闲话部落代码设计文档

Designed by ZhangPeng

一 开发语言

javascript （jxm） ，java

二 主程序设计

1. 程序入口

Index.android.js 入口文件

BackAndroid.addEventListener 定义了返回键的处理

<Navigator  
 style={styles.init}  
 initialRoute={{name: defaultName,component: defaultComponent}}  
 configureScene={() => Navigator.SceneConfigs.FadeAndroid}  
 renderScene={  
 (route,navigator)=>{  
 **let** Component = route.component;  
 \_navigator = navigator;  
 **return** <Component {...route.params} navigator={navigator}/>  
 }  
 }  
/>

定义了路由规则，初始化界面，参数的传递

1. 主界面

.app/main.js

主框架TabNavigator TabNavigator.Item 用来实现底部四个可点击的item来切换页面，类似android 的tabhost+fragment.

<TabNavigator >  
 <TabNavigator.Item  
 selected={**this**.state.selectedTab === 'gallery'}  
 title="小小画廊"  
 renderIcon={() => <Image style={styles.gallery\_bottom\_icon}  
 source={require('../res/image/tab\_weixin\_normal.png')}/>}  
 renderSelectedIcon={() => <Image style={styles.gallery\_bottom\_icon}  
 source={require('../res/image/tab\_weixin\_pressed.png')}/>}  
 onPress={**this**.setSelectedGallery.bind(**this**)}>  
 {GalleryView}  
 </TabNavigator.Item>  
  
 <TabNavigator.Item  
 selected={**this**.state.selectedTab === 'movement'}  
 title="活动"  
 renderIcon={() => <Image style={styles.gallery\_bottom\_icon}  
 source={require('../res/image/tab\_address\_normal.png')}/>}  
 renderSelectedIcon={() => <Image style={styles.gallery\_bottom\_icon}  
 source={require('../res/image/tab\_address\_pressed.png')}/>}  
 onPress={**this**.setSelectedMovementTab.bind(**this**)}>  
 {movementView}  
 </TabNavigator.Item>  
  
 <TabNavigator.Item  
 selected={**this**.state.selectedTab === 'program'}  
 title="节目"  
 renderIcon={() => <Image style={styles.gallery\_bottom\_icon}  
 source={require('../res/image/tab\_find\_frd\_normal.png')}/>}  
 renderSelectedIcon={() => <Image style={styles.gallery\_bottom\_icon}  
 source={require('../res/image/tab\_find\_frd\_pressed.png')}/>}  
 onPress={**this**.setSelectedProgram.bind(**this**)}>  
 {programView}  
 </TabNavigator.Item>  
  
 <TabNavigator.Item  
 selected={**this**.state.selectedTab === 'personal'}  
 title="个人专区"  
 renderIcon={() => <Image style={styles.gallery\_bottom\_icon}  
 source={require('../res/image/tab\_settings\_normal.png')}/>}  
 renderSelectedIcon={() => <Image style={styles.gallery\_bottom\_icon}  
 source={require('../res/image/tab\_settings\_pressed.png')}/>}  
 onPress={() => **this**.setState({selectedTab: 'personal'})}>  
 {personalView}  
 </TabNavigator.Item>  
</TabNavigator>

1. 小小画廊

.app/main.js 中 GalleryView

使用IndicatorViewPager 来实现viewpager左右滑动效果

函数\_renderTitleIndicator()定义了viewpager的头布局

子布局直接嵌套在IndicatorViewPager 内部，是两个Listview.

<IndicatorViewPager  
 style={{flex: 1}}  
 indicator={**this**.\_renderTitleIndicator()}>  
 <View>  
 <ListView  
 dataSource={**this**.state.dataSource}  
 renderRow={**this**.renderGallery.bind(**this**)}  
 refreshControl={**this**.getRefreshControl(CHOICE)  
 }/>  
 <View style={styles.floating}>  
 <TouchableOpacity onPress={**this**.\_addProduct.bind(**this**)}>  
 <Image source={require('../res/image/add.png')} style={styles.floatingImage}/>  
 </TouchableOpacity>  
 <Text style={styles.floatingText}>发布作品</Text>  
 </View>  
 </View>  
 <View>  
 <ListView  
 dataSource={**this**.state.dataSource}  
 renderRow={**this**.renderGallery.bind(**this**)}  
 refreshControl={  
 **this**.getRefreshControl(LATEST)}/>  
 </View>  
</IndicatorViewPager>

获得网络数据

setGalleryData()函数中fetch()函数是用来获得网络网络数据

1. 活动

.app/main.js 中movementView

主要使用Listview来布局

<ListView  
 dataSource={**this**.state.movementDataSource}  
 renderRow={**this**.renderMovieMent.bind(**this**)}  
 refreshControl={  
 **this**.getRefreshControl(MOVEMENT)}  
/>

1. 节目

.app/main.js 中programView

主要使用Listview来布局

<ListView  
 dataSource={**this**.state.programDataSource}  
 renderRow={**this**.renderProgram.bind(**this**)}  
 refreshControl={  
 **this**.getRefreshControl(PROGRAM)}/>

Listview的Item是视频播放器，使用类库react-native-media-kit来实现

<Video  
 style={styles.program\_video}  
 src={data}  
 autoplay={**false**}  
 preload={'none'}  
 loop={**false**}  
 controls={**true**}  
 muted={**false**}  
 poster={'https://www.baidu.com/img/bd\_logo1.png'}/>

1. 图片浏览节目

.app/product/ProductPriview.js

点击小小画廊中的图片，跳到图片浏览界面

使用react-native-image-zoom来实现手势缩放图片功能

**import** Image **from** 'react-native-image-zoom'

<View style={styles.container}>  
 <Image scaleType={'fitCenter'} style={styles.image} source={{uri: **this**.state.imagesource}}/>  
</View>

1. 作品添加

.app/product/AddProduct.js

主要功能从系统图库或相机获取图片，显示到当前界面

使用react-native-image-picker来实现，options是弹出对话框要显示的数据信息

//跳转到相机或者图库  
ImagePicker.showImagePicker(options, (response) => {  
 //类似onActivityResult  
 **if** (response.didCancel) {  
 }  
 **else if** (response.error) {  
 }  
 **else** {  
 //获得照相或图库返回的数据  
 // const source = {uri: 'data:image/jpeg;base64,' + response.data, isStatic: true};  
 **const** source = {uri: response.uri, isStatic: **true**};  
 **this**.setState({  
 avatarSource: source,  
 hasImageData: **true**,  
 DELETE\_HEIGHT : 30,  
 DELETE\_WIDTH : 30,  
 });  
 }  
});

**var** options = {  
 title: '选择作品',  
 takePhotoButtonTitle: '拍照',  
 chooseFromLibraryButtonTitle: '从相册中选取',  
 cancelButtonTitle: '取消',  
 storageOptions: {  
 skipBackup: **true**,  
 path: 'images'  
 }  
};

指定Textinput的输入类型，例如只输入数字，设置InputText的属性

keyboardType={'numeric'}

设置TextInput可以多行显示

multiline={**true**}

原生对话的显示

采用类库react-native-dialogs ，dialogOptions是对话框要显示的数据

**import** DialogAndroid **from** 'react-native-dialogs';

**var** dialog = **new** DialogAndroid();  
dialog.set(dialogOptions);  
dialog.show();

**var** dialogOptions = {  
 title: '作品信息未填写完整',  
 positiveText: '知道了',  
};

8.活动详情

./app/product/MovementDetials.js

在活动列表点击item跳转到活动详情界面

主要布局是webview

<WebView  
 ref={WEBVIEW\_REF}  
 automaticallyAdjustContentInsets={**false**}  
 style={styles.webView}  
 source={{uri: **this**.state.url}}  
 javaScriptEnabled={**true**}  
 domStorageEnabled={**true**}  
 decelerationRate="normal"  
 startInLoadingState={**true**}  
 scalesPageToFit={**this**.state.scalesPageToFit}/>

9.搜索界面

./app/search/SearchActive.js

主要是TextInput

<View style={styles.searchRoot}>  
 <TouchableOpacity style={styles.searchCancel} onPress={**this**.\_cancel.bind(**this**,1)}>  
 <Text style={styles.searchCancelText}>取消</Text>  
 </TouchableOpacity>  
 <View style={styles.search\_icon\_root}>  
 <Image style={styles.search\_icon} source={require('../../res/image/search\_icon.png')}/>  
 </View>  
 <View style={styles.searchInputRoot}>  
 <TextInput style={styles.searchInput} keyboardType={'web-search'}  
 autoFocus={**this**.state.focus}  
 placeholder={'编号/作品名/作者名/创作年龄/培训机构'}  
 placeholderTextColor={'#ADADAD'}  
 onChangeText={(searchText)=>**this**.setState({searchText})}  
 ref={(ref) => **this**.myTextInput = ref}/>  
 </View>  
 <View style={styles.delete\_icon\_root}>  
 <TouchableWithoutFeedback onPress={**this**.\_delete.bind(**this**)}>  
 <Image style={styles.delete\_icon} source={require('../../res/image/delete.png')}/>  
 </TouchableWithoutFeedback>  
 </View>  
 <TouchableOpacity style={styles.searchGo} onPress={**this**.\_go.bind(**this**,1)}>  
 <Text style={styles.searchGoText}>搜索</Text>  
 </TouchableOpacity>  
  
</View>

模拟点击TextInput右侧按钮清空输入框内容

**if** (**this**.myTextInput !== **null**) {  
 **this**.myTextInput.clear();  
}

首先通过ref获得TextInput的对象，然后调用对象的内部函数clea()来清空输入框。

10.搜索结果

./app/search/SearchResult.js

主要界面布局和小小画廊界面类似，是listview的布局。

11.分享功能

点击小小画廊或者搜索结果中的分享按钮

使用rn-umeng-share 来实现

**import** UmengShare **from** 'rn-umeng-share'

//友盟分享设置qq和微信key  
**static** *initShare*(){  
 UmengShare.*setQQZone*('1105543048','JFBbZIJ21sD8uhKP');  
 UmengShare.*setWXAppId*('wx967daebe835fbeac','5bb696d9ccd75a38c8a0bfe0675559b3');  
}

//分享功能  
\_share(action) {  
 UmengShare.*openShareAction*('欢迎使用','分享',"http://www.baidu.com",{uri:"https://www.baidu.com/img/bd\_logo1.png"});  
}