2025年夏季《移动软件开发》实验报告

姓名：潘奕霖 学号：23020007094

姓名和学号?	潘奕霖，23020007094
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?	中国海洋大学2025年夏季《移动软件开发》实验报告——实验4-CSDN博客
Github仓库地址?	https://github.com/PYL1024/mobile-software-development

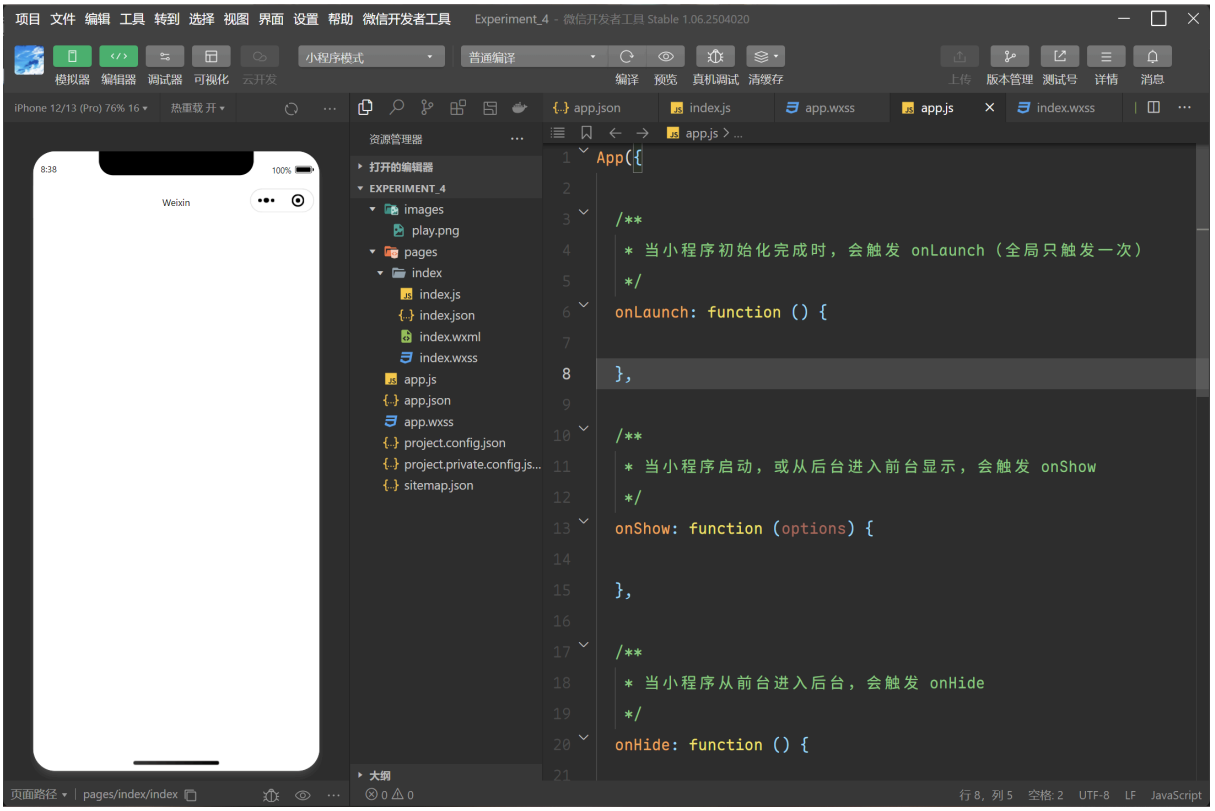
（备注：将实验报告发布在博客、代码公开至 github 是 **加分项**，不是必须做的）

一、实验目标

1、掌握视频API的操作方法；2、掌握如何发送随机颜色的弹幕。

二、实验步骤

1、新建小程序项目，并按要求修改原有的文件。新建 images 文件夹，并将 play.png 放入。



2、主页面分为三个区域：视频播放器，弹幕发送区域和视频列表。用 <video> 组件实现播放器，用 <view> 实现弹幕发送区域和视频列表 。

```
<!--区域1：视频播放器-->
<video id='myVideo' controls></video>

<!--区域2：弹幕控制-->
<view class='danmuArea'>
  <input type='text' placeholder='请输入弹幕内容'></input>
  <button>发送弹幕</button>
</view>

<!--区域3：视频列表-->
<view class='videoList'>
  <view class='videoBar'>
    <image src='/images/play.png'></image>
    <text>这是一个测试标题</text>
  </view>
</view>
```

为了使视频组件的宽度和屏幕一致，设置 `width: 100%;` 。用 flex 布局指定弹幕区域元素按行排列，并且设定 <input> 组件的 `flex-grow: 1;` ，使其占据一行中除发送按钮外的其他空间。类似设定的还有视频列表中的每一视频项，由视频标题占据每一行中除视频图标外的其余空间。

```
/*视频组件样式*/
video {
  width: 100%;
  /*视频组件*/
}

/*区域2：弹幕控制样式*/
/*2-1弹幕区域样式*/
.danmuArea {
  display: flex;
  /*flex模型布局*/
  flex-direction: row;
  /*水平方向排列*/
}

/*2-2文本输入框样式*/
input {
  border: 1rpx solid #987938;
  /*1rpx宽的实线棕色边框*/
  flex-grow: 1;
  /*扩张多余空间宽度*/
  height: 100rpx;
  /*高度*/
}

/*2-3按钮样式*/
button {
  color: white;
  /*字体颜色*/
  background-color: #987938;
  /*背景颜色*/
}
```

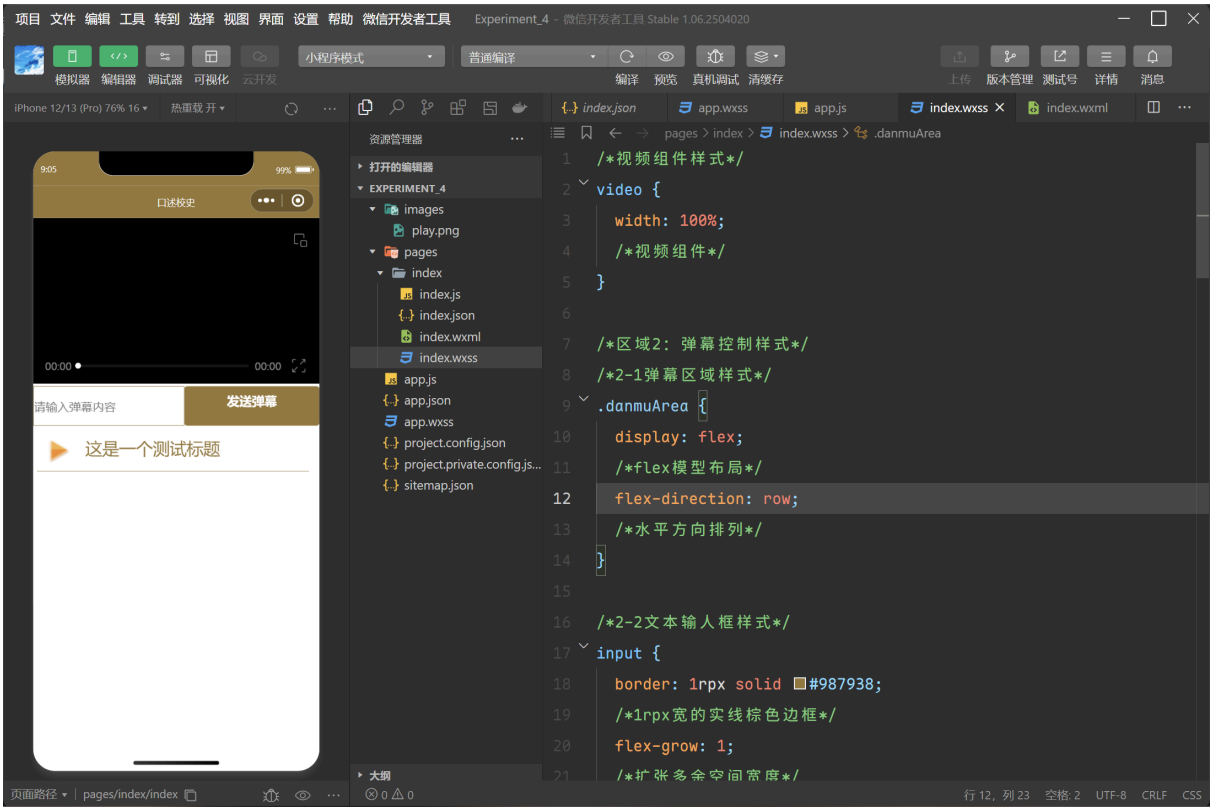
```
/*区域3：视频列表样式*/
/*3-1视频列表区域样式*/
.videoList {
  width: 100%;
  /*宽度*/
  min-height: 400rpx;
  /*最小高度*/
}

/*3-2单行列表区域样式*/
.videoBar {
  width: 95%;
  /*宽度*/
  display: flex;
  /*flex模型布局*/
  flex-direction: row;
  /*水平方向布局*/
  border-bottom: 1rpx solid #987938;
  /*1rpx宽的实线棕色边框*/
  margin: 10rpx;
  /*外边距*/
}

/*3-3播放图标样式*/
image {
  width: 70rpx;
  /*宽度*/
  height: 70rpx;
  /*高度*/
  margin: 20rpx;
  /*外边距*/
}

/*3-4文本标题样式*/
text {
  font-size: 45rpx;
  /*字体大小*/
  color: #987938;
```

```
  /*字体颜色为棕色*/
  margin: 20rpx;
  /*外边距*/
  flex-grow: 1;
  /*扩张多余空间宽度*/
}
```



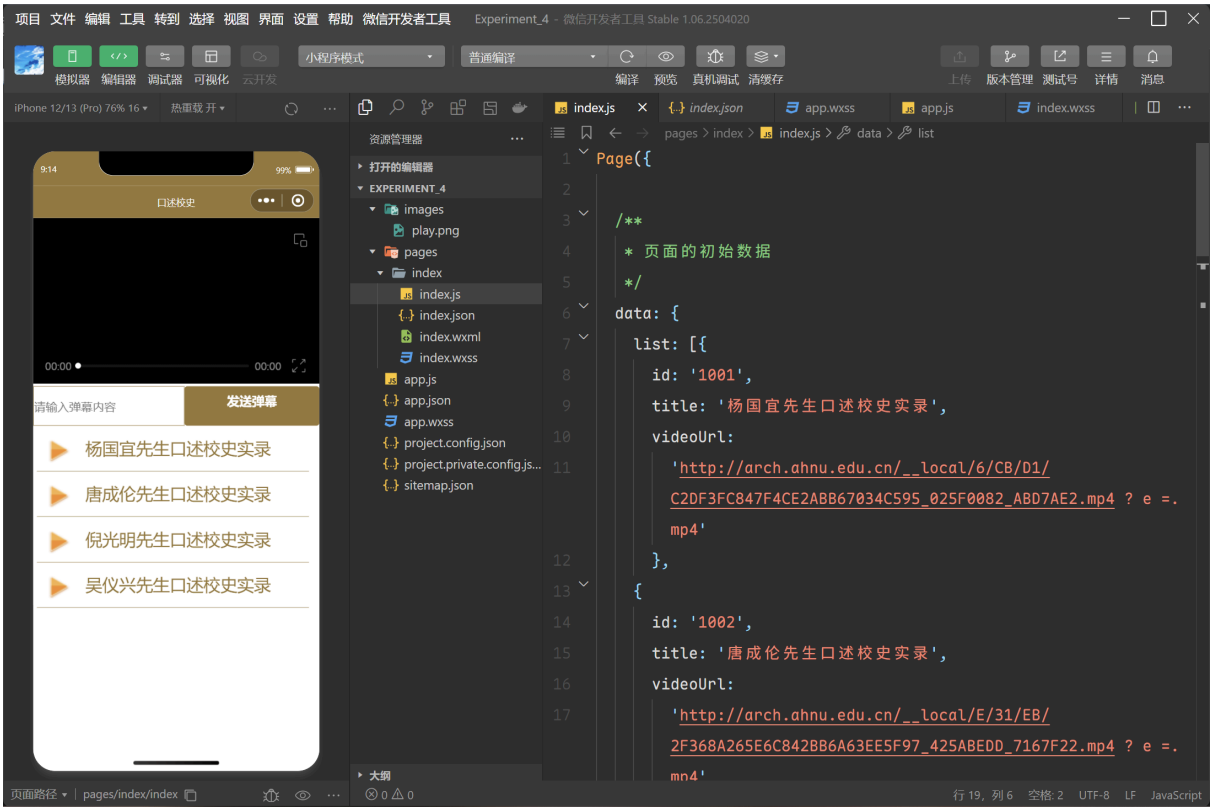
3、为视频播放器添加 `src='{{src}}'` 属性，这样才能获取视频 url，视频才能正常播放。接着对视频列表的 <view class=videoBar'> 组件添加 `wx:for` 属性，改写为循环展示列表。

```
<video id='myVideo' src = '{{src}}' controls></video>

<view class='videoBar' wx:for = '{{list}}' wx:key = 'video{{index}}'>
  <image src='/images/play.png'></image>
  <text>{{item.title}}</text>
</view>
```

并在 index.js 的 `data` 属性中添加存有视频信息的 `list` 数组，作为循环展示列表的内容来源。

```
list: [{
    id: '1001',
    title: '杨国宜先生口述校史实录',
    videoUrl:
     'http://arch.ahnu.edu.cn/__local/6/CB/D1/C2DF3FC847F4CE2ABB670
34C595_025F0082_ABD7AE2.mp4 ? e =.mp4'
    },
  {
    id: '1002',
    title: '唐成伦先生口述校史实录',
    videoUrl:
     'http://arch.ahnu.edu.cn/__local/E/31/EB/2F368A265E6C842BB6A63E
E5F97_425ABEDD_7167F22.mp4 ? e =.mp4'
    },
  {
    id: '1003',
    title: '倪光明先生口述校史实录',
    videoUrl:
     'http://arch.ahnu.edu.cn/__local/9/DC/3B/35687573BA2145023FDAEB
AFE67_AAD8D222_925F3FF.mp4 ? e =.mp4'
    },
  {
    id: '1004',
    title: '吴仪兴先生口述校史实录',
    videoUrl: 'http://arch.ahnu.edu.cn/__local/5/DA/BD/7A27865731CF2B09
6E90B522005_A29CB142_6525BCF.mp4 ? e =.mp4'
     }
  ]
```



4、对视频列表的 <view class=videoBar'> 组件再次新增记录视频对应播放地址的 `data-url` 属性和用于触发播放的 `bindtap` 属性。

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

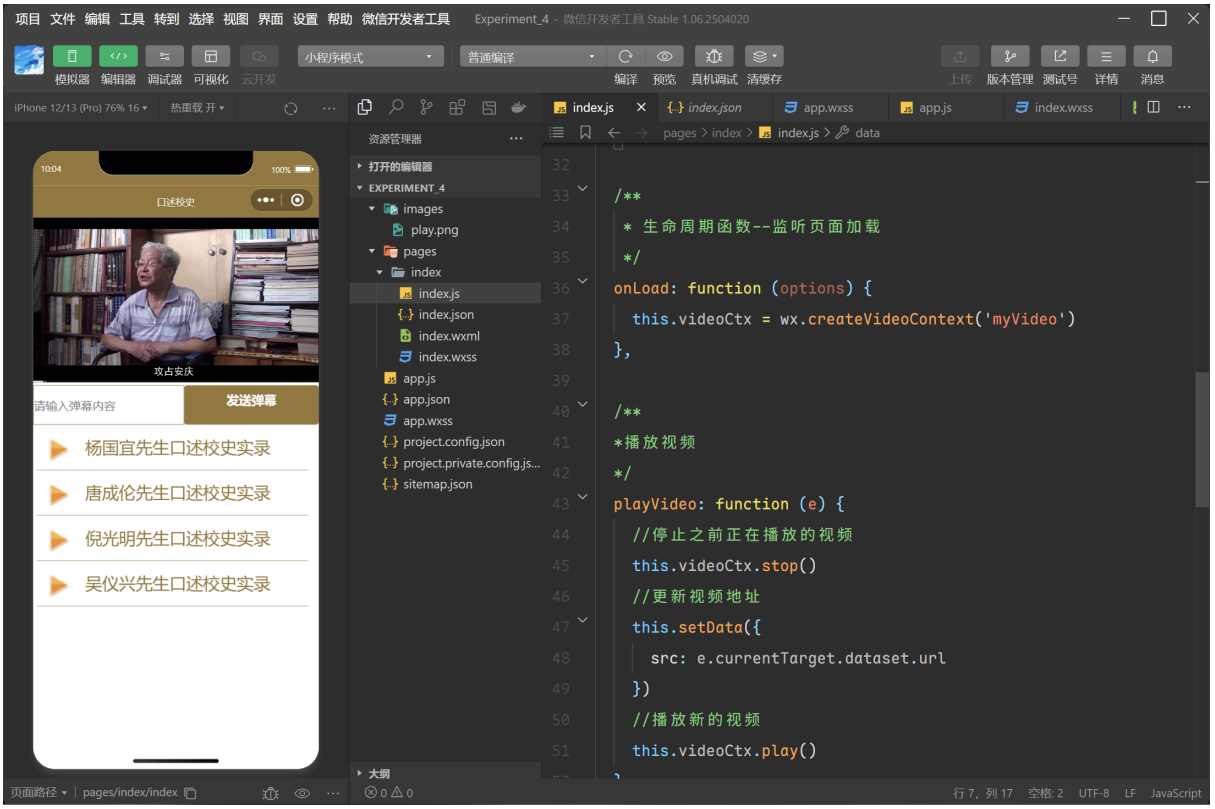
在 index.js 的 `onLoad` 函数中创建视频上下文 videoCtx ，控制视频的播放和停止。

```
onLoad: function (options) {
  this.videoCtx = wx.createVideoContext('myVideo')
},
```

接着添加自定义函数 playVideo，控制视频播放的具体逻辑。

```
/**
*播放视频
*/
playVideo: function (e) {
```

```
    //停止之前正在播放的视频
    this.videoCtx.stop()
    //更新视频地址
    this.setData({
      src: e.currentTarget.dataset.url
    })
    //播放新的视频
    this.videoCtx.play()
  }
```



5、在 <video> 组件处添加 `enable-danmu` 和 `danmu-btn`，允许发送弹幕并隐藏播放器中发送弹幕的按钮。接着为文本输入框添加 `bindinput` 获取弹幕文本，并为发送按钮绑定 `sendDanmu` 函数。

```
<!--区域1：视频播放器-->
<video id='myVideo' src = '{{src}}' controls enable-danmu danmu-btn></video>

<!--区域2：弹幕控制-->
<view class='danmuArea'>
```

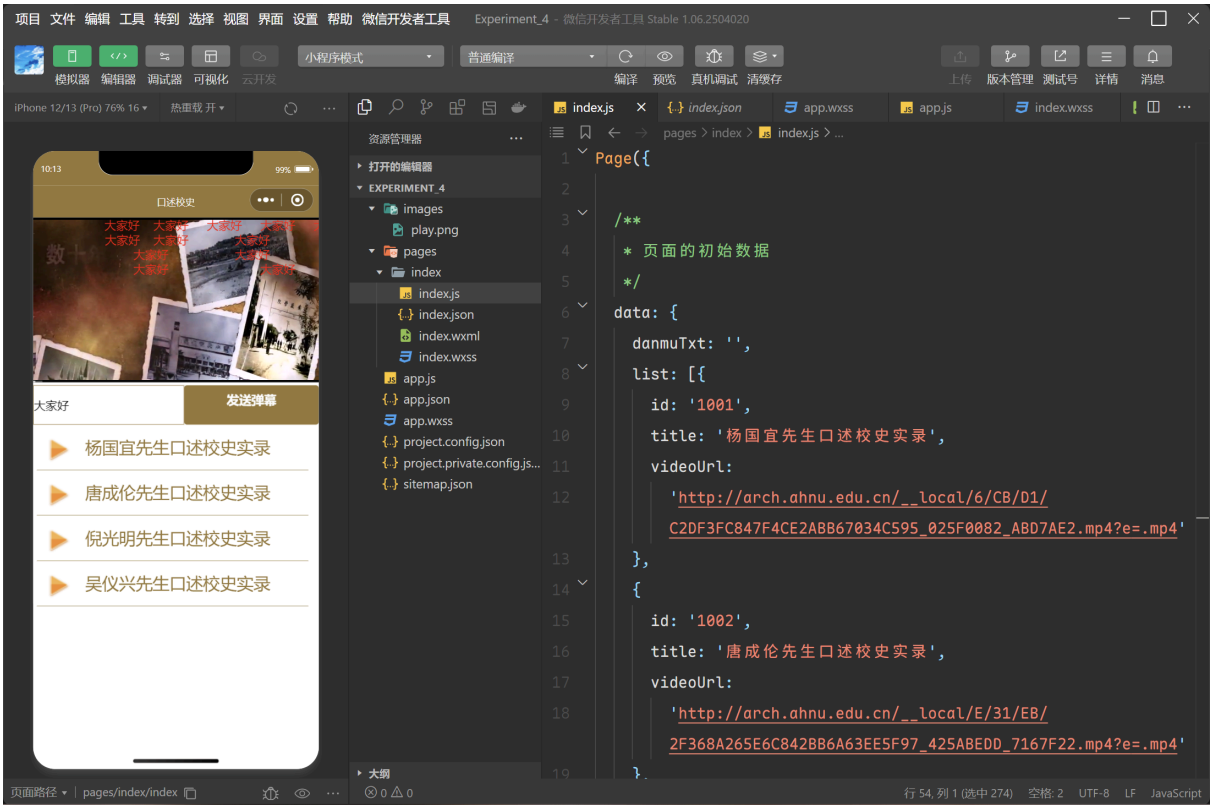
```
  <input type='text' placeholder='请输入弹幕内容' bindinput="getDanmu">
</input>
  <button bind:tap='sendDanmu'>发送弹幕</button>
</view>
```

在 index.js 的 data 项下添加 `danmuTxt` 用于存储弹幕文本，并添加 `getDanmu` 和 `sendDanmu` 函数。

```
/**
 * 更新弹幕内容
 */
getDanmu: function (e) {
  this.setData({
    danmuTxt: e.detail.value
  })
},

/**
*发送弹幕
*/
sendDanmu: function (e) {
  let text = this.data.danmuTxt;
  this.videoCtx.sendDanmu({
    text: text,
    color: 'red'
  })
},
```

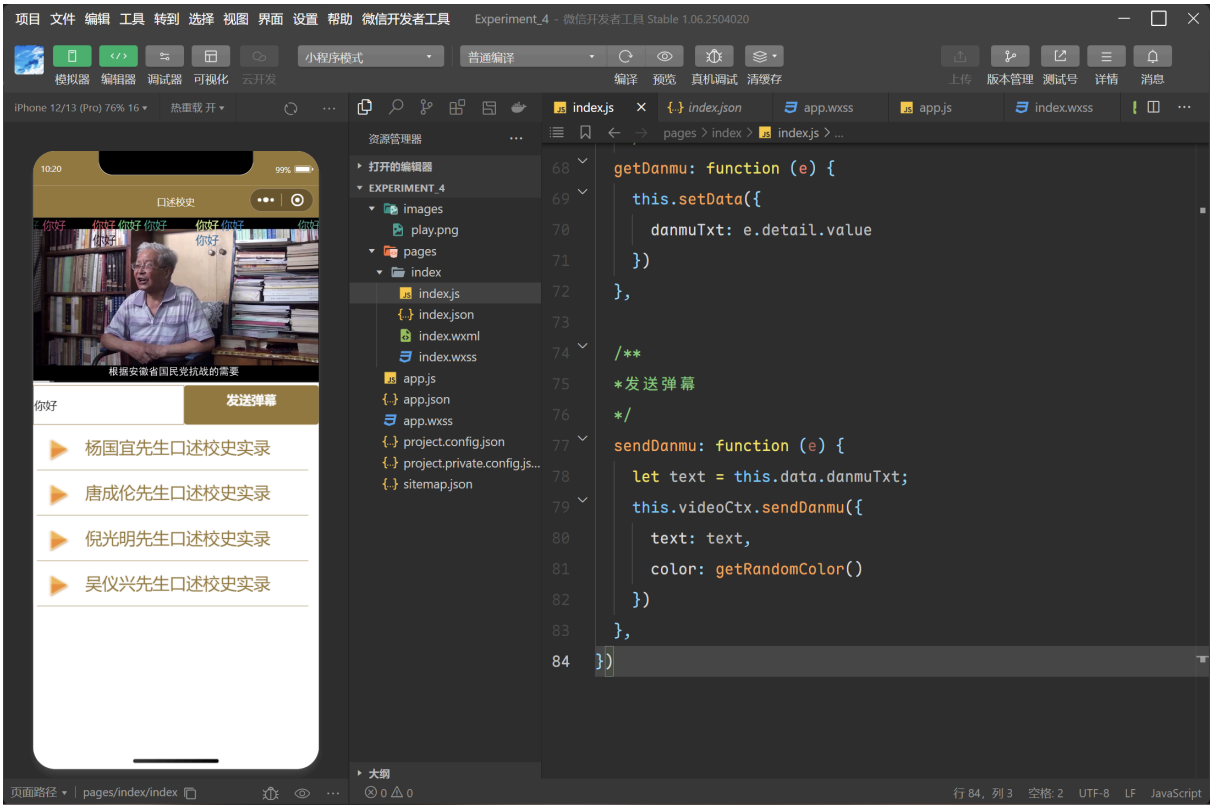
此时能够发送红色的弹幕。



6、在 index.js 中 `Page` 对象之外新增一个 `getRandomColor()` 函数，用于实现颜色的随机生成。

```
//生成随机颜色
function getRandomColor() {
  let rgb = []
  for (let i = 0; i < 3; ++i) {
    let color = Math.floor(Math.random() * 256).toString(16)
    color = color.length == 1 ? '0' + color : color
    rgb.push(color)
  }
  return '#' + rgb.join('')
}
```

修改 `sendDanmu` 函数中 `color: red` 为 `color: getRandomColor()` ,即可实现彩色弹幕的效果。



三、程序运行结果

真机运行的效果如下图。

四、问题总结与体会

遇到的几个问题：

按照实验文档操作后发现无法播放视频，一直处于加载中。检查很久才发现是 <video id='myVideo' src = '{{src}}' controls></video> 中 `src = '{{src}}'` 没有添加，播放器一直无法获得视频 url。

添加 `getRandomColor` 函数时一直存在语法错误，检查后发现这个函数要放在 `Page` 对象之外，而不能放在对象内部。

体会：

本次实验学习了如何使用视频控件播放视频，发送弹幕，以及循环列表如何使用。其中比较重要的收获是知道了如何获取列表中的视频 url 并传递给视频控件，使其能够播放视频。同样比较重要的是弹幕组件的使用，包括怎么从输入框中获取弹幕文本，并显示在视频播放器中。其他的收获还包括 .js 文件中全局函数和对象函数的定义方式存在不同，以及怎么实现随机颜色。