微信小程序

小程序页面代码构成
* js ---- 脚本逻辑文件
* json ---- 项目配置文件
* wxml ---- 模版文件，描述页面结构类似HTML
* wxss ---- 样式文件，秒速页面样式类似css

> 注意：为了方便开发者减少配置项，描述页面的四个文件必须具有相同的路径与文件名。

JSON配置

app.json
app.json 是对当前小程序的**全局配置**，包括了小程序的所有页面路径、界面表现、网络超时时间、底部 tab 等。

```
{
  "pages": [
    "pages/index/index",
    "pages/logs/index"
  ],
  "window": {
    "navigationBarTitleText": "Demo"
  },
  "tabBar": {
    "list": [{
      "pagePath": "pages/index/index",
      "text": "首页"
    }, {
      "pagePath": "pages/logs/logs",
      "text": "日志"
    }]
  },
  "networkTimeout": {
    "request": 10000,
    "downloadFile": 10000
  },
  "debug": true
}
```

属性	类型	必填	描述	
pages	StringArray	是	设置页面路径	对应
window	object	否	设置默认页面的窗口表现	
tabBar	Object	否	设置底部 tab 的表现	
networkTimeout	Object	否	设置网络超时时间	类型：
debug	Boolean	否	设置是否开启 debug 模式	

window

用于设置小程序的状态栏、导航条、标题、窗口背景色。

属性	类型	默认值	
navigationBarBackgroundColor	HexColor	#000000	导航栏背景颜色
navigationBarTextStyle	String	white	标题颜色，仅支
navigationBarTitleText	String		导航栏标题文字
navigationStyle	String	default	样式仅支持defau
backgroundColor	HexColor	#ffffff	窗口的背景色
backgroundTextStyle	String	dark	拉背景字体、loa
enablePullDownRefresh	Boolean	false	是否开启下拉刷
onReachBottomDistance	Number	50	页面上拉触底事

> **HexColor** 十六进制颜色值
> **navigationStyle** 只在 `app.json` 中生效。开启 `custom` 后，低版本客户端需要做好兼容。

tabBar

> 设置 `position` 为 `top` 时，不会显示**icon**。
> `tabBar` 中的 `list` 是一个数组，能配置最少2个、最多5个tab，且按顺序排

> 序。

属性	类型	必填	默认值	描述
color	HexColor	是		tab 上的文字默认颜色
selectedColor	HexColor	是		tab 上的文字选中时的颜
backgroundColor	HexColor	是		tab 的背景色
borderStyle	String	否	black	tabbar上边框的颜色，仅
list	Array	是		tab 的列表
position	String	否	bottom	可选值 bottom/top

JS逻辑层

通过编写 `JS` 脚本文件来处理交互逻辑。

app.js

```
App({
  onLaunch: function(options) {
    // Do something initial when launch.
  },
  onShow: function(options) {
      // Do something when show.
  },
  onHide: function() {
      // Do something when hide.
  },
  onError: function(msg) {
    console.log(msg)
  },
  globalData: 'I am global data'
})
```

生命周期

属性	类型	描述	
onLaunch	Function	生命周期函数--监听小程序初始化	当初始化完成时

onShow	Function	生命周期函数--监听小程序显示	当启动，或从后
onHide	Function	生命周期函数--监听小程序隐藏	从前台进入后台
onError	Function	错误监听函数	当发生脚本错误 调用失败时触发

getApp()

全局的`getApp()`函数可以用来获取到小程序**实例**。

```
// other.js
 onShow: function () {
    var appInstance = getApp();
    this.setData({
      text: appInstance.testData
    })
  },
```

- `App()`必须在`app.js`中注册且唯一。
- 不要在`App()`内调用`getApp()`，使用`this`就可以拿到app实例。
- 不要在`onLaunch()`的时候调用`getCurrentPages()`，此时page没有生成。
- 通过`getApp()`获取实例后，不要调用生命周期函数。

page

`Page()`指定页面的初始数据、生命周期函数、事件处理函数等。

属性	类型	描述
data	Object	页面的初始数据，data将会以`JSON`的形式
onLoad	Function	页面加载且唯一
onReady	Function	页面初次渲染完成且唯一
onShow	Function	页面显示
onHide	Function	页面隐藏，当navigateTo或底部tab切换时
onUnload	Function	页面卸载，当redirectTo或navigateBack的
onPullDownRefresh	Function	监听用户下拉动作

onReachBottom	Function	上拉触底事件onShareAppMessage
onPageScroll	Function	滚动触发事件
onTabItemTap	Function	当前是 `tab` 页时，点击 `tab` 时触发

路由方式

路由方式	api	组件	备
打开新页面	wx.navigateTo	`<navigator open-type="navigateTo"/>`	只跳非tabB
页面重定向	wx.redirectTo	`<navigator open-type="redirectTo"/>`	只跳非tabB
Tab 切换	wx.switchTab	`<navigator open-type="switchTab"/>`	只跳tabBar
重启动	wx.reLaunch	`<navigator open-type="reLaunch"/>`	可以打开任

> 调用页面路由带的参数可以在目标页面的`onLoad`中获取。

navigator.to传参数

```
<!--1 wxml 跳转-->
<navigator url="/pages/index/index">

<!--2 js 跳转-->
itemTap(event){
let item = event.currentTarget.dataset.item;

<!--简单参数-->
 wx.navigateTo({
        url: "/pages/index/index?id=" + item.id     })

<!--复杂参数-->
 wx.navigateTo({
        url: "/pages/index/index?item=" + JSON.stringify(item)
,
      })
    }
```

```
  }

<!--接收-->
 onLoad: function (options) {
    let option = JSON.parse(options.item);
  },
```

视图层

将逻辑层的数据反应成视图，同时将视图层的事件发送给逻辑层。

`wx:for` **列表渲染**

默认数组的当前项的下标变量名默认为index，数组当前项的变量名默认为item。

- `wx:for-item` 可以指定数组当前**元素**的变量名
- `wx:for-index` 可以指定数组当前**下标**的变量名

WXML页面结构

WXML是框架设计的一套标签语言，结合基础组件、事件系统，可以构建出页面的结构。

WXSS页面样式

`wxss` 是样式语言，**是CSS的拓展**，对比 `css` 新增了两个特性：

- 尺寸单位 `rpx`
- 样式导入

rpx

根据屏幕宽度的不同 `rpx` 代表的实际 `px` 也不同。 `WXSS` 规定一个屏幕的宽度为750rpx，是**iphone6**的物理宽度大小。

设备	rpx换算px (屏幕宽度/750)
Phone5	1rpx = 0.42px
iPhone6	1rpx = 0.5px
iPhone6s	1rpx = 0.552px

样式导入

当使用一些其他库的时候可以直接导入第三方的wxss文件

```
@import "common.wxss";
.middle-p {
  padding:15px;
}
```

> wxss支持`class`和`style`两种样式，如果样式中存在**动态内容**，将其写到`style`中，其他的都放到`class`文件。

css样式

种类	属性	介
文本样式	font-size	9pt
	color	#999999
	line-height(行高)	单位：PX、PD、EM
	text-align(水平对齐)	left/right/center/justify
	vertical-align(垂直对齐)	inherit/top/bottom/baseline/middle
框架样式	border-style	dotted/dashed...
	border-width	border-top-width/border-right-width/b
	border-color	「简写」border：width/style/color
	padding-left	「简写」padding：上右下左
	z-index	设置元素的堆叠顺序--高在低前面

> 顺序：上右下左

背景图片

布局

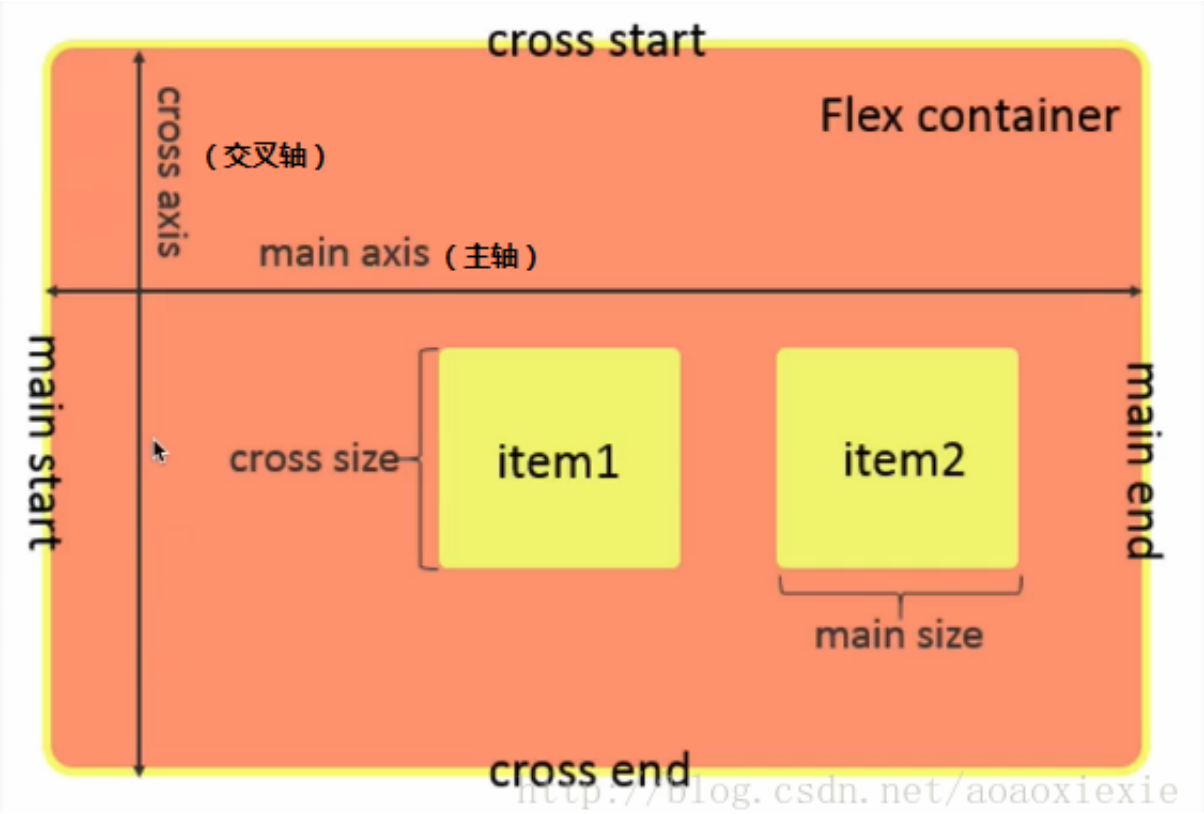
微信小程序页面布局方式采用的是`Flex`布局。

`display`

- `block` 块内容器模式--新行开始布局（view、scroll-view、swiper默认设置为block）

- `flex` 行内容器模式--一行内列出子元素。使用 `flex-wrap` (nowrap/wrap/wrap-reverse)指定换行。

主轴的方向可以使用 `flex-direction` (row、row-reverse、column、column-reverse)属性控制



子元素有两种对齐方式

- `justify-content` -- 主轴对齐(flex-start、flex-end、center、space-between、space-around)
- `align-items` -- 侧轴对齐(stretch、flex-start、flex-end、center、baseline)

自定义组件

Component构造器

定义段	类型	描述
properties	Object Map	组件的对外属性
data	Object	组件的内部数据
methods	Object	组件事件

created	Function	进入页面节点树时执行，此时不能调用 `setData`
attached	Function	进入页面节点树时执行
detached	Function	从页面节点树移除时执行

```
  properties: {
     property: {
 //目前类型：String, Number, Boolean, Object, Array, null（表示任意类型）
      type: String,
 //属性初始值（可选）
      value: '',
 //当属性被改变时执行的函数（可选）
    observer: function(newVal, oldVal){}
          },
         }
```

> 使用时在`page.json`注册
> `"usingComponents": {`
> `"component": "/components/component/component"`
> `}`

组件事件

Component组件基本事件

```
`Component.wxml`

<view bindtap='hideToast'/>
```

```
`Component.js`

methods: {
    hideToast: function () {
      setTimeout(() => {
        this.setData({
          hidden: true,
        })
      }, 1500)
    }
  }
```

Component提供事件监听函数

`Component.wxml`

```
<view bindmyevent="_onMyEvent" />
//or
<view bind:myevent="_onMyEvent" />
```

`Component.js`

```
Component({
  properties: {}
  methods: {
    _onMyEvent: function(e){
   //提供给事件监听函数
      this.triggerEvent('bindmyevent', {e:e.detail})
    }
  }
})
```

`page.wxml`

```
<component  property="XXX"  bind:bindmyevent="bindEvent"/>
```

`page.js`

```
  bindChange: function (e) {
    let val = e.detail.e.value
    ...
  }
```

Page调用Component的内部方法

`Component.wxml`

```
<view bindtap='hideToast'/>
```

`Component.js`

```
methods: {
    hideToast: function () {
      setTimeout(() => {
        this.setData({
          hidden: true,
        })
      }, 1500)
    }
  }
```

`page.wxml`

```
<component id="component"/>
```

`page.js`

```
 //生命周期函数--监听页面初次渲染完成
  onReady: function () {
    this.component = this.selectComponent("#component");
  },
 bindHide() {
    this.component.hideToast();
  },
```