爱创课堂前端培训

# 微信小程序

第3天课堂笔记（本课程共4天）

班级：北京前端训练营19期

讲师：李兰波

日期：2019年2月24日

爱创课堂官网 ：[www.icketang.com](http://www.icketang.com)

# 复习

指令：

wx:for 用于循环的指令

书写方式： wx:for="{{arr || obj}}";

书写位置： 组件上 与属性是一个写法

在该组件内部我们可以使用{{index}} 表示索引 使用{{item}} 表示成员

如果在某种情况下，发生了嵌套 ，则想要在内部使用外部的成员就不方便。

此时我们可以通过wx:for-index="xxx" 来指定索引名称 可以通过wx:for-item="xxx"来指定成员名称

wx:key 用于跟踪数据

微信小程序是需要追踪每一个信息的状态。我们直接书写{{title}}那么可以在编译阶段就将数据与视图绑定在一起。但是如果数据是被循环渲染到视图上。此时，在编译阶段就无法完成对照。我们就需要给定一个wx:key 来指定对照关系。

wx:if 用于控制组件的显示隐藏

页面：微信小程序中，页面分两类。

一类是在Pages数组中，但是不在tabBar的list中。这类页面叫做普通页面。

一类是在Pages数组中，又在tabBar的list中。这类页面叫做tabBar页面。

路由：在微信小程序中，路由指的是页面之间的跳转。

跳转类型：

navigateTo: 跳转到普通页面。 当前页面：hide 目标页面：load、show、ready

swtichTab：跳转到tabBar页面。

如果是第一次：当前页面hide 目标页面：load、show、ready

如果不是第一次：当前页面hide 目标页面show

redirectTo：跳转到普通页面。 当前页面：unload 目标页面：load、show、ready

navigateBack: 后退。 当前页面：unload 上一个页面：show

跳转方式：

组件跳转： 通过navigator组件

url属性：指定目标页面的路径

open-type：指定跳转类型

代码跳转：

wx.navigateTo()

wx.switchTab();

wx.redirectTo();

wx.navigateBack();

wx.getSystemInfoSync(); 获取系统信息API。

模板：事先定义好的一些组件结构。

模板有两种：一种是通过template定义的模板

定义： <template name="xxx">模板内容</template>

使用： <template is="xxx"></template>

另外一种模板就是wxml文件自身的内容。

引入模板有两个组件：

import: 引入template定义的模板

include: 引入wxml文件的内容。

三种跨页面传递数据的方式：

第一种： 通过全局中转

在页面.js中，我们可以通过getApp() 获取到App.js中的那个对象。 我们可以在A页面中获取并设置内容。我们可以在B页面中获取并得到内容。

第二种： 通过挂载URL的query

在页面跳转的时候，可以在URL的后面通过？挂载query字符串

在新页面中，onLoad事件中，我们可以得到格式化好的query对象。

因为该方式依赖onLoad事件，所以switchTab跳转类型不支持。

第三种： 通过本地存储对象

在微信小程序中，也有本地存储对象。10M。

以下方法都是异步方法

存储信息

wx.setStorage({

key: 存储的键值对的键名称

data: 存储的值

success: 存储成功之后的回调函数

})

获取信息

wx.getStorage({

key: 要获取的值所对应的键名称

success: function(res) {

res.data是返回回来的值

}

})

删除信息

wx.removeStorage({

key: 要删除的值所对应的键名称

success: 删除成功之后的回调函数

})

清空信息

wx.clearStorage({

success: 清空之后的回调函数

})

wx.onCompassChange: 当手机转向之后触发的事件 它可以监听到手机跟南北方向磁场的角度。

# 表单组件

input组件 该组件用于输入内容

type: 决定弹出的键盘种类

text 文本输入键盘

number 数字输入键盘

idcard 身份证输入键盘

digit 带小数点的输入键盘

password: 决定是否是密码输入框

confirm-type: 决定键盘的回车按钮是什么文字

next：下一项

go：前往

down：完成

search：搜索

demo:

|  |
| --- |
| 1. <form> 2. <input type='text' confirm-type='go'></input> 3. </form> |

checkbox组件 复选框组件

注：该组件必须放在 checkbox-group内

demo:

|  |
| --- |
| 1. <checkbox-group> 2. <checkbox value=''>你好</checkbox> 3. <checkbox value=''>你好</checkbox> 4. <checkbox value=''>你好</checkbox> 5. <checkbox value=''>你好</checkbox> 6. <checkbox value=''>你好</checkbox> 7. <checkbox value=''>你好</checkbox> 8. <checkbox value=''>你好</checkbox> 9. </checkbox-group> |

radio组件 单选框组件

注：在同一个radio-group内的radio组件视为一组

demo:

|  |
| --- |
| 1. <radio-group> 2. <radio>男</radio> 3. <radio>女</radio> 4. </radio-group> |

textarea组件 多行文本框

demo:

|  |
| --- |
| 1. <textarea> 2. </textarea> |

输入内容：

|  |
| --- |
|  |

button组件 按钮组件

type: 当前组件的样式

primary：绿色

warn：红色

default：白色 默认值

form-type: 当作为form的子组件时决定提交类型

sumbit：触发form的submit事件

reset: 触发form的reset事件 清空表单中的内容

demo:

|  |
| --- |
| 1. <view> 2. <button type='warn' form-type='submit'>我是一个按钮</button> 3. </view> |

样式：

|  |
| --- |
|  |

switch 组件 开关组件

checked 选中 打开该组件

disabled 禁用该组件 该组件的状态不可变

demo:

|  |
| --- |
| 1. <switch bindchange='change' disabled> 2. </switch> |

form 表单组件

bindsubmit: 用于绑定提交事件函数

在事件函数中，获取值的方式： e.detail.value.xxx

e.detail.value是一个对象，该对象的每一个属性名都是表单中的name属性值 所以想要获取数据，就必须给form的子组件name属性

demo：

|  |
| --- |
|  |

# web-view

web-view: 组件 用于将HTML5页面嵌入到微信小程序中

属性： src 指向的值是一个H5页面。

注：必须在微信小程序该账户的后台配置中配置。

demo: 如下，是没有配置过 会报提示信息。

|  |
| --- |
|  |

如果想要体验，可以在开发者工具的右上角的详情按钮的下拉菜单中，勾选：

|  |
| --- |
|  |

此时就可以体验了：

|  |
| --- |
|  |

# 开放组件

微信小程序中，获取用户信息的方式：

在较早的时候，使用wx.getUserInfo这个API来获取用户信息。

现在，该API已经不会再弹出提示用户的弹窗。

demo:

|  |
| --- |
|  |

我们可以使用开放组件：open-data

属性：type

属性有效值：



demo:

|  |
| --- |
| 1. <open-data type="userNickName"></open-data> 2. <open-data type="userAvatarUrl"></open-data> |

输出：

|  |
| --- |
|  |

# WXS

该组件用于在wxml文件中定义功能函数，并可以在该组件所在的页面中使用。

demo: 定义一个wxs 它是一个JS模块。通过module属性定义模块名称 该模块的名称就是暴露出来的内容

|  |
| --- |
| 1. <wxs module="ickt"> 2. function toUpperCase(str) { 3. return str.toUpperCase(); 4. } 5. module.exports.toUpperCase = toUpperCase; 6. </wxs> |

使用该模块：

|  |
| --- |
| 1. <text>{{ickt.toUpperCase(str)}}</text> |

结果：

|  |
| --- |
|  |

# 自定义组件

微信小程序提供了许多的组件，并且能够允许开发者自定义组件。

该自定义组件包含四个文件：

组件.wxml

组件.wxss

组件.js

组件.json

同样的，它们四个也被放在一个文件夹中。

区分一个文件夹中的四个文件组成的内容到底是一个页面还是一个组件：

## 1、区分JSON文件内容

组件的JSON

|  |
| --- |
| 1. { 2. "component": true, 3. "usingComponents": {} 4. } |

页面的JSON

|  |
| --- |
| 1. { 2. "usingComponents": {} 3. } |

组件的JS

|  |
| --- |
| 1. // components/model.js 2. Component({ 3. /\*\* 4. \* 组件的属性列表 5. \*/ 6. properties: { 7. }, 8. /\*\* 9. \* 组件的初始数据 10. \*/ 11. data: { 12. }, 13. /\*\* 14. \* 组件的方法列表 15. \*/ 16. methods: { 17. } 18. }) |

页面的JS

|  |
| --- |
| 1. // pages/index/index.js 2. Page({ 3. /\*\* 4. \* 页面的初始数据 5. \*/ 6. data: { 7. }, 8. /\*\* 9. \* 生命周期函数--监听页面加载 10. \*/ 11. onLoad: function (options) { 12. }, 13. /\*\* 14. \* 生命周期函数--监听页面初次渲染完成 15. \*/ 16. onReady: function () { 17. }, 18. /\*\* 19. \* 生命周期函数--监听页面显示 20. \*/ 21. onShow: function () { 22. }, 23. /\*\* 24. \* 生命周期函数--监听页面隐藏 25. \*/ 26. onHide: function () { 27. }, 28. /\*\* 29. \* 生命周期函数--监听页面卸载 30. \*/ 31. onUnload: function () { 32. }, 33. /\*\* 34. \* 页面相关事件处理函数--监听用户下拉动作 35. \*/ 36. onPullDownRefresh: function () { 37. }, 38. /\*\* 39. \* 页面上拉触底事件的处理函数 40. \*/ 41. onReachBottom: function () { 42. }, 43. /\*\* 44. \* 用户点击右上角分享 45. \*/ 46. onShareAppMessage: function () { 47. } 48. }) |

## 如何在页面中使用组件

组件定义完毕之后，需要使用。

方式：

在页面的JSON中定义该组件的引用声明

|  |
| --- |
| 1. { 2. "usingComponents": { 3. // key表示当前组件在当前页面中的组件名称 4. // value 表示当前组件的实际所在位置 5. "model": "/components/model/model" 6. } 7. } |

在页面中使用： 页面.wxml

|  |
| --- |
| 1. <model></model> |

注：如果usingComponents中的key改为module 则使用时如下

|  |
| --- |
| 1. <module></module> |

## 定义自定义组件的属性

自定义组件的属性，定义在Component()所接收的对象中的properties属性中。

该值是一个对象，对象的key表示自定义属性的名称

对象的value是对该属性的描述

value: 默认值

type: 类型

observer: 当属性值发生变化之后触发的函数

注：每一个属性一旦定义之后，就会自动注册到data属性中。

demo:

|  |
| --- |
| 1. properties: { 2. // 这是一个对象 该对象的key 就是自定义组件的属性名 3. hello: { 4. // 值 5. value: true, 6. // 值的类型 7. type: "boolean", 8. // 当该值发生变化时 执行的函数 9. observer: function() { 10. console.log("该值发生变化了"); 11. } 12. } 13. }, |

## 自定义组件的数据

自定义组件可以拥有数据源。也是data对象。存在于该对象中的数据可以通过插值语法将值插入到自定义组件的wxml中。

demo:

|  |
| --- |
| 1. data: { 2. username: "wanglaowuchongya" 3. }, |

插值语法代码：

|  |
| --- |
| 1. <view hidden="{{hello}}">你好，我是一个自定义组件 {{username}}</view> |

最终结果：

|  |
| --- |
|  |

## 自定义组件的方法

与页面不同，页面的所有方法都定义在Page()所接收的对象上

但是，自定义组件的属性定义在Component()所接收的对象的methods对象中。

demo:

|  |
| --- |
| 1. /\*\* 2. \* 组件的方法列表 3. \*/ 4. methods: { 5. click: function() { 6. console.log("宝刀屠龙，点击就送"); 7. } 8. } |

绑定该事件函数:

|  |
| --- |
| 1. <view bindtap="click" hidden="{{hello}}">你好，我是一个自定义组件 {{username}}</view> |

点击：

|  |
| --- |
|  |

## 从自定义组件内部向外传递数据

第一步：在自定义组件内的任何一个函数中调用triggerEvent方法

|  |
| --- |
| 1. methods: { 2. click: function() { 3. console.log("宝刀屠龙，点击就送"); 4. this.triggerEvent("tulong", 213123, 123); 5. } 6. } |

第二步：在外部绑定事件

triggerEvent的第一个参数是被触发的事件名称 第二个参数是传递的值

在外部的自定义组件身上绑定bindtulong事件

|  |
| --- |
| 1. <module bindtulong="dragon" hello="{{a}}"></module> |

第三步：在页面的JS文件中定义bindtulong所指向的事件函数

|  |
| --- |
| 1. dragon: function(e) { 2. console.log(e.detail) 3. } |

我们可以在该函数内通过e.detail获取传递过来的数据

|  |
| --- |
|  |

从外部向内部传递数据，就是修改自定义组件的属性即可。

## 单插槽

自定义组件允许我们预留位置，在使用的时候确定下来。

预留方式：单插槽 在自定义组件中使用<slot></slot>来定义位置

|  |
| --- |
| 1. <form> 2. <view>哈哈哈哈哈哈</view> 3. <view class="title"> 4. <slot></slot> 5. </view> 6. <view> 7. <label>用户名</label><input type="text" placeholder='请输入用户名'></input> 8. </view> 9. <view> 10. <button>go</button> 11. </view> 12. </form> |

使用时：

|  |
| --- |
| 1. <module bindtulong="dragon" hello="{{a}}"> 2. <view>登录</view> 3. </module> |

## 多插槽

使用方式： 也是使用slot 但是因为要区分多个slot 要给slot添加name属性

定义自定义组件时： 此时要给slot组件添加name属性

|  |
| --- |
| 1. <view> 2. <slot name="A"></slot> 3. <view>--------------------</view> 4. <slot name="B"></slot> 5. </view> |

还要在js文件中，配置options

|  |
| --- |
| 1. Component({ 2. options: { 3. multipleSlots: true 4. } 5. }) |

使用时：给子组件添加slot属性 值就是A或者B

|  |
| --- |
| 1. <module bindtulong="dragon" hello="{{a}}"> 2. <view slot="B">登录</view> 3. <view slot="A">denglu1</view> 4. </module> |

最终结果:

|  |
| --- |
|  |

# API

## 6.1 打开位置

wx.openLocation: 指定位置 打开地图

参数是对象 对象有两个属性

longitude: 经度

latitude: 纬度

## 6.2 获取位置

wx.getLocation: 获取当前手机的经纬度

参数是对象 对象有success属性 表示成功时的回调函数

demo:

|  |
| --- |
| 1. wx.getLocation({ 2. complete: function() { 3. console.log(123) 4. }, 5. fail: function(res) { 6. console.log(res); 7. }, 8. success: function(res) { 9. console.log(res); 10. } 11. }) |

console:

|  |
| --- |
|  |

## 6.3 选取位置

wx.chooseLocation: 该方法用于选择位置 并获取经纬度以及其它信息

参数是对象

对象有success方法

注：该方法只有当用户选择并确定之后才会success。

demo:

|  |
| --- |
| 1. wx.chooseLocation({ 2. success: function(res) { 3. console.log(res); 4. }, 5. }) |

console:

|  |
| --- |
|  |

## 6.4 上传文件

wx.uploadFile() 该方法用于上传文件

参数是对象 对象有success属性 url属性 filePath属性

demo:

|  |
| --- |
| 1. wx.uploadFile({ 2. url: '', // 后台接口 3. filePath: '', // 要上传的文件的路径 4. name: '' // 上传的文件的名称 等价于input[type=file] 的name 5. }) |

## 6.5 选择图片

wx.chooseImage()方法用于选择图片 当然也可以选择文件

参数是对象 对象有success属性

demo:

|  |
| --- |
| 1. wx.chooseImage({ 2. success: function(res) { 3. console.log(res); 4. }, 5. }) |

console:

|  |
| --- |
|  |

## 6.6 微信行走

wx.getWeRunData 注该API必须要求用户关注微信运动公众号才能够有数据。

## 6.7 微信支付

wx.requestPayment 该API用于微信付款

## 6.8 选择地址

wx.chooseAddress 用于获取用户设置的地址

## 6.9 发送请求

wx.request 注该请求方式只能够是HTTPS

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |