其自动补全函数。



新建image文件夹，在<https://gaopursuit.oss-cn-bejing.aliyuncs.com/2022/images_play.zip>里下载一个播放的图片，命名为play.png存入

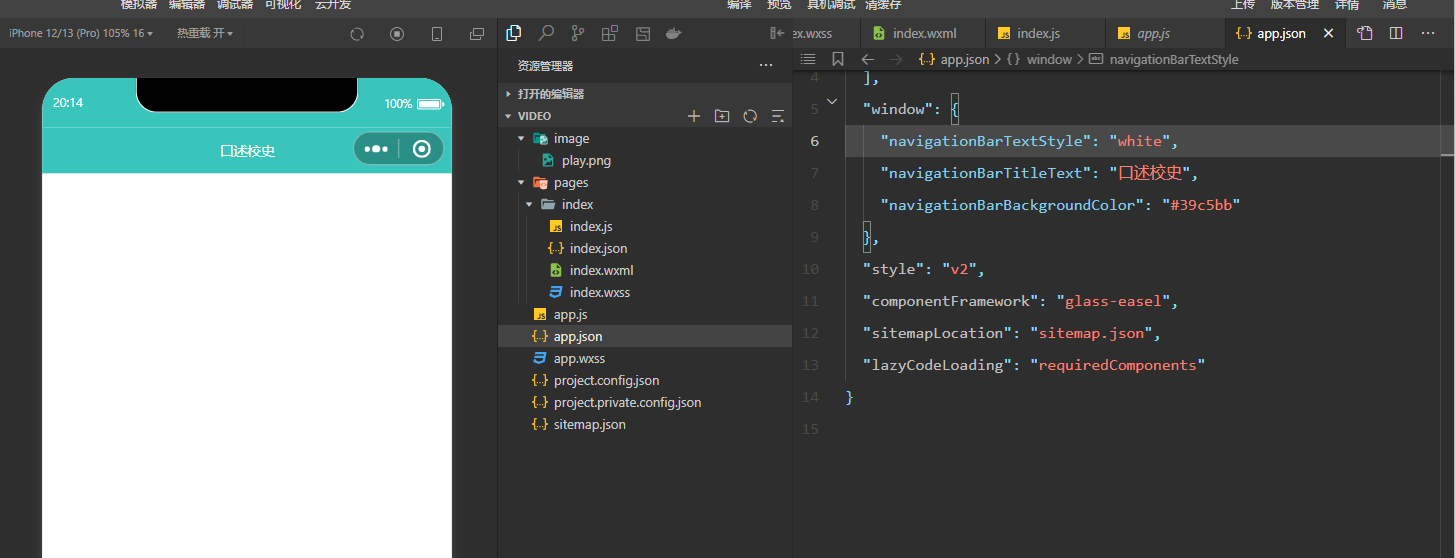
再在index.js文件中加入如下要用的的视频文件地址：

 data:{  
    danmuTxt:'',  
    list:[  
   {  
    id:'299371',  
    title:'杨国宜先生口述校史实录',  
    videoUrl:'http://arch.ahnu.edu.cn/\_\_local/E/31/EB/2F368A265E6C842BB6A63EE5F97\_425ABEDD\_7167F22.mp4?e=.mp4'  
       },  
   {  
      id:'299396',  
      title:'唐成伦先生口述校史实录',  
      videoUrl:'http://arch.ahnu.edu.cn/\_\_local/E/31/EB/2F368A265E6C842BB6A63EE5F97\_425ABEDD\_7167F22.mp4?e=.mp4'  
   },  
   {  
    id:'299378',  
    title:'倪光明先生口述校史实录',  
    videoUrl:'http://arch.ahnu.edu.cn/\_\_local/9/DC/3B/35687573BA2145023FDAEBAFE67\_AAD8D222\_925F3FF.mp4?e=.mp4'  
   },{  
    id:'299392',  
    title:'吴兴仪先生口述校史实录',  
    videoUrl:'http://arch.ahnu.edu.cn/\_\_local/9/DC/3B/35687573BA2145023FDAEBAFE67\_AAD8D222\_925F3FF.mp4?e=.mp4'  
   }  
   ],  
 },



### **2.页面设计**

将页面的上方改变颜色，然后放入我们想要的标题（及其颜色）



页面上主要包含3个区域,具体内容如下图



区域1:视频播放器,用于播放指定的视频，使用<video >组件

区域2:弹幕发送区域,包含文本输入框和发送按钮，使用<view>组件，并定义class='danmuArea'；区域 2内部:<input>和<button >组件

区域3:视频列表,垂直排列多个视频标题,点击不同的标题播放对应的视频内容。面板之间需要有一定的间隔距离，内单元行:<view>组件,并定义class='videoBar'区域3单元行内:每行一个<image>组件用于显示播放图标、一个<text>组件用于显示视频标题。

将他们如下写入wxml和wxss文件

<video id="myvideo"></video>  
​  
<view class="danmuArea">  
  <input type="text" placeholder="请输入弹幕内容"></input>  
  <button>发送弹幕</button>  
</view>  
​  
<view class="videoList">  
  <view class="videoBar">  
    <image src="/image/play.png"></image>  
    <text>{{item.title}}</text>  
  </view>  
</view>  
​  
​

video{  
  width:100%;  
}  
.danmuArea{  
  display: flex;  
  flex-direction: row;  
}  
input {  
  border: 3rpx solid #39c5bb;  
  flex-grow:1 ;  
  height: 80rpx;  
}  
button{  
  height: 80rpx;  
  font: 1em sans-serif;  
  color: white;  
  background-color: #39c5bb;  
}  
.videoBar{  
  width: 100%;  
  min-height: 100rpx;  
}  
.videoBar{  
  width: 95%;  
  display: flex;  
  flex-direction: row;  
  border-bottom: 1rpx solid #39c5bb;  
  margin: 10rpx;  
}  
image{  
  width: 70rpx;  
  height: 70rpx;  
  margin: 20rpx;  
}  
text{  
  font-size: 35rpx;  
  color: #39c5bb;  
  margin: 20rpx;  
  flex-grow: 1;  
}

最后我们可以得到的效果是这样的：



### **3.逻辑实现**

### **更新播放列表**

在区域3对<view class='videoBar'>组件添加 wx:for 属性,改写为循环展示列表。

<view class="videoList">  
  <view class="videoBar" wx:for="{{list}}" wx:key="video{{index}}" data-url="{{item.videoUrl}}">  
    <image src="/image/play.png"></image>  
    <text>{{item.title}}</text>  
  </view>  
</view>  
​

在 WXML 里，wx:for 可以根据数组，把某个组件重复渲染多次。

比如我们现在有 list 数据：

data: {  
  list: [  
   { id: '299371', title: '杨国宜先生口述校史实录', videoUrl: 'xxx1.mp4' },  
   { id: '299396', title: '唐成伦先生口述校史实录', videoUrl: 'xxx2.mp4' },  
   { id: '299378', title: '倪光明先生口述校史实录', videoUrl: 'xxx3.mp4' }  
 ]  
}

如果不用 wx:for，就得手动写一堆重复的 <view>：

<view class="videoBar">  
  <text>杨国宜先生口述校史实录</text>  
</view>  
<view class="videoBar">  
  <text>唐成伦先生口述校史实录</text>  
</view>  
<view class="videoBar">  
  <text>倪光明先生口述校史实录</text>  
</view>

这样很麻烦。

用for只要一行，就能自动循环：

<view class="videoBar"   
      wx:for="{{list}}"   
      wx:key="id"   
      data-url="{{item.videoUrl}}" >  
  <image src="/image/play.png"></image>  
  <text>{{item.title}}</text>  
</view>

wx:for="{{list}}" ：遍历 data 里的 list 数组

item ：当前循环的元素（默认变量名是 item）

index ：当前循环的下标（默认变量名是 index）

wx:key="id" ：指定唯一标识，提高渲染性能

{{item.title}} ：取出每个视频的标题

data-url="{{item.videoUrl}}" ：把视频地址存到自定义属性，点击时传给 playVideo

运行效果如图：



### **点击播放视频**

在区域 3 对< view class='videoBar'>组件添加 data-url 属性和 bindtap属性。其中 data-url用于记录每行视频对应的播放地址,bindtap 用于触发点击事件。

<view class="videoList">  
  <view class="videoBar" wx:for="{{list}}" wx:key="video{{index}}" data-url="{{item.videoUrl}}" bindtap="playVideo">  
    <image src="/image/play.png"></image>  
    <text>{{item.title}}</text>  
  </view>  
</view>  
​

在js文件中写入播放逻辑代码：

 playVideo: function(e){  
    this.setData({  
      src:e.currentTarget.dataset.url  
   })  
    this.videoCtx.play()  
 },  
​  
  onReady: function () {  
    this.videoCtx  = wx.createVideoContext('myvideo', this);  
 },

页面渲染完成 → onReady 执行 → 创建 videoCtx 控制器。

用户点击视频列表 → 触发 playVideo。

从点击的 <view> 里拿到视频地址（data-url）。

setData 更新 src，视频组件的地址发生改变。

调用 videoCtx.play()，新的视频立刻播放。

onReady: function () { ... }:

生命周期函数

小程序的页面生命周期里，onReady 会在 首次渲染完成 后触发。

这时候页面的节点已经生成，可以安全地操作组件。

wx.createVideoContext('myvideo', this)

作用：创建一个 视频上下文对象，用来控制 <video> 组件（播放、暂停、发弹幕等）。

'myvideo' 对应的是在 WXML 里写的 <video id="myvideo">。

this 指当前页面实例（Page），这样上下文对象和当前页面绑定。

赋值给 this.videoCtx

保存到当前页面对象里，后面就可以通过 this.videoCtx.xxx() 来操作视频。

playVideo: function(e) { ... }:

这是点视频列表时触发的事件处理函数。

e.currentTarget.dataset.url

e 是事件对象。

currentTarget 指触发事件的那个组件（ <view data-url="{{item.videoUrl}}" bindtap="playVideo">）。

dataset.url 就是 data-url 里的值，也就是视频的真实地址。

this.setData({ src: ... })

把获取到的视频地址设置到页面的 data.src 里。

因为 <video> 组件写了 src="{{src}}"，所以 setData 会触发数据绑定，让视频切换到新地址。

this.videoCtx.play()

调用视频上下文的 play() 方法，让 <video> 开始播放。

这样我们就成功播放了：



### **发送弹幕**

在区域1对<video>组件添加 enable-danmu 和 danmu-btn属性,用于允许发送弹幕和显示“发送弹幕”按钮。然后在区域2为文本输入框追加bindinput属性,用于获取弹幕文本内容;为按钮追加bindtap属性,用于触发点击事件。

<video id="myvideo" src="{{src}}" controls enable-danmu danmu-btn></video>  
​  
<view class="danmuArea">  
  <input type="text" placeholder="请输入弹幕内容" bindinput="getDanmu"></input>  
  <button bindtap="sendDanmu">发送弹幕</button>  
</view>

js对应的文件为：

 getDanmu : function(e){  
    this.setData({  
      danmuTxt: e.detail.value  
   })  
 },  
  sendDanmu() {  
    let text = this.data.danmuTxt;  
    if (!text) return;  
    this.videoCtx.sendDanmu({  
      text: text,  
      color: this.getRandomColor(),  
   });  
    this.setData({ danmuTxt: '' });  
 },  
  getRandomColor: function() {  
    let rbg = []  
    for(let i=0;i<3;++i){  
      let color = Math.floor(Math.random()\*256).toString(16)  
      color = color.length == 1 ? '0' + color : color  
      rbg.push(color)  
   }  
    return '#' + rbg.join('')  
 },

我们在输入框里输入文字 → getDanmu 被触发 → 把内容保存到 danmuTxt。

点击“发送”按钮 → sendDanmu 被执行。

读取 danmuTxt，生成随机颜色 → 调用 videoCtx.sendDanmu 把弹幕推送到视频。

清空输入框。

#### **1. getDanmu**

getDanmu : function(e){  
  this.setData({  
    danmuTxt: e.detail.value  
 })  
}

这个函数绑定在 <input bindinput="getDanmu"> 上。

每次输入框内容变化，都会触发 bindinput，e.detail.value 就是最新的输入内容。

用 setData 把输入的内容保存到页面的 data.danmuTxt 里。

相当于 实时保存输入的弹幕文本。

#### **2. sendDanmu**

sendDanmu() {  
  let text = this.data.danmuTxt;  
  if (!text) return;  
  this.videoCtx.sendDanmu({  
    text: text,  
    color: this.getRandomColor(),  
 });  
  this.setData({ danmuTxt: '' });  
}

从 data.danmuTxt 里取出当前的输入内容。

如果没输入 (!text)，直接返回，不发弹幕。

this.videoCtx.sendDanmu({ ... })：调用视频上下文的 API，在视频上发送一条弹幕。

text: 弹幕的文字。

color: 弹幕的颜色（调用下面的 getRandomColor() 随机生成）。

this.setData({ danmuTxt: '' })：把 danmuTxt 清空，输入框的内容也会被清空。

相当于 点击按钮 → 发送弹幕 → 清空输入框。

#### **3. getRandomColor**

getRandomColor: function() {  
 let rbg = []  
 for(let i=0;i<3;++i){  
   let color = Math.floor(Math.random()\*256).toString(16)  
   color = color.length == 1 ? '0' + color : color  
   rbg.push(color)  
 }  
 return '#' + rbg.join('')  
}

随机生成 RGB 颜色值：

Math.floor(Math.random()\*256) 生成 0–255 的随机整数。

.toString(16) 转成十六进制。

如果只有一位数，就补 0。

最后拼接成一个 #rrggbb 的字符串。

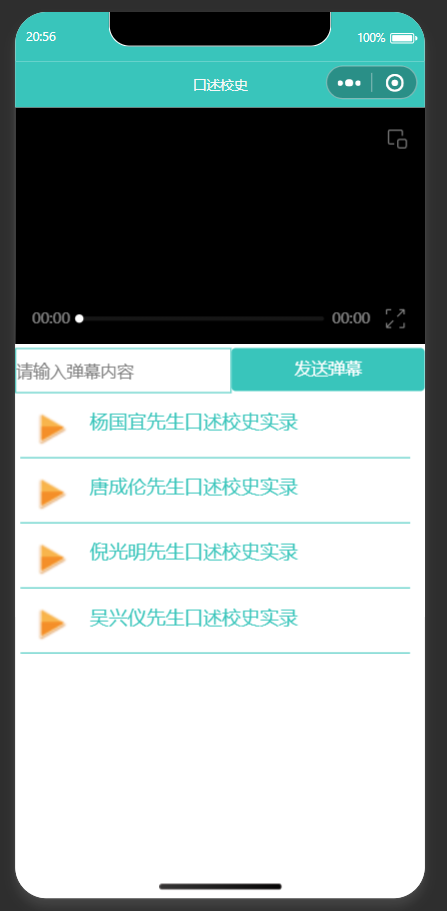
这样我们就成功的发送弹幕了：



## **三、程序运行结果**

列出程序的最终运行结果及截图。

一开始的时候，加载初始播放列表：



点击一个视频：



点击三角形▲就可以播放了：



在文本框里输入字，点击按钮，就可以发送弹幕了：



## **四、问题总结与体会**

描述实验过程中所遇到的问题，以及是如何解决的。有哪些收获和体会，对于课程的安排有哪些建议。

1.弹幕 API 报错

报错信息：Cannot read property 'sendDanmu' of undefined

原因：我在 onReady 里定义的是 this.videoContext，但在 sendDanmu 里用的是 this.videoCtx，名字不一致 → 导致找不到 sendDanmu 方法。

解决：统一用一个名字，比如 this.videoCtx = wx.createVideoContext('myvideo')。

2.WXML 输入框不触发事件

问题：getDanmu 没有执行。

原因：<input> 里拼写错了，写成了 biidndinput。

解决：改成正确的 bindinput="getDanmu"。

3.视频切换播放不生效

问题：点击视频列表时不能正常播放。

原因：没有给 <view> 正确加上 wx:for 和 data-url，并且 this.videoCtx 没有初始化。

解决：在 onReady 里创建 videoCtx，在 <view> 上用 data-url="{{item.videoUrl}}" 传视频地址，playVideo 里用 setData({src: ...}) 更新。