

创新创业实践课（移动应用开发） — 微信小程序

刘 健 培

ljp@bupt.edu.cn

北京邮电大学 计算机学院

课程内容

课程概述

- 课程安排
- 实验内容
- 考核方式
- 参考资料

微信小程序介绍

- 工具
- 框架
 - 逻辑层
 - 视图层
- 组件
- API
- 技术
 - HTML
 - CSS
 - JavaScript
 - JSON

课程概述

实验
考核

课程安排

- 时间
 - 第 4 周：课程入门介绍
 - 第5、7周：实验练习
- 地点
 - 本机房

实验内容

• 分8步完成一个简化的商城类微信小程序

• 主要功能

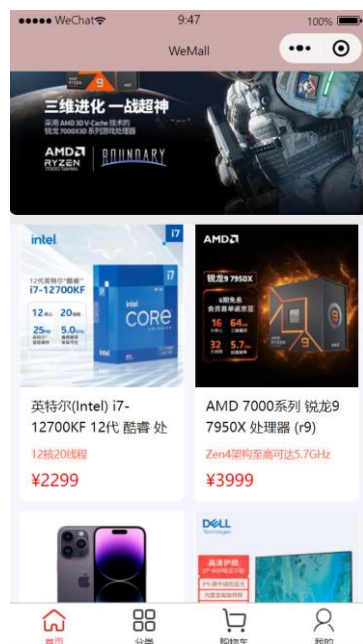
- 商品展示
- 购物车
- 订单

• 主要技术内容

- 小程序UI设计
- 数据绑定
- JS事件逻辑处理

- ▷ 1 实验1 搭建实验环境
- ▷ 2 实验2 商城首页
- ▷ 3 实验3 商品详情页面
- ▷ 4 实验4 分类页面
- ▷ 5 实验5 购物车页面
- ▷ 6 实验6 下单与支付页面
- ▷ 7 实验7 个人中心页
- ▷ 8 实验8 数据接口同步转异步（可选）

后端数据在程序内模拟



考核方式

- 方式1：提交课后作业

- 从实验指导书中挑选2个实验完成，并完成题后的作业（实验讲义中每个实验后有作业问题），将程序运行截图+作业答案 写在文档中。
- 实验自主完成，老师不验收（只需要交作业）
- 提交作业：将所有实验的答案写到一个文档内，上传到教学云平台
 - 文档命名：学号-姓名.docx

- 方式2：自主实验

- 使用微信小程序实现新功能。包括但不限于：
 - 丰富现有实验功能
 - 如增加订单展示、订单状态跟踪、店铺展示、优惠券、库存检查、商品搜索、用户管理、加购物车时选择商品规格和数量等功能
 - or 实现一个全新的微信小程序
- 提交 <小程序源码+实验文档/PPT> 到教学云平台
- 无需再提交作业
- 自主完成，抄袭无效

参考资料

- 实验讲义
- 微信小程序开发文档与图书

<https://developers.weixin.qq.com/miniprogram/dev/framework/>



- 技术资料
 - HTML
 - CSS
 - JavaScript



微信小程序介绍

特点
框架
组件
API

小程序功能特点

微信小程序简介（Wechat）

微信小程序是一种开放能力，是一种无需下载安装即可使用的应用，随时可用。

小程序可理解为“嵌在微信的APP”，与订阅号、服务号和企业号属于同级体系，由此，小程序、订阅号、服务号、企业号形成了并行的微信生态四大体系。



关于小程序和公众号的区别，首先登录的平台地址不一样，其次都是属于微信平台的一个应用，可以设置关联，前提是注册公众号与小程序的主体信息（即身份信息）需一致。小程序的名字不可和非同主体的公众号名字一样。

小程序可以扫码、搜索、公众号关联、微信中“小程序”查找、附近推荐、分享等入口方式体验各类小程序。

微信小程序功能

- 分享页
- 分享对话
- 线下扫码进入微信小程序
- 支持挂起状态
- 消息通知
- 实时音视频录制播放
- 硬件连接
- 小游戏
- 公众号关联
- 搜索查找与历史列表
- 识别二维码

微信小程序特点

- 运行在微信里面的移动网页应用



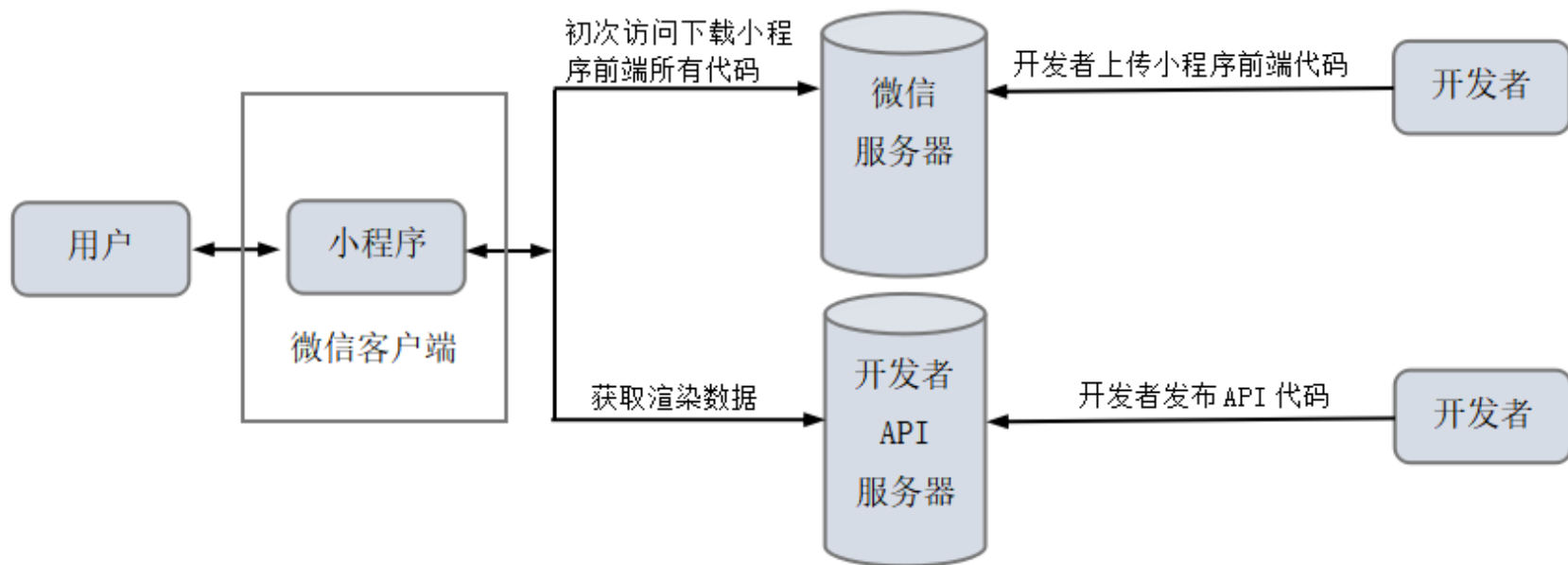
无需下载，即用即走，随手可得
简单框架，工具轻量
上手容易，开发逻辑较简单
基于微信，跨平台
体验接近Native App，可访问原生组件
依托微信社交链，推广容易

小

开发环境封闭
很多前端类库无法使用

微信小程序的工作流程

- 小程序前端代码统一上传到微信服务器，
- 用户访问时，微信客户端自动拉取小程序前端所有代码，
- 小程序代码里再调用API从开发则服务器取回数据，并把数据渲染到页面，然后展示给用户。



小程序开发简介

微信小程序开发流程

注册账号

- 1. 在微信公众平台注册小程序，完成AppID注册后可以同步进行信息完善和开发。

完善信息

- 2. 填写小程序基本信息，包括名称、头像、介绍及服务范围等

开发代码

- 3. 完成小程序开发者绑定、开发信息配置后，开发者可**下载开发者工具**、参考**开发文档**进行小程序的开发、调试和预览
- <https://mp.weixin.qq.com/cgi-bin/wx>

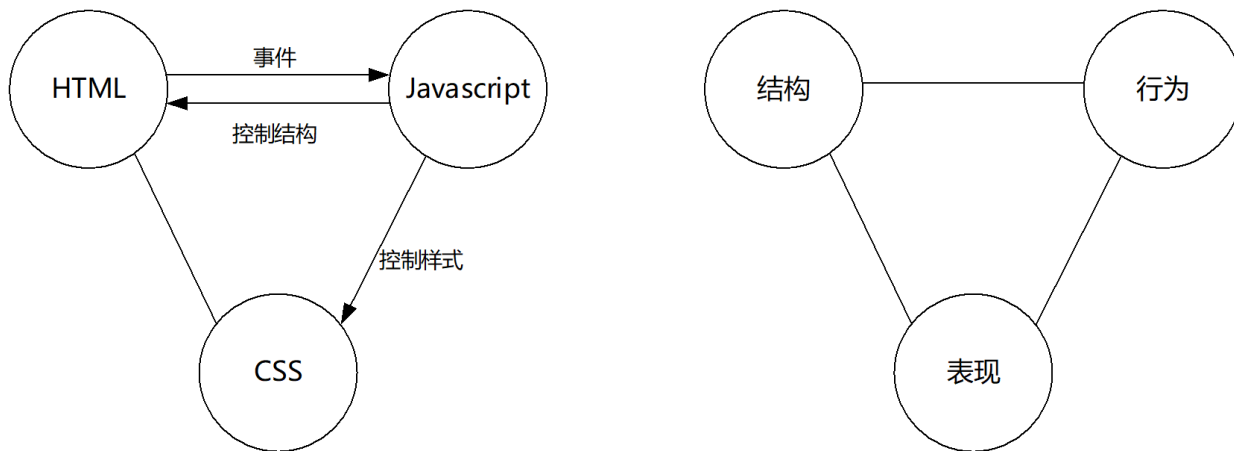
实验使用测试号
无需注册AppID
无需其余步骤

审核发布

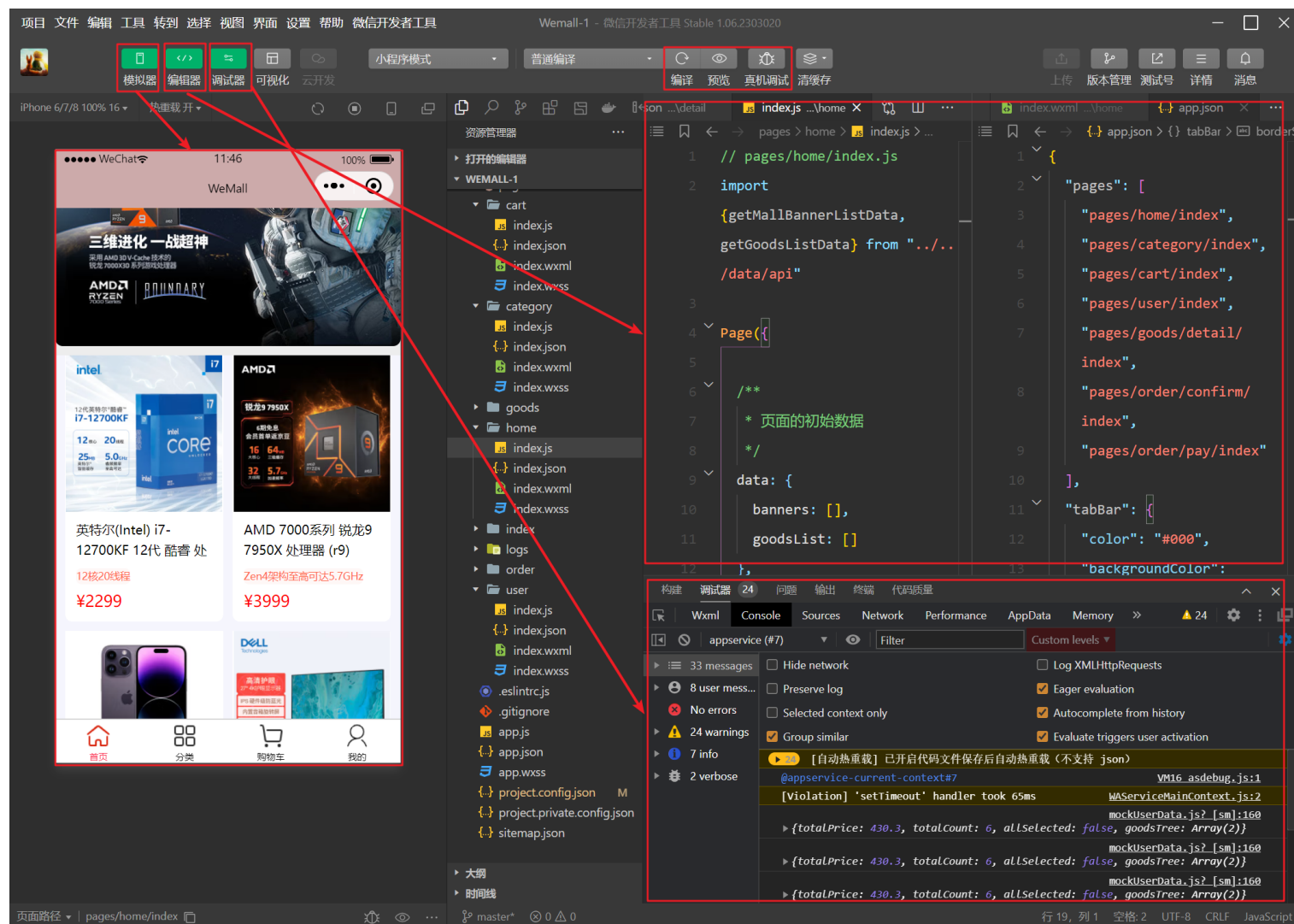
- 4. 完成小程序开发后，提交代码至微信团队审核，审核通过后即可发布

基础技术

- 微信小程序自定义了一套语言，称为**WXML**微信标记语言，它的使用方法类似于**HTML**语言。
- 微信小程序还定义了自己的样式语言**WXSS**，它兼容**CSS**样式，并在CSS样式的基础上做了扩展；
- 微信小程序使用**JavaScript**来进行业务处理，兼容了大部分**JavaScript**功能，但也有一些功能无法使用。
- 所以有一定HTML、CSS、JavaScript技术功底的人学习微信小程序开发会容易很多。

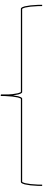


微信开发者工具



小程序全栈开发

- 小程序的全栈开发即从数据库,服务器到前端页面的一个完整技术栈的开发。

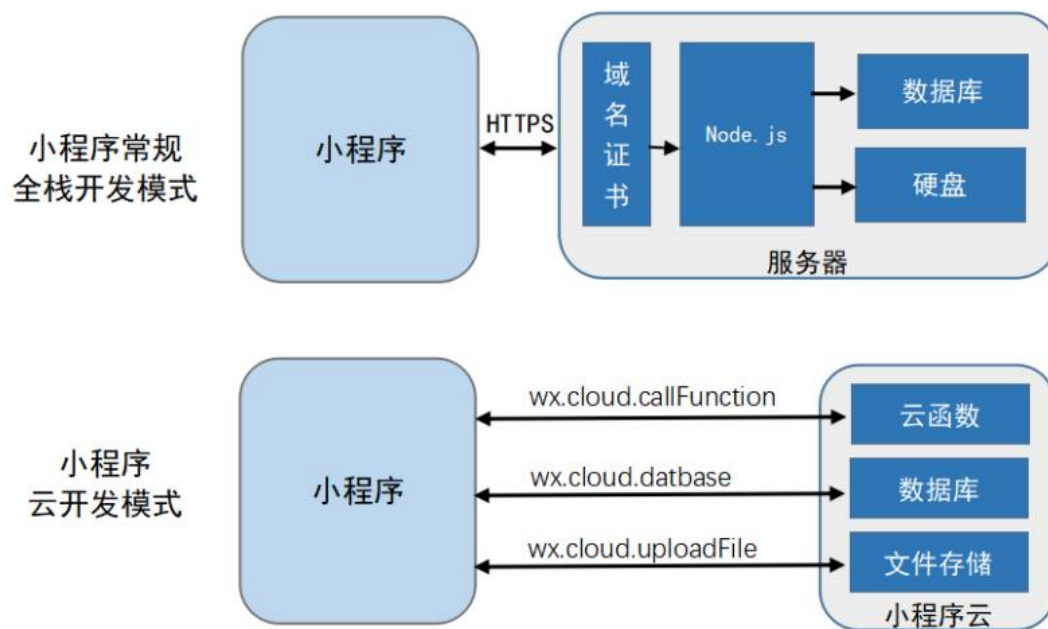
- 小程序全栈开发大概分为两种：
 - 小程序常规全栈开发模式
 - 小程序云开发

- 常规的全栈开发中，目前流行的是后端服务器采用node.js技术，云开发则提供了较为完整的服务器架构，也可省去不少的开发部署和维护成本。

小程序云开发

云开发为开发者提供了云函数、数据库、存储、云调用、HTTP API 五大基础服务端能力，让开发者无需搭建服务器，弱化服务端开发以及运维概念，只专注于小程序开发。

- 采用云函数，无需部署服务器、域名和证书。
- 前端可以直接查询有权限的数据库。
- 封装统一的上传文件API，无需开发后端接口。
- 控制台轻松测试API、监控云函数和查看日志。



全栈小程序传统开发模式与云开发模式对比

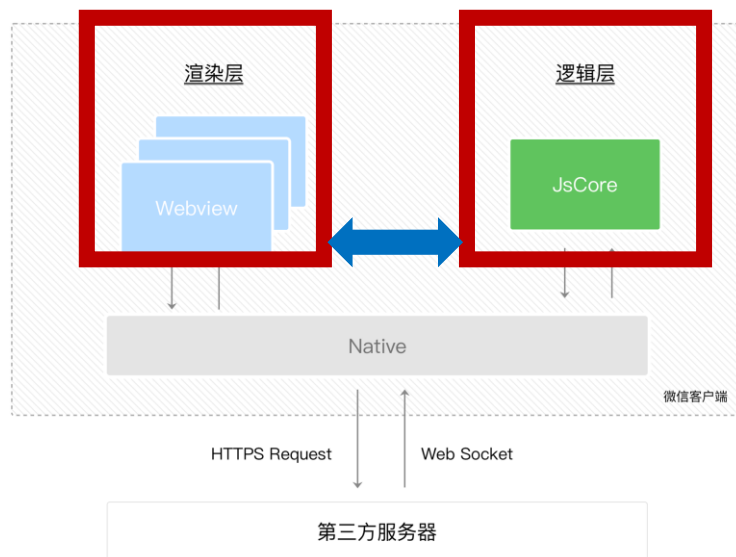
小程序开发框架

逻辑层

视图层

小程序的开发框架

- 小程序开发框架的目标是通过尽可能简单、高效的方式让开发者可以在微信中开发具有原生 APP 体验的服务。
- 整个小程序框架分为两部分：**逻辑层（App Service）**和**视图层（View）**。
 - 小程序提供了自己的视图层描述语言 WXML 和 WXSS，以及基于 JavaScript 的逻辑层框架，并在视图层与逻辑层间提供了数据传输和事件系统，让开发者能够专注于数据与逻辑。
- 视图层描述语言WXML和视图样式WXSS，再加上JavaScript逻辑层语言和JSON配置文件，一同构筑起了微信小程序框架。

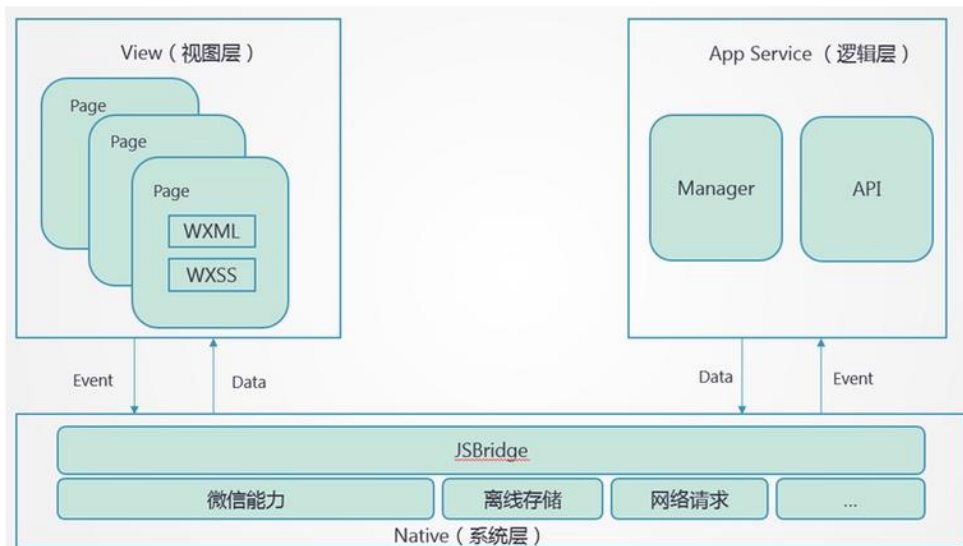


框架功能

- 数据绑定
 - 框架的核心是一个响应的数据绑定系统，可以让数据与视图非常简单地保持同步。当数据修改的时候，只需要在逻辑层修改数据，视图层就会做相应的更新。
- 页面管理
 - 框架管理了整个小程序的页面路由，可以做到页面间的无缝切换，并给以页面完整的生命周期。开发者需要做的只是将页面的数据、方法、生命周期函数注册到框架中，其他的一切复杂的操作都交由 框架 处理。
- 基础组件
 - 框架提供了一套基础的组件，这些组件自带微信风格的样式以及特殊的逻辑，开发者可以通过组合基础组件，创建出强大的微信小程序。
- 丰富的 API
 - 框架提供丰富的微信原生 API，可以方便的调起微信提供的能力，如获取用户信息，本地存储，支付功能等。

小程序框架的双线程运行机制

- 逻辑层：创建一个单独的线程去执行 JavaScript，在这里执行的都是有关小程序业务逻辑的代码，负责逻辑处理、数据请求、接口调用等
- 视图层：界面渲染相关的任务全都在 WebView 线程里执行，通过逻辑层代码去控制渲染哪些界面。一个小程序存在多个界面，所以视图层存在多个 WebView 线程
- JSBridge 起到架起上层开发与Native（系统层）的桥梁，使得小程序可通过API使用原生的功能，且部分组件为原生组件实现，从而有良好体验



小程序框架-逻辑层

小程序App

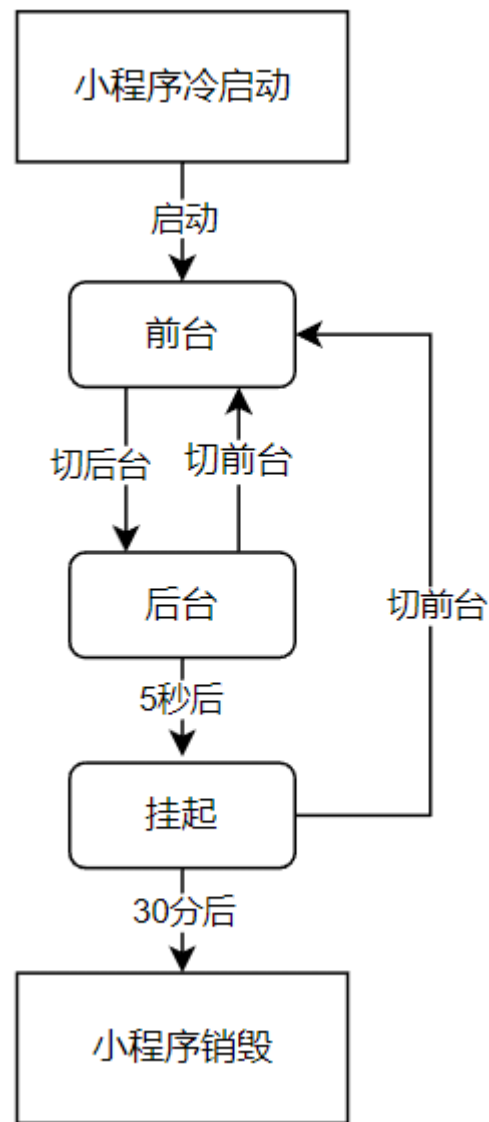
页面Page

逻辑层 App Service

- 小程序开发框架的逻辑层使用 JavaScript 引擎
- 逻辑层将数据进行处理后发送给视图层，同时接受视图层的事件反馈。
- 开发者写的所有代码最终将会打包成一份 JavaScript 文件，并在小程序启动的时候运行，直到小程序销毁。这一行为类似 ServiceWorker，所以逻辑层也称之为 App Service。
- 在 JavaScript 的基础上，小程序增加了一些功能，以方便开发：
 - 增加 App 和 Page 方法，进行程序注册和页面注册。
 - 增加 getApp 和 getCurrentPages 方法，分别用来获取 App 实例和当前页面栈。
 - 提供丰富的 API，如微信用户数据，扫一扫，支付等微信特有功能。
 - 提供模块化能力，每个页面有独立的作用域。
- 注意：小程序框架的逻辑层并非运行在浏览器中，因此 JavaScript 在 web 中一些能力都无法使用，如 window，document 等。

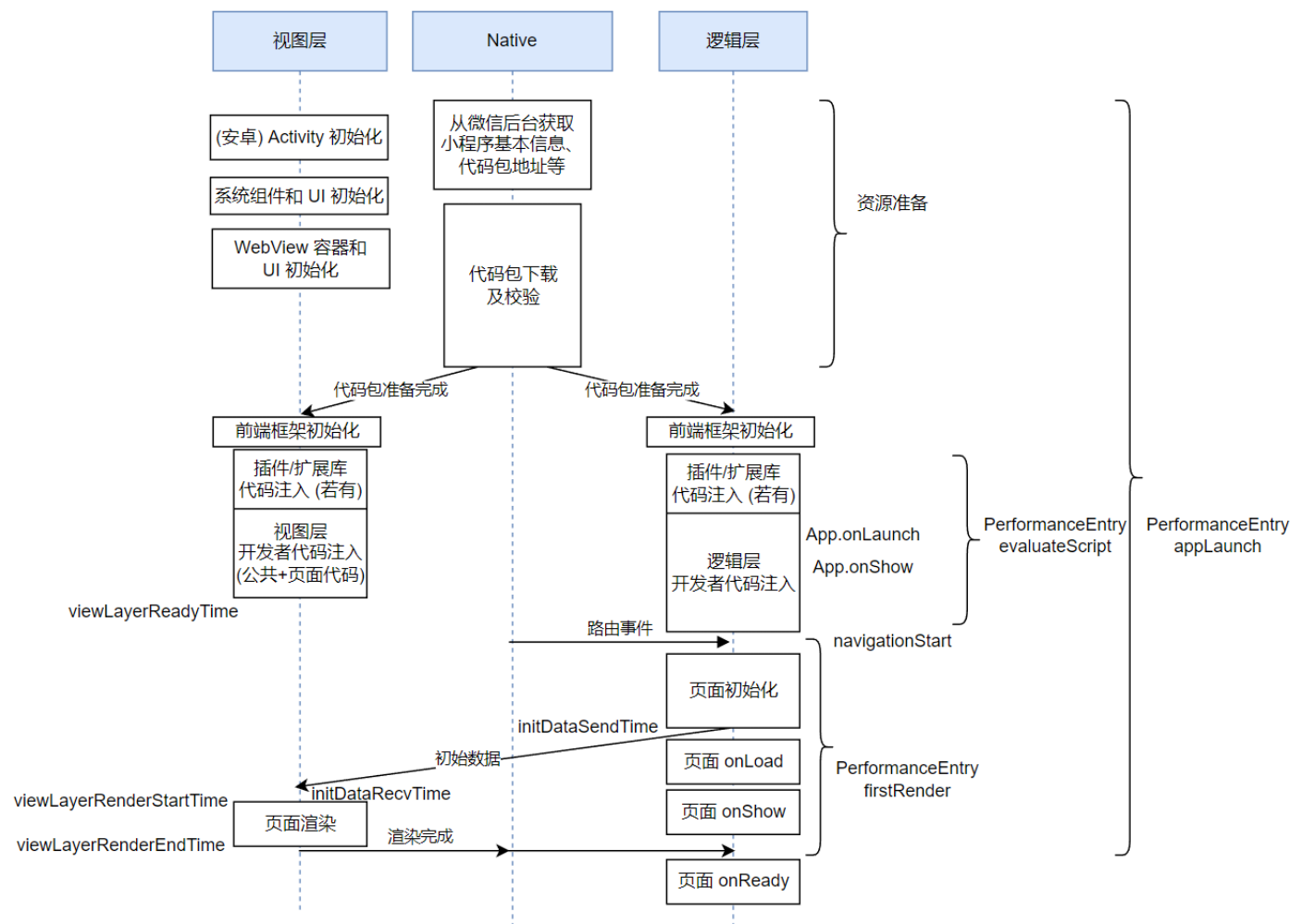
小程序生命周期

- 小程序从启动到最终被销毁，会经历很多不同的状态，小程序在不同状态下会有不同的表现。
- 小程序启动会有两种情况，「冷启动」与「热启动」。
 - 冷启动：用户首次打开或小程序被微信主动销毁后再次打开。
 - 热启动：用户已经打开过小程序，在一定时间内再次打开
- 当小程序进入后台，客户端会维持一段时间的运行状态，超过一定时间后（目前是30分钟）会被微信主动销毁
- 当短时间内（5s）连续收到两次以上收到系统内存告警，会进行小程序的销毁

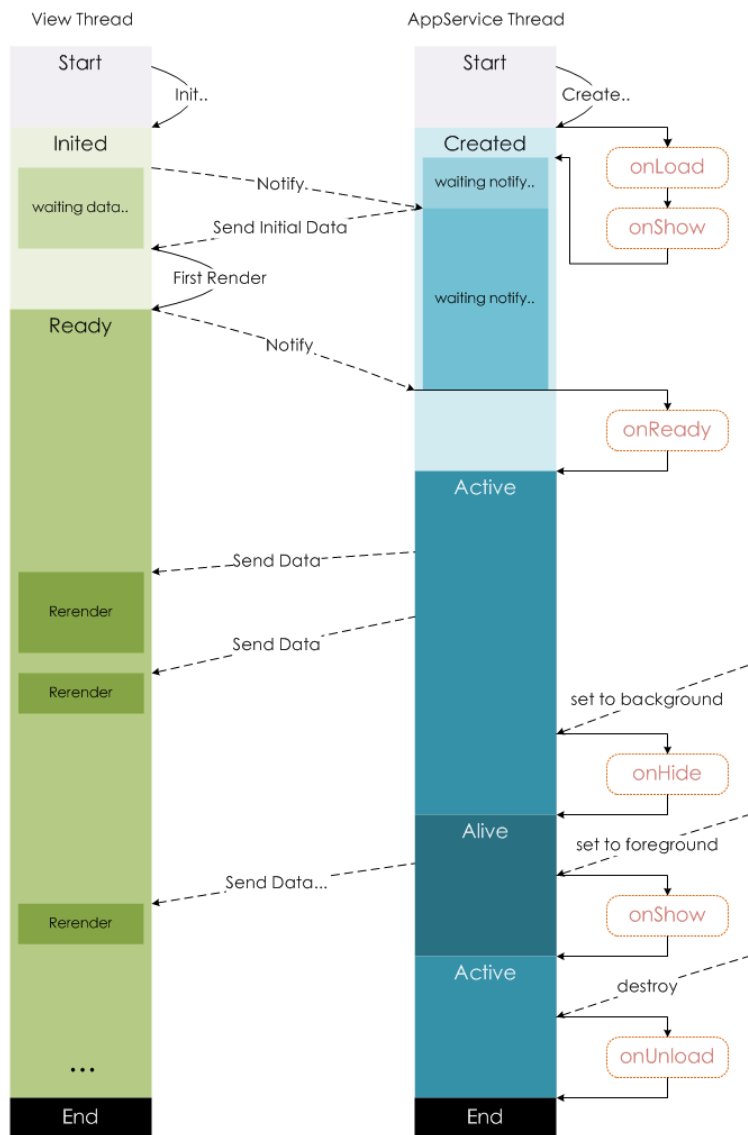


小程序启动详细流程

图1: 小程序启动流程示意图 (用户首次访问或小程序同步更新时, 命中环境预加载)



页面生命周期



逻辑层开发

框架全局文件

- 框架全局文件必须放在项目的根目录中。
- 框架全局文件对所有页面都有效

文件	是否必填	作用
app.js	是	装载小程序逻辑
app.json	是	装载小程序公共设置
app.wxss	否	装载小程序公共样式
project.config.json	是	装载小程序项目个性化配置
sitemap.json	是	配置小程序及其页面是否允许被微信索引

```
project
├── pages
│   ├── index
│   │   ├── index.json  index 页面配置
│   │   ├── index.js   index 页面逻辑
│   │   ├── index.wxml  index 页面结构
│   │   └── index.wxss  index 页面样式表
│   └── log
│       ├── log.json    log 页面配置
│       ├── log.wxml    log 页面逻辑
│       ├── log.js      log 页面结构
│       └── log.wxss    log 页面样式表
├── app.js             小程序逻辑
├── app.json           小程序公共设置
└── app.wxss           小程序公共样式表
```

框架页面文件

- 微信小程序的框架页面文件都放置在pages文件夹下面
- 一个页面包含多个相同文件名、不同类型的文件

文件类型	是否必填	作用
JS	是	页面逻辑
JSON	否	页面配置
WXML	是	页面结构
WXS	否	小程序脚本语言
WXSS	否	页面样式表



注册页面处理程序

对于小程序中的每个页面，都需要在页面对应的 js 文件中进行注册，指定页面的初始数据、生命周期回调、事件处理函数等。

```
//index.js
Page({
  data: {
    text: "This is page data."
  },
  onLoad: function(options) {    // 页面创建时执行
  },
  onShow: function() {    // 页面出现在前台时执行
  },
  onReady: function() {    // 页面首次渲染完毕时执行
  },
  onHide: function() {    // 页面从前台变为后台时执行
  },
  onUnload: function() {    // 页面销毁时执行
  },
  onPullDownRefresh: function() {    // 触发下拉刷新时执行
  },
  onReachBottom: function() {    // 页面触底时执行
  },
  onShareAppMessage: function () {    // 页面被用户分享时执行
  },
  onPageScroll: function() {    // 页面滚动时执行
  },
  onResize: function() {    // 页面尺寸变化时执行
  },
  onTabItemTap(item) {    // tab 点击时执行
  },
  // 事件响应函数
  viewTap: function() {
    this.setData({
      text: 'Set some data for updating view.'
    }, function() {
      // this is setData callback
    })
  },
  // 自由数据
  customData: {
    hi: 'MINA'
  }
})
```


页面路由

页面栈

框架以栈的形式维护了当前的所有页面。当发生路由切换的时候，页面栈的表现如下：

路由方式	页面栈表现
初始化	新页面入栈
打开新页面	新页面入栈
页面重定向	当前页面出栈，新页面入栈
页面返回	页面不断出栈，直到目标返回页
Tab 切换	页面全部出栈，只留下新的 Tab 页面
重加载	页面全部出栈，只留下新的页面

路由方式

对于路由的触发方式以及页面生命周期函数如下：

路由方式	触发时机	路由前页面	路由后页面
初始化	小程序打开的第一个页面		onLoad, onShow
打开新页面	调用 API wx.navigateTo 使用组件 <code><navigator open-type="navigateTo"/></code>	onHide	onLoad, onShow
页面重定向	调用 API wx.redirectTo 使用组件 <code><navigator open-type="redirectTo"/></code>	onUnload	onLoad, onShow
页面返回	调用 API wx.navigateBack 使用组件 <code><navigator open-type="navigateBack"></code> 用户按左上角返回按钮	onUnload	onShow
Tab 切换	调用 API wx.switchTab 使用组件 <code><navigator open-type="switchTab"/></code> 用户切换 Tab		各种情况请参考下表
重启动	调用 API wx.reLaunch 使用组件 <code><navigator open-type="reLaunch"/></code>	onUnload	onLoad, onShow

API

小程序开发框架提供丰富的微信原生 API，可以方便的调起微信提供的能力，如获取用户信息，本地存储，支付功能等。

详细介绍请参考 API 文档。

<https://developers.weixin.qq.com/miniprogram/dev/api/>

- › 基础
- › 路由
- › 跳转
- › 转发
- › 界面
- › 网络
- › 支付
- › 数据缓存
- › 数据分析
- › XR-FRAME Beta
- › 画布
- › 媒体
- › 位置
- › 文件
- › 开放接口
- › 设备
- › AI
- › Worker
- › WXML
- › 第三方平台
- › 广告

API类型

- 事件监听 API
 - 约定以 **on 开头** 的 API 用来监听某个事件是否触发，如：wx.onCompassChange等
- 同步 API
 - 约定以 **Sync 结尾** 的 API 都是同步 API（并非全部），如 wx.setStorageSync等
- 异步 API
 - **大多数** API 都是异步 API，如 wx.request 等，通常都接受一个 Object 类型的参数，执行结果通过Object 类型的参数中传入的**回调函数**获取。
- 异步 API 返回 Promise
 - 当接口参数 Object 对象中不包含 success/fail/complete 时将默认返回 **Promise**，否则仍按回调方式执行，无返回值。
- 云开发 API
 - 开通并使用微信云开发，即可使用云开发API，在小程序端直接调用**服务端的云函数**。

小程序框架-视图层

数据绑定

事件

视图层 View

- 框架的视图层由 **WXML** 与 **WXSS** 编写，由**组件**来进行展示。
- 将逻辑层的**数据**反映成视图，同时将视图层的**事件**发送给逻辑层。
- WXML(WeiXin Markup language) 用于描述页面的结构。
- **WXS**(WeiXin Script) 是小程序的一套脚本语言，结合 WXML，可以构建出页面的结构。
- WXSS(WeiXin Style Sheet) 用于描述页面的样式。
- **组件**(Component)是视图的基本组成单元。

视图层 - WXML

数据绑定

列表渲染

条件渲染

WXML

- WXML（WeiXin Markup Language）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。
- WXML相当于web前端页面语言HTML，用来描述整个页面的结构。
- 要完整了解 WXML 语法，请参考WXML 语法参考文档。

<https://developers.weixin.qq.com/miniprogram/dev/reference/wxml/>

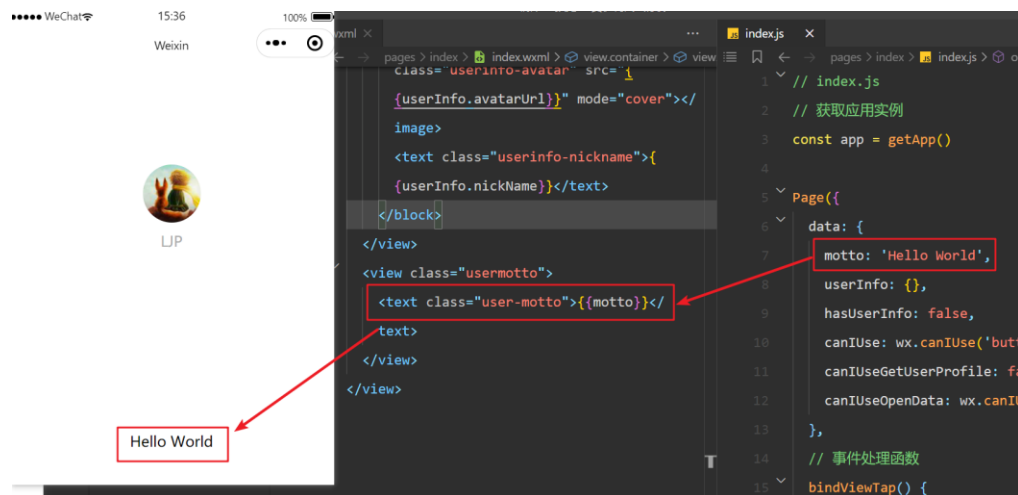
- WXML 的能力
 - 数据绑定
 - 列表渲染
 - 条件渲染
 - 制作模板

WXML - 数据绑定

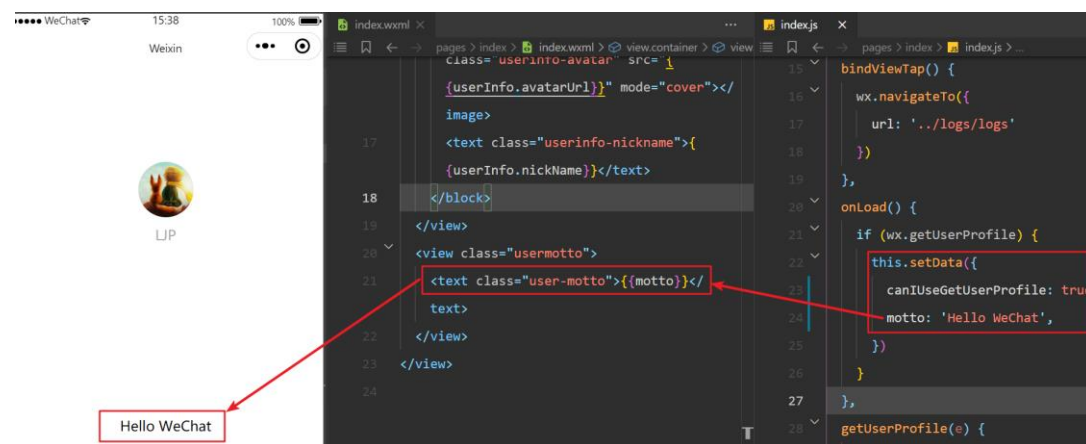
数据绑定

- 小程序中使用 WXML 语言所提供的数据绑定功能，来动态改变页面内容
- 视图层.wxml页面中的动态数据均来自对应逻辑层.js文件的data对象
- 如何将data中的数据“填充”到页面中？ -》2种办法

初始化时的在page的构造函数中传入data对象完成数据绑定



运行过程中使用setData函数动态更新data数据
*直接修改data变量无效，会造成数据不一致，
需要调用setData才会发送数据到页面并触发页面重新渲染



数据绑定作用

- 数据绑定使用 Mustache 语法（双大括号）将变量包起来，可以作用于4处：
内容、组件属性、控制属性、关键字

内容

```
<view> {{ message }} </view>
```

```
Page({  
  data: {  
    message: 'Hello MINA!'  
  }  
})
```

组件属性(需要在双引号之内)

```
<view id="item-{{id}}"> </view>
```

```
Page({  
  data: {  
    id: 0  
  }  
})
```

控制属性(需要在双引号之内)

```
<view wx:if="{{condition}}"> </view>
```

```
Page({  
  data: {  
    condition: true  
  }  
})
```

关键字(需要在双引号之内)

true : boolean 类型的 true, 代表真值。

false : boolean 类型的 false, 代表假值。

```
<checkbox checked="{{false}}"> </checkbox>
```

不要直接写 checked="false",
其计算结果是一个字符串, 转
成boolean类型后代表真值。

WXML - 渲染控制

条件渲染

- wx:if判断单个组件

在微信小程序框架里，使用 `wx:if="{{condition}}"` 来判断是否需要渲染该代码块。

```
<view wx:if="{{condition}}"> True </view>
```

使用wx:elif和wx:else来添加一个else块。

1. `<view wx:if="{{length > 5}}"> 1 </view>`
2. `<view wx:elif="{{length > 2}}"> 2 </view>`
3. `<view wx:else> 3 </view>`

条件渲染

- block wx:if判断多个组件

wx:if是一个控制属性，需要将它添加到一个标签上。但是如果想要一次性判断多个组件标签，就可以使用一个<block/>标签将多个组件包装起来，并在上面使用wx:if控制属性。

1. `<block wx:if="{{true}}">`
2. `<view> view1 </view>`
3. `<view> view2 </view>`
4. `</block>`

<block/>并不是一个组件，它仅仅是一个包装元素，不会在页面中做任何渲染，只接受控制属性。

列表渲染

- wx:for列表渲染单个组件

在组件上使用wx:for控制属性绑定一个数组，即可使用数组中各项的数据重复渲染该组件。数组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。

```
1. <view wx:for="{{array}}">
2.   {{index}}: {{item.message}}
3. </view>
4. Page({
5.   data: {
6.     array: [{
7.       message: 'foo',
8.     }, { message: 'bar'  }]
9.   }
10. })
```

使用wx:for-item可以指定数组当前元素的变量名，使用 wx:for-index 可以指定数组当前下标的变量名。

```
1. <view wx:for="{{array}}" wx:for-index="idx" wx:for-item="itemName">
2.   {{idx}}: {{itemName.message}}
3. </view>
```

列表渲染

- block wx:for列表渲染多个组件

wx:for应用在某一个组件上，如果想渲染一个包含多节点的结构块，wx:for需要应用在<block/>标签上。

1. `<block wx:for="{{[1, 2, 3]]}">`
2. `<view> {{index}}: </view>`
3. `<view> {{item}} </view>`
4. `</block>`

列表渲染

- wx:key指定唯一标识符

如果列表中项目的位置会动态改变或者有新的项目添加到列表中，并且希望列表中的项目保持自己的特征和状态（如中的输入内容，<switch/>的选中状态），需要使用wx:key来指定列表中项目的唯一的标识符。

wx:key的值以如下两种形式提供。

- ❑ 字符串，代表在 for 循环的 array 中 item 的某个 property，该 property 的值需要是列表中唯一的字符串或数字，且不能动态改变。
- ❑ 保留关键字 *this 代表在 for 循环中的 item 本身，这种表示需要 item 本身是一个唯一的字符串或者数字。

```
1. <switch wx:for="{{objectArray}}" wx:key="unique" style="display: block;"> {{item.id}} </switch>
2. Page({
3.   data: {
4.     objectArray: [ {id: 3, unique: 'unique_3'},
5.                     {id: 2, unique: 'unique_2'},
6.                     {id: 1, unique: 'unique_1'},
7.                     {id: 0, unique: 'unique_0'}, ]
8.   }
9. }
```

注意：如不提供wx:key，会报一个warning。如果明确知道该列表是静态的，或者不必关注其顺序，可以选择忽略。

WXML - 模板template

定义模板

在<template/>内定义代码片段，使用name属性作为模板的名字。

1. `<template name="msgItem">`
2. `<view>`
3. `<text> {{index}}: {{msg}} </text>`
4. `<text>Time: {{time}} </text>`
5. `</view>`
6. `</template>`

使用模板

- 在WXML文件里，使用is属性声明需要使用的模板，然后将模板所需要的 data 传入。

```
1. <template is="msgItem" data="{{item}}"/>
2. Page({
3.   data: {
4.     item: {
5.       index: 0,
6.       msg: 'this is a template',
7.       time: '2016-09-15'
8.     }
9.   }
10. })
```

- is属性可以使用三元运算语法，来动态决定具体需要渲染哪个模板。

```
1. <template name="odd">
2.   <view> odd </view>
3. </template>
4. <template name="even">
5.   <view> even </view>
6. </template>
7. <block wx:for="{{[1, 2, 3, 4, 5]]}">
8.   <template is="{{item % 2 == 0 ? 'even' : 'odd'}}"/>
9. </block>
```

视图层 - WXSS

WXSS

- WXSS (WeiXin Style Sheets)是一套样式语言，用于描述 WXML 的组件样式，也就是视觉上的效果。WXSS 用来决定 WXML 的组件应该怎么显示。
- WXSS文件组成
 - 项目公共样式：根目录中的app.wxss为项目公共样式，它会被注入到小程序的每个页面。
 - 页面样式：与app.json注册过的页面同名且位置同级的WXSS文件。
 - 其它样式：其它样式可以被项目公共样式和页面样式引用
- 为了适应广大的前端开发者，WXSS 具有 CSS 大部分特性。同时为了更适合开发微信小程序，WXSS 对 CSS 进行了扩充以及修改。与 CSS 相比，WXSS 扩展的特性有：
 - 尺寸单位
 - 样式导入

WXSS 样式

- 项目公共样式：根目录中的app.wxss为项目公共样式，它会被注入到小程序的每个页面。
- 页面样式：与app.json注册过的页面同名且位置同级的WXSS文件。
- 其它样式：其它样式可以被项目公共样式和页面样式引用

尺寸单位 - rpx

- 在WXSS中，引入了rpx（responsive pixel）尺寸单位。可适配不同宽度的屏幕，rpx可以根据屏幕宽度进行自适应。规定屏幕宽为750rpx。
 - 如在 iPhone 6上，屏幕宽度为375px，共有750个物理像素，则 $750\text{rpx} = 375\text{px} = 750$ 物理像素， $1\text{rpx} = 0.5\text{px} = 1$ 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	$1\text{rpx} = 0.42\text{px}$	$1\text{px} = 2.34\text{rpx}$
iPhone6	$1\text{rpx} = 0.5\text{px}$	$1\text{px} = 2\text{rpx}$
iPhone6 Plus	$1\text{rpx} = 0.552\text{px}$	$1\text{px} = 1.81\text{rpx}$

样式导入

- 使用 @import 语句可以导入外联样式表。
 - @import 后跟需要导入的外联样式表的相对路径。用英文分号 (;) 表示语句结束。

```
1. /** common.wxss **/  
2. .small-p {  
3.   padding:5px;  
4. }  
  
5. /** app.wxss **/  
6. @import "common.wxss";  
7. .middle-p {  
8.   padding:15px;  
9. }
```

上述代码表示在app.wxss文件里，将common.wxss文件样式引入使用。

选择器

选择器	样例	样例描述
.class	.intro	选择所有拥有class="intro" 的组件
#id	#firstname	选择拥有id="firstname" 的组件
element	view	选择所有view 组件
element, element	view, checkbox	选择所有view组件和所有checkbox组件
::after	view::after	在view组件后边插入内容
::before	view::before	在view组件前边插入内容

! important	style=""	#id	.class	element	选择器优先级
∞	1000	100	10	1	

官方样式库 - WeUI

- 为了减轻开发者样式开发的工作量，小程序提供了WeUI.wxss基础样式库。
- **WeUI**是一套与微信原生视觉体验一致的基础样式库，由微信官方设计团队为微信内网页和微信小程序量身设计，令用户的使用感知更加统一。包含button、cell、dialog、progress、toast、article、actionsheet、icon等各式原生。
- 具体使用文档可参考：
<https://github.com/Tencent/weui-wxss>



事件系统

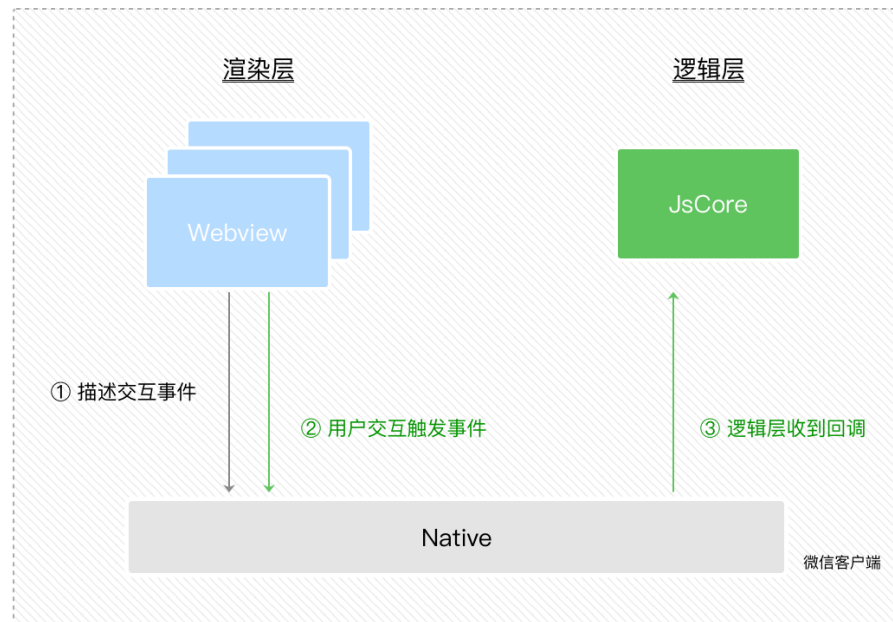
事件

- 什么是事件

- 事件是视图层到逻辑层的通讯方式。
- 事件可以将用户的行为反馈到逻辑层进行处理。
- 事件可以绑定在组件上，当达到触发事件，就会执行逻辑层中对应的事件处理函数。
- 事件对象可以携带额外信息，如 id, dataset, touches。

- 事件的使用方式

- 在组件中绑定一个事件处理函数。



事件类型和事件对象

常见的事件类型

```
<!-- page.wxml -->
<view id="outer" catchtap="handleTap">
  <view id="inner">点击我</view>
</view>
```

触发事件

```
// page.js
Page({
  handleTap: function(evt) {
    // 当点击inner节点时
    // evt.target 是inner view组件
    // evt.currentTarget 是绑定了handleTap的outer view组件
    // evt.type == "tap"
    // evt.timeStamp == 1542
    // evt.detail == {x: 270, y: 63}
    // evt.touches == [{identifier: 0, pageX: 270, pageY: 63, clientX: 270, clientY: 63}]
    // evt.changedTouches == [{identifier: 0, pageX: 270, pageY: 63, clientX: 270, clientY: 63}]
  }
})
```

事件处理函数

事件对象

类型	触发条件
touchstart	手指触摸动作开始
touchmove	手指触摸后移动
touchcancel	手指触摸动作被打断，如来电提醒，弹窗
touchend	手指触摸动作结束
tap	手指触摸后马上离开
longpress	手指触摸后，超过350ms再离开，如果指定了事件回调函数并触发了这个事件，tap事件将不被触发
longtap	手指触摸后，超过350ms再离开（推荐使用longpress事件代替）
transitionend	会在 WXSS transition 或 wx.createAnimation 动画结束后触发
animationstart	会在一个 WXSS animation 动画开始时触发
animationiteration	会在一个 WXSS animation 一次迭代结束时触发
animationend	会在一个 WXSS animation 动画完成时触发

事件属性

当事件回调触发的时候，会收到一个事件对象对象的详细属性如下

属性	类型	说明
type	String	事件类型
timeStamp	Integer	页面打开到触发事件所经过的毫秒数
target	Object	触发事件的组件的一些属性值集合
currentTarget	Object	当前组件的一些属性值集合
detail	Object	额外的信息
touches	Array	触摸事件，当前停留在屏幕中的触摸点信息的数组
changedTouches	Array	触摸事件，当前变化的触摸点信息的数组

target和currentTarget对象的详细参数如下

属性	类型	说明
id	String	当前组件的id
tagName	String	当前组件的类型
dataset	Object	当前组件上由data-开头的自定义属性组成的集合

*currentTarget为当前事件所绑定的组件，而target则是触发该事件的源头组件

事件自定义属性

在 JavaScript 自定义 data 属性的属性名与没有 data- 前缀的 HTML 属性相同，并且在移除单个破折号（-）后，大写之后的字母以获得属性的“驼峰”命名。

例如，

html 中的 `data-good-id` =》 JavaScript 中的 `goodId`

JavaScript 中读取自定义属性 `data-xxx-yyy`:
`event.currentTarget.dataset.xxxYyy`

```
<view class="section goods-list">
```

```
<block wx:for="{{goodsList}}" wx:for-item="good" wx:key="goodId">
```

```
<view wx:if="{{good !== null}}" class="goods-card" data-good-id="{{
```

```
{good.goodId}}" bindtap="onClickGoodDetail">
```

```
76 //点击了商品卡
```

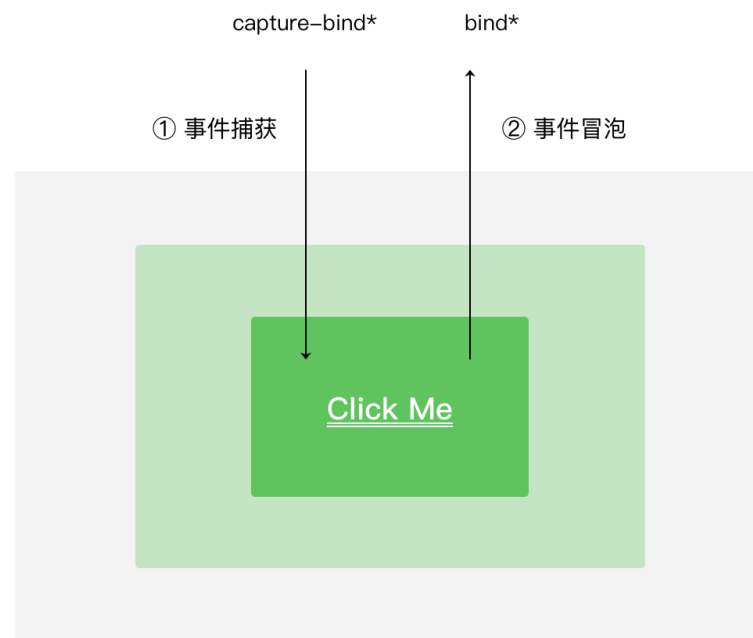
```
77 ✓ onClickGoodDetail: function(event) {
```

```
78   let goodId = event.currentTarget.dataset.goodId;
```

```
79   // console.log(event);
```

事件绑定与冒泡捕获

- 事件绑定的写法和组件属性一致，以`key="value"`的形式，其中：
 - key以bind或者catch开头，然后跟上事件的类型，如bindtap、catchtouchstart。自基础库版本1.5.0起，bind和catch后可以紧跟一个冒号，其含义不变，如bind:tap、catch:touchstart。同时bind和catch前还可以加上capture-来表示捕获阶段。
- value是一个字符串，需要在对应的页面Page构造器中定义同名的函数，否则触发事件时在控制台会有报错信息。
- bind和capture-bind的含义分别代表事件的冒泡阶段和捕获阶段
- 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。



小程序组件

基础组件

- 框架为开发者提供了一系列基础组件，开发者可以通过组合这些基础组件进行快速开发。

- 详细介绍请参考组件文档。

<https://developers.weixin.qq.com/miniprogram/dev/component/>

- 什么是组件：

- 组件是视图层的**基本组成单元**。
 - 组件自带一些功能与微信风格一致的样式。
 - 一个组件通常包括 **开始标签** 和 **结束标签**，**属性** 用来修饰这个组件，**内容** 在两个标签之内。

- `<tagname property="value">` 与 `</tagname>` 通过“-” 连接

Content goes here ...

`</tagname>`

› 视图容器

› 基础内容

› 表单组件

› XR-FRAME Beta

› 导航

› 媒体组件

› 地图

› 画布

› 开放能力

› 原生组件说明

› 无障碍访问

› 导航栏

› 页面属性配置节点

组件属性数据类型

1.Boolean:

布尔值，组件写上该属性，不管该属性等于什么，其值都为true，只有组件上没有写该属性时，属性值才为false。如果属性值为变量，变量的值会被转换为Boolean。

2.Number: 数字

3.String: 字符串

8.Any: 任意属性

7.Eventhandler: 事件处理函数名

6.Object: 对象

5.Array: 数组

4.Value : 是一个字符串，需要在对应的 Page 中定义同名的函数，否则当触发事件的时候会报错。



公共/特殊组件属性

公共属性（所有组件都有以下属性）

属性名	类型	描述	注解
id	String	组件的唯一标示	保持整个页面唯一
class	String	组件的样式类	在对应的 WXSS 中定义的样式类
style	String	组件的内联样式	可以动态设置的内联样式
hidden	Boolean	组件是否显示	所有组件默认显示
data-*	Any	自定义属性	组件上触发的事件时，会发送给事件处理函数
bind* / catch*	EventHandler	组件的事件	详见 事件

特殊属性

所有组件都有各自定义的属性，可以对该组件的功能或样式进行修饰，请参考各个组件的定义。