第2章 小程序框架分析

第3节 注册页面的使用

谢涛

内容提要

- 页面初始化数据
- 生命周期函数
- 页面相关事件处理函数
- 页面路由管理
- 自定义函数
- setData设值函数

准备

- 每个页面文件夹里，都有一个页面对应的js文件
- 这个文件里的Page()函数用来注册页面
- 接受一个object参数，其指定页面的初始数据、生命周期函数、事件处理函数等页面的所有业务逻辑处理都放在这个文件里

object参数说明

属性	类型	描述
data	Object	页面的初始数据
onLoad	Function	监听页面加载
onReady	Function	监听页面初次渲染完成
onShow	Function	监听页面显示
onHide	Function	监听页面隐藏
onUnload	Function	监听页面卸载
onPullDownRefresh	Function	监听用户下拉动作
onReachBottom	Function	页面上拉触底事件的处理函数
onShareAppMessage	Function	用户单击右上角分享
onPageScroll	Function	页面滚动触发事件的处理函数
onTabItemTap	Function	当前是tab页时，点击tab时触发
其他	Any	开发者可以添加任意的函数或数据到object参数中，在页面的函数中用this可以访问

Page()函数代码

```
Page({
  data: {
    text: 'Hello World!'
  },
  onLoad: function (options) {

  },
  onReady: function (options) {

  },
  onShow:function (options) {

  },
 onShareAppMessage:function (options) {

  },
})
```

1、页面初始化数据

- data为页面初始化数据
- 初始化数据将作为页面的第一次渲染
- data会以json的形式由逻辑层传到渲染层
- 其数据必须是可以转成json的格式
  - 字符串、数字、布尔值、对象或数组
- 渲染界面可以通过WXML对数据进行绑定

页面初始化数据代码示例

```
//此段代码在.js文件中
Page({
  data: {
    motto: ‘我的第一个微信小程序',
    userInfo: {},
  } })

//下面代码在.wxml文件中
<text class="user-motto">{{motto}}</text>
```

2、生命周期函数

- onLoad页面加载：一个页面只调用一次，接收页面参数可以获取wx.navigateTo和wx.redirectTo及<navigator>中的query
- onShow页面显示：每次打开页面都会调用一次
- onReady页面初次渲染完成:一个页面只调用一次，可和视图层进行交互，对界面的设置如wx.setNavigationBarTitle请在onReady之后设置
- onHide页面隐藏：当调用navigateTo或底部tab切换时调用
- onUnload页面卸载：当调用redirectTo或navigateBack的时候调用

3、页面相关事件处理函数

- onPullDownRefresh下拉刷新：监听用户下拉刷新事件，需要在config的windows选项中开启enablePullDownRefresh。当处理完数据刷新后，wx.stopPullDownRefresh可以停止当前页面的下拉刷新
- onShareAppMessage用户分享：只有定义了此事件处理函数，右上角菜单才会显示“分享”按钮，用户点击“分享”按钮的时候会调用此函数。此事件需要返回一个Object参数，用于自定义分享内容。

分享参数及代码

```
Page({
  onShareAppMessage:function() {
      return {
          title: ‘自定义分享标题’
          desc: ‘自定义分享描述’
          path: ‘/page/user?id=123’
                //必须是以/开头的完整路径
          },
    }
})
```

4、页面路由管理

- 微信小程序的页面路由都是由微信小程序框架来管理的
- 框架以栈的形式维护了所有页面
- 栈：一种数据结构
  - ✓ 只能在一端进行插入和删除操作
  - ✓ 按“后进先出”的原则存储数据
- 小程序页面交互也是通过栈来完成
  - ✓ 小程序初始化时，新页面入栈
  - ✓ 打开新页面时，新页面入栈
  - ✓ 页面重定向时，当前页面出栈
  - ✓ 页面返回时，页面不断出栈，直到新页面入栈
  - ✓ tab切换时，页面全部出栈，只留下新的tab页面
  - ✓ 重新加载时，页面全部出栈，只留下新的页面

路由触发方式及页面生命周期函数

页面路由方式	触发时机	路由后页面	路由前页面
初始化	小程序打开的第一个页面	onLoad，onShow	
打开新页面	调用API wx.navigateTo或使用组件<navigator open-type=“navigate”/>	onLoad, onShow	onHide
页面重定向	调用API wx.redirectTo或使用组件<navigator open-type=“redirect”/>	onLoad, onShow	onUnload
页面返回	调用API wx.navigateBack或用户按左上角返回按钮	onShow	onUnload(多层页面返回，每个页面都会按顺序触发onUnload)
tab切换	Function	调用API wx.switchTab或使用组件<navigator open-type=“switchTab”/>或用户切换tab	

页面路由管理注意事项

① navigateTo，redirectTo只能打开非tabBar页面

② switchTab只能打开tabBar页面

③ reLaunch可以打开任意页面

④ 页面底部的tabBar由页面决定，即只要是定义为tabBar的页面，底部都有tabBar

⑤ 调用页面路由带的参数可以在目标页面的onLoad中获取

5、自定义函数

- 除了初始化数据和生命周期函数，Page中还可以定义一些特殊的函数：事件处理函数
- 在渲染层的组件中可以加入事件绑定
- 当达到触发事件时，就会执行Page中定义的事件处理函数

自定义函数示例代码

```
//wxml文件
<view bindtap=“clickMe”>点我</view>

//js文件
Page({
  clickMe:function() {
      console.log(‘view tap’)
   }
})
```

6、setData设值函数

- Page.prototype.setData()设值函数用于将数据从逻辑层发送到视图层
- 同时改变对应的this.data的值
- setData()参数格式:
  - ✓ 接受一个对象，以key, value的形式表示将this.data中的key对应的值改变成value
- 其中key非常灵活，以数据路径的形式给出，如

  array[2].message，a.b.c.d
- 并且不需要在this.data中预先定义

setData设值函数示例代码

```
<!—wxml-->
<view>{{title}}</view>
<button bindtap=“changeText”>改变正常数据</button>
<view>{{array[0].text}}</view>
<button bindtap=“changeItemInArray”>改变数据数据</button>
<view>{{object.text}}</view>
<button bindtap=“changeItemInObject”>改变对象数据</button>
<view>{{newField.text}}</view>
<button bindtap=“addNewField”>增加新数据</button>
```

```
//js
Page({
  data: {
    text: ‘init data’,
    array:[{text:’init data’}],
    object:{ text:’init data’}  },
  changeText:function() {
   //this.data.text=‘changed data’//错误，不能工作
    this.setData({ text:’changed data’})  },
  changeItemInArray:function() {
    this.setData({ ‘array[0].text’:’changed data’})  } ,
  changeItemInObject:function() {
    this.setData({ ‘object.text’:’changed data’})  } ,
  addNewField:function() {
    this.setData({ ‘newField.text’:’new data’})  }
})
```

注意事项

- 直接修改this.data无效
- 无法改变页面的状态
- 还会造成数据不一致
- 单次设置的数据不能超过1024KB，尽量避免一次性设置过多的数据