

微信小程序开发文档

你的新陪伴，收藏起来！





Easy Tutorial

Framework

Component

API

Tool

Directory Structure

Settings

Global configuration

Page configuration

Logic Layer

View layer

Custom component

Plugin

Basic Capabilities

Hardware Capabilities

Open Capabilities

Base Library

Runtime Environment

Operating mechanism

Performance

Framework

The goal of the Mini Program development framework is to enable developers to develop services with a native APP experience on WeChat in a simplest and most efficient way possible.

The framework provides its own view layer description languages `WXML` and `WXSS`, as well as a `JavaScript`-based logical layer framework, and provides a data transfer and event system between the view layer and the logical layer, allowing developers to focus on data and logic.

Responsive data binding

The core of the framework is a responsive data binding system.

The entire Mini Program framework system is divided into two parts, including a view layer (View) and a logic layer (App Service).

The framework keeps data and views in sync in a simple way. When the data needs to be changed, you only need to modify it in the logical layer, and the view layer will update accordingly.

Look at this simple example:

[Preview with Developer Tool](#)

```
<!-- This is our View -->
<view> Hello {{name}}! </view>
<button bindtap="changeName"> Click me! </button>
```

```
// This is our App Service.
// This is our data.
var helloData = {
  name: 'WeChat'
}

// Register a Page.
Page({
  data: helloData
```



微信小程序UI库 (WeUI)

微信扫描二维码



课程计划

创建一个微信小程序

"F*** My Code"



如何setup一个小程序

使用微信的开发工具



>_

下载stable build



Mini Program Project


Mini Program

Mini Game

Mini Code

WeChat Web Project

WeChat Web

 Logout >

New ProjectImport Project

ProjectName

FMC

Directory

/Users/thibaultgenaitay/code/tgenaitay/FMC

▼

AppID

▼

If no AppID, please [Register](#)
Or use [Test Account](#)

Dev Mode

Mini Program

▼

Backend Service

☒ Use no cloud service

☐ Mini Program Cloud Base

Mini Program Cloud Base provides sophisticated cloud support. By using the Cloud API provided by wechat mini program platform, we can develop and deploy mini program in no time, and enjoy fast iterative development [Learn More](#)

☐ Tencent Cloud

Language

JavaScript

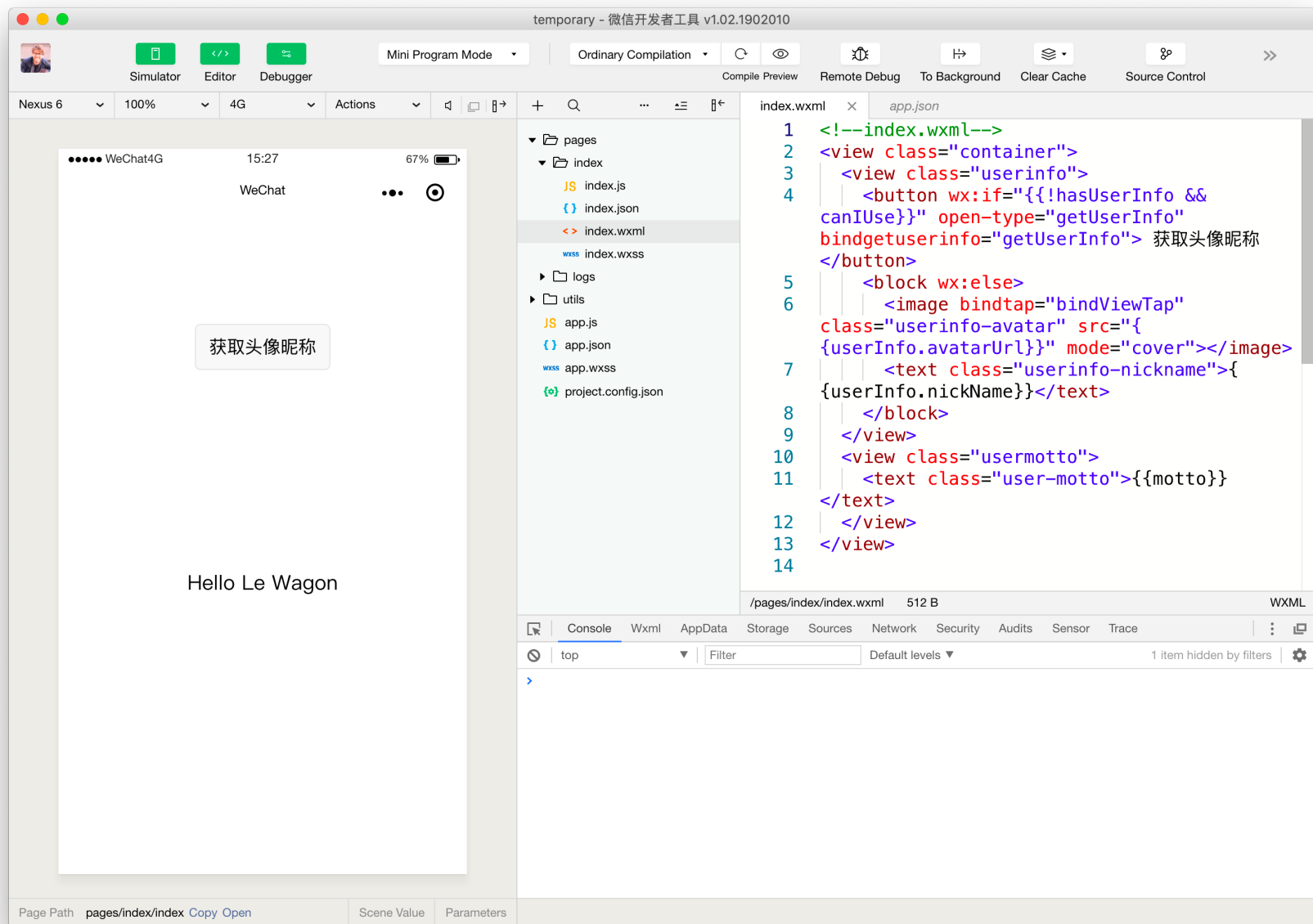
▼

Cancel

Create

使用touristID作为applD





快速预览



微信小程序代码结构

小程序的4种代码文件类型:

- `.wxml` = HTML文件
- `.wxss` = CSS文件
- `.js` 逻辑文件
- `.json` 配置文件 (configuration)

小程序由以下文件运行:

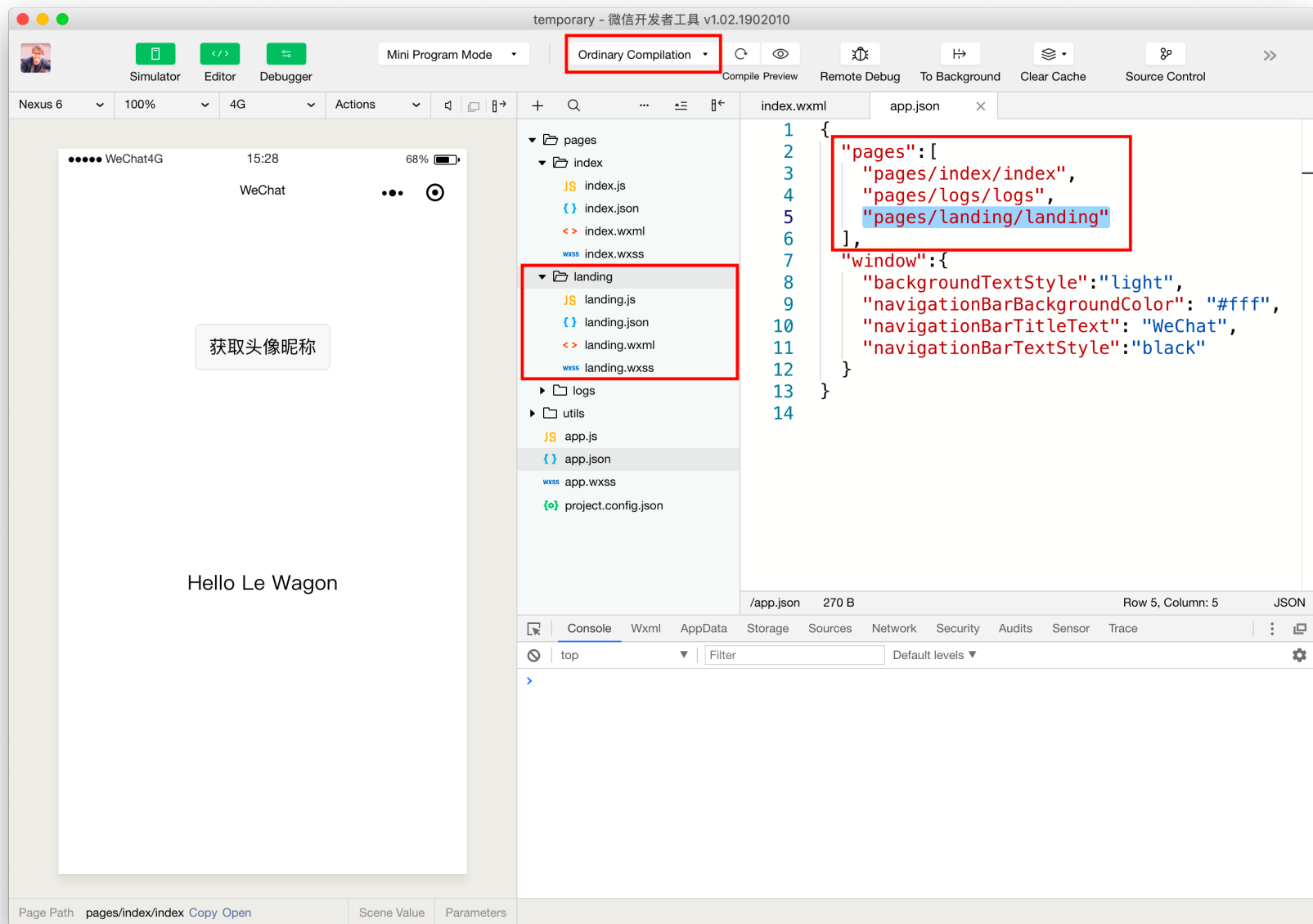
- `app.js` 定义函数
- `app.json` 公共设置
- `app.wxss` 公共样式表



如何创建微信小程序页面

直接在app.json的pages配置中添加新路由注册即可，将生成page的四个文件（`.wxml`，`.wxss`，`.js`，and `.json`）





提示：可以使用开发工具的编译模式（compilation mode）设置启动页面（startup page）



WXML语法（大同小异）



>_

WXML tags	HTML tags	Examples
-----------	-----------	----------

<view>	<div>	<view class="header">...</view>
--------	-------	---------------------------------

<navigator>	<a>	<navigator url="/pages/about/about">...</navigator>
-------------	-----	---

<image>		<image src="/image/logo.png"></image>
---------	-------	---------------------------------------

<text>	<p>	<text> ... </text>
--------	-----	--------------------



JSON文件进行配置

- 定制小程序（Eg: app/page title, navigation bar color）
- 设置标签页
- 设置组件（components）

[查看所有选项](#)



现场编程 1：落地页

使用Le Wagon设定的banner组件.



现场编程 2: 故事页面

为了节省时间，使用Le Wagon 预设的卡片组件（不需要产品图片）



不够有挑战性吗？
我们才刚刚开始...



让我们开始编程

框架概念：

- 生命周期
- 数据存储
- 数据绑定 (data binding)
- 逻辑控制



查看JS的结构

- 主要函数： **Page({...})** 或者 **App({...})**
- 数据存储： **data: {key: value}** 或者 **globalData: {key: value}**
- 生命周期： **onLoad, onLaunch, ...**
- 自定义函数



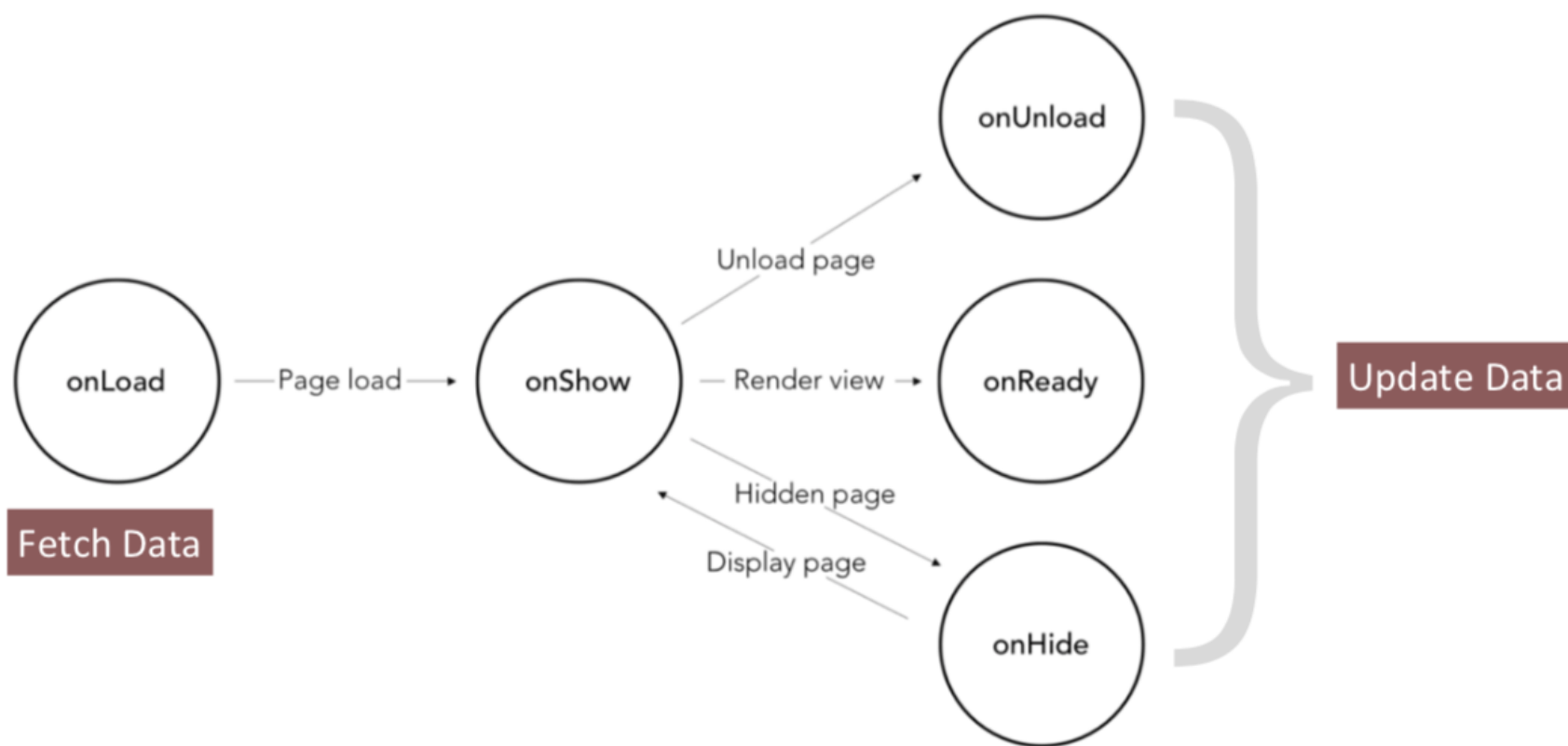
小程序生命周期

可以`console.log`每个函数，以查看它们被调用的顺序。
试试 `onLoad` , `onShow` , `onReady` 看看有什么不同。



>_

生命周期函数是指一个小程序从创建到销毁的一系列过程



例如：

可以在onReady生命周期方法中创建并触发函数

```
//index.js
Page({
  testFunction: function() {
    console.log('test')
  },
  onReady: function() {
    this.testFunction()
  }
})
```

要访问该函数，用以下语法： `this.functionName()`



在哪里存储数据？

1. 本地数据 (local data) : 页面的内部数据
2. 全局数据 (global data) : 小程序所有页面中的共享数据
3. 缓存 (cache) : 用户手机本地
4. 服务器 (server) : 通过API!



查看WXML中的本地数据

可以将数据存储在JS文件中，并在WXML文件中访问。这称为**数据绑定（data binding）**。

```
//index.js
Page({
  data: { name: 'Allen' }
})
<!-- index.wxml -->
<text> My name is {{name}}</text>
```

与HTML相比，WXML更多样化：是一种**模板语言（templating language）**！



setData({})

可以更新本地数据存储中的一些值

```
//index.js
Page({
  data: { text: "Original data" },
  testFunction: function() {
    console.log(this.data.text)
    this.setData({
      text: 'F My Code!'
    })
  },
  onReady: function() {
    this.testFunction()
  }
})
<!-- index.wxml -->
<text> {{text}}</text>
```



this.data

可以访问JS代码中的本地数据

```
//index.js
Page({
  data: { text: "original data" },
  onReady: function() {
    console.log(this.data.text)
  }
})
```



触发函数（example 1）

可以通过向元素添加**bindtap**中的参数从 `.wxml` 文件触发函数

```
//index.js
Page({
  testFunction: function() {
    console.log('Trigger testFunction from Button')
  }
})
<!-- index.wxml -->
<button bindtap="testFunction">OK</button>
```



触发函数 (example 2)

```
//index.js
Page({
  myToast: function() {
    wx.showToast({
      title: 'SUCCESS'
    })
  }
})
<!-- index.wxml -->
<button bindtap="myToast">Show Success Toast</button>
```

[showToast API文档](#)



WXML一个高级视图层

可以在 `<view>` 和 `<block>` 上使用特殊属性

1. **wx:for** control属性：绑定数组（array）
2. **wx:if** 条件属性：绑定语句



wx:for

Example 1: 简单版本（`item` 是默认的）

```
<!-- index.wxml -->  
<view wx:for="{{['Restaurant 1','Restaurant 2','Restaurant 3']}}"  
  <view>{{item}}</view>  
</view>
```



Example 2: 帶有自定义index和item的完整版

```
<view wx:for="{{['Restaurant 1','Restaurant 2','Restaurant 3']}"
wx:for-index="index" wx:for-item="restaurant">
  <view>{{index}}: {{restaurant}}</view>
</view>
```



wx:for

还可以直接从页面的数据中获取数据！

```
//index.js
Page({
  data: {
    thingsIcoded:
    [
      { module: "Programming", name: "Black Jack" },
      { module: "OOP", name: "Food Delivery" },
      { module: "Databases", name: "Hacker News" },
      { module: "Front", name: "Chatroom" },
      { module: "WeChat", name: "FMC" }
    ]
  },
})
<!-- index.wxml -->
<view wx:for="{{thingsIcoded}}" wx:for-item="thing">
  <view>
    {{thing.module}} | {{thing.name}}
  </view>
</view>
```



wx:if

可以使用wx:if隐藏或显示卡片组件

```
<view wx:if="{{true}}">...</view>  
<view wx:if="{{false}}">...</view>
```



wx:if

逻辑可以直接写在WXML文件中

```
<view wx:if="{{true}}">Fuck My Code 1</view>  
<view wx:if="{{1 === 1}}">Fuck My Code 2</view>  
<view wx:if="{{1 === 2}}">Fuck My Code 3</view>
```



wx:if

还可以直接从页面的数据对象获取语句

```
//index.js
Page({
  data: {
    trueStatement: true,
    falseStatement: false
  }
})
<!-- index.wxml -->
<view wx:if="{{trueStatement}}">Fuck My Code 1</view>
<view wx:if="{{falseStatement}}">Fuck My Code 2</view>
```



Live code 3: 改善设计（添加添加多个卡片）

在stories页面中显示多个stories，而不重复相同的WXML标记。



全局数据（Global Data）

在页面中存储数据比较mainstream

- 所有Javascript页面可以从 `app.js` 文件访问全局数据（`globalData`）对象
- 但WXML文件不可直接访问全局数据（`globalData`） ...

```
//app.js
App({
  globalData: {
    userInfo: { nickName: "salmon", gender: 1 }
  }
})
//index.js
let app = getApp()

Page({
  data: { userInfo: app.globalData.userInfo }
})
<!-- index.wxml -->
<view>Hello {{userInfo.nickName}}</view>
```



在哪里存储数据？



遵循以下指南..

- 数据验证 缓存 (cache)
- 小程序中多次使用的数据 (ex: userId) 全局数据 (globalData)
- 获取仅与页面相关的数据 本地数据 page.js



Live code 4: 创建一个Post页面

使用一个表单在全局数据存储中添加新的FMC stories



Congratulations, you're a mini program developer!