《微信小程序开发》实验报告五

小程序实战 —— 音乐小程序制作

班级 计科 1902

1. 实验目的

通过本次实验综合运用网页前端知识，完成一个综合的音乐小程序设计制作全过程。

掌握使用 wxml 对小程序进行基本页面布局的基本方法。

能熟练使用 wxss 对小程序的页面进行合理布局。

掌握小程序生命周期函数的用法，能使用 JavaScript 在小程序页面加载前对资源进行初始化。

掌握模块化的基本技能，能进行组件复用。

掌握小程序页面基本属性的配置方法，能对页面进行个性化设置。

掌握小程序静态数据加载的方法。

2. 实验内容及要求

准备工作：为了方便后续的学习和实践，建议同学们自带电脑，在自己电脑上进行实验内容，并完成实验报告。本次实验为基础内容，认真完成本次实验内容即可掌握开发微信小程序的基本工具使用。

（一） 实验内容

1、 对音乐小程序进行功能分析，理清楚小程序需要实现的功能模块，找到小程序编写入手点

2、 对小程序页面数据进行渲染，掌握小程序页面数据的渲染方式，并对相应的布局设置样式；

3、 学习小程序组件的相关内容，将相同逻辑抽象成组件，通过父子组件传值和循环渲染完成复杂页面

4、 掌握小程序中数据的渲染方式，自定义数据配置，通过引入本地自定义数据配置，对页面进行渲染

（二）实验步骤（仔细阅读，按照步骤做完）

1. 对音乐小程序进行功能分析，为实际页面编写做准备

（1） 查看音乐小程序效果图；

（2） 对效果图进行分析，确定小程序的页面结构

通过效果图可以看出，该小程序主要包含三个页面，在 pages 目录下分别新建三个页面对该页面进行编写：

--pages

|--hot

|--broadcast

|--user

（3） 完成页面的基本设置，为页面开发做准备

a) 在 app.json 中 pages 数组中对三个页面进行注册

b) 在 app.json 中 tabBar 数组中对定义导航栏

2. 编写「热歌推荐」页面

（1） 首先确保点击 tabBar 中按钮，三个页面能够正常进行跳转

（2） 分析页面数据结构

通过观察，页面包括三位歌手、他们的代表作歌曲和图片封面，首先编写周杰伦相关的内容

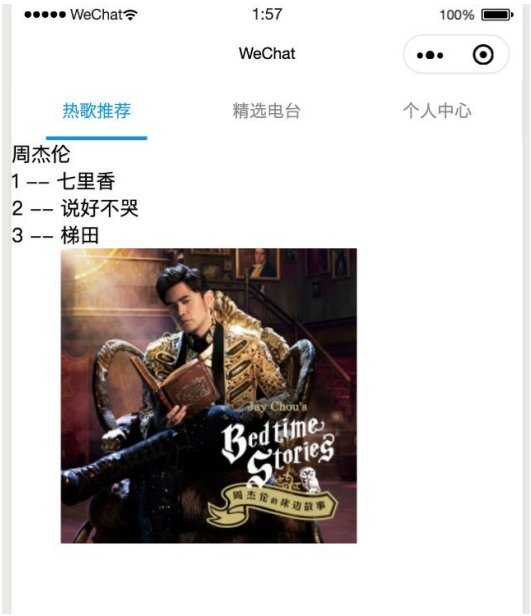
（3） 在 hot.js 文件中定义好所需要的数据，如下图所示：

```
data: {
    "id": 1,
    "name": "周杰伦",
    "src": "http://cdnmusic.migu.cn/picture/2019/1031/0254/ASd6c2d9697d2a4f5f96508c8a7ec8b1a8.jpg",
    "songs": [
        { "id": 1, "title": "七里香" },
        { "id": 2, "title": "说好不哭" },
        { "id": 3, "title": "梯田" }
    ]
},
```

（4） 编写 hot.wxml 文件，先不考虑样式，使数据能够正常显示，可参考下图：

```
<!--pages/hot/hot.wxml-->
<view >
  <view class="author">{{name}}</view>
  <view class="content">
    <view class="song-list">
      <view class="song" wx:for="{{songs}}" wx:key="{{index}}" wx:for-item="song">
        <text >{{song.id}} -- </text>
        <text >{{song.title}}</text>
      </view>
    </view>
      <view class="profile">
        <image class="banner" mode="aspectFit" src="{{src}}"/>
      </view>
  </view>
</view>
```

编写完成后，此时的「热歌推荐」页面应该如下图所示：



3. 为「热歌推荐」页面添加样式

（1） 给歌手增加样式：

```
.author{
    font-weight: bold;
    font-size: 40rpx;
    background-color: #f5f5f5;
}
```

（2） 调整歌手封面大小

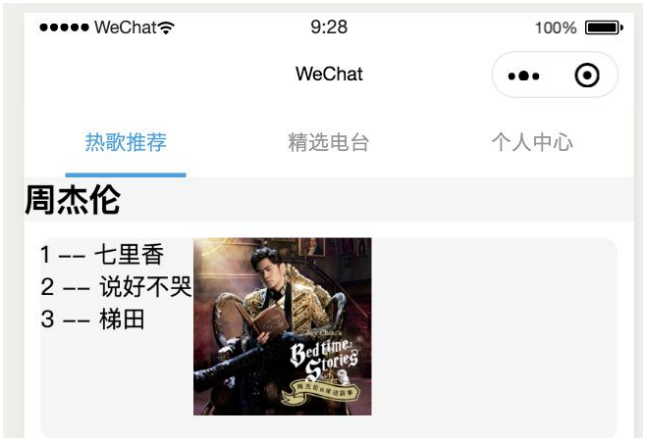
```
.profile{
    height: 250rpx;
    width: 250rpx;
}
```

```
.banner{
    width: 220rpx;
    height: 220rpx;
}
```

（3） 使用弹性布局（flex 布局），使封面显示在歌曲的右侧：

在 .content 中添加：`display: flex;` 使整个容器都采用弹性布局。

完成后的效果应该如下图所示：



（4） 使封面显示在最右侧并对齐

在 .profile 中添加属性：`margin-left: auto;`

完成后的效果显示封面已对齐在了最右侧：

（5） 为页面添加背景色

在 hot.wxss 最上方添加：

```
page{background-color: darkslategrey;}
```

此时效果如下：

（6） 调整歌手名与页面顶部边距

为 .author 添加 `margin: 20rpx;` ，效果如下图所示：

（7） 使歌曲名填充整个空间

为歌曲名 .song 添加样式：

```
.song{
    padding: 20rpx;
}
```

完成后的效果如下：

（8） 让封面照片显示在上下边距的正中间

为 .profile 增加样式：

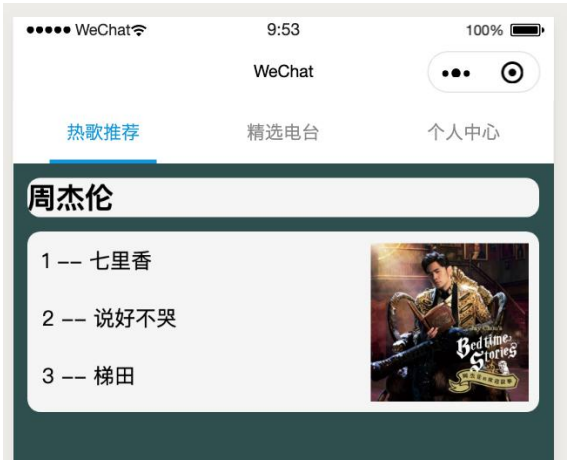
```
display: flex;
align-items: center;
justify-content: center;
```

完成后的效果如下图所示：

可以看到封面已居中显示

（9） 为边框增加圆角，使得卡片更加美观

分别为 .author 和 .content 添加样式：`border-radius: 10px;`，完成效果如图：



（10） 调整歌手名左边距

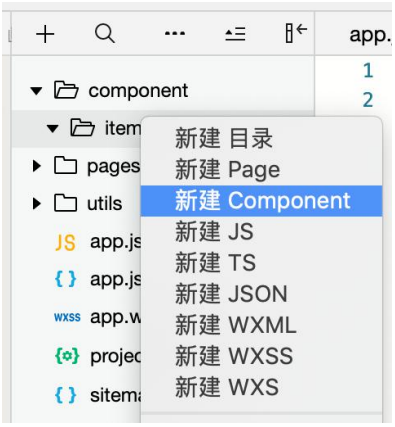
为 .author 添加样式 `padding-left: 10px;`，效果如下：

4. 实现组件复用，将相同逻辑抽象成一个组件并引入

（1） 将周杰伦及其歌曲和封面抽象成组件

观察到其余歌手的显示方式与周杰伦的相同，应该将刚才的内容抽象成为一个可复用的组件模板，对数据循环渲染

（2） 新建 component 文件夹，并在其下新建 itemCard 组件：



（3） 将原来 hot.wxml 和 hot.wxss 中的内容拷贝到 itemCard.wxml 和 itemCard.wxss 中（除了 `page{background-color:darkslategrey;}` ）

（4） 删掉原来 hot.wxml 和 hot.wxss 中的内容，并引入 itemCard 这个组件：在 hot.json 中配置引入该组件：

```
{
  "usingComponents": {
    "itemCard": "../../components/itemCard/itemCard"
  }
}
```

（5） 在 hot.wxml 中引入该组件

```
<!--pages/hot/hot.wxml-->
<view class="music-hot">
    <itemCard />
</view>
```

但此时因为没有传递数据，所以显示结果如下图所示：

（6） 为 itemCard 组件传递数据。

a) 修改 hot.js 中的数据，将它们写在一个对象中

```
data: {
    musics: {
        "id": 1,
        "name": "周杰伦",
        "src": "http://cdnmusic.migu.cn/picture/2019/1031/0254/ASd6c2d9697d2a4f5f96508c8a7ec8b1a8.jpg",
        "songs": [
            { "id": 1, "title": "七里香" },
            { "id": 2, "title": "说好不哭" },
            { "id": 3, "title": "梯田" }
        ]
    }
},
```

b) 在 hot.wxml 中引入该数据

```
<!--pages/hot/hot.wxml-->
<view class="music-hot">
    <itemCard item="{{musics}}"/>
</view>
```

c) 配置 itemCard，使它能够接受到该值：在 itemCard.js 中设置 propertities：

```
/**
 * 组件的属性列表
 */
properties: {
    item: {
        type: Object,
        value: {}
    }
},
```

d) 此时还无法正常显示，还需要对 itemCard 再进行修改

```
<!--component/itemCard/itemCard.wxml-->
<view >
  <view class="author">{{item.name}}</view>
  <view class="content">
    <view class="song-list">
      <view class="song" wx:for="{{item.songs}}" wx:key="{{index}}" wx:for-item="song">
        <text >{{song.id}} -- </text>
        <text >{{song.title}}</text>
      </view>
    </view>
      <view class="profile">
        <image class="banner" mode="aspectFit" src="{{item.src}}"/>
      </view>
  </view>
</view>
```

（7） 查看显示结果

5. 引入自定义数据，使用生命周期函数加载自定义数据，并通过自定义数据进行渲染

（1） 回顾之前的数据获取方式

之前的数据获取方式将数据集定义在 hot.js 中的 data 对象中，当数据量很大或者需要修改时，hot.js 维护将变的十分不方便。

（2） 新建配置文件，将数据写在该配置文件中

a) 新建 datas 文件夹，并在其下建立 musicList.js 文件

b) 将数据写入配置文件

c) 在 musicList 文件中将数据写入一个对象：

```
var json = [
    {
        "id": 1,
        "name": "周杰伦",
        "src": "http://cdnmusic.migu.cn/picture/2019/1031/0254/ASd6c2d9697d2a4f5f96508c8a7ec8b1a8.jpg",
        "songs": [
            { "id": 1, "title": "七里香" },
            { "id": 2, "title": "说好不哭" },
            { "id": 3, "title": "梯田" }
        ]
    },
    {
        "id": 2,
        "name": "李荣浩",
        "src": "http://cdnmusic.migu.cn/picture/2019/1031/0130/AM5251251132424657bd7190cc3690fa40.jpg",
        "songs": [
            { "id": 1, "title": "太坦白" },
            { "id": 2, "title": "不将就" },
            { "id": 3, "title": "老街" }
        ]
    },
    {
        "id": 3,
        "name": "许嵩",
        "src": "http://cdnmusic.migu.cn/picture/2019/0422/0849/AS1704070955546876.jpg",
        "songs": [
            { "id": 1, "title": "有何不可" },
            { "id": 2, "title": "雅俗共赏" },
            { "id": 3, "title": "素颜" }
        ]
    }
]
```

d) 将该对象导出，使得外部可以引入该数据

在 musicList.json 文件末尾添加：

```
module.exports = {
    musics: json
}
```

则外部文件便可通过 musics 引入该数据

（3） 在 hot.js 中引入刚定义好的数据

a) 在 hot.js 开头添加引入语句：

```
4   var musicList = require("../../datas/musicList.js");
5
6
7   Page({
8
9       /**
0       * 页面的初始数据
```

b) 将原来 data 中 musics 的值删去：

```
data: {
    musics: {

    }
},
```

c) 在生命周期函数 onLoad 中更新 data 数据：

```
/**
* 生命周期函数--监听页面加载
*/
onLoad: function (options) {
    this.setData({
        musics: musicList.musics
    })
},
```

d) 在生命周期函数 onLoad 中添加辅助信息，查看是否正确获取到数据：

```
/**
* 生命周期函数--监听页面加载
*/
onLoad: function (options) {
    this.setData({
        musics: musicList.musics
    })
    console.log(musicList)
    console.log(this.data.musics)
},
```

e) 此时还不能正确显示，因为此时返回的数据是一个数组，需要对 hot.wxml 进行修改。

```
<!--pages/hot/hot.wxml-->
<view class="music-hot">
    <view wx:for="{{musics}}" wx:for-item="item" wx:for-index="index" wx:key="index">
        <itemCard item="{{item}}"/>
    </view>
</view>
```

修改之后的显示结果如下图所示：

f) 为顶部添加边距：

```
<!--pages/hot/hot.wxml-->
<view style="padding-bottom: 20px"></view>
<view class="music-hot">
    <view wx:for="{{musics}}" wx:for-item="item" wx:for-index="index" wx:key="index">
        <itemCard item="{{item}}"/>
    </view>
</view>
```

添加之后的显示效果如下图所示：

5、 编写「精选电台」界面

（1） 准备数据源：

在 datas 文件夹下新建 broadcastList.js 文件，在其中写入所需要的配置，并在最后对数据进行导出：

```
var json = [
    {
        name: "漫步春天",
        thumb: "http://hiphotos.qianqian.com/ting/pic/item/838ba61ea8d3fd1fb4912e42354e251f95ca5f2a.jpg"
    },
    {
        name: "私日冬雨",
        thumb: "http://hiphotos.qianqian.com/ting/pic/item/b3119313b07eca80b93485cf932397dda14483e1.jpg"
    },
    {
        name: "温暖冬日",
        thumb: "http://hiphotos.qianqian.com/ting/pic/item/728da9773912b31b106b9e4e8518367adab4e156.jpg"
    },
    {
        name: "热歌",
        thumb: "http://hiphotos.qianqian.com/ting/pic/item/a2cc7cd98d1001e92b3c4768bb0e7bec54e79750.jpg"
    },

    {
        name: "KTV金曲",
        thumb: "http://hiphotos.qianqian.com/ting/pic/item/838ba61ea8d3fd1f0b0bf15d334e251f95ca5f52.jpg"
    },
    {
        name: "成名曲".
```

```
module.exports = {
    broadcasts: json
}
```

（2） 新建 broadItem 组件，并在 broadcast 页面中进行引入：

```
<!--pages/broadcast/broadcast.wxml-->
<text>pages/broadcast/broadcast.wxml</text>
<view class="page">
    <broadItem item="{{item}}"></broadItem>
</view>
```

```
{
  "usingComponents": {
      "broadItem": "../../components/broadItem/broadItem"
  }
}
```

当页面能正常显示出 broadcast 和 broadItem 时说明引入成功：

（3） 引入电台数据：

在 broadcast.js 中引入数据源，并在说明周期函数中对数据进行初始化：

```
// pages/broadcast/broadcast.js
var broadcastList = require("../../datas/broadcastList.js");
Page({

    /**
     * 页面的初始数据
     */
    data: {
        broadcasts: []
    },

    /**
     * 生命周期函数--监听页面加载
     */
    onLoad: function (options) {
        this.setData({
            broadcasts: broadcastList.broadcasts
        })
    },
```

（4） 在 broadcast.wxml 中循环渲染 broadItem：

```
<!--pages/broadcast/broadcast.wxml-->
<text>pages/broadcast/broadcast.wxml</text>
<view class="page">
    <view class="content" wx:for="{{broadcasts}}" wx:for-item="item" wx:for-index="index" wx:key="index">
        <broadItem item="{{item}}"></broadItem>
    </view>
</view>
```

当页面显示如下时，说明正常引入数据和组件，并可正常渲染

（5） 开始编写 broadItem 组件：

a) 编写 broadItem.wxml 文件：

```
<!--components/broadItem/broadItem.wxml-->
<view class='broad-item' style="background-image: url('{{item.thumb}}');" >
    <text>{{item.name}}</text>
</view>
```

b) 设置 broadItem.js 文件，接收父组件传递的属性：

```
// components/broadItem/broadItem.js
Component({
    /**
     * 组件的属性列表
     */
    properties: {
        item: {
            type: Object,
            value: {}
        }
    },
```

c) 查看显示结果：

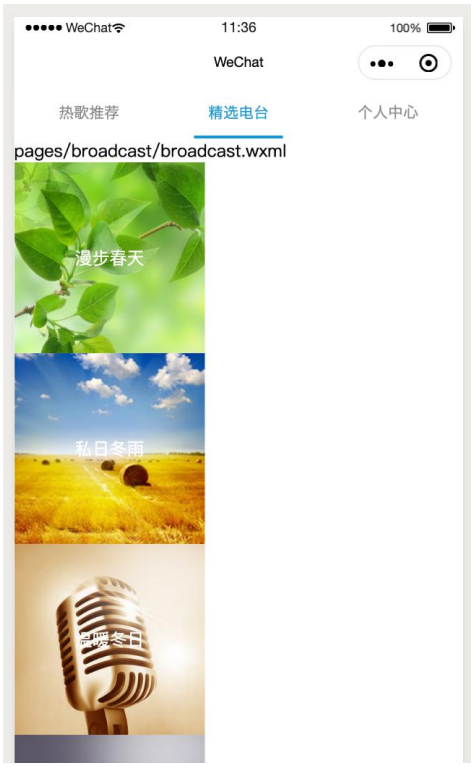
（6） 为 broadItem 组件配置样式：

a) 设置电台封面图大小和文字样式：

```
/* components/broadItem/broadItem.wxss */
.broad-item{
  width: 320rpx;
  height: 320rpx;
  display: flex;
  justify-content: center;
  align-content: center;
  background-size: 100% 100%;
  position: relative;
}

.broad-item text{
  color: white;
  display: flex;
  align-items: center;
  z-index: 1;
}
```

查看效果如下图所示：



b) 为了使得封面文字能更清楚显示，继续调整给图片增加阴影效果：

```
.broad-item::after{
  position: absolute;
  top: 0;
  left: 0;
  content: '';
  width: 100%;
  height: 100%;
  background-color: rgba(0, 0, 0, .5);
  z-index: 0;
}
```

修改之后效果如下：

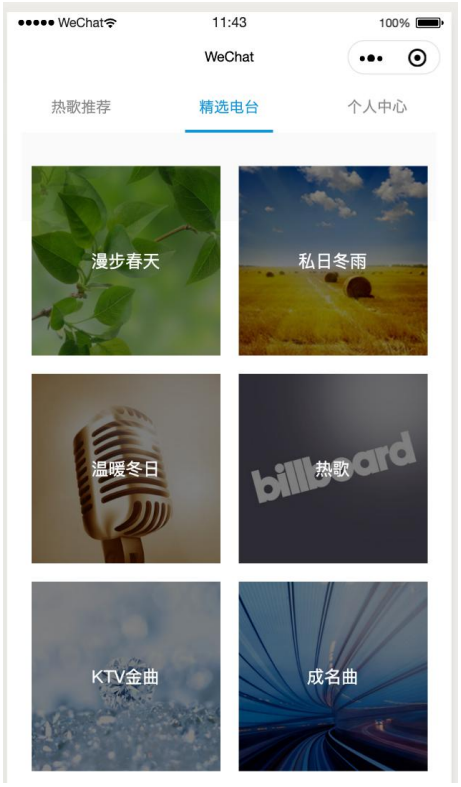
c) 调整外部容器，使得封面图片能够每行两个对齐显示

为父组件 broadcast 添加样式，在 broadcast.wxss 中添加样式：

```
/* pages/broadcast/broadcast.wxss */
.page{
  background-color: #fafafa;
  width: 700rpx;
  margin: 0 auto;
  padding-top: 40rpx;
  padding-bottom: 110rpx;
}

.content {
    width: 350rpx;
    height: 350rpx;
    float: left;
    display: flex;
    justify-content: center;
    align-items: center;
}
```

调整之后的截图如下：



作业上交内容与事项：

1．实验报告文档：填写（或截图）「实验记录和作业」后，上交原实验报告的 PDF 版本；

2．程序代码文件：上交「实验记录和作业」中所要求的源代码文件；

3．将 2 中文件打包为压缩包，并在提交作业时分别提交 1 和 2 的文件。

本期实验作业上期限：

请在实验设置的截止日期内提交实验报告结果，若逾期提交，成绩会适当被打折。

本次作业上交内容：

注：本次实验报告上交格式为 PDF 文件 ，具体的上交内容包括「实验记录和作业 」中所要求的：

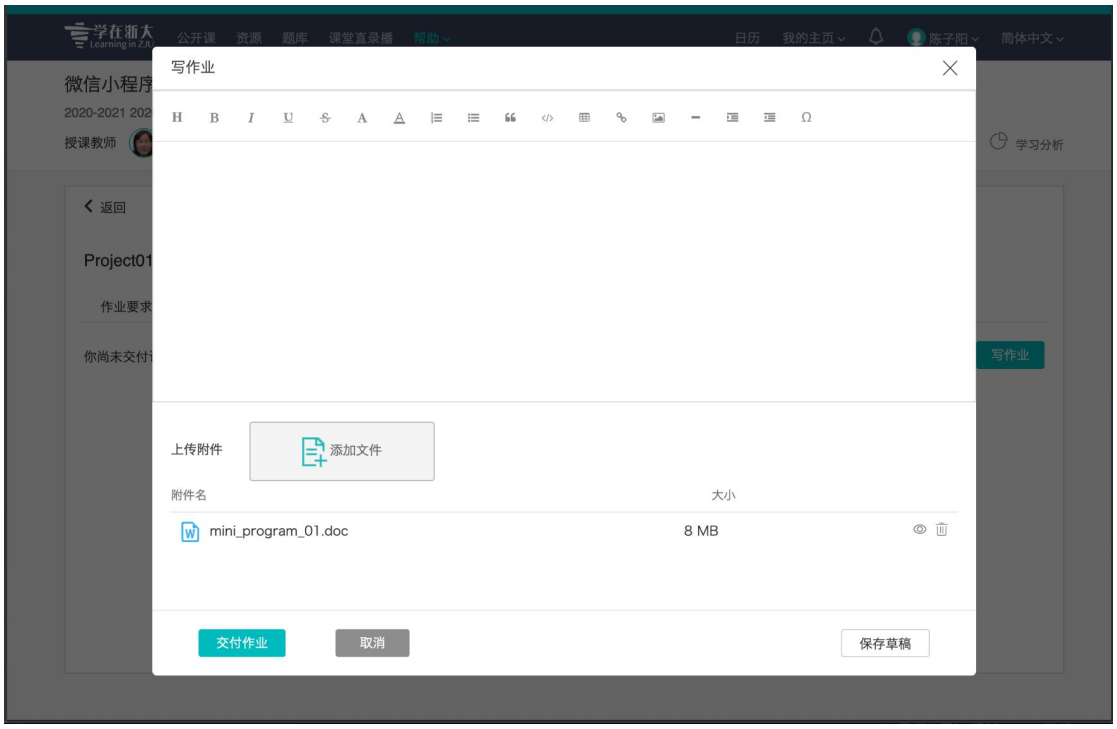
1. 问题解答
2. 要求的效果截图
3. 要求的源代码截图

按作业序号作答并写到指定位置，最终保存为 PDF 文件并上传

3．实验感受及记录

1.实验感受（本次实验遇到的疑难问题及你的主要收获，即写你掌握了什么，问题还有哪些）

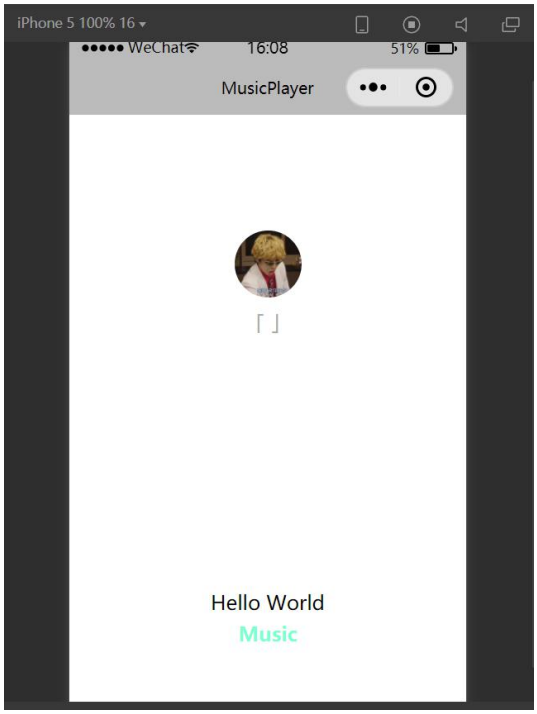
说明：请将此部分内容写到“写作业”下方的文本框中。



2．实验记录和作业（实验作业完成好的源程序及运行结果放在此处）

（1） 独立完成实验步骤的内容，将所有示例进行复现（不允许重复，要有适当改动）。

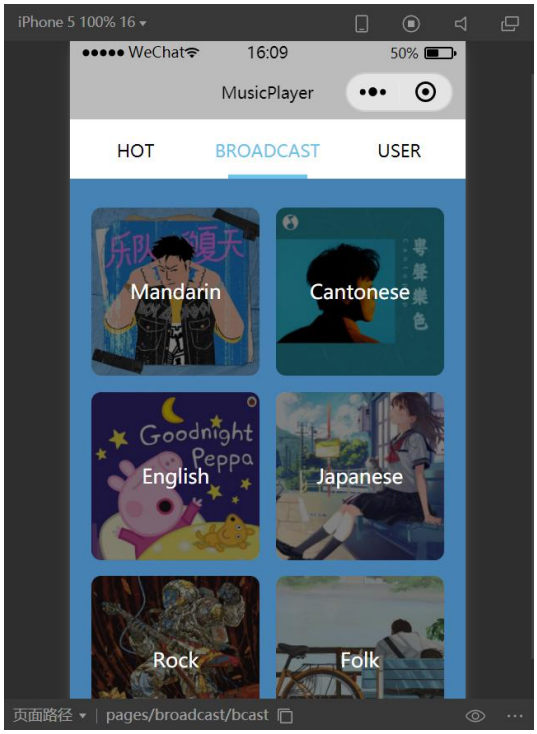
单击 Music 进入热播页面：



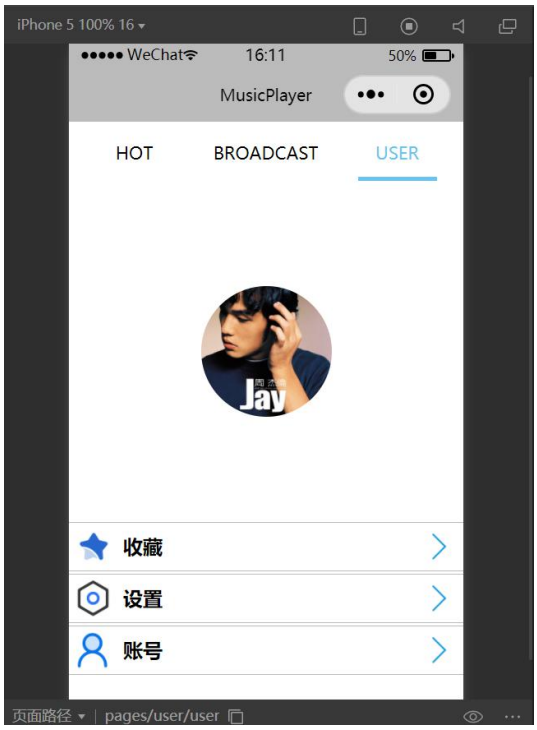
热播：


iPhone 5 100% 16
WeChat
16:09
51%
MusicPlayer
HOT
BROADCAST
USER
周杰伦
1 -- 黑色幽默
2 -- 世界末日
3 -- 困兽之斗
李荣浩
1 -- 太坦白
2 -- 不将就
3 -- 老街
许嵩
1 -- 有何不可

电台：

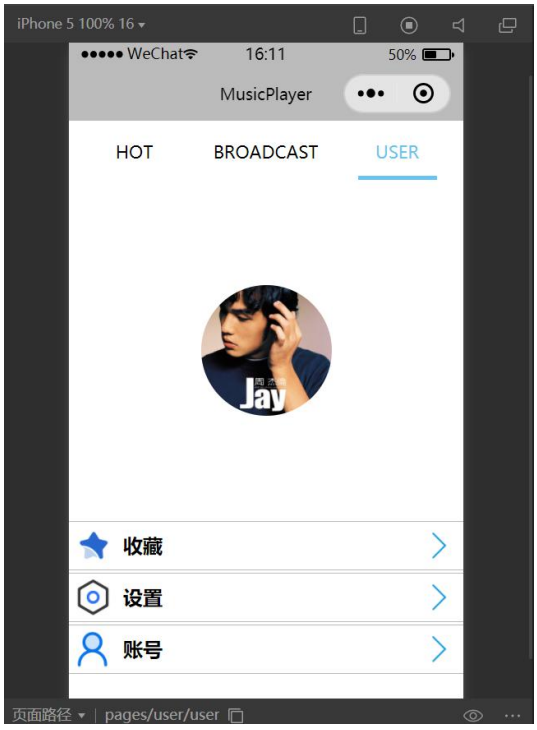

iPhone 5 100% 16
WeChat
16:09
50%
MusicPlayer
HOT
BROADCAST
USER
Mandarin
Cantonese
English
Japanese
Rock
Folk
页面路径
pages/broadcast/bcast

用户：



（2） 实验中三个页面中「个人中心」页面还没有完成，请发挥创造力在完成作业（1）的基础上，对「个人中心」页面进行编写。

用户页面：



（3） 将完成作业（2）后的源代码进行提交，并在下方附上三个页面的截图