

程序员开发实战系列《三》App()和 Page()

从上篇文章中，我们了解：小程序主要由两个部分构成，app 和 page。

app，就是小程序的框架。支撑 pages，逻辑层的调用，对数据，wxss，wxml 的解析；page，主要是业务层，用于实现界面化操作功能，是程序员使用频率最高的部分。

本篇文章简单介绍 App() 和 Page()函数；

详细的可以参数微信小程序 api 文档。

一，App()

用来注册一个小程序。在小程序启动的时候调用，并创建小程序，直到销毁。在整个小程序的生命周期过程中，它都是存在的。很显然它是单例的，全局的。所以，

1) 只能在 app.js 中注册一次。

2) 在代码的任何地方都可以通过 getApp() 获取这个唯一的小程序单例，
比如 var appInstance = getApp();

App()的参数是 object 类型 {}，指定了小程序的声明周期函数。

onLaunch 函数

监听小程序初始化。

当小程序初始化完成时，会触发 onLaunch（全局只触发一次）。

onShow 函数

监听小程序显示。

当小程序启动，或从后台进入前台显示，会触发。

onHide 函数

监听小程序隐藏。

当小程序从前台进入后台，会触发。

所谓前后台的定义，类似于手机上的 **app**，比如当不在使用微信时，就进入了后台。

globalData 对象

全局数据。

代码和日志参考，如下动图：

```
2 1 //app.js
3 App({
4
5     //小程序初始化时调用
6     onLaunch: function () {
7         var logs = wx.getStorageSync('logs') || []
8         logs.unshift('小程序初始化时调用')
9         wx.setStorageSync('logs', logs)
10    },
11
12    //小程序显示时调用
13    onShow: function () {
14        var logs = wx.getStorageSync('logs') || []
15        logs.unshift('小程序显示时调用')
16        wx.setStorageSync('logs', logs)
17    },
18
19    //小程序隐藏时调用
20    onHide: function() {
21        var logs = wx.getStorageSync('logs') || []
22        logs.unshift('小程序隐藏时调用')
23        wx.setStorageSync('logs', logs)
24    },
25
26    //全局数据
27   .globalData: {
28        userInfo: {username: 'jack'}
29    }
30 })
```

二、Page()

通过 App() 注册完成小程序之后，框架就开始注册页面。所以不要在 App() 的 onLaunch 中调用 getCurrentPage() 方法，因为此时页面还没有注册完成。

同样的 `Page()`也是有生命周期的。当页面注册完成之后，可以在 `page.js` 文件中调用 `getCurrentPage()` 方法，获取当前页面对象。

2.1，`Page()`的参数也是 `Object` 类型。

`onLoad`

监听页面加载

页面刚开始加载的时候触发。只会调用一次。

`onReady`

监听页面初次渲染完成

类似于 html 的 `onReady`。只会调用一次。

`onShow`

监听页面显示

页面显示的时候触发，比如页面切换

`onHide`

监听页面隐藏

和 `onShow` 对应

`onUnload`

监听页面卸载

在 `redirectTo` 或 `navigateBack` 的时候调用

`onPullDownRefresh`

监听用户下拉动

- 1) 需要在 `config` 的 `window` 选项中开启 `enablePullDownRefresh`。
- 2) 当处理完数据刷新后, `wx.stopPullDownRefresh` 可以停止当前页面的下拉刷新。

onReachBottom

页面上拉触底事件的处理函数

data

页面的初始数据

2.2, Page.prototype.setData()

`Page` 的函数 `setData()` 用于页面初始数据 `data` 的修改。如果该数据绑定到了视图层 `wxml` 中展示, 那么无须刷新, 视图层就会反映出修改。

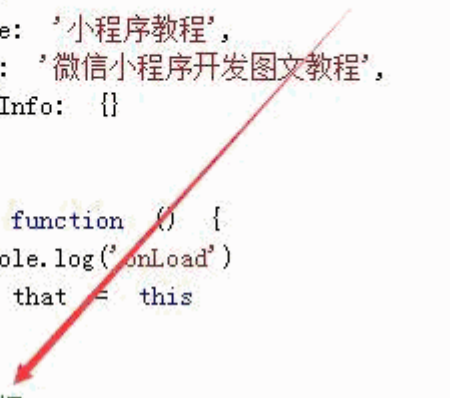
对于 `data` 的修改, 只能使用 `setData()`, 不能直接通过 `this.data` 进行修改。数据量限制在 1024 kb 以内。

2.3, getCurrentPages()

,获取当前页面栈的实例, 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。

2.4, 案例动图

```
1 //index.js
2 //获取应用实例
3 var app = getApp()
4 Page({
5   data: {
6     title: '小程序教程',
7     desc: '微信小程序开发图文教程',
8     userInfo: {}
9   },
10
11   onLoad: function () {
12     console.log('onLoad')
13     var that = this
14   },
15
16   //修改数据
17   changeData: function() {
18     this.setData({
19       title: 'o(∩_∩)o 小程序教程'
20     })
21   },
22
23   //事件处理函数，进入log页面
24   bindViewTap: function() {
25     wx.navigateTo({
26       url: '../logs/logs'
27     })
28   }
29 })
```



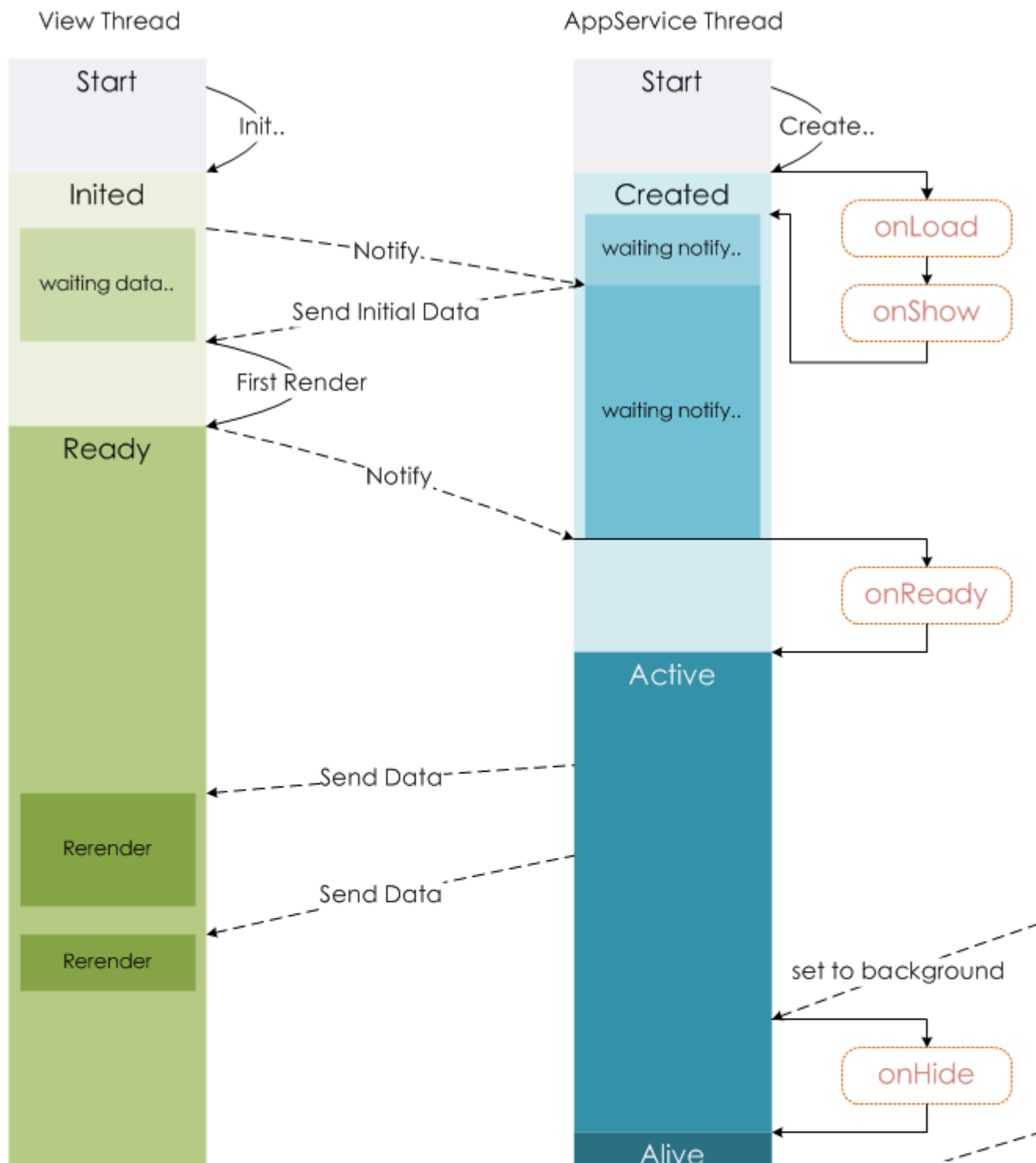
三、页面栈

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，直到目标返回页面， 新页面入栈
Tab 切换	当前页面出栈， 新页面入栈

四、生命周期

下图说明了 Page 实例的生命周期。



五，页面路由

路由方式	路由后页面	路由前页面
初始化	onLoad， onShow	
打开新页面	onLoad， onShow	onHide
页面重定向	onLoad， onShow	onUnload
页面返回	onShow	onUnload（多层页面返回每个页面都会按顺序触发 onU nload）
Tab 切换	第一次打开 onLoad，onshow； 否则 onShow	onHide