小程序的结构

App（程序）->Page（页面）->Component（组件）

App

app.js：创建App实例的代码以及一些全局相关的内容、

app.json：全局的一些配置、

app.wxss：全局的样式

Page

page.js：创建page实例的代码，以及页面相关的内容、

page.json：业务单独的配置，比如页面对应window配置、

page.wxml：页面的wxml布局代码、

page.wxss：页面的样式配置

Component

component.js：创建Component实例的代码，以及组件内部的内容

component.json：组件内部的配置，比如当前组件使用了别的组件

component.wxml：组件的wxml布局代码

component.wxss：组件的样式配置

快速生成结构

1. 通过微信开发者工具开发，保留project.config.json、sitemap.json其他可以全部删除
2. 创建app相关的三个文件，根据错误提示编写app.json的结构

```
{
"pages": [
"pages/home/home",
"pages/about/about"
]
}
```

3. 新版的工具需要在Pages目录中才能“新建page”，通过这个方式会自动生成Page默认的结构（4个文件），而且会自动在app.json中注册路径，虽然说app.js和app.wxss不是必须的，但我们会在后面使用到它们。
4. 保存后，自动更新显示界面，界面上显示的内容，就是home.wxml中的结构，修改标签中的内容，会时是更新显示，可以尝试用其他标签（也称为内置组件）生成元素，如果希望添加样式，则可以直接在home.wxss中设置，样式会自动渲染home.wxml文件中的标签。

项目托管

git标签管理项目版本

知识点保存在标签中，生成tag1、tag2

知识点结束回退到初始化状态，git reset --hard 初始化

查看某个知识点，git checkout tag1

操作步骤：

1. 在MiniProgram文件夹下，通过git bash 初始化仓库，输入以下命令初始化仓库

```
git init
```

2. 将所有文件添加到暂缓区，"."表示所有文件

```
git add .
```

3. 提交并打上初始化标记

```
git commit -m '初始化项目'
```

4. 打开github.com，创建远程仓库，"MiniProgram"，通过生成后的提示继续将远程仓库与本地仓库关联起来

> ```
> git init
> git add README.md
> git commit -m "first commit"
>
> git remote add origin https://github.com/5631723/MiniProgram.git
> git push -u origin master
> ```

5. 第一次提交可能需要登录github账户，输入当前仓库的账户即可。

> MI@SuperRao MINGW64 ~/Desktop/小程序/代码/MiniProgram (master)
> $ git push -u origin master
> git: 'credential-cache' is not a git command. See 'git --help'.
> Enumerating objects: 16, done.
> Counting objects: 100% (16/16), done.
> Delta compression using up to 8 threads
> Compressing objects: 100% (13/13), done.
> Writing objects: 100% (16/16), 1.98 KiB | 225.00 KiB/s, done.
> Total 16 (delta 1), reused 0 (delta 0)
> remote: Resolving deltas: 100% (1/1), done.
> To https://github.com/5631723/MiniProgram.git
>
> - [new branch] master -> master
>   Branch 'master' set up to track remote branch 'master' from 'origin'.

6. 学习添加标签，随意修改小程序中的代码，将home.wxml修改如下：

```
<view>1.学习添加标签，用于保存当前学习进度</view>
```

7. 将修改的文件添加到暂存区，可以将全部内容添加到暂存区

```
git add .
```

```
$ git add .
warning: LF will be replaced by CRLF in pages/home/home.wxml.
The file will have its original line endings in your working directory
```

8. 提交并打上标记

```
git commit -m '学习添加标签'
```

```
$ git commit -m '学习添加标签'
[master 321bbf8] 学习添加标签
1 file changed, 1 insertion(+)
```

9. 创建标签“01_添加标签”

```
git tag 01_添加标签
```

10. 查看已有标签

```
git tag
```

```
$ git tag
01_添加标签
```

11. 查看提交历史

```
git log
```

```
$ git log
commit 321bbf8170ebd9860aaf5304be04f3a22d9a779c (HEAD -> master, tag: 01_添加标
签)
Author: 5631723 5631723@qq.com
Date:   Sat May 9 12:23:20 2020 +0800

    学习添加标签

commit 2fedfb8dceed8e831cbf0fab709c35902b176b22 (origin/master)
Author: 5631723 5631723@qq.com
Date:   Sat May 9 11:53:14 2020 +0800

    初始化项目
```

12. 回退提交版本，不需要输入全部版本号，输入前六位即可

```
git reset 2fedfb
```

```
$ git reset 2fedfb
Unstaged changes after reset:
M       pages/home/home.wxml
```

如果修改了文件，但没有新的提交，但又希望回退到某个版本，则可以添加"--hard"参数强制回退。

```
git reset --hard 2fedfb
```

```
$ git reset --hard 2fedfb
HEAD is now at 2fedfb8 初始化项目
```

13. git status命令可以列出当前目录所有还没有被git管理的文件和被git管理且被修改但还未提交(git commit)的文件，此命令可以更新微信开发者工具一直提示文件被修改的bug，清除“m”图标提示。

```
git status
```

14. 将本地的tag推送到远程仓库，这样就可以在远程仓库中，查看到已有的Tag标签

```
git push --tags
```

15. 此时已经可以在远程仓库中查看到同步后的标签。如果需要将某个标签获取到本地编辑或查看，可以在本地重新创建一个新目录，打开git bash，输入克隆命令。克隆地址可以直接从远程仓库网站上“clone or download”处获取。

```
git clone https://github.com/5631723/MiniProgram.git
```

```
$ git clone https://github.com/5631723/MiniProgram.git
Cloning into 'MiniProgram'...
remote: Enumerating objects: 21, done.
remote: Counting objects: 100% (21/21), done.
remote: Compressing objects: 100% (15/15), done.
remote: Total 21 (delta 4), reused 20 (delta 3), pack-reused 0
Unpacking objects: 100% (21/21), done.
```

16. 在微信开发者工具中直接导入小程序，但是并没有看到标签中“01_添加标签”版本中的内容，此时还需要将标签CHECKOUT出来。

```
git checkout 01_添加标签
```

```
$ git checkout 01添加标签
Note: switching to '01添加标签'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

git switch -c

Or undo this operation with:

git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at 321bbf8 学习添加标签
M       app.json
M       project.config.json
```

17. 查看微信开发者工具中的页面，此时home.wxml中的标签1.学习添加标签，用于保存当前学习进度又恢复出来了。

小程序体验

数据绑定

是一个显示组件，就是一个容器。通过mustache语法来调用变量，在home.js中声明data属性，保存定义的变量。

home.js:

```
// pages/home/home.js
Page({
  data:{
    name:'raoqi',
    age:33
  }
})
```

home.wxml:

```
<!--pages/home/home.wxml-->
<view>Hello {{name}}</view>
<view>age:{{age}}</view>
```

列表渲染

通过wx:for来实现组件的遍历，默认通过item来遍历数组中的元素

```
// pages/home/home.js
Page({
  data: {
    name: 'raoqi',
    age: 33,
    students: [
        {id: 1,name: '小张',age: 18},
        {id: 2,name: '小王',age: 17},
        {id: 3,name: '小李',age: 16},
        {id: 4,name: '小陈',age: 19}
    ]
})
```

```
<!--pages/home/home.wxml-->
<!-- 数据绑定 -->
<view>Hello {{name}}</view>
<view>age:{{age}}</view>
<!-- 列表展示 -->
<view wx:for='{{students}}'>{{item.name}}的年龄是{{item.age}}岁</view>
```

事件监听

再小程序开发中，将定义的方法直接复制给事件，但并不能通过方法直接修改data中的数据，因为修改的数据并不能及时渲染到页面，所以需要通过this.setData方法将需要渲染的数据更新。

```
// pages/home/home.js
Page({
```

```
  data: {
    name: 'raoqi',
    age: 33,
    students: [
        {id: 1,name: '小张',age: 18},
        {id: 2,name: '小王',age: 17},
        {id: 3,name: '小李',age: 16},
        {id: 4,name: '小陈',age: 19}
    ],
    counter: 0
  },
  handleBtnClick: function () {
    console.log('点击测试')
    //不能直接更新数据
    // this.data.counter++;
    //需要通过this.setData来渲染页面上的数据更新
    this.setData({
      counter: this.data.counter + 1
    })
  },
  handleSubtraction() {
    this.setData({
      counter: this.data.counter - 1
    })
  }
})
```

```
<!--pages/home/home.wxml-->
<!-- 数据绑定 -->
<view>Hello {{name}}</view>
<view>age:{{age}}</view>
<!-- 列表展示 -->
<view wx:for='{{students}}'>{{item.name}}的年龄是{{item.age}}岁</view>
<!-- 事件监听 -->
<view>当前计数：{{counter}}</view>
<button size="mini" bindtap="handleSubtraction">-</button>
<button size="mini" bindtap="handleBtnClick">+</button>
```

MVVM

小程序也是使用MVVM架构。但与Vue（viewModel）不同的是，它是使用MINA框架（ViewModel）来连接视图层（VIew）和模型/逻辑层（Model）。

MVVM的两个作用是：

- DOM Listeners：ViewModel层可以将DOM的监听绑定到Model层
- Data Bindings：ViewModel层可以将数据的变化，响应式的渲染到View层

MVVM架构将我们从“命令式编程”（js/Jquery等）转移到“声明式编程”（React/Angular/vue等）。

配置小程序

小程序很多开发需求被规定在配置文件中。

- 这样做可以更有利于提高开发效率
- 可以保障开发出来的小程序的某些风格是比较统一的，例如导航栏、TapBar、页面路由等等。

常见配置文件包含以下文件。

- project.config.json：项目配置文件，比如项目名称、appid等，通常在微信开发者工具的"详情"中配置一次，保证团队开发的配置一致性。https://developers.weixin.qq.com/miniprogram/dev/devtools/projectconfig.html
- sitemap.json：小程序搜索相关。https://developers.weixin.qq.com/miniprogram/dev/framework/sitemap.html
- app.json：全局配置
- page.json：页面配置

全局配置app

文档地址：https://developers.weixin.qq.com/miniprogram/dev/reference/configuration/app.html

常用配置如下：

属性	类型	必填	描述	最低版本
pages	string[]	是	页面路径列表	
window	Object	否	全局的默认窗口表现	
tabBar	Object	否	底部 `tab` 栏的表现	

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（含文件名） 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`, `.wxss` 四个文件进行处理。

数组的第一项代表小程序的初始页面（首页）。小程序中新增/减少页面，都需要对 pages 数组进行修改。

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"]
}
```

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	描述	最低版本
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	
navigationStyle	string	default	导航栏样式，仅支持以下值：`default` 默认样式 `custom` 自定义导航栏，只保留右上角胶囊按钮。参见注 2。	微信客户端 6.6.0
backgroundColor	HexColor	#ffffff	窗口的背景色	
backgroundTextStyle	string	dark	下拉 loading 的样式，仅支持 `dark` / `light`	
backgroundColorTop	string	#ffffff	顶部窗口的背景色，仅 iOS 支持	微信客户端 6.5.16
backgroundColorBottom	string	#ffffff	底部窗口的背景色，仅 iOS 支持	微信客户端 6.5.16
enablePullDownRefresh	boolean	false	是否开启全局的下拉刷新。 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	
pageOrientation	string	portrait	屏幕旋转设置，支持 `auto` / `portrait` / `landscape` 详见 响应显示区域变化	2.4.0 (auto) / 2.5.0 (landscape)

注 1：HexColor（十六进制颜色值），如"#ff00ff"

注 2：关于 `navigationStyle`

1. 客户端 7.0.0 以下版本，`navigationStyle` 只在 `app.json` 中生效。
2. 客户端 6.7.2 版本开始，`navigationStyle: custom` 对 web-view 组件无效
3. 开启 custom 后，低版本客户端需要做好兼容。开发者工具基础库版本切到 1.7.0（不代表最低版本，只供调试用）可方便切到旧视觉

如：

```
{
  "window": {
    "navigationBarBackgroundColor": "#0094ff",
    "navigationBarTextStyle": "white",
    "navigationBarTitleText": "小程序",
    "backgroundColor": "#eeeeee",
    "backgroundTextStyle": "light",
    "enablePullDownRefresh": false
  }
}
```

注3：`backgroundColor` 属性只有在IOS系统中自带下拉弹性效果时看到背景，其他系统需要通过设置 `enablePullDownRefresh:true` 来开启下拉加载更多功能查看背景颜色。

tabBar

如果小程序是一个多 tab 应用（客户端窗口的底部或顶部有 tab 栏可以切换页面），可以通过 tabBar 配置项指定 tab 栏的表现，以及 tab 切换时显示的对应页面。

属性	类型	必填	默认值	描述	最低版本
color	HexColor	是		tab 上的文字默认颜色，仅支持十六进制颜色	
selectedColor	HexColor	是		tab 上的文字选中时的颜色，仅支持十六进制颜色	
backgroundColor	HexColor	是		tab 的背景色，仅支持十六进制颜色	
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`	
custom	boolean	否	false	自定义 tabBar，见详情	2.5.0

其中 list 接受一个数组，**只能配置最少 2 个、最多 5 个 tab。**tab 按数组的顺序排序，每个项都是一个对象，其属性值如下：

属性	类型	必填	说明
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，不支持网络图片。 **当 `position` 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 `position` 为 `top` 时，不显示 icon。**

```
{
    "tabBar": {
      "list": [{
          "pagePath": "pagePath",
          "text": "text",
          "iconPath": "iconPath",
          "selectedIconPath": "selectedIconPath"
        }]
    }
}
```

> Sat May 09 2020 16:18:10 GMT+0800 (中国标准时间) app.json 文件内容错误
> VM824:1 app.json: ["tabBar"]["list"] 需至少包含 2 项

通过在app.json中键入tabBar来生成tabBar的结构，但直接保存时会报错，提示字少要包含两项。在根目录下创建assets文件夹，拷贝任意图标文件到该文件夹下，重新修改tabBar属性内容如下：

```
  "tabBar": {
    "selectedColor": "#0094ff",
    "list": [{
      "pagePath": "pages/home/home",
      "text": "首页",
      "iconPath": "/assets/1.jpg",
      "selectedIconPath": "/assets/1.jpg"
    },{
      "pagePath": "pages/about/about",
      "text": "关于",
      "iconPath": "/assets/2.jpg",
      "selectedIconPath": "/assets/2.jpg"
    }]
  }
```

`selectedColor`选中时文字的颜色，`pagePath`表示跳转的页面名称（完整路径），`text`显示文字，`iconPath`表示显示的图标（完整路径），`selectedIconPath`表示选中时显示的图标（完整路径）。

页面配置

- 单独配置about页面

在设置全局配置后，我们可以为每个页面单独配置某一个页面，，打开about目录下的对应文件设置如下代码，about.json:

```
{
  "usingComponents": {},
  "navigationBarBackgroundColor": "#0094ff",
  "navigationBarTextStyle": "white",
  "navigationBarTitleText": "关于我们"
}
```

```
/* pages/about/about.wxss */
text{
  font-size: 32px;
  color: cornflowerblue;
}
```

```
<!--pages/about/about.wxml-->
<text>关于页面独立配置</text>
```

运行后会发现页面配置的内容，覆盖掉了全局配置的内容，所以页面配置的优先级要高于全局配置。`usingComponents`用来设置自定义组件功能，后面讲组件功能时再详细描述。

- 添加编译模式

如果不是制作首页，每次保存编译后都需要点击调试的页面，这样比较麻烦，我们可以通过添加编译模式，自定义编译条件。选中“普通编译”旁边的小三角，在弹出的菜单中“点击添加编译模式”。这是程序会自动帮我们设置正在编辑的页面作为下次直接打开的地址。

小程序的双线程模型

“微信客户端”是小程序的宿主环境，宿主环境为了执行小程序的各种文件（.wxml/.wxss/.js），提供了小程序的双线程模型。

- 渲染层：通过WebView执行wxml布局文件，wxss样式文件。
- 逻辑层：通过JsCore执行JS文件（逻辑层）。

这两个线程都会经由微信客户端（Native）进行中转交互。

关于wxml布局文件，我们可以想象到，每个wxml等价于一颗DOM数，所以可以使用一个JS对象来模拟（虚拟DOM）。例如：

```
<!-- wxml布局 -->
<view>
  <view>a</view>
  <view>b</view>
  <view>c</view>
</view>
```

可以用下面的JS对象来表示。

```
//js对象
{
  name:"view",
  children:[
    {name:"view",children:[{text:"a"}]},
    {name:"view",children:[{text:"b"}]},
    {name:"view",children:[{text:"c"}]},
  ]
}
```

那么，WXML可以先转成JS对象，再渲染出真正的DOM树。通过setData把页面上的变量渲染成新的数据，数据发生变化的过程如下：

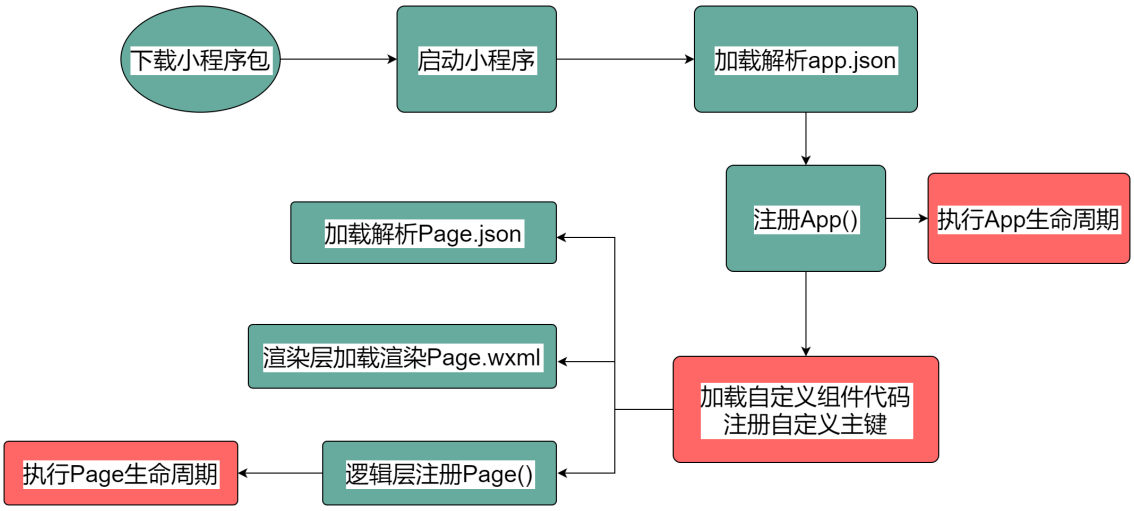
1. 产生的JS对象对应的节点就会发生变化
2. 此时可以对比前后两个JS对象得到变化的部分
3. 然后把这个差异应用到原来的DOM树上
4. 从而达到更新UI的目的，这就是“数据驱动”的原理

界面渲染整理流程：

1. 在渲染层，宿主环境会把WXML转化成对应的JS对象
2. 将JS对象再次转成真实DOM树，交由渲染层线程渲染
3. 数据变化时，逻辑层提供最新的变化数据，JS对象发生变化比较进行diff算法对比
4. 将最新变化的内容反映到真实的DOM树中，更新UI。

小程序启动流程

我们一起来看一下一个小程序的启动流程，通过了解小程序的启动流程，我们就能知道代码的执行顺序：



App(Object object)

注册小程序。接受一个 `Object` 参数，其指定小程序的生命周期回调等。

App() 必须在 `app.js` 中调用，必须调用且只能调用一次。不然会出现无法预期的后果。

app.js中的app()包含以下常用生命周期函数。

```
App({

  /**
   * 当小程序初始化完成时，会触发 onLaunch（全局只触发一次）
   */
  onLaunch: function () {

  },

  /**
   * 当小程序启动，或从后台进入前台显示，会触发 onShow
   */
  onShow: function (options) {

  },

  /**
   * 当小程序从前台进入后台，会触发 onHide
   */
  onHide: function () {

  },

  /**
   * 当小程序发生脚本错误，或者 api 调用失败时，会触发 onError 并带上错误信息
   */
  onError: function (msg) {

  },
  //自定义的一个全局数据对象
  globalData: 'I am global data'
})
```

`globalData`变量可以修改成任意名称，它是一个自定义的全局对象，在早期时被创建在项目中使用。

除了自动生成的函数外，还包括其他一些函数，参考文档：https://developers.weixin.qq.com/miniprogram/dev/reference/api/App.html

App()参数

App(Object object)，App函数的参数是对象类型，包含以下内容：

属性	类型	默认值	必填	说明	最低版本
onLaunch	function		否	生命周期回调——监听小程序初始化。	
onShow	function		否	生命周期回调——监听小程序启动或切前台。	
onHide	function		否	生命周期回调——监听小程序切后台。	
onError	function		否	错误监听函数。	
onPageNotFound	function		否	页面不存在监听函数。	1.9.90
onUnhandledRejection	function		否	未处理的 Promise 拒绝事件监听函数。	2.10.0
onThemeChange	function		否	监听系统主题变化	2.11.0
其他	any		否	开发者可以添加任意的函数或数据变量到 `Object` 参数中，用 `this` 可以访问	

我们将在onLaunch函数中尝试获取用户信息，下面用一个例子来说明这些生命周期函数的作用。

- 例子，修改app.js中App(object)中的`onLaunch`函数：

```
onLaunch: function () {
  wx.getUserInfo({
    complete: (res) => {
      console.log(res);
    },
  })
}
```

当再次启动程序时，将会异步请求当前登录小程序的用户信息，控制台将显示当前登录的用户信息。

> {errMsg: "getUserInfo:ok", rawData: "{"nickName":"饶麒","gender":1,"language":"zh_CN","ci...yHPV9p7sibs7PPVADPbl4xV7adVIAF7oB91V27AskBw/132"}", userInfo: {...}, signature: "f166f68136bb35e5e757dc3babfe6b424c8ecf54", encryptedData: "yEiTxJ32iEZBxnxzlxs+fY/usFOfMO4lClUBNq/U3oHzOBKuRW...Zht9jfQKgmUAcGzz87x6zB2oRBDLNvtL/XS+0WVUpmcJ86g==", ...}

- 思考？在注册App()时，我们通常会做以下事情呢?
    1. 判断小程序的进入场景

2. 监听生性周期函数，在生命周期中执行对应的业务逻辑，比如在某个生命周期函数中获取微信用户的信息。
3. 因为App()实例只有一个，并且是全局共享的（单例对象），所以我们可以将一些共享数据存放其中。

- 前台/后台状态

1. 小程序启动后，界面被展示给用户，此时小程序处于**前台**状态。
2. 当用户点击右上角胶囊按钮关闭小程序，或者按了设备 Home 键离开微信时，小程序并没有完全终止运行，而是进入了**后台**状态，小程序还可以运行一小段时间。
3. 当用户再次进入微信或再次打开小程序，小程序又会从后台进入**前台**。但如果用户很久没有再进入小程序，或者系统资源紧张，小程序可能被**销毁**，即完全终止运行。

- 小程序启动

  小程序启动可以分为两种情况，一种是**冷启动**，一种是**热启动**。

  - 冷启动：如果用户首次打开，或小程序销毁后被用户再次打开，此时小程序需要重新加载启动，即冷启动。
  - 热启动：如果用户已经打开过某小程序，然后在一定时间内再次打开该小程序，此时小程序并未被销毁，只是从后台状态进入前台状态，这个过程就是热启动。

- 小程序销毁时机

  只有当小程序进入后台一定时间，或者系统资源占用过高，才会被销毁。具体而言包括以下几种情形：

  - 当小程序进入后台，可以维持一小段时间的运行状态，如果这段时间内都未进入前台，小程序会被销毁。
  - 当小程序占用系统资源过高，可能会被系统销毁或被微信客户端主动回收。
    - 在 iOS 上，当微信客户端在一定时间间隔内连续收到系统内存告警时，会根据一定的策略，主动销毁小程序，并提示用户 「运行内存不足，请重新打开该小程序」。具体策略会持续进行调整优化。
    - 建议小程序在必要时使用 wx.onMemoryWarning 监听内存告警事件，进行必要的内存清理。

小程序的打开场景

- 用户打开小程序时，场景可分为以下场景，我们通过在App()中的`onLaunch`或`onShow`方法中获取`options`参数查看`scene`属性即可获得对应的场景值。

```
/**
 * 当小程序启动，或从后台进入前台显示，会触发 onShow
 */
onShow: function (options) {
  console.log(options)
}
```

  重新运行小程序，控制台上会打印以下内容， **1001**代表场景值ID，可以参考对应的说明，了解小程序打开的场景。

> {path: "pages/about/about", query: {...}, scene: 1001, shareTicket: undefined, referrerInfo: {...}}

  文档地址：https://developers.weixin.qq.com/miniprogram/dev/reference/scene-list.html

场景值ID	说明	图例
1000	其他	/
1001	发现栏小程序主入口，「最近使用」列表（基础库2.2.4版本起包含「我的小程序」列表）	/
1005	微信首页顶部搜索框的搜索结果页	查看
1006	发现栏小程序主入口搜索框的搜索结果页	查看
1007	单人聊天会话中的小程序消息卡片	查看
1008	群聊会话中的小程序消息卡片	查看
1011	扫描二维码	查看
1012	长按图片识别二维码	查看
1013	扫描手机相册中选取的二维码	查看
1014	小程序模板消息	查看
1017	前往小程序体验版的入口页	查看
1019	微信钱包（微信客户端7.0.0版本改为支付入口）	查看
1020	公众号 profile 页相关小程序列表（已废弃）	查看
1022	聊天顶部置顶小程序入口（微信客户端6.6.1版本起废弃）	/
1023	安卓系统桌面图标	查看
1024	小程序 profile 页	查看
1025	扫描一维码	查看
1026	发现栏小程序主入口，「附近的小程序」列表	查看
1027	微信首页顶部搜索框搜索结果页「使用过的小程序」列表	查看

场景值 ID	说明	图例
1028	我的卡包	查看
1029	小程序中的卡券详情页	查看
1030	自动化测试下打开小程序	/
1031	长按图片识别一维码	查看
1032	扫描手机相册中选取的一维码	查看
1034	微信支付完成页	查看
1035	公众号自定义菜单	查看
1036	App 分享消息卡片	查看
1037	小程序打开小程序	查看
1038	从另一个小程序返回	查看
1039	摇电视	查看
1042	添加好友搜索框的搜索结果页	查看
1043	公众号模板消息	查看
1044	带 shareTicket 的小程序消息卡片 详情	查看
1045	朋友圈广告	查看
1046	朋友圈广告详情页	查看
1047	扫描小程序码	查看
1048	长按图片识别小程序码	查看
1049	扫描手机相册中选取的小程序码	查看

场景值ID	说明	图例
1052	卡券的适用门店列表	查看
1053	搜一搜的结果页	查看
1054	顶部搜索框小程序快捷入口（微信客户端版本6.7.4起废弃）	/
1056	聊天顶部音乐播放器右上角菜单	查看
1057	钱包中的银行卡详情页	查看
1058	公众号文章	查看
1059	体验版小程序绑定邀请页	/
1064	微信首页连Wi-Fi状态栏	查看
1067	公众号文章广告	查看
1068	附近小程序列表广告（已废弃）	/
1069	移动应用	查看
1071	钱包中的银行卡列表页	查看
1072	二维码收款页面	查看
1073	客服消息列表下发的小程序消息卡片	查看
1074	公众号会话下发的小程序消息卡片	查看
1077	摇周边	查看
1078	微信连Wi-Fi成功提示页	查看
1079	微信游戏中心	查看
1081	客服消息下发的文字链	查看

场景值ID	说明	图例
1082	公众号会话下发的文字链	查看
1084	朋友圈广告原生页	查看
1088	会话中查看系统消息，打开小程序	/
1089	微信聊天主界面下拉，「最近使用」栏（基础库2.2.4版本起包含「我的小程序」栏）	查看
1090	长按小程序右上角菜单唤出最近使用历史	查看
1091	公众号文章商品卡片	查看
1092	城市服务入口	查看
1095	小程序广告组件	查看
1096	聊天记录，打开小程序	查看
1097	微信支付签约原生页，打开小程序	查看
1099	页面内嵌插件	/
1102	公众号 profile 页服务预览	查看
1103	发现栏小程序主入口，「我的小程序」列表（基础库2.2.4版本起废弃）	/
1104	微信聊天主界面下拉，「我的小程序」栏（基础库2.2.4版本起废弃）	/
1106	聊天主界面下拉，从顶部搜索结果页，打开小程序	/
1107	订阅消息，打开小程序	/
1113	安卓手机负一屏，打开小程序（三星）	/
1114	安卓手机侧边栏，打开小程序（三星）	/
1124	扫“一物一码”打开小程序	/
1125	长按图片识别“一物一码”	/
1126	扫描手机相册中选取的“一物一码”	/
1129	微信爬虫访问 详情	/
1131	浮窗打开小程序	/

场景值 ID	说明	图例
1135	小程序资料页打开小程序	查看
1146	地理位置信息打开出行类小程序	查看
1148	卡包-交通卡，打开小程序	/
1150	扫一扫商品条码结果页打开小程序	查看
1153	“识物”结果页打开小程序	查看

对应的场景值非常的多，如果需要切换不同场景开发，可以点击微信开发者工具模拟器上的关闭圆圈按钮，此时会弹出以下界面供开发者选择测试场景。

在实际开发过程中，我们可以用代码来处理不同入口进入小程序的业务；

```
onShow: function (options) {
  switch (options.scene) {
```

```
      case 1001:
        //发现栏小程序入口业务
        break;
      case 1011:
        //扫描二维码业务
        break;
      case 1092:
        //城市服务入口业务
        break;
      ...
    }
  }
```

获取用户信息

1. wx.getUserInfo(Object object)调用前需要 用户授权 scope.userInfo。获取用户信息。

```
// 必须是在用户已经授权的情况下调用
wx.getUserInfo({
  success: function(res) {
    var userInfo = res.userInfo
    var nickName = userInfo.nickName
    var avatarUrl = userInfo.avatarUrl
    var gender = userInfo.gender //性别 0：未知、1：男、2：女
    var province = userInfo.province
    var city = userInfo.city
    var country = userInfo.country
  }
})
```

2. 通过按钮获取用户信息。通过按钮的两个特殊属性（事件），获取用户信息。（注意此处点击不能绑定**bindtap**事件）

```
  <!--pages/about/about.wxml-->
  <text>关于页面通过按钮获取用户信息</text>
  <button size='mini' open-type="getUserInfo"
bindgetuserinfo="handleGetUserInfo">获取用户信息</button>
```

```
// pages/about/about.js
Page({
  handleGetUserInfo(event) {
    console.log(event);

  }
})
```

点击按钮，控制台打印以下内容，其中**detail**属性中包含了用户信息。

> {type: "getuserinfo", timeStamp: 1944, target: {...}, currentTarget: {...}, mark: {...}, ...}
> type: "getuserinfo"
> timeStamp: 1944
> target: {id: "", offsetLeft: 128, offsetTop: 46, dataset: {...}}
> currentTarget: {id: "", offsetLeft: 128, offsetTop: 46, dataset: {...}}

> mark: {}
> detail: {errMsg: "getUserInfo:ok", rawData: "{"nickName":"饶
> 麒","gender":1,"language":"zh_CN","ci...
> yHPV9p7sibs7PPVADPbl4xV7adVIAF7oB91V27AskBw/132"}", signature:
> "f166f68136bb35e5e757dc3babfe6b424c8ecf54", encryptedData:
> "h3rljd15/DiyyVTn3LaTywHe+HPzG3HiBqwOAR98mDZKAJuixU...
> ObhNKKtqEPK9ctg9RnBa+vQUyteJEL2np6CPmhGcAANNRjQ==", iv:
> "KO6GoMk139I2Vk7jlHYtMA==", ...}
> mut: false
> _userTap: false
> **proto**: Object

3. open-data直接展示用户信息

文档位置：https://developers.weixin.qq.com/miniprogram/dev/component/open-data.html

通过在页面中直接声明，用于展示微信开放的数据。

```
<open-data type="userNickName"></open-data>
<open-data type="userAvatarUrl"></open-data>
<open-data type="userGender" lang="zh_CN"></open-data>
```

运行测试，会看到直接在页面上展示了当前用户的三个信息（昵称、头像、性别）。

4. AppObject **getApp(Object object)**

获取到小程序全局唯一的 `App` 实例。此处的`globalData`在**app.js**中声明了数据。

```
// other.js
var appInstance = getApp()
console.log(appInstance.globalData) // I am global data
```

扩展：我们可以在监听小程序初始化`onLaunch`事件中，获取用户信息，并将值存入`globalData`中。这样就可以在全局通过`getApp()`来获取当前用户信息了。

注册页面

小程序中的每个页面，都有一个对应的js文件，其中调用Page方法注册页面示例。

- 在注册时，可以绑定初始化数据、生命周期回调、事件处理函数等。
- 文档位置：https://developers.weixin.qq.com/miniprogram/dev/reference/api/Page.html

```
// pages/other/other.js
Page({

  /**
   * 页面的初始数据
   */
  data: {
    message:'this a other page'
  },

  /**
   * 生命周期函数--监听页面加载
   */
  onLoad: function (options) {
```

```
    console.log("onLoad");
  },

  /**
   * 生命周期函数--监听页面初次渲染完成
   */
  onReady: function () {
    console.log("onReady");
  },

  /**
   * 生命周期函数--监听页面显示
   */
  onShow: function () {
    console.log("onShow");
  },

  /**
   * 生命周期函数--监听页面隐藏
   */
  onHide: function () {
    console.log("onHide");
  },

  /**
   * 生命周期函数--监听页面卸载
   */
  onUnload: function () {
    console.log("onUnload");
  },

  /**
   * 页面相关事件处理函数--监听用户下拉动作
   */
  onPullDownRefresh: function () {
    console.log("onPullDownRefresh");
  },

  /**
   * 页面上拉触底事件的处理函数
   */
  onReachBottom: function () {
    console.log("onReachBottom");
  },

  /**
   * 用户点击右上角分享
   */
  onShareAppMessage: function () {
    console.log("onShareAppMessage");
  }
})
```

尝试触发上面的监听函数。

发送网络请求、初始化页面

测试网络请求，在页面加载时，获取服务器端数据。在小程序中请求第三方网站，需要在小程序中配置安全域名，也可以在详情->本地设置中，勾选**不校验合法域名。**

例子：

```
Page({
  /**
   * 页面的初始数据
   */
  data: {
    message:'this a other page',
    list:[]
  },
  /**
   * 生命周期函数--监听页面加载
   */
  onLoad: function (options) {
    console.log("onLoad");
    wx.request({
      url: 'http://film.glkjjt.com/api/Movie/New',
      success: (res) => {
        console.log(res);
        this.setData({
          list:res.data
        })
      }
    })
  },
  ...
})
```

> Sat May 09 2020 23:58:06 GMT+0800 (中国标准时间) request 合法域名校验出错
> VM34 asdebug.js:1 如若已在管理后台更新域名配置，请刷新项目配置后重新编译项目，操作路径：“详情-域名信息”
> VM30:1 http://film.glkjjt.com 不在以下 request 合法域名列表中，请参考文档：https://developers.weixin.qq.com/miniprogram/dev/framework/ability/network.html

勾选**不校验合法域名**后显示：

> {data: Array(11), header: {…}, statusCode: 200, cookies: Array(1), errMsg: "request:ok"}
> data: Array(11)
> nv_length: (...)
> 0: {MovieID: 11, Name: "飞屋环游记", Actors: "爱德华·阿斯纳,乔丹·长井,鲍勃·彼德森", Type: "喜剧,动画,冒险", Rate: 8.9, ...}
> 1: {MovieID: 9, Name: "三傻大闹宝莱坞", Actors: "阿米尔·汗,黄渤,卡琳娜·卡普", Type: "剧情,喜剧,爱情,歌舞", Rate: 9.1, ...}
> 2: {MovieID: 7, Name: "泰坦尼克号", Actors: "莱昂纳多·迪卡普里奥,凯特·温丝莱特,比利·赞恩", Type: "历史,爱情,灾难", Rate: 9.5, ...}
> 3: {MovieID: 3, Name: "奇门遁甲", Actors: "大鹏,倪妮,李治廷", Type: "喜剧,动作,奇幻", Rate: 7.7, ...}
> 4: {MovieID: 4, Name: "帕丁顿熊2", Actors: "本·威士肖,杜江,休·格兰特", Type: "喜剧,动画,冒险", Rate: 9.6, ...}
> 5: {MovieID: 5, Name: "霸王别姬", Actors: "张国荣,张丰毅,巩俐", Type: "爱情,剧情", Rate: 9.6, ...}
> 6: {MovieID: 1, Name: "机器之血", Actors: "成龙,罗志祥,欧阳娜娜", Type: "动作,剧情", Rate: 8.8,

> ...}
> 7: {MovieID: 2, Name: "寻梦环游记", Actors: "安东尼·冈萨雷斯,本杰明·布拉特,盖尔·加西亚·贝纳尔", Type: "动画,冒险,家庭", Rate: 9.6, ...}
> 8: {MovieID: 6, Name: "肖申克的救赎", Actors: "蒂姆·罗宾斯,摩根·弗里曼,鲍勃·冈顿", Type: "犯罪,剧情", Rate: 9.5, ...}
> 9: {MovieID: 8, Name: "阿凡达", Actors: "萨姆·沃辛顿,佐伊·索尔达娜,米歇尔·罗德里格兹", Type: "动作,冒险,奇幻", Rate: 9, ...}
> 10: {MovieID: 10, Name: "神偷奶爸", Actors: "史蒂夫·卡瑞尔,杰森·席格尔,拉塞尔·布兰德", Type: "动画,家庭,喜剧", Rate: 9, ...}
> length: 11
> **proto**: Array(0)
> header: {Cache-Control: "no-cache", Pragma: "no-cache", Content-Type: "application/json; charset=utf-8", Expires: "-1", Server: "Microsoft-IIS/7.5", ...}
> statusCode: 200
> cookies: ["security_session_verify=b87ab05a8338540f69e5a584a2...ires=Tue 12-May-20 23:59:29 GMT; path=/; HttpOnly"]
> errMsg: "request:ok"
> **proto**: Object

```
<!--pages/other/other.wxml-->
<text>pages/other/other.wxml</text>
<view wx:for='{{list}}'>{{item.Name}}</view>
```

运行小程序，在other.wxml上查看获取到的服务器数据。

补充：网络请求时使用**箭头函数**方法实现**success**的调用，其中**this**关键字会向上一层一层的寻找data数据，但如果将**success**用`function(res){}`来声明，在function函数中使用**this**，**this**就会变成`undefined`，必须得在网络请求之前，将**this**用**_that**保存起来。

监听事件函数

```
// pages/other/other.js
...
  handleTextClick() {
    this.setData({
      message: '点击文字修改Message'
    })
  }, handleBtnClick() {
    this.setData({
      message: '点击按钮修改Message'
    })
  }
...
```

```
<!--pages/other/other.wxml-->
<text bindtap="handleTextClick">{{message}}</text>
<button bindtap="handleBtnClick">监听点击事件</button>
<view wx:for='{{list}}'>{{item.Name}}</view>
```

运行小程序，测试点击效果。

其他监听

常用的监听还有比如**页面滚动、上拉刷新、下拉加载更多。**

```
// pages/other/other.js
...
  /**
   * 页面相关事件处理函数--监听用户下拉动作
   */
  onPullDownRefresh: function () {
    console.log("onPullDownRefresh");
  },
  /**
   * 页面上拉触底事件的处理函数
   */
  onReachBottom: function () {
    console.log("onReachBottom");
  },
 /**
  * 监听用户滑动页面事件的处理函数
  */
  onPageScroll(obj) {
    console.log(obj);
  },
...
```

模拟对应动作，完成事件的触发。