# WeChat MP Documentation

Your new best friend, bookmark it!

# 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
```

# WeUI for 小程序

(Scan QR with WeChat) https://github.com/Tencent/weui-wxss


## Agenda

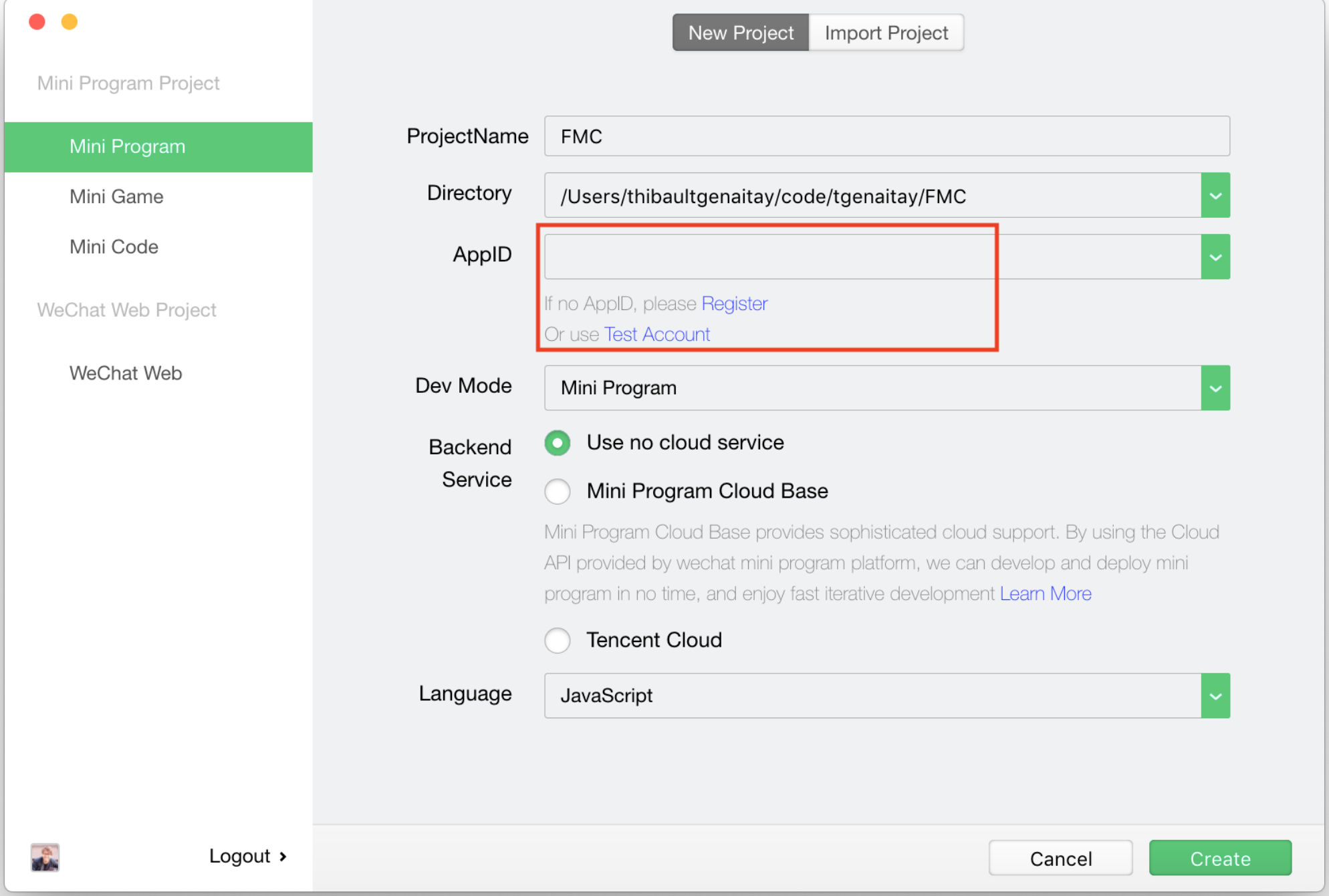
We will build 1 WeChat app in this course

**"F*** My Code"**

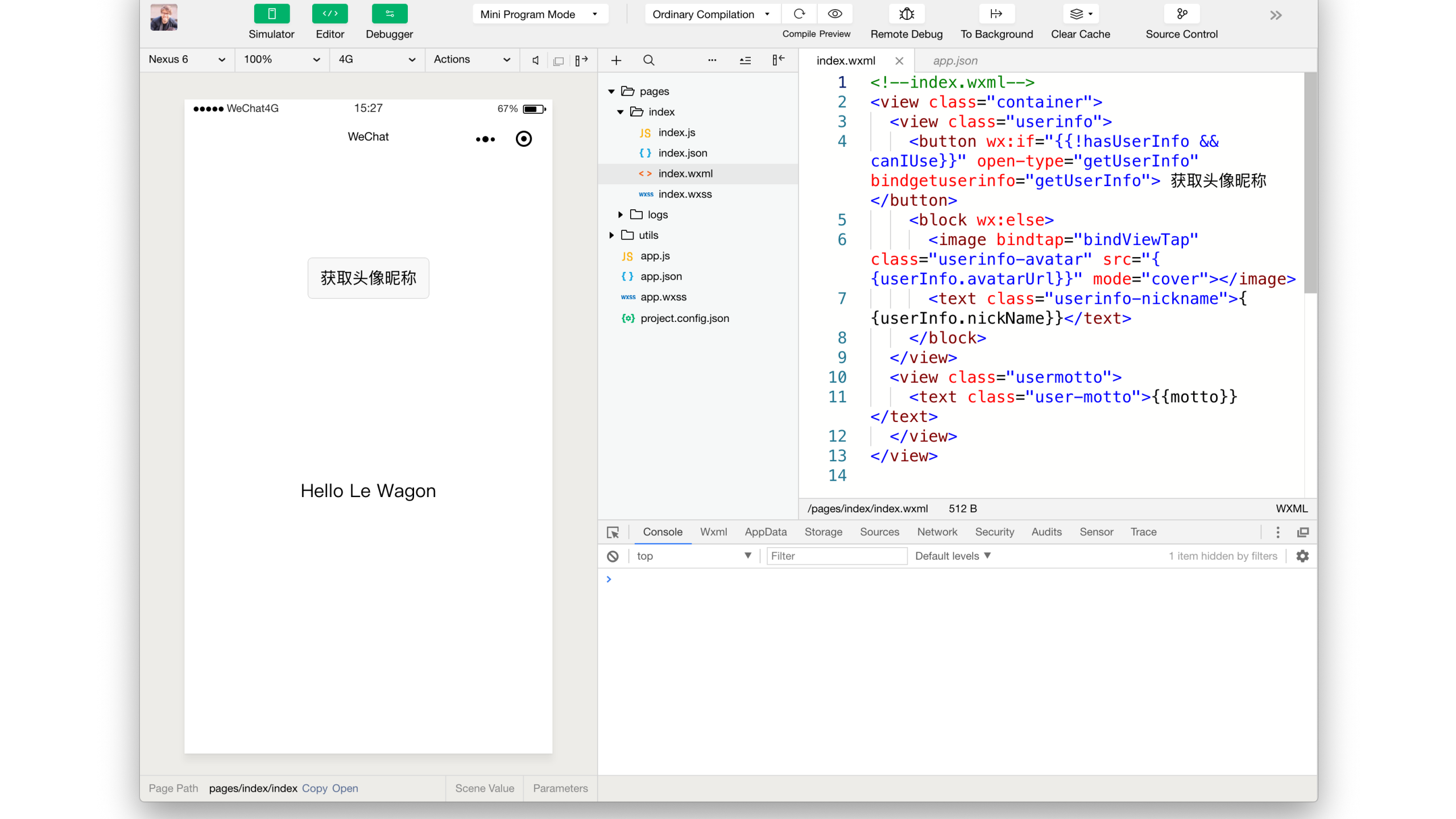
# Let's setup a Mini Program!

We'll use WeChat's developer tools moving forward.

Download the stable build now


Mini Program Project
Mini Program
Mini Game
Mini Code
WeChat Web Project
WeChat Web
Logout
New Project
Import 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

Setup with a tourist appID


Simulator
Editor
Debugger
Mini Program Mode
Ordinary Compilation
Compile
Preview
Remote Debug
To Background
Clear Cache
Source Control
Nexus 6
100%
4G
Actions
WeChat4G
15:27
67%
WeChat
获取头像昵称
Hello Le Wagon
pages
index
index.js
index.json
index.wxml
index.wxss
logs
utils
app.js
app.json
app.wxss
project.config.json
index.wxml
app.json
1 <!--index.wxml-->
2 <view class="container">
3 <view class="userinfo">
4 <button wx:if="{{!hasUserInfo && canIUse}}" open-type="getUserInfo" bindgetuserinfo="getUserInfo"> 获取头像昵称 </button>
5 <block wx:else>
6 <image bindtap="bindViewTap" class="userinfo-avatar" src="{{userInfo.avatarUrl}}" mode="cover"></image>
7 <text class="userinfo-nickname">{{userInfo.nickName}}</text>
8 </block>
9 </view>
10 <view class="usermotto">
11 <text class="user-motto">{{motto}}</text>
12 </view>
13 </view>
14
/pages/index/index.wxml 512 B
WXML
Console
Wxml
AppData
Storage
Sources
Network
Security
Audits
Sensor
Trace
top
Filter
Default levels
1 item hidden by filters
Page Path pages/index/index Copy Open
Scene Value
Parameters

# Quick tour

## Code structure 🤔

There are 4 file types in a WeChat mini program project:

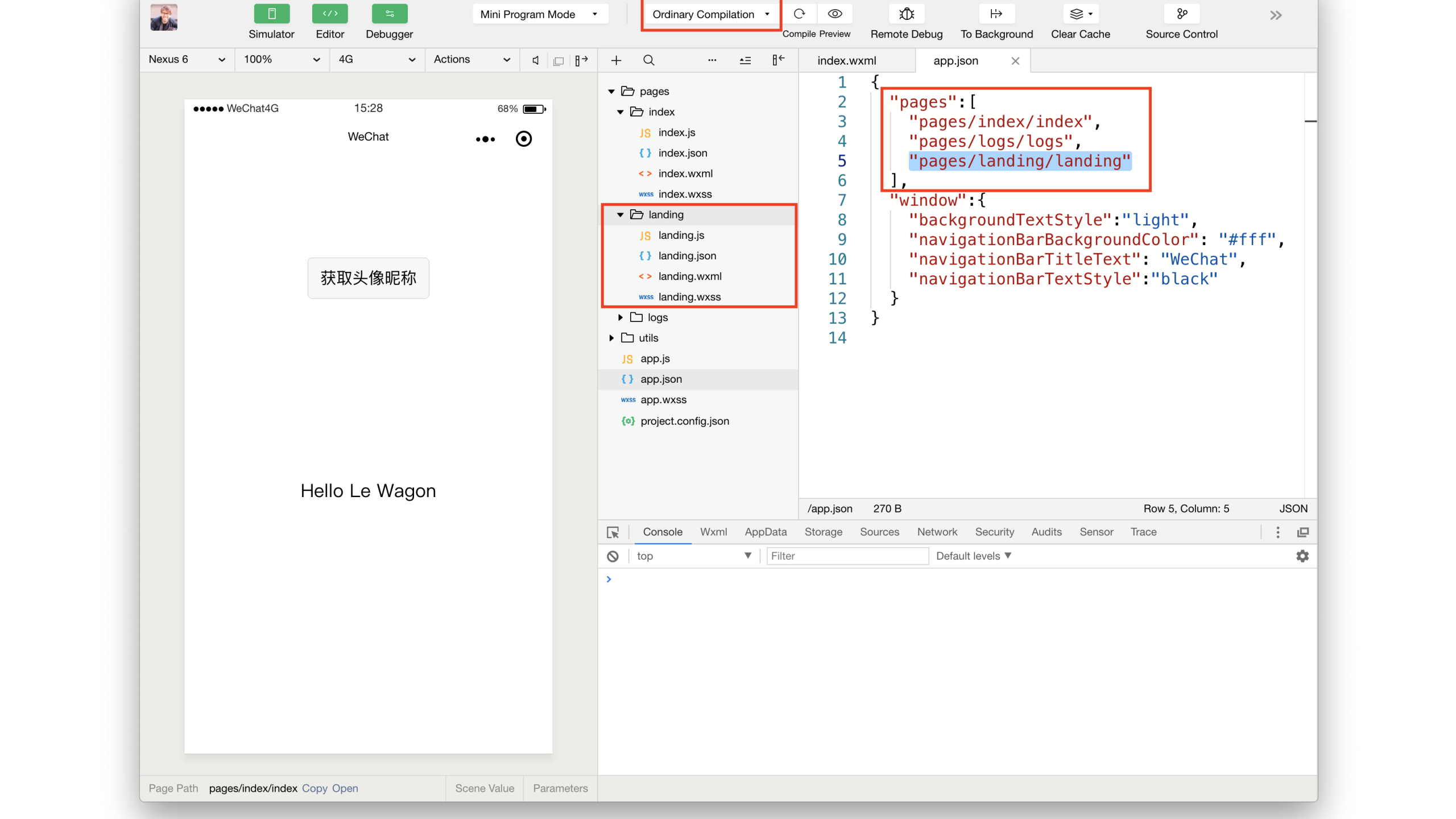
- `.wxml` same as HTML
- `.wxss` same as CSS
- `.js` logic
- `.json` configuration

## Your app is ruled by the following files:

- `app.js` Mini Program functions
- `app.json` Mini Program configuration
- `app.wxss` Global CSS stylesheet

## How to create a new page

Mandatory page files ( `.wxml` , `.wxss` , `.js` , and `.json` ) can be generated by adding a new route inside `app.json`


Simulator
Editor
Debugger
Mini Program Mode
Ordinary Compilation
Compile
Preview
Remote Debug
To Background
Clear Cache
Source Control
Nexus 6
100%
4G
Actions
WeChat4G
15:28
68%
WeChat
获取头像昵称
Hello Le Wagon
pages
index
index.js
index.json
index.wxml
index.wxss
landing
landing.js
landing.json
landing.wxml
landing.wxss
logs
utils
app.js
app.json
app.wxss
project.config.json
index.wxml
app.json
{
"pages":[
"pages/index/index",
"pages/logs/logs",
"pages/landing/landing"
],
"window":{
"backgroundTextStyle":"light",
"navigationBarBackgroundColor": "#fff",
"navigationBarTitleText": "WeChat",
"navigationBarTextStyle":"black"
}
}
/app.json 270 B
Row 5, Column: 5
JSON
Console
Wxml
AppData
Storage
Sources
Network
Security
Audits
Sensor
Trace
top
Filter
Default levels
Page Path pages/index/index Copy Open
Scene Value
Parameters

Tip: You can customize the startup page with a new compilation mode.

## WXML syntax (same same but different)

| WXML tags | HTML tags | Examples |
| --- | --- | --- |
| <view> |  | <view class="header">…</view> |
| <navigator> | <a> | <navigator url="/pages/about/about”>…</navigator> |
| <image> | <img> | <image src="/image/logo.png"></image> |
| <text> | <p> | <text> … </text> |

## JSON files for configuration

- Customize your app (Eg: app/page title, navigation bar color)
- Setup tabs
- Setup components

See all options here

## Live code 1: Landing page 💪

We'll use the Banner Component from Le Wagon UI.

## Live code 2: Stories page 💪

Save time using Le Wagon's card component (but no need of a product image).

Not challenging enough? It was just the beginning...

# Let's make our pages alive!

Framework notions:

- Life cycle
- Data stores
- Data binding
- Logic control

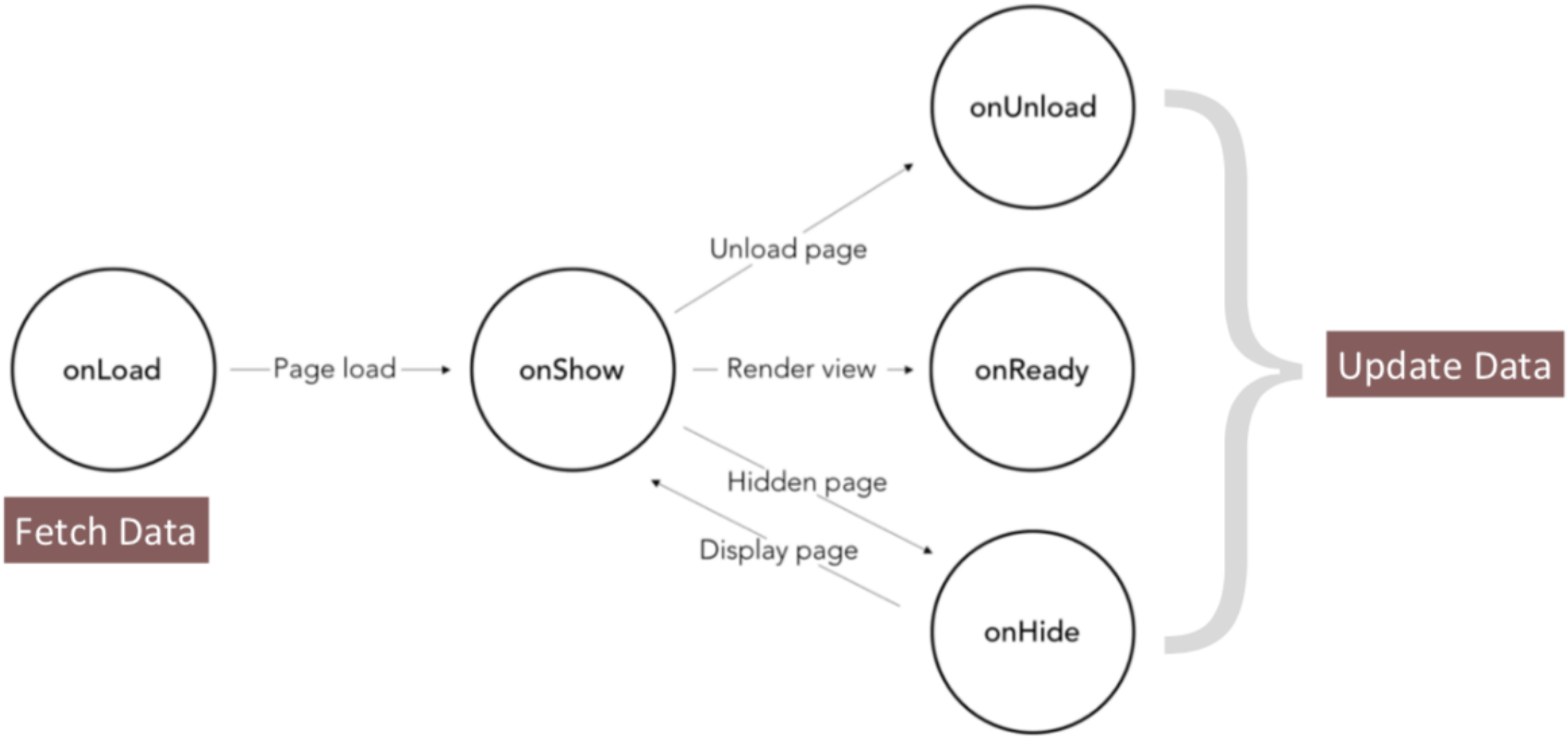
## Look at the JS structure

- 1 main function: **Page({...})** or **App({...})**
- 1 data store: **data: {key: value}** or **globalData: {key: value}**
- Lifecycle functions: **onLoad**, **onLaunch**,...
- Custom functions

## What is the life cycle of an app?

We can **console.log** the name of each function to see in which order they are called. Try with `onLoad`, `onShow`, `onReady` ... see the difference.

Life cycle functions = trigger code at specific time in the application



## Example

We can create functions and trigger them in the onReady life cycle method

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

To access the function, do not forget the syntax: `this.functionName()`

## Where do we store data?

1. **local data**: lives only inside a page
2. **global data**: shared across the whole app
3. **cache**: persists in your user's phone
4. **server**: through APIs!

## View local data inside WXML

We can store data inside a JS file and access it in WXML. This is called **data binding**.

```
//index.js
Page({
    data: { name: 'Allen' }
})
```

```
<!-- index.wxml -->
<text> My name is {{name}}</text>
```

WXML allows much more than HTML: it's a **"templating language"**!

## setData({})

We can update some value in our local data storage

```
//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

We can access the local data inside our JS code

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

## Triggering a function (example 1)

We can trigger a function from our page `.wxml` file by adding the **bindtap** argument to an element

```
//index.js
Page({
    testFunction: function() {
        console.log('Trigger testFunction from Button')
    }
})
```

```
<!-- index.wxml -->
<button bindtap="testFunction">OK</button>
```

## Triggering a function (example 2)

```
//index.js
Page({
  myToast: function() {
      wx.showToast({
          title: 'SUCCESS'
      })
  }
})
```

```
<!-- index.wxml -->
<button bindtap="myToast">Show Success Toast</button>
```

showToast API documentation

## WXML is an advanced view layer

We can use special attributes on `<view>` and `<block>`

1. **wx:for** control attribute: bind an array
2. **wx:if** conditional attribute: bind a statement

## wx:for

Example 1: Simple version ( `item` is default)

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

Example 2: Full version with custom index and item

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

## wx:for

We can also take the data directly from the page's data!

```
//index.js
Page({
  data: {
    thingsIcoded:
    [
        { module: "Programming", name: "Black Jack" },
        { 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

We can use **wx:if** to hide or show card components

```
<!-- index.wxml -->
<view wx:if="{{true}}">...</view>
<view wx:if="{{false}}">...</view>
```

## wx:if

The logic can be directly in the WXML file

```
<!-- index.wxml -->
<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

We can also take statements directly from the page's data object

```
//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: Improve the view (add multiple cards) 💪

We want to show more than one story in the **stories page** without repeating the same WXML markup.

## Global Data

Because storing in pages is too mainstream

- Every Javascript page can access the **globalData** object from `app.js`
- But WXML cannot access directly your **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>
```

## Where to store your data?

## Follow these guidelines..

- Data for authentication 👉 **cache**
- Data used everywhere in the app (ex: userId) 👉 **global Data**
- Data relevant only to the page 👉 **local Data page.js**

## Live code 4: Create a Post Page 💪

We'll use a form to add new FMC stories in a global data storage.

# Congratulations, you're a mini program developer!