

实验 4：视频播放小程序

本实验来自于周文洁老师的《微信小程序开发实战》第六章。主要内容是使用小程序媒体 API 制作一个视频播放小程序，视频素材来自于某高校档案馆的《口述校史》栏目，它录制了多位耄耋之年的老教工回忆工作时期对大学的印象。本实验的学习目标：1、掌握视频列表的切换方法；2、掌握视频自动播放方法；3、掌握视频随机颜色弹幕效果。

注意事项

1、实验中需要使用四个视频的地址，代码如下：

```
JavaScript
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/C2DF3FC847F4CE2ABB67034C5
      95_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/2F368A265E6C842BB6A63EE5F
      97_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/35687573BA2145023FDAEBAFE
      67_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/7A27865731CF2B096E90B5220  
05_A29CB142_6525BCF.mp4?e=.mp4'  
}  
]
```

2、实验中需要使用到一个播放的图片 play.png，图片下载地址为：
https://gaopursuit.oss-cn-beijing.aliyuncs.com/2022/images_play.zip

6.1 项目创建

本项目创建选择空白文件夹 videoDemo，效果如图 6-1 所示。

单击“新建”按钮完成项目创建，然后准备手动修改页面配置文件。



[视频讲解](#)



图 6-1 小程序项目填写效果图

6.2 页面配置

6.2.1 创建页面文件

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(index)即可。

具体操作如下：

- (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。

6.2.2 删除和修改文件

具体操作如下：

- (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(如图 6-2 所示)。

The screenshot shows a code editor window for 'index.js'. The code is as follows:

```
index.js
1 //index.js
2 page
```

A completion dropdown menu is open at the cursor position, showing several options starting with 'Page'. The second option, 'Page', is highlighted and circled in red. Other visible options include 'pageScrollTo', 'getCurrentPages', 'package', 'pauseBackgroundAudio', 'previewImage', and 'canvasPutImageData'. The right side of the editor shows a 'function ()' placeholder.

图 6-2 输入关键词创建 Page 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(如图 6-3 所示)。

The screenshot shows a code editor window for 'app.js'. The code is as follows:

```
app.js
1 //app.js
2 app
```

A completion dropdown menu is open at the cursor position, showing several options starting with 'App'. The first option, 'App', is highlighted and circled in red. Other visible options include 'APP' and 'getApp'. The right side of the editor shows a 'function ()' placeholder.

图 6-3 输入关键词创建 App 函数

6.2.3 创建其他文件

接下来创建其他自定义文件，本项目还需要一个文件夹用于存放播放图标。文件夹名称由开发者自定义（例如 images），单击目录结构左上角的十号创建文件夹并命名为 images。

本项目用到的图标素材如图 6-4 所示。

右击目录结构中的 images 文件夹，选择“硬盘打开”，然后将图标文件复制、粘贴进该目录。完成后的目录结构如图 6-5 所示。



图 6-4 图标素材展示



图 6-5 页面文件创建完成

此时文件配置就全部完成，6.3 节将正式进行页面布局和样式设计。

6.3 视图设计

6.3.1 导航栏设计

小程序默认导航栏是黑底白字的效果，因此需要在 app.json 中自定义导航栏标题和背景颜色。更改后的 app.json 文件代码如下：

```
1.  {
2.    "pages": [
3.      "pages/index/index"
4.    ],
5.    "window": {
6.      "navigationBarBackgroundColor": "#987938",
7.      "navigationBarTitleText": "口述校史"
8.    }
9.  }
```



视频讲解



图 6-6 自定义导航栏效果

上述代码可以更改所有页面的导航栏标题文本为“口述校史”、背景颜色为金棕色（#987938）。预览效果如图 6-6 所示。



6.3.2 页面设计

页面上主要包含 3 个区域，具体内容解释如下。

视频讲解

- 区域 1：视频播放器，用于播放指定的视频；
- 区域 2：弹幕发送区域，包含文本输入框和发送按钮；
- 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，设计图如图 6-7 所示。

计划使用如下组件。

- 区域 1：<video>组件；
- 区域 2：<view>组件，并定义 class='danmuArea'；
- 区域 2 内部：<input>和<button>组件；
- 区域 3：<view>组件，并定义 class='videoList'；
- 区域 3 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。

1 区域 1(视频组件)设计

区域 1 需要使用<video>组件来实现一个视频播放器。

WXML(pages/index/index. wxml)代码片段如下：

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

其中 controls 属性用于显示播放/暂停、音量等控制组件。

WXSS(pages/index/index. wxss)代码片段如下：

```
1. /* 视频组件样式 */
2. video {
3.   width: 100%;           /* 视频组件宽度为 100% */
4. }
```

当前效果如图 6-8 所示。

由图可见，此时视频播放区域已经出现在屏幕的顶端。

2 区域 2(弹幕区域)设计

区域 2 需要使用<view>组件实现一个单行区域，包括文本输入框和发送按钮。

WXML(pages/index/index. wxml)代码片段修改如下：

```
1. <!-- 区域 1：视频播放器 -->
2. ...
3. <!-- 区域 2：弹幕控制 -->
4. <view class='danmuArea'>
5.   <input type='text' placeholder='请输入弹幕内容'></input>
6.   <button>发送弹幕</button>
7. </view>
```

WXSS(pages/index/index. wxss)代码片段如下：

```
1. /* 区域 2：弹幕控制样式 */
2. /* 2-1 弹幕区域样式 */
3. .danmuArea {
4.   display: flex;           /* flex 模型布局 */
```



图 6-7 页面设计图

```

5.     flex-direction: row;           /* 水平方向排列 */
6.   }
7. /* 2-2 文本输入框样式 */
8. input {
9.   border: 1rpx solid #987938;    /* 1rpx 宽的实线棕色边框 */
10.  flex-grow: 1;                /* 扩张多余空间宽度 */
11.  height: 100rpx;              /* 高度 */
12. }
13. /* 2-3 按钮样式 */
14. button {
15.   color: white;               /* 字体颜色 */
16.   background-color: #987938;   /* 背景颜色 */
17. }

```

当前效果如图 6-9 所示。

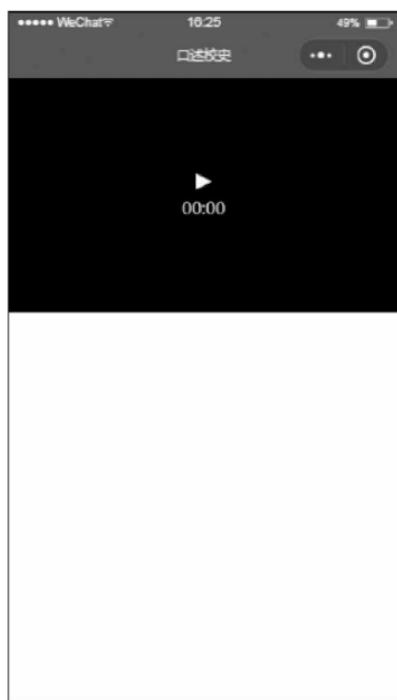


图 6-8 区域 1 预览效果



图 6-9 区域 2 预览效果

3 区域 3(视频列表)设计

区域 3 需要使用`<view>`组件实现一个可扩展的多行区域，每行包含一个播放图标和一个视频标题文本。当前先设计第一行效果，后续使用`wx:for`属性循环添加全部内容。

WXML(pages/index/index.wxml)代码片段修改如下：

```

1. <!-- 区域 1: 视频播放器 -->
2. ...
3. <!-- 区域 2: 弹幕控制 -->
4. ...
5. <!-- 区域 3: 视频列表 -->
6. <view class = 'videoList'>
7.   <view class = 'videoBar'>
8.     <image src = '/images/play.png'></image>

```

```
9.      <text>这是一个测试标题</text>
10.     </view>
11.     </view>
```

WXSS(pages/index/index.wxss)代码片段如下：

```
1. /* 区域 3：视频列表样式 */
2. /* 3-1 视频列表区域样式 */
3. .videoList {
4.   width: 100%; /* 宽度 */
5.   min-height: 400rpx; /* 最小高度 */
6. }
7. /* 3-2 单行列表区域样式 */
8. .videoBar {
9.   width: 95%; /* 宽度 */
10.  display: flex; /* flex 模型布局 */
11.  flex-direction: row; /* 水平方向布局 */
12.  border-bottom: 1rpx solid #987938; /* 1rpx 宽的实线棕色边框 */
13.  margin: 10rpx; /* 外边距 */
14. }
15. /* 3-3 播放图标样式 */
16. image {
17.   width: 70rpx; /* 宽度 */
18.   height: 70rpx; /* 高度 */
19.   margin: 20rpx; /* 外边距 */
20. }
21. /* 3-4 文本标题样式 */
22. text {
23.   font-size: 45rpx; /* 字体大小 */
24.   color: #987938; /* 字体颜色为棕色 */
25.   margin: 20rpx; /* 外边距 */
26.   flex-grow: 1; /* 扩张多余空间宽度 */
27. }
```

当前效果如图 6-10 所示。



图 6-10 区域 3 预览效果

6.4 逻辑实现

6.4.1 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。



视频讲解

WXML(pages/index/index.wxml)代码片段修改如下：

```
1. <! -- 区域 1:视频播放器 -->
2. <video id="myVideo" controls src="{{src}}"/></video>
3. <! -- 区域 2:弹幕控制区域(代码略) -->
4. <! -- 区域 3:视频列表 -->
5. <view class='videoList'>
6.   <view class='videoBar' wx:for='{{list}}' wx:key='video{{index}}'>
7.     <image src='/images/play.png'/>
8.     <text>{{item.title}}</text>
9.   </view>
10. </view>
```

然后在 JS 文件的`data`属性中追加`list`数组,用于存放视频信息。

JS(pages/index/index.js)代码片段修改如下：

```
1. Page({
2.   /**
3.    * 页面的初始数据
4.    */
5.   data: {
6.     list: [
7.       {
8.         id: '299371',
9.         title: '杨国宜先生口述校史实录',
10.        videoUrl: 'http://arch.ahnu.edu.cn/_local/6/CB/D1/C2DF3FC847F4CE2ABB67034C595_
025F0082_ABD7AE2.mp4?e=.mp4'
11.      },
12.      {
13.        id: '299396',
14.        title: '唐成伦先生口述校史实录',
15.        videoUrl: 'http://arch.ahnu.edu.cn/_local/E/31/EB/2F368A265E6C842BB6A63EE5F97_
425ABEDD_7167F22.mp4?e=.mp4'
16.      },
17.      {
18.        id: '299378',
19.        title: '倪光明先生口述校史实录',
20.        videoUrl: 'http://arch.ahnu.edu.cn/_local/9/DC/3B/35687573BA2145023FDAEBAFE67_
AAD8D222_925F3FF.mp4?e=.mp4'
21.      },
22.      {
23.        id: '299392',
24.        title: '吴兴仪先生口述校史实录',
25.        videoUrl: 'http://arch.ahnu.edu.cn/_local/5/DA/BD/7A27865731CF2B096E90B522005_
A29CB142_6525BCF.mp4?e=.mp4'
26.      }
27.   }
28. }
```

```
27.      ]
28.    },
29.  })
```

运行效果如图 6-11 所示。



图 6-11 更新视频列表效果

由图可见,当前已经可以展示全部视频列表。

6.4.2 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。



视频讲解

WXML(pages/index/index.wxml)代码片段修改如下：

```
1. <!-- 区域 3: 视频列表 -->
2. <view class = 'videoList'>
3.   <view class = 'videoBar' wx:for = '{{list}}' wx:key = 'video{{index}}' data - url = '{{item.
videoUrl}}' bindtap = 'playVideo'>
4.     <image src = '/images/play.png'></image>
5.     <text>{{item.title}}</text>
6.   </view>
7. </view>
```

然后在 JS 文件的`onLoad`函数中创建视频上下文,用于控制视频的播放和停止。

JS(pages/index/index.js)代码片段修改如下：

```
1. Page({
2.   /**
3.    * 生命周期函数--监听页面加载
4.    */
5.   onLoad: function(options) {
```

```
6.         this.videoCtx = wx.createVideoContext('myVideo')
7.     },
8. })
```

接着添加自定义函数 playVideo,JS(pages/index/index.js)代码片段如下：

```
1. Page({
2.   /**
3.    * 播放视频
4.    */
5.   playVideo: function(e) {
6.     //停止之前正在播放的视频
7.     this.videoCtx.stop()
8.     //更新视频地址
9.     this.setData({
10.       src: e.currentTarget.dataset.url
11.     })
12.     //播放新的视频
13.     this.videoCtx.play()
14.   },
15. })
```

运行效果如图 6-12 所示。



图 6-12 点击播放视频效果

由图可见,当前已经可以成功播放视频列表中的视频。

6.4.3 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。



视频讲解

WXML(pages/index/index.wxml)代码片段修改如下：

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

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。

WXML(pages/index/index.wxml)代码片段修改如下：

```
1. <!-- 区域 2：弹幕控制 -->
2. <view class='danmuArea'>
3.   <input type='text' placeholder='请输入弹幕内容' bindinput='getDanmu'></input>
4.   <button bindtap='sendDanmu'>发送弹幕</button>
5. </view>
```

对应的 JS(pages/index/index.js)代码片段修改如下：

```
1. Page({
2.   /**
3.    * 页面的初始数据
4.    */
5.   data: {
6.     danmuTxt: '',
7.     list: [ ... ]
8.   }
9.   /**
10.  * 更新弹幕内容
11. */
12. getDanmu: function(e) {
13.   this.setData({
14.     danmuTxt: e.detail.value
15.   })
16. },
17. /**
18.  * 发送弹幕
19. */
20. sendDanmu: function(e) {
21.   let text = this.data.danmuTxt;
22.   this.videoCtx.sendDanmu({
23.     text: text,
24.     color: 'red'
25.   })
26. },
27. })
```

此时可以发出红色文本的弹幕，运行效果如图 6-13 所示。

如果希望发出随机颜色的弹幕内容，可以在 JS 文件中追加自定义函数 getRandomColor。JS(pages/index/index.js)代码片段如下：

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

```

7.     rgb.push(color)
8.   }
9.   return '#' + rgb.join('')
10. }

```

上述代码可以随机生成一个十六进制的颜色，将其在原先需要录入 color 属性的地方调用即可实现彩色弹幕效果。JS(pages/index/index.js)代码片段修改如下：

```

1. Page({
2.   /**
3.    * 发送弹幕
4.    */
5.   sendDanmu: function(e) {
6.     let text = this.data.danmuTxt;
7.     this.videoCtx.sendDanmu({
8.       text:text,
9.       color: getRandomColor()
10.      })
11.    },
12.  })

```

此时可以发出彩色文本的弹幕，运行效果如图 6-14 所示。



图 6-13 发送红色弹幕效果



图 6-14 发送彩色弹幕效果

至此相关的逻辑功能均已实现，项目全部完成。

本实验到这里就全部完成了🎉，你完成的怎么样呢？要继续加油啊！😊