
黑马程序员™
www.itheima.com
传智播客旗下
高端IT教育品牌
小程序的结构与配置

目录 Contents

1. 小程序的结构和组件

1.1 小程序项目的结构

```
├── pages ..... 【目录】存放所有的小程序页面
│   ├── index ..... 【目录】index 页面
│   │   ├── index.wxml ..... 【文件】 index 页面的结构
│   │   ├── index.js ..... 【文件】 index 页面的逻辑
│   │   ├── index.json ..... 【文件】 index 页面的配置
│   │   └── index.wxss ..... 【文件】 index 页面的样式
│   └── logs ..... 【目录】 logs 页面
│       ├── logs.wxml ..... 【文件】 logs 页面的结构
│       └── logs.js ..... 【文件】 logs 页面的逻辑
└── utils ..... 【目录】 存放小程序中用到的工具函数
├── app.js ..... 【文件】 小程序逻辑
├── app.json ..... 【文件】 小程序的公共配置
├── app.wxss ..... 【文件】小程序公共样式表
└── project.config.json ..... 【文件】 开发工具配置文件
```

注意：

- 对于小程序来说：

  app.js 和 app.json 文件是必须的

- 对于小程序的页面来说：

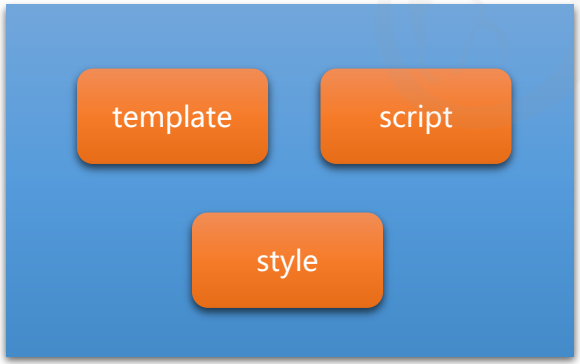
  .js 和 .wxml 文件是必须的

1. 小程序的结构和组件

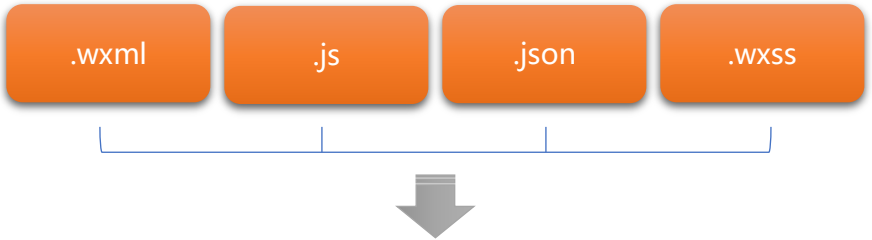
1.2 小程序页面的结构

1. 小程序页面和Vue组件的对比

- 每个.vue组件，是由 template 模板结构、script 行为逻辑、 style 样式3个部分组成的
- 每个小程序的页面，是由 .wxml 结构、.js 逻辑、.json 配置、.wxss 样式表 这4个文件组成的



.vue组件



四个文件组合成一个完整的小程序页面

1. 小程序的结构和组件

1.2 小程序页面的结构

2. 小程序页面中每个组成部分的作用

- **.wxml :** 用来描述当前这个页面的结构，同时提供了类似于Vue中指令的语法
- **.js :** 用来定义当前页面中用到的数据、交互逻辑和响应用户的操作
- **.json :** 用来定义当前页面的个性化配置，例如，为每个页面单独配置顶部颜色、是否允许下拉刷新等
- **.wxss :** 用来定义样式来美化当前的页面

1. 小程序的结构和组件

1.2 小程序页面的结构

3. 创建自己的小程序页面

① 在 pages 目录上右键，选择 “新建目录” ，并将目录命名为 home

② 在新建的 home 目录上右键，选择 “新建page”,并将页面命名为 home

注意：选择“新建page”后，开发者工具会自动创建页面相关的4个文件

1. 小程序的结构和组件

1.2 小程序页面的结构

4. 设置小程序项目的默认首页

1. 打开 app.json 全局配置文件，找到 pages 节点。这个 pages 节点是一个数组，存储了项目中所有页面的访问路径。其中，pages 数组中第一个页面路径，就是小程序项目的默认首页。
2. 修改 pages 数组中路径的顺序，即可修改小程序的默认首页。

app.json

```
{
  "pages": [
    "pages/index/index",
    "pages/logs/logs",
    "pages/home/home"
  ],
  "window": {
    "backgroundTextStyle": "light",
    "navigationBarBackgroundColor": "#fff",
    "navigationBarTitleText": "WeChat",
    "navigationBarTextStyle": "black"
  }
}
```

1.3 小程序常用的UI组件

小程序提供了丰富的基础组件给开发者，开发者可以像搭积木一样，组合各种组件拼合成自己的小程序。

小程序中的组件，就像HTML中的div，p标签的作用一样，用于搭建页面的基础结构。

1.3 小程序常用的UI组件

1. text文本

属性名	类型	默认值	说明
selectable	Boolean	false	文本是否可选，除了text组件之外，其它组件都无法长按选中
space	String	false	显示连续空格，可选值：ensp、emsp、nbsp
decode	Boolean	false	是否解码，可解析 < > & '

1.3 小程序常用的UI组件

2. view视图容器

属性名	类型	默认值	说明
hover-class	String	none	指定按下去的样式类。当 hover-class="none" 时，没有点击态效果
hover-stop-propagation	Boolean	false	指定是否阻止本节点的祖先节点出现点击态
hover-start-time	Number	50	按住后多久出现点击态，单位毫秒
hover-stay-time	Number	400	手指松开后点击态保留时间，单位毫秒

1.3 小程序常用的UI组件

3. button按钮

属性名	类型	默认值	说明
size	String	default	按钮的大小
type	String	default	按钮的样式类型
plain	Boolean	false	按钮是否镂空，背景色透明
disabled	Boolean	false	是否禁用
loading	Boolean	false	名称前是否带 loading 图标

1.3 小程序常用的UI组件

4. input输入框

属性名	类型	默认值	说明
value	String		输入框的初始内容
type	String	"text"	input 的类型
password	Boolean	false	是否是密码类型
placeholder	String		输入框为空时占位符
disabled	Boolean	false	是否禁用
maxlength	Number	140	最大输入长度，设置为 -1 时不限制最大长度

1.3 小程序常用的UI组件

5. image图片

- 常见的属性：
  - ① src ：支持本地和网络上的图片
  - ② mode ：指定图片裁剪、缩放的模式

注意：image组件默认宽度300px、高度225px

- 更多属性用法请翻阅 image 官方文档：

  https://developers.weixin.qq.com/miniprogram/dev/component/image.html

目录 Contents

2. 小程序中的样式

2.1 什么是WXSS

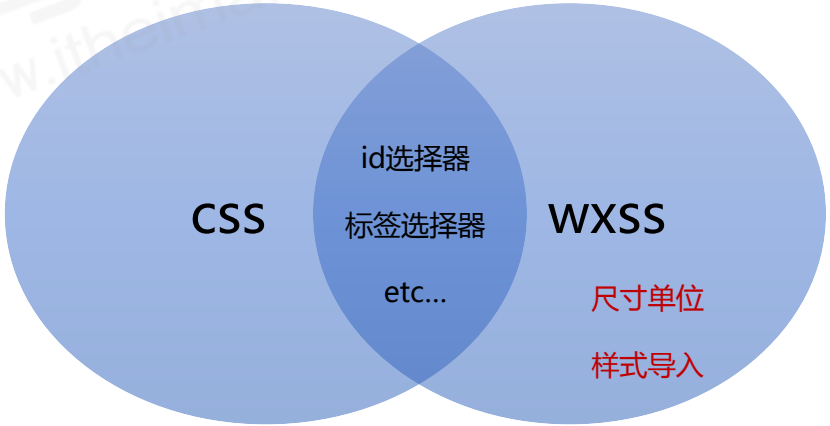
WXSS(WeiXin Style Sheets)是一套样式语言，用来决定 WXML 的组件应该怎么显示；

WXSS 具有 CSS 大部分特性；

WXSS 对 CSS 进行了扩充以及修改，以适应微信小程序的开发；

与 CSS 相比，WXSS 扩展的特性有：

- 尺寸单位
- 样式导入



WXSS与CSS的关系

2. 小程序中的样式

2.2 WXSS目前支持的选择器

- 标签选择器
- id选择器
- class选择器
- 伪类选择器
- data-*属性选择器
- :nth-of-type() 等常用的 css3 选择器
- etc...

2. 小程序中的样式

2.3 什么是rpx尺寸单位

rpx（responsive pixel）：是微信小程序独有的、解决屏幕自适应的尺寸单位。

- 可以根据屏幕宽度进行自适应。不论大小屏幕，**规定屏幕宽为750rpx**。
- 通过rpx设置元素和字体的大小，小程序在**不同尺寸**的屏幕上，**可以实现自动适配。**

2.4 rpx与px之间的换算

以 iPhone6 为例，iPhone6 的屏幕宽度为375px，共有750个物理像素，则750rpx = 375px = 750物理像素，1rpx = 0.5px = 1物理像素。

设备	rpx换算px (屏幕宽度/750)	px换算rpx (750/屏幕宽度)
iPhone5	1rpx = 0.42px	1px = 2.34rpx
iPhone6	**1rpx = 0.5px**	**1px = 2rpx**
iPhone6 Plus	1rpx = 0.552px	1px = 1.81rpx

在iPhone6上，如果要绘制宽100px，高20px的盒子，换算成rpx单位，宽高分别为 200rpx 和 40rpx。

2.5 rpx 和 iPhone6 设计稿的关系

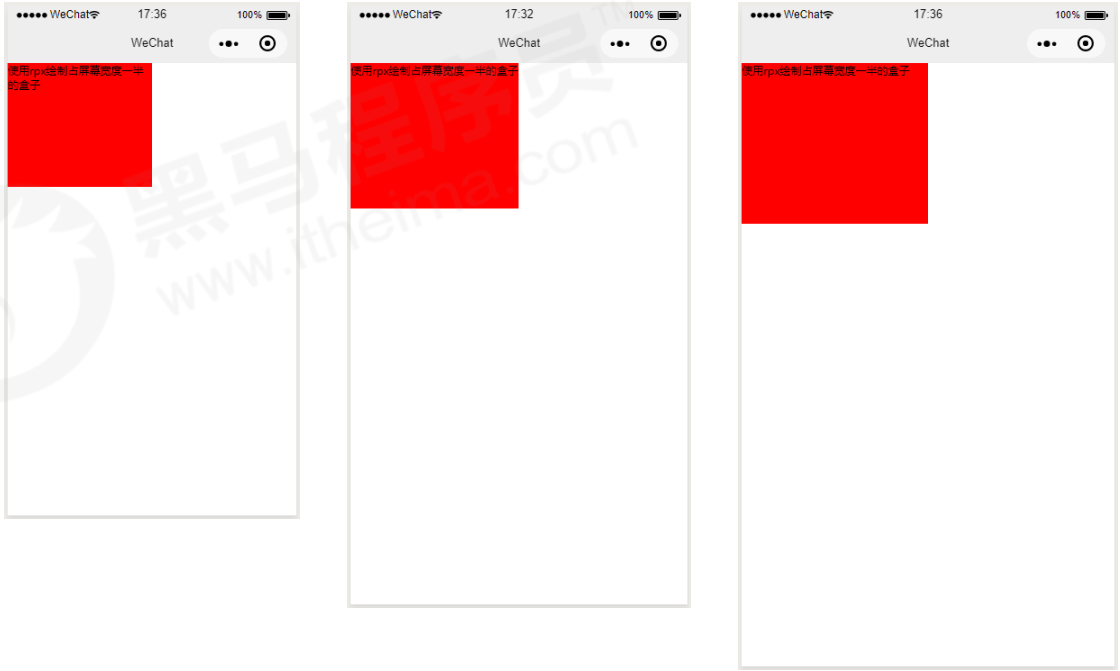
官方建议：开发微信小程序时，设计师可以用 iPhone6 作为视觉稿的标准。如果要根据iPhone6的设计稿，绘制小程序页面，可以直接把单位从 px 替换为 rpx 。例如，假设iPhone6设计稿上，要绘制一个 宽高为 200px 的盒子，换算为 rpx 为 200rpx。

2. 小程序中的样式

案例：使用rpx绘制占屏幕宽度一半的盒子

需求描述：

使用rpx尺寸单位，绘制一个红色背景的view组件，使之在各种尺寸的移动端屏幕上，能够自适应调整自身的宽度，都显示为占屏幕宽度的一半。



2. 小程序中的样式

案例：实现步骤

1. 搭建页面的wxml结构

```
<view class="v1">
    使用rpx绘制占屏幕宽度一半的盒子
</view>
```

通过 class 属性，为 view 组件添加类名为 v1

注意：这里添加的类名，是为了方便后续为组件添加样式

2. 小程序中的样式

案例：实现步骤

2. 编写wxss的样式

```
.v1{
    width: 375rpx;
    height: 375rpx;
    background-color: red;
}
```

- 注意：这里标红的属性 width，它的值被设置为了 375rpx 后，就能够实现需求效果了；
- 原因：小程序中的rpx尺寸单位，把所有宽度尺寸的屏幕，统一划分为了 750 份，不论大屏幕还是小屏幕，375rpx 会被小程序识别，并渲染为屏幕宽度的一半。

2. 小程序中的样式

2.6 @import 样式导入

- 使用 @import 语句可以导入外联样式表；
- 语法格式为：

  @import “wxss样式表的相对路径”；

```
/** common.wxss **/
.small-p {
  padding:5px;
}
```

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

2. 小程序中的样式

2.7 全局样式与局部样式

1. 全局样式

定义在 app.wxss 中的样式为全局样式，作用于每一个页面。

2. 局部样式

在 page 的 wxss 文件中定义的样式为局部样式，只作用在对应的页面，并会覆盖 app.wxss 中相同的选择器。

注意：当局部样式的权重大于或等于全局样式的权重时，才会覆盖全局的样式效果！

目录 Contents

3. 使用全局配置文件app.json

3.1 app.json配置文件的作用

小程序根目录下的 **app.json 文件**用来对微信小程序进行**全局配置**，它决定了页面文件的路径、窗口表现、设置网络超时时间、设置多 tab 等。

在app.json配置文件中，最主要的配置节点是：

- **pages 数组**：配置小程序的页面路径
- **window 对象**：用于设置小程序的状态栏、导航条、标题、窗口背景色
- **tabBar 对象**：配置小程序的tab栏效果

3. 使用全局配置文件app.json

3.2 pages – 配置小程序的页面路径

- pages 用于指定小程序由哪些页面组成，每一项都对应一个页面的 路径+文件名 信息。
- 文件名不需要写文件后缀，框架会自动去寻找对应位置的 .json、 .js、 .wxml 和 .wxss 四个文件进行处理。

如开发目录为：

```
├── app.js
├── app.json
├── app.wxss
├── pages
│   ├── index
│   │   ├── index.wxml
│   │   ├── index.js
│   │   ├── index.json
│   │   └── index.wxss
│   └── logs
│       ├── logs.wxml
│       └── logs.js
└── utils
```

则需要在 app.json 中写

```
{
  "pages":[
    "pages/index/index",
    "pages/logs/logs"
  ]
}
```

3. 使用全局配置文件app.json

3.2 pages – 配置小程序的页面路径

1. 自动创建新页面

- 回顾：之前创建新页面，需要新建页面目录 -> 新建页面文件 -> 修改pages数组
- 现在推荐的方式：打开 app.json -> pages 数组节点 -> 新增页面路径并保存 -> 自动创建路径对应的页面

3. 使用全局配置文件app.json

3.2 pages – 配置小程序的页面路径

2. 设置默认首页

- 打开 app.json -> pages 数组节点
- 按需调整数组中路径的顺序，即可修改默认首页

注意：

① 数组的第一项代表小程序的初始页面（首页）。

② 小程序中新增/减少页面，都需要对 pages 数组进行修改。

3. 使用全局配置文件app.json

3.3 小程序窗口的组成部分

3. 使用全局配置文件app.json

3.4 window节点常用的配置项

属性名	类型	默认值	说明
navigationBarTitleText	String	字符串	导航栏标题文字内容
navigationBarBackgroundColor	HexColor	#000000	导航栏背景颜色，如 #000000
navigationBarTextStyle	String	white	导航栏标题颜色，仅支持 black / white
backgroundColor	HexColor	#ffffff	窗口的背景色
backgroundTextStyle	String	dark	下拉 loading 的样式，仅支持 dark / light
enablePullDownRefresh	Boolean	false	是否全局开启下拉刷新。 详见 Page.onPullDownRefresh
onReachBottomDistance	Number	50	页面上拉触底事件触发时距页面底部距离，单位为px。 详见 Page.onReachBottom

3. 使用全局配置文件app.json

3.4 window节点常用的配置项

1. 设置导航栏标题文字内容

设置步骤：app.json -> window -> navigationBarTitleText

需求：把导航条上的标题，从默认的 **WeChat** 修改为 **黑马程序员** ，效果如图所示：

3. 使用全局配置文件app.json

3.4 window节点常用的配置项

2. 设置导航栏背景色

设置步骤：app.json -> window -> navigationBarBackgroundColor

需求：把导航条上的标题，从默认的 **#fff** 修改为 **#2b4b6b** ，效果如图所示：

3. 使用全局配置文件app.json

3.4 window节点常用的配置项

3. 设置导航栏标题颜色

设置步骤：app.json -> window -> navigationBarTextStyle

需求：把导航条上的标题，从默认的 **black** 修改为 **white** ，效果如图所示：

3. 使用全局配置文件app.json

3.4 window节点常用的配置项

4. 全局开启下拉刷新功能

概念：下拉刷新是移动端的专有名词，通过手指在屏幕上的下拉滑动操作，从而重新加载页面数据的行为；

设置步骤：app.json -> window -> 把 enablePullDownRefresh 的值设置为 true

3. 使用全局配置文件app.json

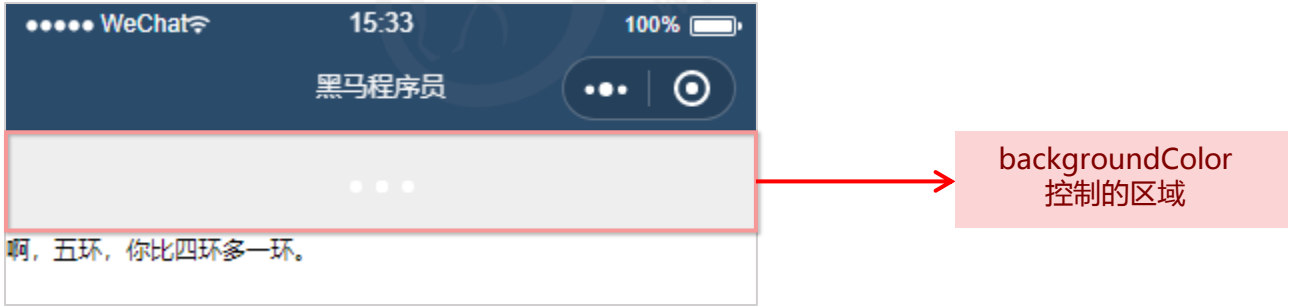
3.4 window节点常用的配置项

5. 设置下拉刷新窗口的背景色

当全局开启下拉刷新功能之后，默认的窗口背景为白色；如果自定义下拉刷新窗口背景色，设置步骤为：

app.json -> window -> 为 backgroundColor 指定16进制颜色值 #eee

效果如下：



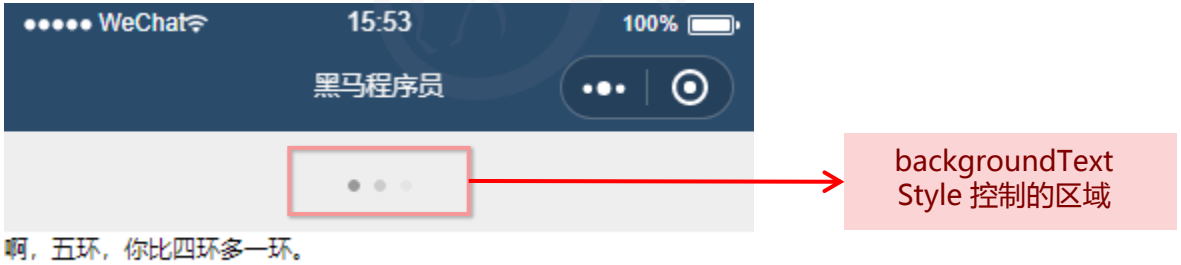
3. 使用全局配置文件app.json

3.4 window节点常用的配置项

6. 设置下拉 loading 的样式

当全局开启下拉刷新功能之后，默认窗口的loading样式为白色，如果要更改loading样式的效果，设置步骤为 app.json -> window -> 为 backgroundTextStyle 指定 dark 值

效果如下：



3. 使用全局配置文件app.json

3.4 window节点常用的配置项

7. 设置上拉触底的距离

概念：上拉触底是移动端的专有名词，通过手指在屏幕上的上拉滑动操作，从而加载更多数据的行为；

设置步骤： app.json -> window -> 为 onReachBottomDistance 设置新的数值

注意：默认距离为50px，如果没有特殊需求，建议使用默认值即可；

3. 使用全局配置文件app.json

3.5 tabBar - 配置Tab栏

1. 什么是tabBar

tabBar 是移动端应用常见的页面效果，用于实现多页面的快速切换；

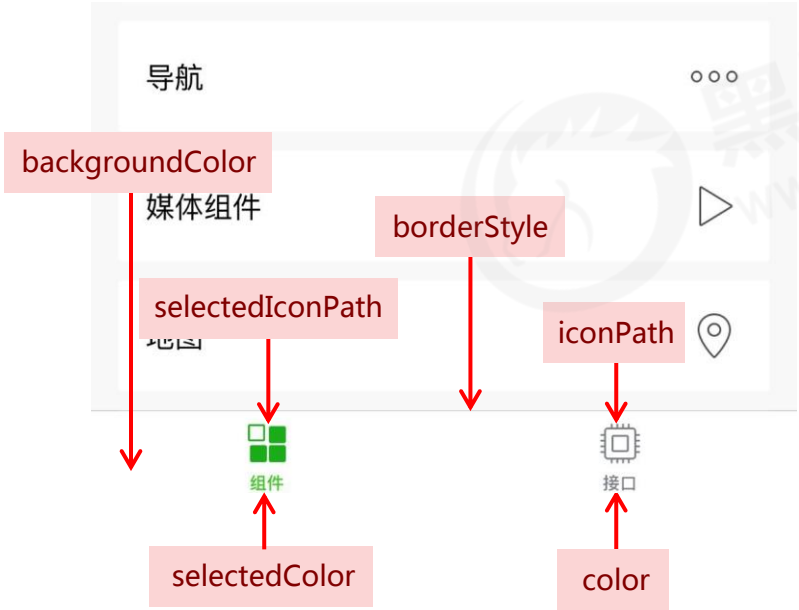
小程序中通常将其分为底部tabBar和顶部tabBar。

注意：tabBar中，只能配置最少2个、最多5个 tab 页签，当渲染顶部tabBar的时候，不显示icon，只显示文本

3. 使用全局配置文件app.json

3.5 tabBar - 配置Tab栏

2. tabBar的组成部分



backgroundColor：导航条背景色

selectedIconPath：选中时的图片路径

borderStyle：tabBar上边框的颜色

iconPath：未选中时的图片路径

selectedColor：tab 上的文字选中时的颜色

color：tab 上的文字默认（未选中）颜色

3. 使用全局配置文件app.json

3.6 tabBar节点的配置项

属性	类型	必填	默认值	描述
color	HexColor	否		tab 上的文字默认颜色
selectedColor	HexColor	否		tab 上的文字选中时的颜色
backgroundColor	HexColor	否		tab 的背景色
borderStyle	String	否	black	tabBar上边框的颜色， 仅支持 black / white
list	Array	是		tab 的列表，详见 list 属性说明， 最少2个、最多5个 tab
position	String	否	bottom	tabBar的位置， 仅支持 bottom / top

3. 使用全局配置文件app.json

3.7 tabBar节点中list的配置项

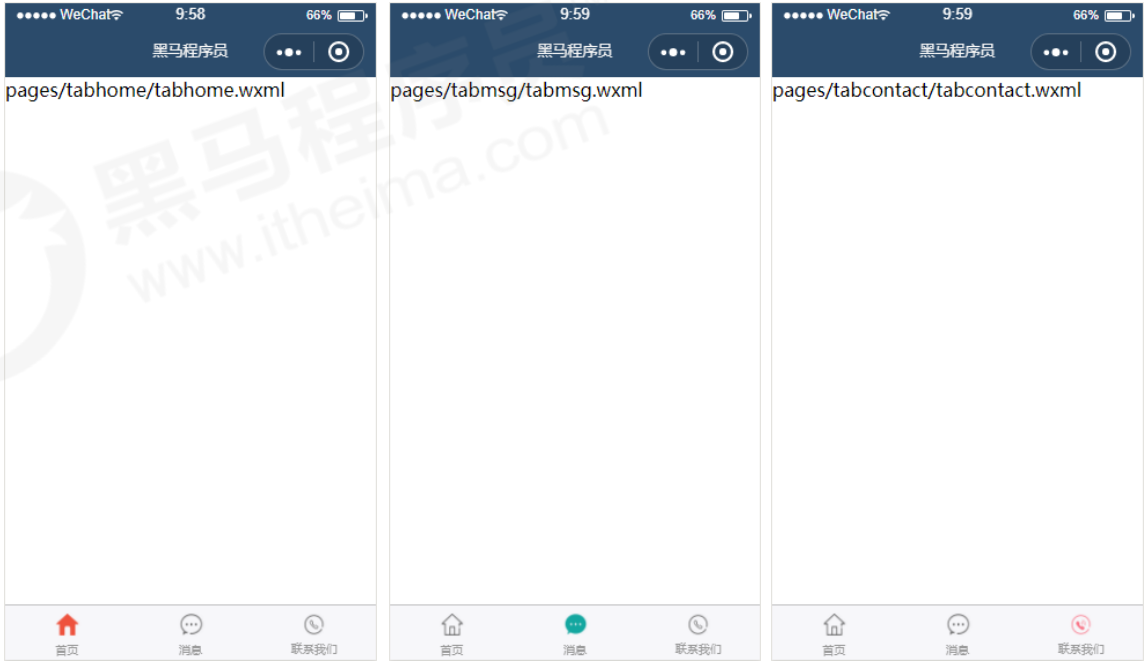
属性	类型	必填	描述
pagePath	String	是	页面路径，必须在 pages 中先定义
text	String	是	tab 上按钮文字
iconPath	String	否	图片路径；icon 大小限制为40kb，建议尺寸为 81px * 81px，不支持网络图。 当 postion 为 top 时，不显示 icon。
selectedIconPath	String	否	选中时的图片路径； icon 大小限制为40kb， 建议尺寸为 81px * 81px，不支持网络图片。 当 postion 为 top 时，不显示 icon。

3. 使用全局配置文件app.json

案例：配置tabBar

需求描述：

根据资料中提供的小图标、在小程序中配置如图所示的tabBar。



3. 使用全局配置文件app.json

案例：实现步骤

1. 通过app.json中的pages数组新建3个tab页面

```
{
  "pages": [
    "pages/tabhome/tabhome",
    "pages/tabmsg/tabmsg",
    "pages/tabcontact/tabcontact"
  ]
}
```

注意：tabhome是首页，tabmsg是消息，tabcontact是联系我们。

3. 使用全局配置文件app.json

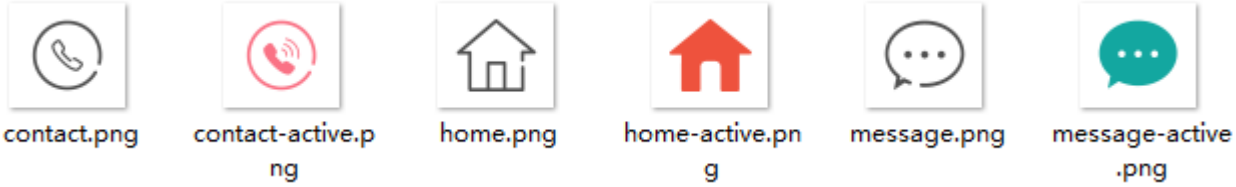
案例：实现步骤

2. 把资料中提供的小图标拷贝到项目指定目录中

① 在小程序根目录中，新建 assets -> images 目录；

② 把资料中提供的6个小图标，拷贝到新建的 images 目录中；

③ 将需要用到的小图标分为3组，每组两个，其中图片名称中包含 -active 的是选中之后的图标，不包含 -active 的是默认图标，截图如下：



3. 使用全局配置文件app.json

案例：实现步骤

3. 配置tabBar选项

① 打开app.json配置文件，和pages、window平级，新增tabBar节点；

② tabBar节点中，新增list数组，这个数组中存放的，是每个tab页的配置对象；

③ 在list数组中，新增每一个tab页的配置对象。对象中包含的属性与作用如下：

- pagePath 指定当前tab对应的页面路径【必填】
- text 指定当前tab上按钮的文字【必填】
- iconPath 指定当前tab未选中时候的图片路径【可选】
- selectedIconPath 指定当前tab被选中后高亮的图片路径【可选】

3. 使用全局配置文件app.json

4. tabBar节点配置的完整代码

```
{
  "tabBar": {
    "list": [{
      "pagePath": "pages/tabhome/tabhome",
      "text": "首页",
      "iconPath": "/assets/images/home.png",
      "selectedIconPath": "/assets/images/home-active.png"
    },{
      "pagePath": "pages/tabmsg/tabmsg",
      "text": "消息",
      "iconPath": "/assets/images/message.png",
      "selectedIconPath": "/assets/images/message-active.png"
    }, {
      "pagePath": "pages/tabcontact/tabcontact",
      "text": "联系我们",
      "iconPath": "/assets/images/contact.png",
      "selectedIconPath": "/assets/images/contact-active.png"
    }]
}
```

目录 Contents

4. 使用页面配置文件page.json

4.1 页面级别和全局级别配置的关系

小程序中，app.json 中的 window节点，可以全局配置小程序中每个页面的窗口表现；

如果某些小程序页面，想要拥有特殊的窗口表现，此时，“页面级别的.json配置文件”就可以实现这种需求；

总结：页面级别配置优先于全局配置生效。

4. 使用页面配置文件page.json

4.2 页面配置文件中可选的配置项列表

属性	类型	默认值	描述
navigationBarBackgroundColor	HexColor	#000000	导航栏背景颜色，如 #000000
navigationBarTextStyle	String	white	导航栏标题颜色，仅支持 black / white
navigationBarTitleText	String		导航栏标题文字内容
backgroundColor	HexColor	#ffffff	窗口的背景色
backgroundTextStyle	String	dark	下拉loading的样式，仅支持 dark / light

注意：页面的.json只能设置 window 相关的配置项，以决定本页面的窗口表现。

4. 使用页面配置文件page.json

4.2 页面配置文件中可选的配置项列表

属性	类型	默认值	描述
enablePullDownRefresh	Boolean	false	是否全局开启下拉刷新。 详见 Page.onPullDownRefresh
onReachBottomDistance	Number	50	页面上拉触底事件触发时距页面底部距离， 单位为px。 详见 Page.onReachBottom
disableScroll	Boolean	false	设置为 true 则页面整体不能上下滚动； 只在页面配置中有效， 无法在 app.json 中设置该项

注意：页面的.json只能设置 window 相关的配置项，以决定本页面的窗口表现；

目录 Contents

5. 小程序中的生命周期

5.1 什么是生命周期

生命周期（Life Cycle）是指一个对象从 创建 -> 运行 -> 销毁 的整个阶段，**强调的是一个时间段**。

例如：

- 张三出生，表示这个人生命周期的开始
- 张三死亡，表示这个人生命周期的结束
- 中间张三的一生，就是张三的生命周期

我们可以把每个小程序运行的过程，也概括为生命周期：

- 小程序的启动，表示生命周期的开始
- 小程序的关闭，表示生命周期的结束
- 中间小程序运行的过程，就是小程序的生命周期

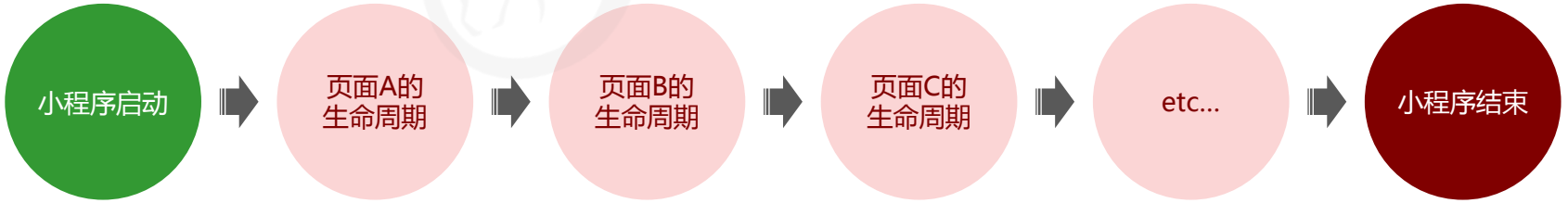
5. 小程序中的生命周期

5.2 生命周期的两种类型

在小程序中，包含两种类型的生命周期：

- 应用生命周期：特指小程序从启动 -> 运行 -> 销毁的过程；
- 页面生命周期：特指小程序中，每个页面的加载 -> 渲染 -> 销毁的过程；

其中，页面的生命周期范围较小，应用程序的生命周期范围较大，如图所示：



5. 小程序中的生命周期

5.3 什么是生命周期函数

生命周期函数：是由小程序框架提供的内置函数，会伴随着生命周期，**自动**按次序执行；

生命周期函数的作用：允许程序员在特定的生命周期时间点上，**执行某些特定的操作**。例如，页面刚加载的时候，在生命周期函数中自动发起数据请求，获取当前页面的数据；

注意：生命周期强调的是时间段，生命周期函数强调的是时间点。

5. 小程序中的生命周期

5.4 生命周期函数的分类

小程序中的生命周期函数，分为两种类型：

- 应用生命周期函数
- 页面生命周期函数

5. 小程序中的生命周期

5.5 应用生命周期函数

app.js 是小程序执行的入口文件，在 app.js 中必须调用 App() 函数，且只能调用一次。其中，App() 函数是用来注册并执行小程序的。

App(Object) 函数接收一个Object参数，可以通过这个Object参数，指定小程序的生命周期函数。

例如：

```
App({
  // 小程序初始化完成时，执行此函数，可以做一些初始化的工作。
  onLaunch: function(options) {},
  // 小程序显示到屏幕上的时候，执行此函数。
  onShow  : function(options) {},
  // 小程序被最小化的时候，执行此函数。
  onHide  : function() {}
})
```

5. 小程序中的生命周期

5.5 应用生命周期函数

应用生命周期函数列表

属性	类型	描述	触发时机
onLaunch	Function	生命周期回调 - 监听小程序初始化	小程序初始化完成时 （全局只触发一次）
onShow	Function	生命周期回调 - 监听小程序显示	小程序启动，或从后台进入前台显示时
onHide	Function	生命周期回调 - 监听小程序隐藏	小程序从前台进入后台时

5. 小程序中的生命周期

5.6 页面生命周期函数

每个小程序页面，必须拥有自己的 .js 文件，且必须调用 Page() 函数，否则报错。其中Page() 函数用来注册小程序页面。

Page(Object) 函数接收一个Object参数，可以通过这个Object参数，指定页面的生命周期函数。

例如：

```
Page({
  onLoad  : function(options) {}, // 监听页面加载
  onShow  : function() {},        // 监听页面显示
  onReady : function() {},        // 监听页面初次渲染完成
  onHide  : function() {},        // 监听页面隐藏
  onUnload: function() {}         // 监听页面卸载
})
```

5. 小程序中的生命周期

5.6 页面生命周期函数

页面生命周期函数列表

属性	类型	描述
onLoad	Function	生命周期回调 - 监听页面加载
onShow	Function	生命周期回调 - 监听页面显示
onReady	Function	生命周期回调 - 监听页面初次渲染完成
onHide	Function	生命周期回调 - 监听页面隐藏
onUnload	Function	生命周期回调 - 监听页面卸载

传智播客旗下高端IT教育品牌