第 7 章　莫凡商城商品详情页设计

课时内容	页面间传递数据与媒体组件及媒体 API	课时	2
教学目标	了解页面间传递数据的方式 掌握媒体组件及媒体 API 的应用		
教学重点	媒体组件及媒体 API		
教学难点	媒体组件及媒体 API		
教学设计	1. 教学思路：介绍页面间传递数据的方式；通过多媒体演示和实机操作讲解媒体组件及媒体 API 的使用。 2. 教学手段：多媒体+计算机 3. 教学资料：教材、多媒体课件		
教学内容			

7.1 页面间传递数据

在首页图书商品列表页面中查看图书商品的详情，需要将图书商品的 id 传递给详情页，在商品详情页根据图书商品的 id 来获取图书商品的具体信息，那么如何在页面间传递数据呢？

下面我们一起来实现将图书商品 id 传递给详情页的功能。

（1）在 app.json 文件里添加意见反馈界面路径“pages/goodsDetail/goodsDetail”。

（2）在 index.js 首页业务逻辑处理文件里，添加跳转商品详情页绑定函数，并且把商品 id 作为参数携带，示例代码如下。

```
var app = getApp();
var host = app.globalData.host;
Page({
  data: {
    indicatorDots: true,
    autoplay: true,
    interval: 5000,
    duration: 1000,
    imgUrls: [
      "/pages/images/haibao/1.jpg",
      "/pages/images/haibao/2.jpg",
      "/pages/images/haibao/3.jpg"
    ],
    hotList:[ ], //热门书籍列表
    spikeList:[ ], //畅销书籍列表
    bestSellerList:[ ], //秒杀时刻列表
    host: host
  },
  onLoad: function (options) {
    var page = this;
    page.getBannerList();
    page.getBookList();
  },
  getBannerList: function () {
    var page = this;
    wx.request({
      url: host + '/api/banner/getBannerList?type=0',
      method: 'GET',
      data: {},
      header: {
        'Content-Type': 'application/json'
      },
      success: function (res) {
```

```
        var code = res.data.code;
        var list = res.data.data;
        if (code == '0000') {
          var code = res.data.code;
          var list = res.data.data;
          if (code == '0000') {
            var imgUrls = new Array();
            for (var i = 0; i < list.length; i++) {
              imgUrls.push(host + "/" + list[i].url);
            }
            page.setData({ imgUrls: imgUrls });
          }
        }
      }
    })
  },
  getBookList: function () {//获取图书列表方法
    var page = this;
    wx.request({
      url: host + '/api/goods/getHomeGoodsList',
      method: 'GET',
      data: {},
      header: {
        'Content-Type': 'application/json'
      },
      success: function (res) {
        var book = res.data.data;
        //将图书列表数据缓存到本地
        wx.setStorage({
          key: 'book',
          data: book,
        })
        //获取缓存到本地图书列表数据
        book = wx.getStorageSync('book');
        console.log(book);
        var hotList = book.rmjs;//热门书籍列表
        var spikeList = book.mssk;//秒杀时刻书籍列表
        var bestSellerList = book.cxsj;//畅销书籍列表
        page.setData({ hotList: hotList });
        page.setData({ spikeList: spikeList });
        page.setData({ bestSellerList: bestSellerList });
      }
    })
  },
  more:function(e){//查看更多
    var id = e.currentTarget.id;
    wx.navigateTo({
      url: '../goods/goods?id='+id,
    })
  },
  seeDetail: function (e) {//查看商品详情
    var goodsId = e.currentTarget.id;
    wx.navigateTo({
      url: '../goodsDetail/goodsDetail?goodsId=' + goodsId,
    })
  },
  searchInput:function(e){//进入搜索页面
    wx.navigateTo({
      url: '../search/search',
    })
  }
})
```

（3）在 goodsDetail.js 商品详情业务逻辑处理文件里，通过 onLoad 生命周期函数来获取传递过来的参数，示例代码如下。

```
var app = getApp();
var host = app.globalData.host;
Page({
  data: {
```

```
  },
  onLoad: function(e) {
    //从参数 e 里面获取上一个页面携带过来的参数
    var goodsId = e.goodsId;
  },
})
```

通过 onLoad: function（e）函数来获取携带过来的参数，携带过来的值都放在参数 e 里面，通过 e.goodsId 或者其他携带过来的值，就可以实现在页面间传递数据了。

7.2 媒体组件及媒体 API 的应用

微信小程序经常需要实现播放音频、播放视频、相机拍照、实时音视频播放、实时音视频录制等功能，我们可以通过 audio 音频组件及 API 实现音频播放功能；通过 video 视频组件及视频 API 实现视频播放功能；通过 camera 相机组件及相机 API 实现相机拍照功能；通过 liver-player 组件实现实时音视频播放功能；通过 live-pusher 组件实现实时音视频录制功能。本节将详细介绍媒体组件及媒体 API 的应用。

7.2.1 audio 音频组件及音频 API

1. audio 音频组件

audio 音频组件需要有唯一的 id，根据 id 使用 wx.createAudioContext（'myAudio'）创建音频播放的环境，从 1.6.0 版本开始，该组件不再维护。建议使用能力更强的 wx.createInnerAudioContext 接口，具体属性如表所示。

audio 音频的属性

属性	类型	默认值	说明
id	string		video 组件的唯一标识符
src	string		要播放音频的资源地址
loop	boolean	false	是否循环播放
controls	boolean	true	是否显示默认控件
poster	string		默认控件上音频封面的图片资源地址，如果 controls 的属性值为 false，则设置 poster 无效
name	string	未知音频	默认控件上的音频名字，如果 controls 的属性值为 false，则设置 name 无效
author	string	未知作者	默认控件上的作者名字，如果 controls 的属性值为 false，则设置 author 无效
binderror	eventHandle		当发生错误时触发 error 事件，detail = {errMsg: MediaError.code}，MediaError.code，错误码：1 为获取资源被用户禁止，2 为网络错误，3 为解码错误，4 为不合适资源
bindplay	eventHandle		当开始/继续播放时触发 play 事件
bindpause	eventHandle		当暂停播放时触发 pause 事件
bindtimeupdate	eventHandle		当播放进度改变时触发 timeupdate 事件，detail = {currentTime, duration}
bindended	eventHandle		当播放到末尾时触发 ended 事件

示例代码如下。

```
<!-- audio.wxml -->
<audio poster="{{poster}}" name="{{name}}" author="{{author}}" src="{{src}}" id="myAudio" controls loop></audio>

<button type="primary" bindtap="audioPlay">播放</button>
<button type="primary" bindtap="audioPause">暂停</button>
<button type="primary" bindtap="audio14">设置当前播放时间为 14 秒</button>
<button type="primary" bindtap="audioStart">回到开头</button>
```

```
// audio.js
Page({
  onReady: function (e) {
    // 使用 wx.createAudioContext 获取 audio 上下文 context
    this.audioCtx = wx.createAudioContext('myAudio')
  },
  data: {
    poster: 'http://y.gtimg.cn/music/photo_new/T002R300x300M000003rsKF44GyaSk.jpg?max_age=2592000',
    name: '此时此刻',
```

```
        author: '许巍',
                        src:      'http://ws.stream.qqmusic.qq.com/M500001VfvsJ21xFqb.mp3?guid=ffffffff82def4af4b12b3cd9337d5e7&uin=346897220&vkey=6292F51E1E384E06DCBDC9AB7C49FD713D632D313AC4858BACB8DDD29067D3C601481D36E62053BF8DFEAF74C0A5CCFADD6471160CAF3E6A&fromtag=46',
      },
      audioPlay: function () {//绑定的播放事件
        this.audioCtx.play()
      },
      audioPause: function () {//绑定的暂停事件
        this.audioCtx.pause()
      },
      audio14: function () {//指定多少秒开始播放
        this.audioCtx.seek(14)
      },
      audioStart: function () {//从头播放
        this.audioCtx.seek(0)
      }
    })
```

2. AudioContext 音频 API

从基础库 1.6.0 开始，AudioContext 接口停止维护，可以使用 wx.createAudioContext（string id, Object this） 创建 audio 上下文 AudioContext 对象。AudioContext 对象提供以下方法。

（1）AudioContext.pause()暂停音频。

（2）AudioContext.play()播放音频。

（3）AudioContext.seek（number position）跳转到指定位置，position 的单位为秒（s）。

（4）AudioContext.setSrc（string src）设置音频地址。

3. InnerAudioContext 音频 API

使用 wx.createInnerAudioContext()可以创建内部 audio 上下文 InnerAudioContext 对象。AudioContext 对象提供以下方法。

（1）InnerAudioContext.play()播放音频。

（2）InnerAudioContext.pause()暂停音频。暂停后的音频再播放会从暂停处开始播放。

（3）InnerAudioContext.stop()停止音频。停止后的音频再播放会从头开始播放。

（4）InnerAudioContext.seek（number position）跳转到指定位置。

（5）InnerAudioContext.destroy()销毁当前实例。

（6）InnerAudioContext.onCanplay（function callback）监听音频进入可以播放状态的事件，但不保证后面可以流畅播放。

（7）InnerAudioContext.offCanplay（function callback）取消监听音频进入可以播放状态的事件。

（8）InnerAudioContext.onPlay（function callback）监听音频播放事件。

（9）InnerAudioContext.offPlay（function callback）取消监听音频播放事件。

（10）InnerAudioContext.onPause（function callback）监听音频暂停事件。

（11）InnerAudioContext.offPause（function callback）取消监听音频暂停事件。

（12）InnerAudioContext.onStop（function callback）监听音频停止事件。

（13）InnerAudioContext.offStop（function callback）取消监听音频停止事件。

（14）InnerAudioContext.onEnded（function callback）监听音频自然播放至结束的事件。

（15）InnerAudioContext.offEnded（function callback）取消监听音频自然播放至结束的事件。

（16）InnerAudioContext.onTimeUpdate（function callback）监听音频播放进度更新事件。

（17）InnerAudioContext.offTimeUpdate（function callback）取消监听音频播放进度更新事件。

（18）InnerAudioContext.onError（function callback）监听音频播放错误事件。

（19）InnerAudioContext.offError（function callback）取消监听音频播放错误事件。

（20）InnerAudioContext.onWaiting（function callback）监听音频加载中事件。当音频因为数据不足，需要停下来加载时会触发。

（21）InnerAudioContext.offWaiting（function callback）取消监听音频加载中事件。

（22）InnerAudioContext.onSeeking（function callback）监听音频进行跳转操作的事件。

（23）InnerAudioContext.offSeeking（function callback）取消监听音频进行跳转操作的事件。

（24）InnerAudioContext.onSeeked（function callback）监听音频完成跳转操作的事件。

（25）InnerAudioContext.offSeeked（function callback）取消监听音频完成跳转操作的事件。

4. BackgroundAudioManager 背景音频 API

BackgroundAudioManager 对象实例，可通过 wx.getBackgroundAudioManager 获取。BackgroundAudioManager 提供以下方法。

（1）BackgroundAudioManager.play()播放音频。

（2）BackgroundAudioManager.pause()暂停音频。

（3）BackgroundAudioManager.seek（number currentTime）跳转到指定位置。

（4）BackgroundAudioManager.stop()停止音频。

（5）BackgroundAudioManager.onCanplay（function callback）监听背景音频进入可播放状态事件，但不保证后面可以流畅播放。

（6）BackgroundAudioManager.onWaiting（function callback）监听音频加载中事件。当音频因为数据不足，需要停下来加载时会触发。

（7）BackgroundAudioManager.onError（function callback）监听背景音频播放错误事件。

（8）BackgroundAudioManager.onPlay（function callback）监听背景音频播放事件。

（9）BackgroundAudioManager.onPause（function callback）监听背景音频暂停事件。

（10）BackgroundAudioManager.onSeeking（function callback）监听背景音频开始跳转操作事件。

（11）BackgroundAudioManager.onSeeked（function callback）监听背景音频完成跳转操作事件。

（12）BackgroundAudioManager.onEnded（function callback）监听背景音频自然播放结束事件。

（13）BackgroundAudioManager.onStop（function callback）监听背景音频停止事件。

（14）BackgroundAudioManager.onTimeUpdate（function callback）监听背景音频播放进度更新事件，只有小程序在前台时会回调。

（15）BackgroundAudioManager.onNext（function callback）监听用户在系统音乐播放面板中单击下一曲事件（仅用于 iOS）。

（16）BackgroundAudioManager.onPrev（function callback）监听用户在系统音乐播放面板中单击上一曲事件（仅用于 iOS）。

7.2.2 video 视频组件及视频 API

1. video 视频组件

video 视频组件是用来播放视频的组件，可以控制是否显示默认播放控件（“播放/暂停”按钮、播放进度、时间），可以发送弹幕信息等，video 组件的默认宽度为 300 px，高度为 225 px，宽、高需要通过用 WXSS 设置 width 和 height 来调整，具体属性如表所示。

video 视频的属性

属性	类型	默认值	说明
src	string		要播放视频的资源地址
controls	boolean	true	是否显示默认播放控件（“播放/暂停”按钮、播放进度、时间）
danmu-list	Object	Array	弹幕列表
danmu-btn	boolean	false	是否显示弹幕按钮，只在初始化时有效，不能动态变更
enable-danmu	boolean	false	是否展示弹幕，只在初始化时有效，不能动态变更
autoplay	boolean	false	是否自动播放
bindplay	eventHandle		当开始/继续播放时触发 play 事件
bindpause	eventHandle		当暂停播放时触发 pause 事件
bindended	eventHandle		当播放到末尾时触发 ended 事件
bindtimeupdate	eventHandle		播放进度变化时触发，event.detail = {currentTime： '当前播放时间'}。触发频率应该在 250 ms/次
objectFit	string	contain	当视频大小与 video 容器大小不一致时，视频的表现形式：contain 为包含，fill 为填充，cover 为覆盖

示例代码如下。

```
<view class="section tc">
        <video id="myVideo" src="http://wxsnsdy.tc.qq.com/105/20210/snsdyvideodownload?filekey=30280201010421301f0201690402534804102ca905ce620b1241b726bc41dcff44e00204012882540400&bizid=1023&hy=SH&fileparam=302c020101042530230204136ffd93020457e3c4ff02024ef202031e8d7f02030f424002040 45a320a0201000400" danmu-list="{{danmuList}}" enable-danmu danmu-btn controls></video>
    <view class="btn-area">
      <button bindtap="bindButtonTap">获取视频</button>
      <input bindblur="bindInputBlur"/>
      <button bindtap="bindSendDanmu">发送弹幕</button>
    </view>
```

```
</view>
```

```
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('')
}

Page({
  onReady: function (res) {
    this.videoContext = wx.createVideoContext('myVideo')
  },
  inputValue: '',
    data: {
        src: '',
    danmuList: [
      {
        text: '第 1s 出现的弹幕',
        color: '#ff0000',
        time: 1
      },
      {
        text: '第 3s 出现的弹幕',
        color: '#ff00ff',
        time: 3
    }]
    },
  bindInputBlur: function(e) {
    this.inputValue = e.detail.value
  },
  bindButtonTap: function() {
    var that = this
    wx.chooseVideo({
      sourceType: ['album', 'camera'],
      maxDuration: 60,
      camera: ['front', 'back'],
      success: function(res) {
        that.setData({
          src: res.tempFilePath
        })
      }
    })
  },
  bindSendDanmu: function () {
    this.videoContext.sendDanmu({
      text: this.inputValue,
      color: getRandomColor()
    })
  }
})
```

界面效果如图所示。

视频播放界面

2. video 视频 API

可以使用 wx.createVideoContext（string id, Object this）创建 video 上下文 VideoContext 对象。VideoContext 对象提供以下方法。

（1）VideoContext.play()播放视频。
（2）VideoContext.pause()暂停视频。
（3）VideoContext.stop()停止视频。
（4）VideoContext.seek（number position）跳转到指定位置。
（5）VideoContext.sendDanmu（Object data）发送弹幕。
（6）VideoContext.playbackRate（number rate）设置倍速播放。
（7）VideoContext.requestFullScreen（Object object）进入全屏。
（8）VideoContext.exitFullScreen()退出全屏。
（9）VideoContext.showStatusBar()显示状态栏，仅在 iOS 全屏下有效。
（10）VideoContext.hideStatusBar()隐藏状态栏，仅在 iOS 全屏下有效。
（11）wx.chooseVideo（Object object）拍摄视频或从手机相册中选视频。
（12）wx.saveVideoToPhotosAlbum（Object object） 保存视频到系统相册，支持 MP4 视频格式，调用前需要用户授权 scope.writePhotosAlbum。

7.2.3 camera 相机组件及相机 API

1. camera 相机组件

camera 是相机组件，在使用的时候需要用户授权 scope.camera。camera 相机组件是由客户端创建的原生组件，它的层级是最高的，不能通过 z-index 控制层级，可使用 cover-view、 cover-image 覆盖在上面，同一页面只能插入一个 camera 组件，不能在 scroll-view、swiper、picker-view、movable-view 中使用 camera 组件。camera 相机组件的属性如表所示。

camera 相机组件的属性

属性	类型	默认值	说明
mode	string	normal	应用模式，只在初始化时有效，不能动态变更， normal 为相机模式、scanCode 为扫码模式
device-position	string	Back	前置或后置，值为 front、back
flash	string	auto	闪光灯，auto 为自动、on 为打开、off 为关闭、torch 为常亮
frame-size	string	medium	指定期望的相机帧数据尺寸， small 为小尺寸、medium 为中尺寸、large 为大尺寸
bindinitdone	eventhandle		用户不允许使用摄像头时触发
bindscancode	eventhandle		在扫码识别成功时触发，仅在 mode="scanCode"时生效
bindstop	eventhandle		摄像头在非正常终止时触发，如退出后台等情况
binderror	eventhandle		用户不允许使用摄像头时触发

示例代码如下。

```
<camera device-position="back" flash="off" binderror="error" style="width: 100%; height: 300px;"></camera>
<button type="primary" bindtap="takePhoto">拍照</button>
<view>预览</view>
```

```
<image mode="widthFix" src="{{src}}"></image>
```

```
Page({
  takePhoto() {
    const ctx = wx.createCameraContext()
    ctx.takePhoto({
      quality: 'high',
      success: (res) => {
        this.setData({
          src: res.tempImagePath
        })
      }
    })
  },
  error(e) {
    console.log(e.detail)
  }
})
```

2. camera 相机 API

可以使用 wx.createCameraContext()创建 camera 上下文 CameraContext 对象，CameraContext 与页面内唯一的 camera 组件绑定，操作对应的 camera 组件。CameraContext 对象提供以下方法。

（1）CameraContext.onCameraFrame（onCameraFrameCallback callback）获取 Camera 实时帧数据。

（2）CameraContext.takePhoto（Object object）拍摄照片。

（3）CameraContext.startRecord（Object object）开始录像。

（4）CameraContext.stopRecord()结束录像。

（5）CameraFrameListener 是 CameraContext.onCameraFrame() 返回的监听器，CameraFrameListener.start()开始监听帧数据，CameraFrameListener.stop()停止监听帧数据。

7.2.4 live-player 实时音视频播放

live-player 为实时音视频播放组件，它的使用是针对特定类目开放的，需要先通过类目审核，再在小程序管理后台中通过“设置”→“接口设置”命令自助开通该组件的权限。目前支持的类目有社交（直播）、教育（在线教育）、医疗（互联网医院、公立医院）、金融[银行、信托、基金、证券/期货、证券/期货投资咨询、保险、征信业务、新三板信息服务平台、股票信息服务平台（港股/美股）、消费金融]、汽车（汽车预售服务）、政府主体账号、工具（视频客服）。live-player 实时音视频播放组件的属性如表所示。

live-player 实时音视频播放组件的属性

属性	类型	默认值	说明
src	string		音视频地址，目前仅支持 flv、rtmp 格式
mode	string	live	live（直播）、RTC（实时通话）
autoplay	boolean	false	自动播放
muted	boolean	false	是否静音
orientation	string	vertical	画面方向，可选值有 vertical、horizontal
object-fit	string	contain	填充模式，可选值有 contain、fillCrop
background-mute	Boolean	false	进入后台时是否静音（已废弃，默认退台静音）
min-cache	number	1	最小缓冲区，单位为秒
max-cache	number	3	最大缓冲区，单位为秒
sound-mode	string	speaker	声音输出方式，speaker 为扬声器、ear 为听筒
auto-pause-if-navigate	boolean	true	当跳转到其他小程序页面时，是否自动暂停本页面的实时音视频播放
auto-pause-if-open-native	boolean	true	当跳转到其他微信原生页面时，是否自动暂停本页面的实时音视频播放
bindstatechange	eventHandle		播放状态变化事件，detail = {code}
bindfullscreenchange	eventHandle		全屏变化事件，detail = {direction, fullScreen}
bindnetstatus	eventHandle		网络状态通知，detail = {info}

示例代码如下。

```
<live-player src="https://domain/pull_stream" mode="RTC" autoplay bindstatechange="statechange" binderror= "error" style="width: 300px; height: 225px;" />
```

```
Page({
  statechange(e) {
```

```
    console.log('live-player code:', e.detail.code)
  },
  error(e) {
    console.error('live-player error:', e.detail.errMsg)
  }
})
```

7.2.5 live-pusher 实时音视频录制

live-pusher 为实时音视频录制组件，它的使用需要取得用户授权 scope.camera、scope.record，针对特定类目开放的，需要先通过类目审核，再在小程序管理后台中通过“设置”→“接口设置”命令自助开通该组件的权限。现在支持的类目有社交（直播）、教育（在线视频课程）、医疗（互联网医院、公立医院）、金融[银行、信托、基金、证券/期货、证券/期货投资咨询、保险、征信业务、新三板信息服务平台、股票信息服务平台（港股/美股）、消费金融（金融产品视频客服理赔、金融产品推广直播等）]、汽车（汽车预售服务）、政府主体账号、工具（视频客服）。live-pusher 实时音视频录制组件的属性如表所示。

live-pusher 实时视频录制组件的属性

属性	类型	默认值	说明
url	string		推流地址，目前仅支持 flv、rtmp 格式
mode	string	RTC	类型，包括 SD（标清）、HD（高清）、FHD（超清）、RTC（实时通话）
autopush	boolean	false	自动推流
muted	boolean	false	是否静音
enable-camera	boolean	true	开启摄像头
auto-focus	boolean	true	自动聚集
orientation	string	vertical	垂直 vertical、水平 horizontal
beauty	number	0	美颜
whiteness	number	0	美白，取值范围为 0～9，0 表示关闭
aspect	string	9：16	宽高比，可选值有 3 ： 4、9 ： 16
min-bitrate	number	200	最小码率
max-bitrate	number	1000	最大码率
audio-quality	string	high	高音质（48 kHz）或低音质（16 kHz），值为 high、low
waiting-image	string		进入后台时推流的等待画面
waiting-image-hash	string		等待画面资源的 MD5 值
zoom	boolean	false	调整焦距
device-position	string	front	前置或后置，值为 front、back
background-mute	boolean	false	进入后台时是否静音
mirror	boolean	false	设置推流画面是否镜像
background-mute	boolean	false	进入后台时是否静音
bindstatechange	eventhandle		状态变化事件，detail = {code}
bindnetstatus	eventhandle		网络状态通知，detail = {info}
binderror	eventhandle		渲染错误事件，detail = {errMsg, errCode}
bindbgmstart	eventhandle		背景音开始播放时触发
bindbgmprogress	eventhandle		背景音进度变化时触发，detail = {progress, duration}
bindbgmcomplete	eventhandle		背景音播放完成时触发

示例代码如下。

```
<live-pusher url="https://domain/push_stream" mode="RTC" autopush bindstatechange="statechange" style="width: 300px; height: 225px;" />

Page({
  statechange(e) {
    console.log('live-pusher code:', e.detail.code)
  }
})
```

小结	本节课主要介绍了页面间传递数据、audio 音频组件及音频 API 、video 视频组件及 API camera 相机组件及相机 API 的应用等知识。
练习	