# 使用UI

## 1、colorUI

/app.wxss添加：

@import "colorui/main.wxss";

@import "colorui/icon.wxss";

@import "colorui/animation.wxss";

从colorUI目录下复制colorui目录

# 二、小程序组件

## 1、组件创建

---------------

1. 组件必须文件index.js、index.wxss、index.json、index.wxml
2. 在 /组件/index.json中声明，这里是组件

{"component": true,}

1. 在 /组件/index.js中添加：

Component({

properties: {

// 这里定义了innerText属性，属性值可以在组件使用时指定

innerText: {

type: String,

value: "default value",

}

},

data: {

// 这里是一些组件内部数据

someData: {}

},

methods: {

// 这里是一个自定义方法

customMethod () {}

}

})

**# 不是在page({})方法中添加**

1. 在 /组件/index.wxml中添加

<view>我是组件</view>

## 2、组件使用

------------------------

1、在/page/index/index.json中添加

"usingComponents": {

//组件位置

"navbottom": "/pages/Component/nav-bottom/index"

}

1. 在/page/index/index.wxml中添加

<navbottom inner-text="Some text"></navbottom>

## 3、父子组件

* **子**组件获取**父页面**中的值
* **子**组件触发**父页面**的方法

1. **childen.js**

//name 这里指明这个触发事件的名称

//param1 携带的data数据

//param2

that.triggerEvent("name", param1, param2)

1. **index.wxml**

这里在父页面调用子组件的标签上添加

// name 是childen中triggerEvent中绑定的事件名

//dataFromParent 子组件携带的参数

// inHidpage 是要触发的函数

<login bind:name='isHidpage'></login>

1. **index.js**

定义isHidpage方法

* **父页面**获取**子**组件中的值

## 4、组件生命周期

created (){...}组件实例化，但节点树还未导入，因此这时不能用setData

attached (){...}节点树完成，可以用setData渲染节点，但无法操作节点

ready (){...}组件布局完成，这时可以获取节点信息，也可以操作节点

moved (){...}组件实例被移动到树的另一个位置

detached (){...}组件实例从节点树中移除

# 三、app对象

1. 在app.js中定义要使用的变量
2. 在使用的时候

var app = getApp(); //必要条件

app.appData.user\_info = {"username"："jane"}; //赋值

user\_info = app.appData.user\_info; //取值

# 四、使用全局样式全局样式表

/page/组件/index.js

Component({

options: {

addGlobalClass: true //这里是设置项

},

data: {}

})

# 五、loyout布局

实现对页面的拆分，根据条件加载不同的页面

1. 制作组件，将不同的页面变成组件添加进inde.wxml页面中
2. 确定首页

/page/index/index.wxml

1. 组册组件，参考组件使用
2. 在 index.wxml 中添加

// 用于确定什么时候显示

<index wx:if="{{PageCur == "index"}}"></index>

//绑定函数

// bindtap绑定一个函数NavChange方法

// data-cur是携带的值，用于确定加载哪个组件

<view bindtap="NavChange" data-cur="index">...</view>

1. index.js

Page({

data: {

//用于确定加载哪个插件

PageCur: 'index'

},

//事件处理函数

NavChange(e) {

//设置

this.setData({

PageCur: e.currentTarget.dataset.cur

})

})}

# 六、持久缓存

## 1、插入

wx.setStorage //异步

wx.setStorageSync //同步

## 2、读取

wx.getStorage //异步

wx.getStorageSync //同步

## 3、删除

wx.removeStorage //异步

wx.removeStorageSync //同步

## 4、清空

wx.clearStorage //异步

wx.clearStorageSync //同步

## 5、获取缓存信息

wx.getStorageInfo

wx.getStorageInfoSync

以Sync结尾都是同步方法。同步方法和异步方法的区别是：

* 同步方法会堵塞当前任务，直到同步方法处理返回。
* 异步方法不会塞当前任务。
* 一般选用异步方法

## 6、保存信息 ex

//保存信息

wx.setStorage({

key: 'key1',

data: 'data1',

success: function(res){

console.log('异步保存成功')

}})

-----------------------------------------------------------------------------------

//保存信息

wx.setStorageSync('key2', 'data2')

# 七、微信功能标签

bindtap 绑定点击事件

----------------------

bindtap=”NavChange” //绑定NavChange方法

data-cur=”xx” //此参数传递给bindtap绑定的方法，NavChange()通过参数接收

-----------------------

open-type="getUserInfo" //新版微信接口修改，与bindgetuserinfo配合使用

bindgetuserinfo="onGetUserInfo" //调用onGetUserInfo函数

# 八、微信函数简记

wx.getSetting() //获取设置设置项，包含认证状态

wx.getUserInfo() //用户信息，头像，名称，

wx.login() //这个主要是获取登录的code用于换取信息

# 九、功能简记

## 1、登录流程code

1. 是否注册验证
2. wx.login获取**临时登录凭证code** ，**临时登录凭证code**
3. **使用临时登录凭证code换取openID和session\_key**
4. **将什么返回前端**
5. **验证openID是否存在于数据库中（验证是否注册）**
6. 返回前端信息
7. 引导用户注册
8. 通过<button>绑定事件

<button open-type="getUserInfo" bindgetuserinfo="onGetUserInfo">

1. 方法 onGetUserInfo( e ) {....} 的 e 中有用户基本信息。
2. wx.request发送信息到服务器，注册

## 2、请求验证

设置请求头

getRequestHeader () {

return {

‘content-type’:’application/x-www-form-urlencoded’,

‘Authorization’: this.getCache(“token”) //服务器返回的token

}

}

## 3、获取页面ID

onload：(e)=>{

this.setData({

id=e.id

}

## 4、跳转



**tabBar页需要在app.json中注册**

