目录

[基础 3](#_Toc518923571)

[文件 3](#_Toc518923572)

[json 3](#_Toc518923573)

[wxml 3](#_Toc518923574)

[wxss 3](#_Toc518923575)

[其他 4](#_Toc518923576)

[大小 4](#_Toc518923577)

[逻辑层 4](#_Toc518923578)

[注册程序 4](#_Toc518923579)

[App 4](#_Toc518923580)

[getApp 4](#_Toc518923581)

[scene场景值 5](#_Toc518923582)

[page（object）页面 5](#_Toc518923583)

[object 5](#_Toc518923584)

[Page.prototype.setData 5](#_Toc518923585)

[生命周期 6](#_Toc518923586)

[this.data 8](#_Toc518923587)

[Page.prototype.route 8](#_Toc518923588)

[路由 8](#_Toc518923589)

[模块化 8](#_Toc518923590)

[组件 8](#_Toc518923591)

[json 9](#_Toc518923592)

[wxss 9](#_Toc518923593)

[wxml 9](#_Toc518923594)

[<slot> 9](#_Toc518923595)

[js 10](#_Toc518923596)

[使用 10](#_Toc518923597)

[Component() 10](#_Toc518923598)

[properties 10](#_Toc518923599)

[组件事件 11](#_Toc518923600)

[behaviors 11](#_Toc518923601)

[内置behaviors 11](#_Toc518923602)

[组件间关系 11](#_Toc518923603)

[wxml 11](#_Toc518923604)

[数据绑定 11](#_Toc518923605)

[位置 12](#_Toc518923606)

[列表渲染 12](#_Toc518923607)

[key 12](#_Toc518923608)

[条件渲染 12](#_Toc518923609)

[block 12](#_Toc518923610)

[对比hiddden 13](#_Toc518923611)

[模板 13](#_Toc518923612)

[事件 13](#_Toc518923613)

[事件冒泡 13](#_Toc518923614)

[事件捕获 13](#_Toc518923615)

[自定义事件 13](#_Toc518923616)

[引用 14](#_Toc518923617)

[import 14](#_Toc518923618)

[include 14](#_Toc518923619)

[wxs 14](#_Toc518923620)

# 基础

## 文件

### json

#### app.json

pages

#### page.json

### wxml

区别Html:

1）有一些封装的组件，例如view/text。

2) 有一些数据绑定的指令。

节点标签名只能是小写字母、中划线和下划线的组合，所以自定义组件的标签名也只能包含这些字符。

### wxss

与css区别

1、增加尺寸单位rpx, 换算采用的浮点数运算，所以运算结果会和预期结果有一点点偏差。规定屏幕宽为750rpx。

2、模块化。app.wxss全局样式，page.wxss局部样式。

3、仅支持部分css选择器。

## 其他

### 大小

整个小程序所有分包大小不超过 4M

单个分包/主包大小不能超过 2M

所以一共4m。

# 逻辑层

## 注册程序

### App

#### onLaunch

此时还没有生成page

### getApp

App() 必须在 app.js 中注册，且不能注册多个。

不要在定义于 App() 内的函数中调用 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 还没有生成。

通过 getApp() 获取实例之后，不要私自调用生命周期函数。

## scene场景值

由于Android系统限制，目前还无法获取到按 Home 键退出到桌面，然后从桌面再次进小程序的场景值，对于这种情况，会保留上一次的场景值。

## page（object）页面

object 内容在页面加载时会进行一次深拷贝

### object

可以自定义函数，在wxml中调用。

#### data

初始化数据将作为页面的第一次渲染

要可以转成json(字符串，数字，布尔值，对象，数组)

### Page.prototype.setData

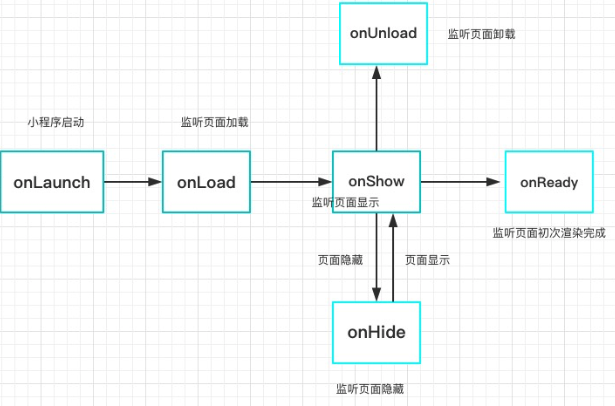
类似setState,不用对象整体替换。

直接修改 this.data 而不调用 this.setData 是无法改变页面的状态的，还会造成数据不一致。

单次设置的数据不能超过1024kB，请尽量避免一次设置过多的数据。

请不要把 data 中任何一项的 value 设为 undefined ，否则这一项将不被设置并可能遗留一些潜在问题。

### 生命周期





### this.data

其中 key 可以非常灵活，以数据路径的形式给出

### Page.prototype.route

## 路由

只有打开新页面，原页面不出栈（即onhide）。

只有页面返回不执行onload。

跨tab打开，tab级页面onhide,非tab级页面逐层unload。

跨tab重定向，

* navigateTo, redirectTo 只能打开非 tabBar 页面。
* switchTab 只能打开 tabBar 页面。
* reLaunch 可以打开任意页面。
* 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。
* 调用页面路由带的参数可以在目标页面的onLoad中获取。

## 模块化

通过 module.exports 或者 exports 才能对外暴露接口。和node一样。

小程序目前不支持直接引入 node\_modules , 开发者需要使用到 node\_modules 时候建议拷贝出相关的代码到小程序的目录中。

require 暂时不支持绝对路径

# 组件

1.6.3开始支持。

自定义组件和使用自定义组件的页面所在项目根目录名不能以“wx-”为前缀，否则会报错。

旧版本的基础库不支持自定义组件，此时，引用自定义组件的节点会变为默认的空节点。

## json

类似Page,需要申明

{

"component": true

}

## wxss

在组件wxss中不应使用ID选择器、属性选择器和标签名选择器。

组件和引用组件的页面中使用后代选择器（.a .b）在一些极端情况下会有非预期的表现，如遇，请避免使用。

子元素选择器（.a>.b）只能用于 view 组件与其子节点之间，用于其他组件可能导致非预期的情况。

继承样式，如 font 、 color ，会从组件外继承到组件内。

除继承样式外， app.wxss 中的样式、组件所在页面的的样式对自定义组件无效。

组件可以指定它所在节点的默认样式，使用 :host

## wxml

### <slot>

类似react的children

一个组件的wxml中只能有一个slot。需要使用多slot时，可以在组件js中声明启用。多个slot用name区分

组件

<slot name="before"></slot>

父级

<view slot="before">这里是插入到组件slot name="before"中的内容</view>

## js

使用 Component() 来注册组件

## 使用

在page或component中。

{

"usingComponents": {

"component-tag-name": "path/to/the/custom/component"

}

}

## Component()

### properties

在 properties 定义段中，属性名采用驼峰写法（propertyName）；在 wxml 中，指定属性值时则对应使用连字符写法（component-tag-name property-name="attr value"），应用于数据绑定时采用驼峰写(attr="{{propertyName}}"

* Component 构造器构造的组件也可以作为页面使用。
* 使用 this.data 可以获取内部数据和属性值，但不要直接修改它们，应使用 setData 修改。
* 生命周期函数无法在组件方法中通过 this 访问到。
* 属性名应避免以 data 开头，即不要命名成 dataXyz 这样的形式，因为在 WXML 中， data-xyz="" 会被作为节点 dataset 来处理，而不是组件属性。
* 在一个组件的定义和使用时，组件的属性名和data字段相互间都不能冲突（尽管它们位于不同的定义段中）。

### 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(不是onReady) 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

## 组件事件

意思是说可以给组件绑定事件。像正常标签一样。

## behaviors

类似mixins

使用module规则引入引出。

每个组件可以引用多个 behavior 。 behavior 也可以引用其他 behavior。

如果有同名的属性或方法，组件本身的属性或方法会覆盖 behavior 中的属性或方法，如果引用了多个 behavior ，在定义段中靠后 behavior 中的属性或方法会覆盖靠前的属性或方法；

如果有同名的数据字段，如果数据是对象类型，会进行对象合并，如果是非对象类型则会进行相互覆盖；

生命周期函数不会相互覆盖，而是在对应触发时机被逐个调用。如果同一个 behavior 被一个组件多次引用，它定义的生命周期函数只会被执行一次。

### 内置behaviors

## 组件间关系

# wxml

## 数据绑定

双大括号加data/properties中的变量，顶层key开始一层层向下。

{{}}:并不是用js执行其中的内容，有一套规则。

### 位置

数据可用于内容，组件属性/控制属性（id="item-{{id}}"）

其他需要大括号：关键字（true之类），运算，逻辑判断，字符串运算，

对象：data="{{for: a, bar: b}}" **不需要三个大括号，如有存在变量名相同的情况，后边的会覆盖前面**

花括号和引号之间如果有空格，将最终被解析成为字符串

## 列表渲染

<view wx:for="{{array}}">

{{index}}: {{item.message}} // 默认index item

</view>

使用 wx:for-item wx:for-index 可以自定义。

当 wx:for 的值为字符串时，会将字符串解析成字符串数组

也可以用block。

### key

用字符串表示item的属性，或者\*this表示item 本身。

## 条件渲染

### block

特殊组件，如果有多个在一个逻辑里有多个标签，可以用这个包裹起来。

### 对比hiddden

if需要显示才渲染，每次显示都会渲染。

hidden初始化时渲染，后面只是显示隐藏。

## 模板

模板需要的数据用data传入data="{{a,b,c}}"

模板拥有自己的作用域，只能使用 data 传入的数据以及模版定义文件中定义的 <wxs /> 模块。

## 事件

使用bind/catch capture-bind capture- catch绑定函数，在后面加上对应的事件，有无冒号都可以bind:tap,bindtap

函数写在page的对象里。

事件对象可以携带额外信息，如 id, dataset, touches。

### 事件冒泡

bind事件绑定不会阻止冒泡事件向上冒泡，catch事件绑定可以阻止冒泡事件向上冒泡。

### 事件捕获

触摸类事件支持捕获阶段

capture-bind、capture-catch关键字，后者将中断捕获阶段和取消冒泡阶段。

### 自定义事件

用triggerEvent触发。

this.triggerEvent('myevent', myEventDetail, myEventOption)

## 引用

WXML 提供两种文件引用方式import和include。

### import

只是import template。

import 有作用域的概念，即只会 import 目标文件中定义的 template，而不会 import 目标文件 import 的 template。

### include

目标文件除了 <template/> <wxs/> 外的整个代码引入，相当于是拷贝到 include 位置

# wxs

# 坑

小程序能力所致，bind 和 catch 事件同时绑定时候，只会触发 bind ,catch 不会被触发，要避免踩坑。