第6章：小程序入门

本章我们开始小程序的学习，随着各个知名大厂的加入，小程序也越来越热门，了解和学习小程序变得十分必要。

6.1 认识小程序

我们先了解下什么是小程序，有一个整体的概念。

6.1.1 小程序简介

微信小程序是腾讯于 2017 年 1 月 9 日推出的一种不需要下载安装即可在微信平台上使用的应用。引用一句张小龙的话：小程序是一种不需要下载安装即可使用的应用，它实现了应用「触手可及」的梦想，用户扫一扫或搜一下即可打开应用。也体现了「用完即走」的理念，用户不用关心是否安装太多应用的问题。应用将无处不在，随时可用，但又无需安装卸载。

我们可以在微信聊天列表中下拉，或者在 `发现` -> `小程序` 中都找到小程序。相对于原生 App 来说小程序的用户可便捷地获取服务，不用安装，即开即用，用完就走。省流量，省安装时间，不占用桌面；小程序还可以跨平台（同时支持 iOS 和 Android），降低开发成本，推广更容易更简单。

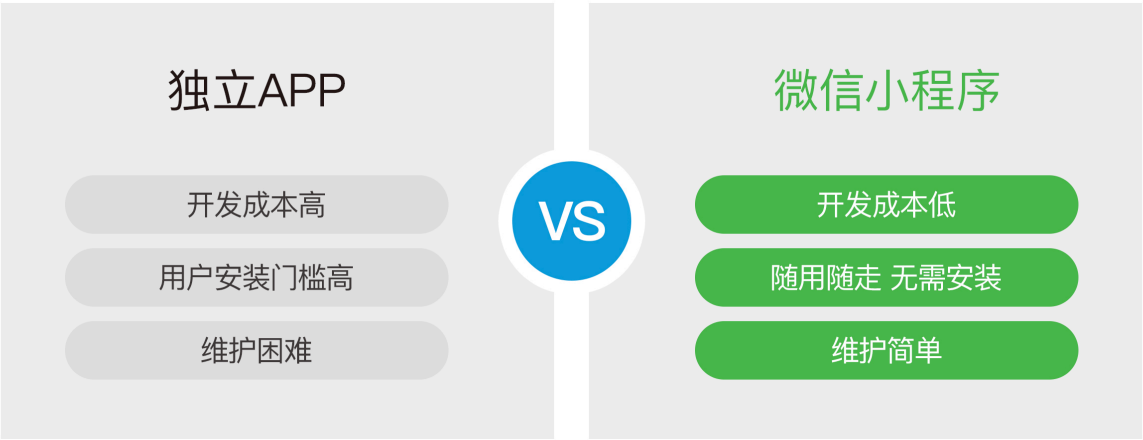

图6-1 App 对比微信小程序图

简单来说微信小程序就是依托微信公众平台开发并在微信上使用的应用，用户只需要下载微信即可使用小程序。同时微信还为微信小程序提供了一套基础组件库以及 API，可以满足开发的基础开发需求，从而实现简单的快速开发。

6.1.2 开发前的准备

1. 注册小程序开发者

首先我们需要注册一个微信小程序开发者账号，在微信公众平台（https://mp.weixin.qq.com）点击右上角的 `立即注册`，然后选择 `小程序` 填写相关信息即可，最后在 `信息登记` 部分需要填写相关信息。（注意：本书推荐 `主体类型` 选择 `个人` 来进行学习）

2. 下载微信开发者工具

注册完微信小程序开发者账号后，我们需要登录小程序管理平台，在首页我们可以看到 `小程序发布流程`。我们需要根据提示先完善小程序信息，然后下载并安装微信开发者工具。

图6-2 小程序发布流程图

下载并安装好微信开发者工具后的效果图如下：

图6-3 微信开发者工具效果图

6.1.3 Hello 小程序

接下来开始创建我们的第一个微信小程序项目，我们在开发者工具中选择 `小程序项目`，然后选择加号新建一个小程序项目，选择项目目录、填写 AppID、项目名称。

图6-4 创建小程序项目效果图

AppID 我们可以在 `小程序管理平台` 中的 `设置` -> `开发设置` 中找到。

图6-5 开发设置效果图

创建好项目之后就能看到开发工具的全貌了，到这里我们已经完成了第一个项目 `Hello World`。

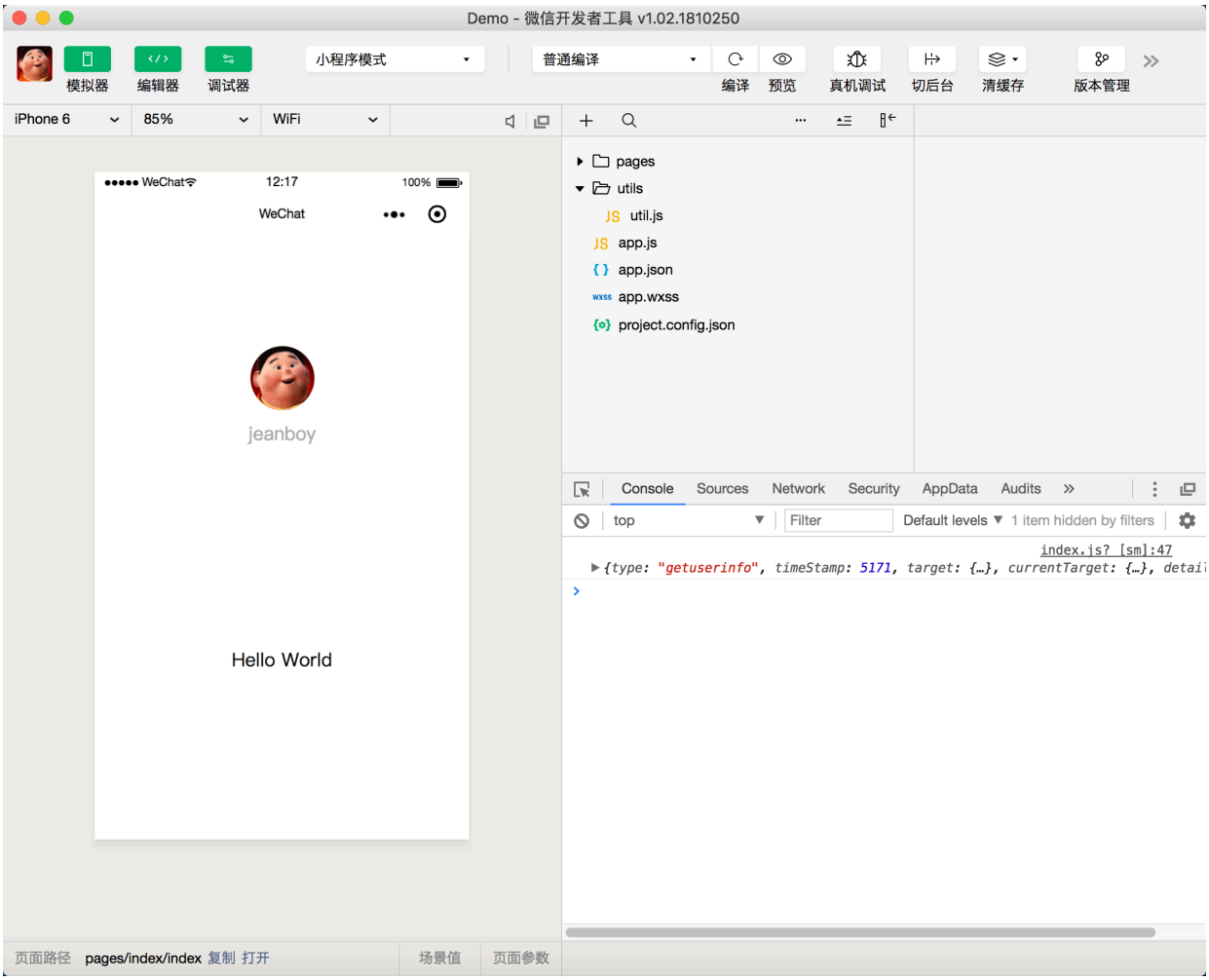

图6-6 开发工具项目效果图

6.1.4 代码构成

在上一节中，我们使用开发者工具创建了一个项目，目录结构如下：

```
|-ProjectName
    |-pages
        |-home
            |-home.js
            |-home.json
            |-home.wxml
            |-home.wxss
    |-utils
        |-util.js
    |-app.js
    |-app.wxss
    |-app.json
    |-project.config.json
```

通过项目可以看到里面生成了不同的文件，总共有 4 种类型的文件。

1. `.json` 后缀的 `JSON` 配置文件
2. `.wxml` 后缀的 `WXML` 模板文件，可以理解为 HTML 文件
3. `.wxss` 后缀的 `WXSS` 样式文件，可以理解为 CSS 文件
4. `.js` 后缀的 `JS` 脚本逻辑文件

我们可以看到在项目的根目录有一个 `app.json` 和 `project.config.json`，此外在 `pages/home` 目录下还有一个 `home.json`。其中：

1. `app.json` 是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。
2. `project.config.json` 是当前项目对于开发工具的配置，在工具上做的任何配置都会写入到这个文件。
3. `home.json` 是每个独立页面的配置，包括了页面导航栏的配置、是否允许下拉刷新等。

看过前面章节的读者应该知道，网页编程采用的是 HTML + CSS + JS 这样的组合，其中 HTML 是用来描述当前这个页面的结构，CSS 用来描述页面的样式，JS 通常是用来处理页面交互的。

同样道理，在小程序中也有同样的角色，其中 `WXML` 充当的就是类似 `HTML` 的角色，和 `HTML` 非常相似，`WXML` 也是由标签、属性等等构成。但是 `WXML` 也有自己独特的特性，这些差异我们将在后面的章节详细介绍。

`WXSS` 充当的是类似 `CSS` 的角色，`WXSS` 具有 `CSS` 大部分的特性，小程序在 `WXSS` 也做了一些扩充和修改。更详细的差别我们将在后面的部分为大家展示。在 `WXSS` 中推荐使用的尺寸单位是 `rpx` （responsive pixel）：可以根据屏幕宽度进行自适应。规定屏幕宽为 750rpx。如在 iPhone6 上，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。

小程序中的 `JS` 与网页编程中的 `JS` 角色相同， `JS` 主要处理用户页面的交互，响应用户的点击。此外你还可以在 `JS` 中调用小程序提供的丰富的 API，利用这些 API 可以很方便的调起微信提供的能力，例如获取用户信息、本地存储、微信支付等。

6.1.5 小程序的能力

小程序的特点：

1. 微信客户端在打开小程序之前，会把整个小程序的代码包下载到本地。
2. 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。
3. 为了让开发者可以很方便的调起微信提供的能力，例如获取用户信息、微信支付等等，小程序提供了很多 API 给开发者去使用。
4. 服务端接口全部使用 HTTPS，确保传输中安全。
5. View 层和逻辑层分离，通过数据驱动去更新视图。

小程序的不足：

1. 小程序仍然是使用 WebView 渲染，并非原生渲染。
2. 需要独立开发，不能在非微信环境运行。
3. 依赖浏览器环境的 js 库不能使用，因为是 JSCore 执行的，没有 window、document 对象。
4. WXSS 中无法使用本地图片、字体等。
5. WXSS 转化成 JS 而不是 CSS，为了兼容 rpx 和使用原生组件。
6. 小程序无法打开原生页面和网页（可使用 WebView 组件承载），无法拉起 App。

6.2 小程序框架

这一章节我们来了解下小程序的整体框架，了解下小程序的工作流程。

6.2.1 小程序配置

在小程序项目的根目录下有一个 `app.json`，这是小程序项目的全局配置文件，主要配置页面的路径、窗口样式、设置多 tab 等。我们来看下文件里面的内容：

```
{
    "pages": [//所有的页面都需要在这里配置
        "pages/home/home"
    ],
    "window": {//全局页面顶部样式
        "backgroundTextStyle": "dark",//下拉 loading 的样式
        "navigationBarBackgroundColor": "white",//导航栏背景颜色
        "navigationBarTextStyle": "black",//导航栏标题颜色
        "navigationBarTitleText": "Demo",//默认页面标题
        "enablePullDownRefresh": false//是否可以下拉刷新
    },
```

```
    "tabBar": {//底部 tab 的样式配置
        "color": "#989898",//tab 名称未选中时的颜色
        "selectedColor": "#2F96F9",//tab 名称未选中时的颜色
        "backgroundColor": "#FFFFFF",//tab 背景颜色
        "borderStyle": "white",//tab 顶部边框颜色
        "list": [
            {
                "pagePath": "pages/home/home",//tab 的页面路径
                "iconPath": "",//tab 未选中时的图标
                "selectedIconPath": "",//tab 选中时的图标
                "text": "tab1"//tab 名称
            }
        ]
    }
}
```

在每个页面文件夹下都有一个与页面同名的 JSON 文件，例如 home.json，我们来看下文件里面的内容：

```
{
    "navigationBarBackgroundColor": "#ffffff",//导航栏背景颜色
    "navigationBarTextStyle": "black",//导航栏标题颜色
    "navigationBarTitleText": "首页",//导航栏标题文字内容
    "backgroundColor": "#eeeeee",//窗口的背景色
    "backgroundTextStyle": "light"//下拉 loading 的样式
}
```

这里列出了常用的配置并备注了功能，更详细的配置可以去查看微信小程序的开发文档，在 `小程序开发` -> `框架` -> `配置` -> `全局配置` 中可以找到。

6.2.2 小程序生命周期

我们在项目的根目录还可以看到 `app.js`，这是小程序的入口，管理所有页面和全局数据，以及提供生命周期方法。它也是一个构造方法，生成 App 实例。一个小程序就是一个 App 实例。

在 `app.js` 提供了一些方法，用来处理小程序的生命周期：

```
App({
    onLaunch: function (options) {
        //当小程序初始化完成时触发，全局只触发一次
    },
    onShow: function () {
        //当小程序启动，或从后台进入前台显示时触发
    },
    onHide: function () {
```

```
        //当小程序从前台进入后台时触发
    },
    onError: function () {
        //当小程序发生 js 错误时触发
    },
    globalData: {//全局数据
    }
})
```

前台与后台定义： 当用户点击左上角关闭，或者按了设备 Home 键离开微信，小程序并没有直接销毁，而是进入了后台；当再次进入微信或再次打开小程序，又会从后台进入前台。需要注意的是：只有当小程序进入后台一定时间，或者系统资源占用过高，才会被真正的销毁。

我们可以使用 `Page(Object)` 函数来注册一个页面。接收的是 `Object` 类型参数，其指定页面的初始数据、生命周期回调、事件处理函数等。比如我们在 home 目录下看到的 `home.js` 的作用就是注册一个页面。

```
Page({
    data: {//页面第一次渲染使用的初始数据
    },
    onLoad: function (options) {
        //页面加载时触发，一个页面只会调用一次
    },
    onShow: function () {
        //页面显示/切入前台时触发
    },
    onReady: function () {
        //页面初次渲染完成时触发，一个页面只会调用一次，代表页面已经准备妥当，可以和视图层进行交互
    },
    onHide: function () {
        //页面隐藏/切入后台时触发
    },
    onUnload: function () {
        //页面卸载时触发
    },
})
```

在 Page 中我们可以使用 `getApp()` 函数来获取到小程序 App 的实例，比如我们获取全局数据就可以这样获取 `getApp().globalData`。

我们来通过一张图看一下 Page 实例的生命周期：

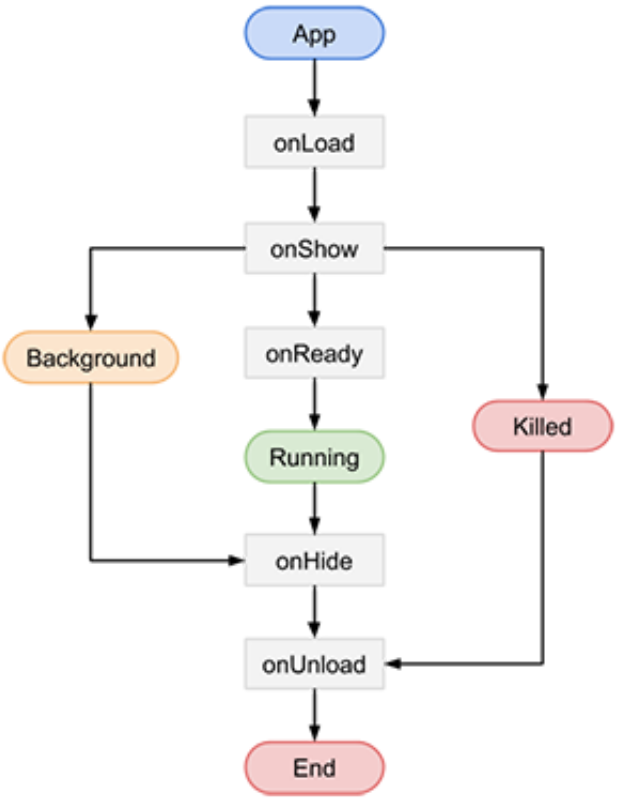

图6-7 页面生命周期图

6.2.3 路由

在小程序中所有页面的路由全部由框架进行管理的，框架以栈的形式维护了当前的所有页面。

我们可以在 Page 页面中使用 `getCurrentPages()` 函数，用于获取当前页面栈的实例，该函数以数组形式按栈的顺序给出，第一个元素为首页，最后一个元素为当前页面。

1. 打开新页面：调用 API `wx.navigateTo` 或使用组件 `<navigator open-type="navigateTo"/>`，作用是保留当前页面，跳转到应用内的某个指定页面。
2. 页面重定向：调用 API `wx.redirectTo` 或使用组件 `<navigator open-type="redirectTo"/>`，作用是关闭当前页面，跳转到应用内的某个指定页面。
3. 页面返回：调用 API `wx.navigateBack` 或使用组件 `<navigator open-type="navigateBack">` 或用户按左上角返回按钮，作用是关闭当前页面，返回上一级或多级页面。
4. Tab 切换：调用 API `wx.switchTab` 或使用组件 `<navigator open-type="switchTab"/>` 或用户切换 Tab，作用是跳转到指定 tabBar 页面，并关闭其他所有非 tabBar 页面。
5. 重启动：调用 API `wx.reLaunch` 或使用组件 `<navigator open-type="reLaunch"/>`，作用是关闭当前所有页面，打开应用内的某个指定页面。

6.2.4 视图层

视图层主要由 WXML 与 WXSS 编写，由组件来进行展示。视图层的主要任务就是展示逻辑层提供的数据，同时将用户操作的事件发送给逻辑层。

1. 数据绑定

数据绑定就是将 Page 中的 data 数据在 WXML 中展示，当我们操作 Page 中的 data 时 WXML 中会动态更新。我们来看下示例：

WXML：

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

Page：

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

示例中 WXML 通过 `{{ message }}` 的方式获取了 Page 中 data 名称为 message 的数据，当我们在 Page 中修改 message 的值时 WXML 中的值也会动态更新。

2. 列表渲染

列表渲染就是我们在展示一组相同数据结构的数据时用到的渲染方式。我们可以在组件上使用 `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。

WXML：

```
<!-- 这里的 index 和 item 是默认的变量名，也就是说使用 wx:for 来渲染都会默认有这两个变量 -->
<view wx:for="{{array}}">
  {{index}}: {{item.message}}
</view>
```

Page：

```
Page({
  data: {
    array: [{
      message: 'foo',
    }, {
      message: 'bar'
    }]
  }
})
```

这里仅做简单的介绍，详情请参考 `微信公众平台` -> `小程序开发` -> `框架` -> `视图层` -> `列表渲染` 。

3. 条件渲染

跟列表渲染的语法很相似，WXML 中还支持 `wx:if` 、 `wx:else` 、 `wx:elif` 等条件渲染方式。

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

4. 模板

WXML 还提供模板（template），可以在模板中定义代码片段，然后在不同的地方调用。

定义模板：

```
<template name="test">
  <view>
    <text> 内容: {{content}} </text>
  </view>
</template>
```

这里使用 `<template/>` 标签的 name 属性定义了一个名为 test 的模板，模板接受的参数为 content。

使用模板：

```
<template is="test" data="{{content:'哈哈'}}"/>
```

模板的使用也很简单，只需要引入模板文件使用 `<template/>` 标签指定模板的名称，通过 data 属性来传入参数即可。

5. 事件

什么是事件：

> 1. 事件是视图层到逻辑层的通讯方式。
> 2. 事件可以将用户的行为反馈到逻辑层进行处理。
> 3. 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。
> 4. 事件对象可以携带额外信息，如 id, dataset, touches。

事件的使用方式：

如 `bindtap` ，当用户点击该组件的时候会在该页面对应的 Page 中找到相应的事件处理函数。

```
<view id="tapTest" data-hi="WeChat" bindtap="tapName"> Click me!
</view>
```

在相应的 Page 定义中写上相应的事件处理函数，参数是 event。

```
Page({
  tapName: function(event) {
    console.log(event)
  }
})
```

可以看到 log 出来的信息大致如下：

```
{
  "type":"tap",
  "timeStamp":895,
  "target": {//点击对象
    "id": "tapTest",//对象 ID
    "dataset":  {//data-hi 的数据
      "hi":"WeChat"
    }
  },
  "currentTarget":  {
    "id": "tapTest",
    "dataset": {
      "hi":"WeChat"
    }
  },
  "detail": {
    "x":53,
    "y":14
  },
  "touches":[{
    "identifier":0,
    "pageX":53,
    "pageY":14,
    "clientX":53,
    "clientY":14
  }],
  "changedTouches":[{
    "identifier":0,
    "pageX":53,
    "pageY":14,
    "clientX":53,
    "clientY":14
  }]
```

```
}
```

当然微信小程序支持的事件还有很多，具体可在 `微信公众平台` -> `小程序开发` -> `框架` -> `视图层` -> `事件` 中找到。

6.2.5 动画

在小程序中，通常可以使用 CSS 来创建简易的界面动画。同时，还可以使用 `wx.createAnimation` 接口来动态创建简易的动画效果。

wx.createAnimation 参考官方文档：https://developers.weixin.qq.com/miniprogram/dev/api/wx.createAnimation.html

6.3 常用组件

这一章节我们来学习下小程序开发项目中常用的组件。

6.3.1 视图容器

1. view

相当于 HTML 中的 `<div>`，我们来看下它的用法：

```
<view>
    <view class="title">横向布局</view>
    <view class="flex-row">
        <view class="a">A</view>
        <view class="b">B</view>
        <view class="c">C</view>
    </view>
    <view class="title">纵向布局</view>
    <view class="flex-column">
        <view class="a">A</view>
        <view class="b">B</view>
        <view class="c">C</view>
    </view>
</view>
```

效果图如下，我们可以看到 HTML 中 `<div>` 可以实现的的效果 `<view>` 都可以实现。

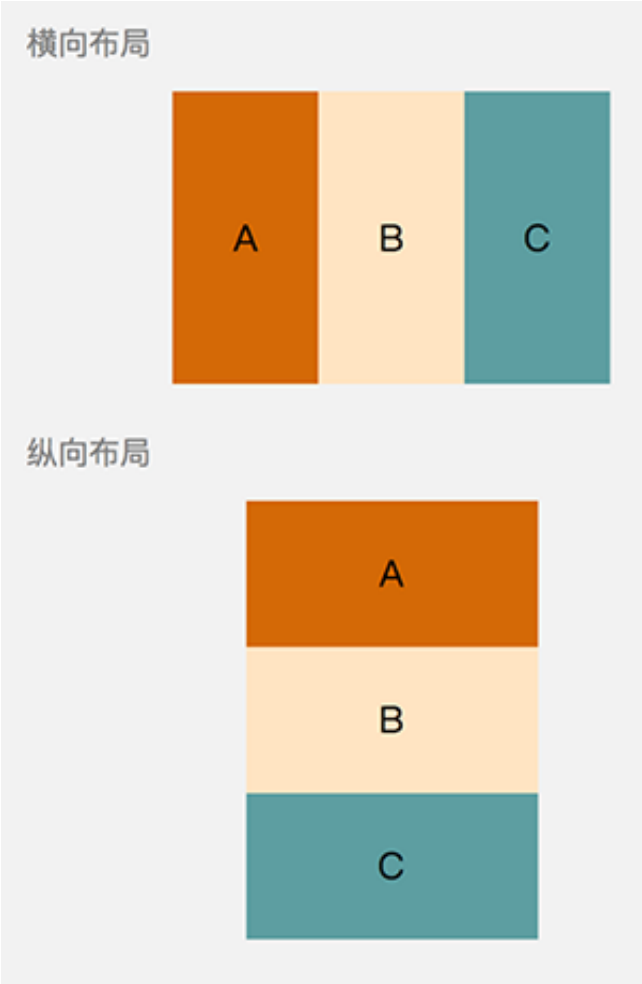

图6-8 view 示例图

2. scroll-view

可滚动的视图区域，也就是可以水平或者垂直滚动的容器。

```
<view>
    <view class="title">横向滚动</view>
    <scroll-view scroll-x class="scroll-view">
        <view class="flex-row">
            <view class="a">A</view>
            <view class="b">B</view>
            <view class="c">C</view>
        </view>
    </scroll-view>
    <view class="title">纵向滚动</view>
    <scroll-view scroll-y class="scroll-view">
        <view class="flex-column">
            <view class="a">A</view>
            <view class="b">B</view>
            <view class="c">C</view>
        </view>
    </scroll-view>
```

```
  </view>
```

`<scroll-view>` 组件可以很方便的做出横向或纵向滚动的效果，常见的导航栏、滑动列表等都可以实现。

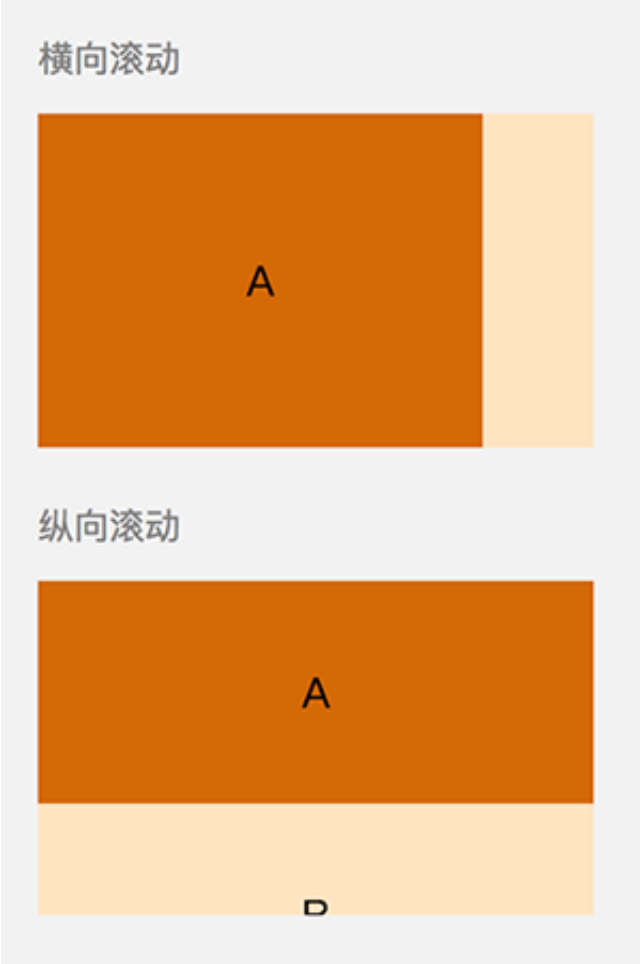

图6-9 scroll-view 示例图

3. swiper

滑块视图容器，也就我们常见的 banner 轮播图效果。

```
<view>
    <swiper indicator-dots="true" autoplay="true" interval="2000"
duration="1000">
        <swiper-item>
            <view class="a" />
        </swiper-item>
        <swiper-item>
            <view class="b" />
        </swiper-item>
        <swiper-item>
            <view class="c" />
        </swiper-item>
    </swiper>
</view>
```

`<swiper>` 组件最常见的使用场景就是轮播图效果了，当然我们也可以使用 swiper 实现引导页等效果。

图6-10 swiper 示例图

6.3.2 基础内容

1. text

文本可以理解为 HTML 中的 `<span>`。

```
<text>{{text}}</text>
```

2. progress

进度条。

```
<progress percent="20" show-info />
<progress percent="40" stroke-width="12" />
<progress percent="60" color="pink" />
<progress percent="80" active />
```

`<progress>` 组件可以帮助我们快速实现进度加载的效果，可以根据需求自定义样式。

图6-11 text、progress 示例图

6.3.3 表单组件

1. button

按钮。

```
<button type="default">default</button>
<button type="primary">primary</button>
<button type="warn">warn</button>
```

`<button>` 组件虽然使用起来很方便，但是当需要自定义样式时却不是很方便，我们可以使用 `<view>` 组件代替。

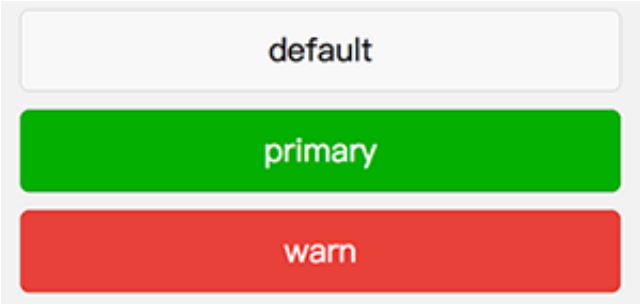

图6-12 button 示例图

2. checkbox

复选框。

```
<checkbox-group bindchange="checkboxChange">
    <label class="checkbox" wx:for="{{checkBoxItems}}"
wx:key="index">
        <checkbox value="{{item.value}}" checked="{{item.checked}}"
/>
        {{item.name}}
    </label>
</checkbox-group>
```

```
Page({
    data: {
        checkBoxItems: [
            { name: 'USA', value: '美国' },
            { name: 'CHN', value: '中国', checked: 'true' },
            { name: 'BRA', value: '巴西' },
            { name: 'JPN', value: '日本' },
            { name: 'ENG', value: '英国' },
            { name: 'TUR', value: '法国' }]
    }
})
```

图6-13 checkbox 示例图

3. radio

单选按钮。

```
<radio-group class="radio-group" bindchange="radioChange">
    <label class="radio" wx:for="{{radioItems}}" wx:key="index">
        <radio value="{{item.value}}" checked="{{item.checked}}" />
        {{item.name}}
    </label>
</radio-group>
```

```
Page({
    data: {
         radioItems: [
            { name: '男', value: '1' },
            { name: '女', value: '2', checked: 'true' }]
    }
})
```

图6-14 radio 示例图

4. input

输入框。

```
<input placeholder="这是一个 input" />
```

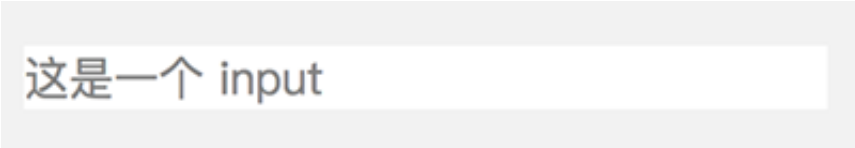

图6-15 input 示例图

5. textarea

多行输入框。

```
<textarea placeholder="这是一个 textarea" />
```

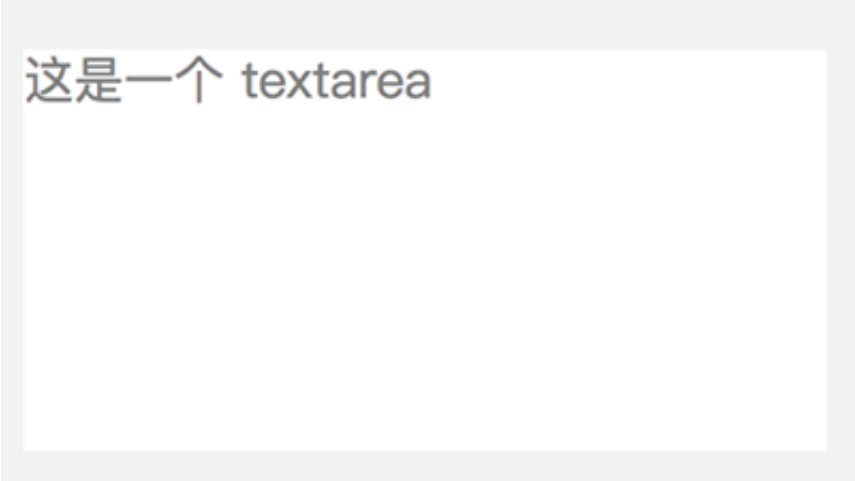

图6-16 textarea 示例图

6. form

表单，就是将组件内的用户输入的 `<switch/>` `<input/>` `<checkbox/>` `<slider/>` `<radio/>` `<picker/>` 的内容提交。

当点击 `<form/>` 表单中 formType 为 submit 的 `<button/>` 组件时，会将表单组件中的 value 值进行提交，需要在表单组件中加上 name 来作为 key。

```
<form bindsubmit="formSubmit" bindreset="formReset">
     <button formType="submit">Submit</button>
     <button formType="reset">Reset</button>
</form>
```

7. slider

滑动选择器。

```
<slider bindchange="sliderChange" step="5" />
```


图6-17 slider 示例图

8. switch

开关选择器。

```
<switch bindchange="switchChange" />
```

图6-18 switch 示例图

9. picker

从底部弹起的滚动选择器。

```
<picker bindchange="bindPickerChange" value="{{index}}" range="
{{array}}">
     <view class="picker">当前选择：{{array[index]}}</view>
</picker>
```

```
Page({
    data: {
        array: ['美国', '中国', '巴西', '日本'],
        index: 1
    }
})
```

图6-19 picker 示例图

6.3.4 媒体组件

1. image

图片。

```
<image src="" mode="scaleToFill"></image>
```

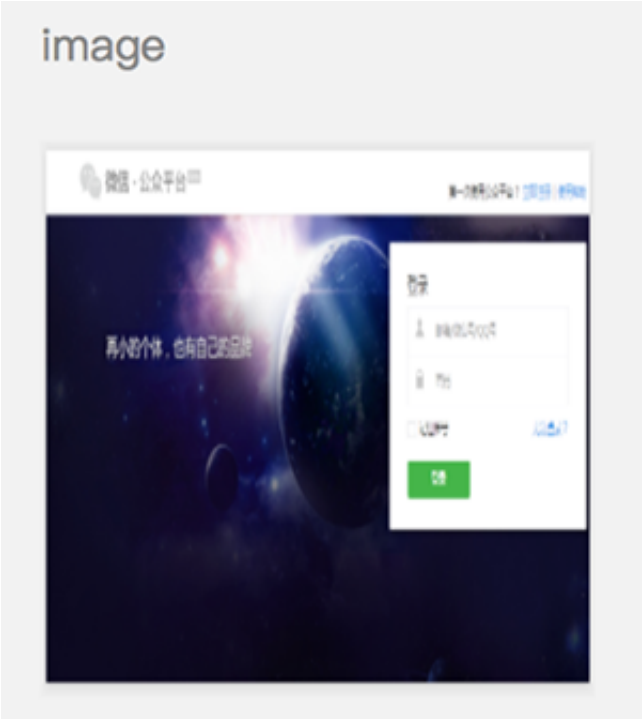

图6-20 image 示例图

我们可以看到图片是变形的，这里与 `mode="scaleToFill"` 有关，常用的 mode 属性如下：

模式	值	说明
缩放	scaleToFill	不保持纵横比缩放图片，使图片的宽高完全拉伸至填满 image 元素
缩放	aspectFit	保持纵横比缩放图片，使图片的长边能完全显示出来。也就是说，可以完整地将图片显示出来。
缩放	aspectFill	保持纵横比缩放图片，只保证图片的短边能完全显示出来。也就是说，图片通常只在水平或垂直方向是完整的，另一个方向将会发生截取。

2. audio

播放音频。

属性名	类型	默认值	说明
id	String		audio 组件的唯一标识符
src	String		要播放音频的资源地址
loop	Boolean	false	是否循环播放
controls	Boolean	false	是否显示默认控件
poster	String		默认控件上的音频封面的图片资源地址，如果 controls 属性值为 false 则设置 poster 无效
name	String	未知音频	默认控件上的音频名字，如果 controls 属性值为 false 则设置 name 无效
author	String	未知作者	默认控件上的作者名字，如果 controls 属性值为 false 则设置 author 无效

```
<audio poster="{{poster}}"
       name="{{name}}"
       author="{{author}}"
       src="{{audioSrc}}"
       id="myAudio"
       controls
       loop></audio>
<button type="primary" bindtap="audioPlay">播放</button>
<button type="primary" bindtap="audioPause">暂停</button>
<button type="primary" bindtap="audio14">设置当前播放时间为14秒</button>
<button type="primary" bindtap="audioStart">回到开头</button>
```

图6-21 audio 示例图

3. video

播放视频。

属性名	类型	默认值	说明
src	String		要播放视频的资源地址
danmu-list	Object Array		弹幕列表
enable-danmu	Boolean	false	是否展示弹幕
danmu-btn	Boolean	false	是否显示弹幕按钮
controls	Boolean	true	是否显示默认播放控件

```
<video id="myVideo"
       src=""
       danmu-list="{{danmuList}}"
       enable-danmu
       danmu-btn
       controls></video>
<button bindtap="bindButtonTap">获取视频</button>
```

图6-22 video 示例图

6.3.5 地图

属性名	类型	默认值	说明
longitude	Number		中心经度
latitude	Number		中心纬度
scale	Number	16	缩放级别，取值范围为5 - 18
controls	Array		控件
bindcontroltap	EventHandle		点击控件时触发，会返回 control 的 id
markers	Array		标记点
bindmarkertap	EventHandle		点击标记点时触发，会返回 marker 的 id
polyline	Array		路线
bindregionchange	EventHandle		视野发生变化时触发
show-location	Boolean		显示带有方向的当前定位点

```
<map id="map"
      longitude="113.324520"
      latitude="23.099994"
      scale="14"
      controls="{{controls}}"
      bindcontroltap="controltap"
      markers="{{markers}}"
      bindmarkertap="markertap"
      polyline="{{polyline}}"
      bindregionchange="regionchange"
      show-location
      style="width: 100%; height: 300px;"></map>
```

图6-23 map 示例图

6.3.6 web-view

展示网页。由于目前不对个人小程序开放这里不做展示。

```
<web-view src="https://jeanboy.cn"></web-view>
```

6.3.7 总结

这里仅仅示例了常用的组件，其他组件的介绍可以在 `微信公众平台` -> `小程序开发` -> `组件` 中找到。

本章节所有的示例代码详见：https://github.com/jeanboydev/Wechat-Demo

6.4 常用 API

这一章节我们来学习下小程序项目开发中常用的 API。

6.4.1 网络

1. 上传

就是将本地资源（图片或者文件）上传到服务器端。

```
wx.chooseImage({//从手机中选取文件
    success: function (res) {//选取到的文件
        const tempFilePaths = res.tempFilePaths;
        wx.uploadFile({//开始上传文件
            url: 'https://example.weixin.qq.com/upload', //仅为示例，非真实的接口地址
            filePath: tempFilePaths[0],//设置文件地址
            name: 'file',
            formData: {
                'user': 'test'
            },
            success: function (res) {//图片上传成功
                const data = res.data;
                //do something
            }
        });
    }
});
```

2. 下载

就是从服务器下载文件资源到本地。

```
wx.downloadFile({
    url: 'https://example.com/audio/123', //仅为示例，并非真实的资源
    success: function (res) {
        // 只要服务器有响应数据，就会把响应内容写入文件并进入 success 回调
        // 业务需要自行判断是否下载到了想要的内容
        if (res.statusCode === 200) {
            let filePath = res.tempFilePath;// 下载成功的文件路径
        }
    }
});
```

3. 请求

发起 HTTPS 网络请求。

```
 wx.request({
     url: 'test.php', //仅为示例，并非真实的接口地址
     data: {// 请求参数
         x: ''
     },
     header: {// 请求头
         'content-type': 'application/json' // 默认值
     },
     success: function (res) {// 请求成功
         console.log(res.data);
         //请求成功后，这里将收到服务器返回的数据
     },
     fail: function (res) {// 请求失败
         console.log(res)
         //请求失败后，这里将收到失败信息，包括状态码、错误消息等
     }
 });
```

6.4.2 数据缓存

微信小程序的数据缓存主要是使用 localStorage。

```
wx.clearStorage();// 清空所有数据，异步处理
wx.clearStorageSync();// 清空所有数据，同步处理

wx.getStorage({// 获取数据，异步处理
    key: 'key',
    success: function (res) {
        console.log(res.data);
    }
});
let data = wx.getStorageSync('key');// 获取数据，同步处理

wx.removeStorage({// 移除数据，异步处理
    key: 'key',
    success: function (res) {
        console.log(res.data);
    }
});
wx.removeStorageSync('key');// 移除数据，同步处理

wx.setStorage({// 保存数据，异步处理
    key: 'key',
    value: 'value'
```

```
});
wx.setStorageSync('key', 'value');// 保存数据，同步处理
```

6.4.5 位置

获取当前的地理位置、速度。当用户离开小程序后，此接口无法调用。

```
wx.getLocation({
    type: 'wgs84',
    success: function (res) {
        const latitude = res.latitude;
        const longitude = res.longitude;
        const speed = res.speed;
        const accuracy = res.accuracy;
    }
});
```

6.4.6 设备

1. 网络

获取用户网络状态。

```
wx.getNetworkType({
    success: function (res) {
        const networkType = res.networkType;
    }
});
```

2. 电话

拨打电话。

```
wx.makePhoneCall({
    phoneNumber: '1340000' //仅为示例，并非真实的电话号码
});
```

3. 扫码

```
wx.scanCode({// 允许从相机和相册扫码
    success: function (res) {
        console.log(res);
    }
})

wx.scanCode({// 只允许从相机扫码
    onlyFromCamera: true,
    success: function (res) {
        console.log(res);
    }
});
```

6.4.7 开放接口

1. 授权

向用户发起授权请求。调用后会立刻弹窗询问用户是否同意授权小程序使用某项功能或获取用户的某些数据，但不会实际调用对应接口。如果用户之前已经同意授权，则不会出现弹窗，直接返回成功。

```
// 可以通过 wx.getSetting 先查询一下用户是否授权了 "scope.record" 这个
scope
wx.getSetting({
    success: function (res) {
        if (res.authSetting['scope.record']) {// 已授权
            // 用户已经同意小程序使用录音功能，不会弹窗询问
            wx.startRecord();
        } else {// 没有权限
             wx.authorize({// 请求权限会弹窗询问
                 scope: 'scope.record',
                 success: function () {
                     // 用户已经同意小程序使用录音功能，后续调用
wx.startRecord 接口不会弹窗询问
                     wx.startRecord();
                 }
             });
        }
    }
});
```

微信小程序中常用的 scope 如下：

scope	对应接口	描述
scope.userInfo	wx.getUserInfo	用户信息
scope.userLocation	wx.getLocation, wx.chooseLocation, wx.openLocation	地理位置
scope.address	wx.chooseAddress	通讯地址

2. 支付

发起微信支付。

```
 wx.requestPayment({
     timeStamp: '',// 从服务器端获取
     nonceStr: '',// 从服务器端获取
     package: '',// 从服务器端获取
     signType: '',// 从服务器端获取
     paySign: '',// 从服务器端获取
     success: function (res) {
         // 支付成功
     },
     fail: function (res) {
         // 支付失败
     }
 });
```

3. 小程序跳转

小程序之间相互跳转。

```
wx.navigateToMiniProgram({// 打开另一个小程序
    appId: '',// 要打开的小程序 appId
    path: 'page/index/index?id=123',//打开的页面路径，如果为空则打开首页
    extraData: {//需要传递给目标小程序的数据
        foo: 'bar'
    },
    envVersion: 'develop',
    success: function (res) {
        // 打开成功
    }
});

wx.navigateBackMiniProgram({//返回到上一个小程序
```

```
    extraData: {//需要返回给上一个小程序的数据
        foo: 'bar'
    },
    success: function (res) {
        // 返回成功
    }
});
```

4. 数据分析

自定义分析数据上报接口。使用前，需要在小程序管理后台自定义分析中新建事件，配置好事件名与字段。

```
wx.reportAnalytics('purchase', {
    price: 120,
    color: 'red'
});
```

6.4.8 更新

用于检查并管理小程序的更新。

```
//微信小程序检查更新
const updateManager = wx.getUpdateManager();
updateManager.onCheckForUpdate(function (res) {
    //请求完新版本信息的回调
    console.log("onCheckForUpdate:" + res.hasUpdate);
});

updateManager.onUpdateReady(function () {
    //updateManager 会自动下载更新
    //新的版本已经下载好，调用 applyUpdate 应用新版本并重启
    console.log("onUpdateReady");
    updateManager.applyUpdate();
});

updateManager.onUpdateFailed(function () {
    //新的版本下载失败
    console.log("onUpdateFailed");
});
```

6.4.9 总结

这里仅仅示例了常用的 API，其他开放 API 的介绍可以在 `微信公众平台` -> `小程序开发` -> `API` 中找到。