# 3.1 网络接口简介：
# 如何在小程序中封装 wx.request 请求？（一）

极客时间

扫码试看/订阅

《微信小程序全栈开发实战》视频课程

RequestTask wx.request(Object object)

RequestTask

RequestTask.abort()

RequestTask.onHeadersReceived(function callback)

RequestTask.offHeadersReceived(function callback)

```
login(e) {
  ...
  const requestLoginApi = (code)=>{
    wx.request({
      url: 'http://localhost:3000/user/wexin-login2',
      ...
      success(res) {
        let token = res.data.data.authorizationToken
        wx.setStorageSync('token', token)
        onUserLogin(token)
      }
    })
  }
  const onUserLogin = (token)=>{
    getApp().globalData.token = token
    ...
  }
  wx.checkSession({
    success () {
      let token = wx.getStorageSync('token')
      if (token) onUserLogin(token)
    },
    fail () {
      wx.login({
        success(res0) {
          if (res0.code) {
            requestLoginApi(res0.code)
          }
        }
      })
    }
  })
}
```

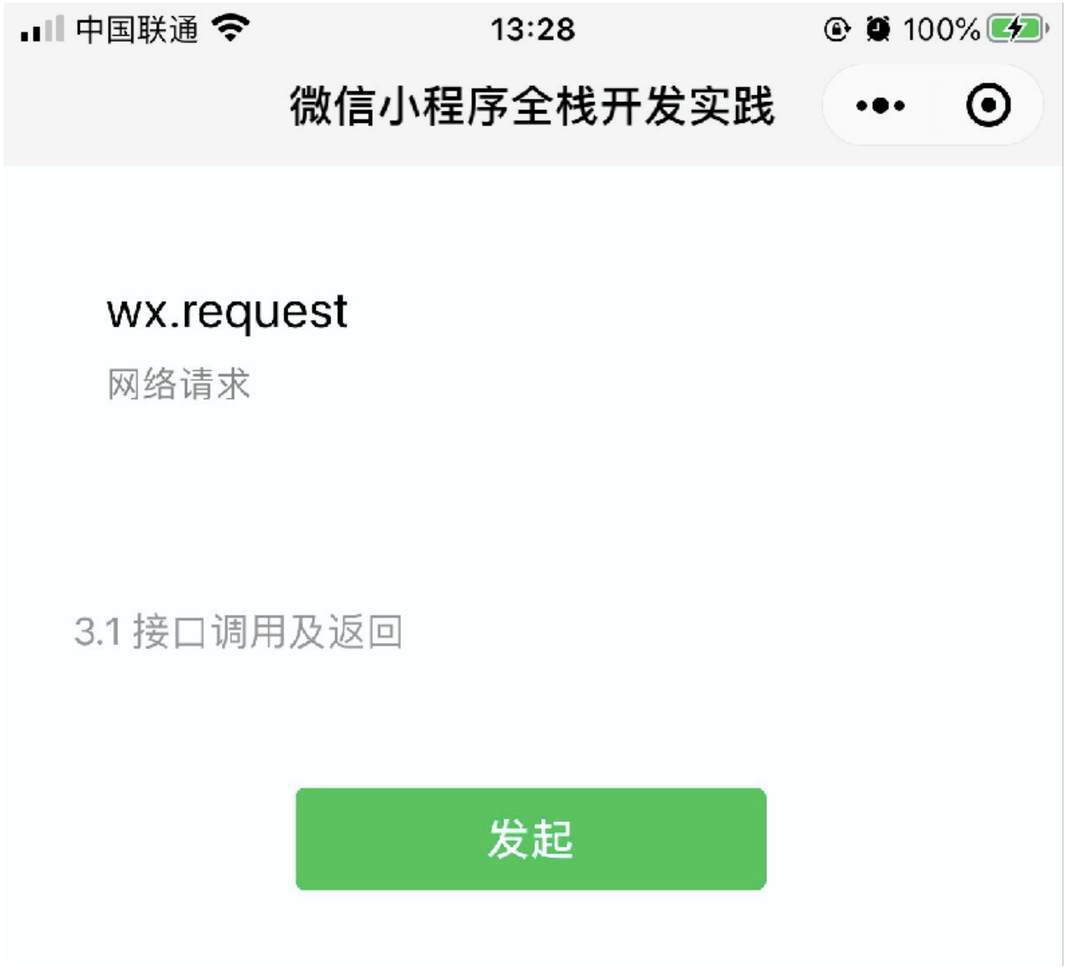
参数是一个对象

都有success、fail、complete三个回调属性

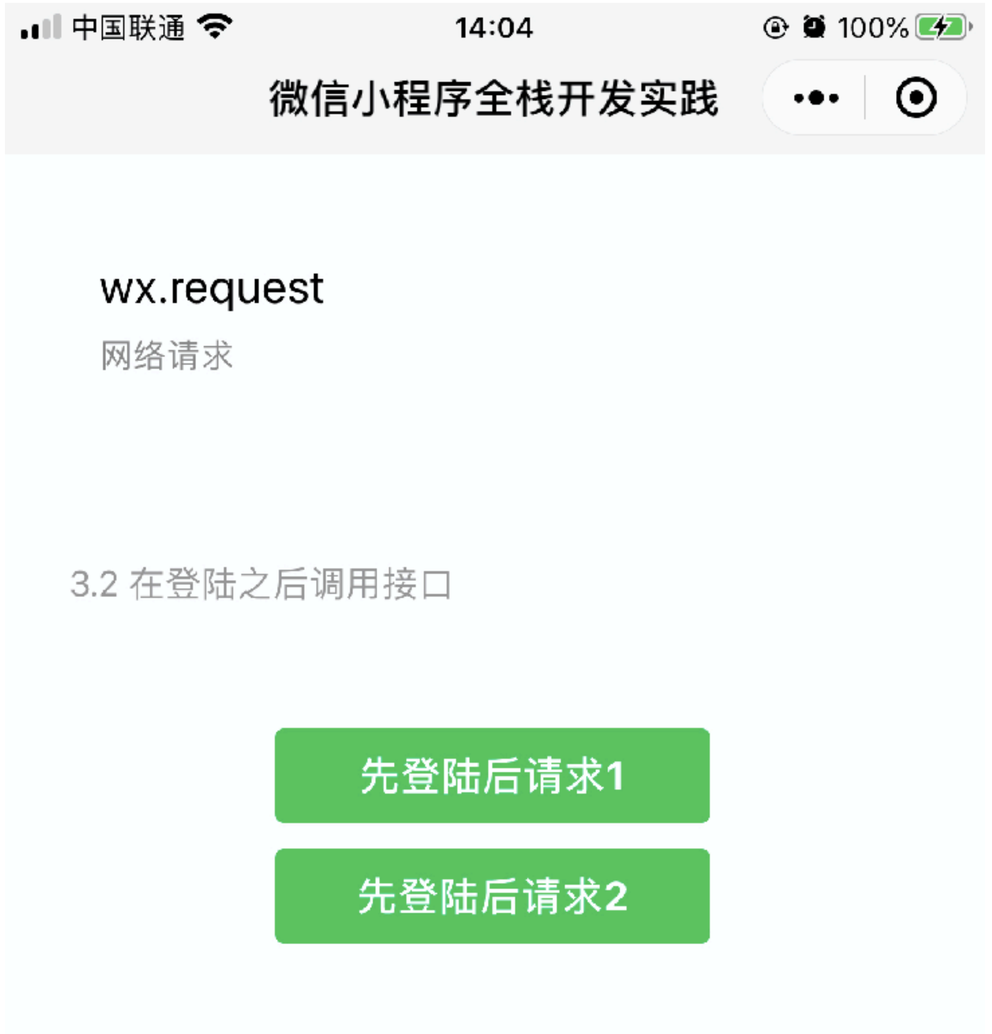
{errMsg: "request:ok"...}

{errMsg: "request:fail"...}

{errMsg: "request:fail abort"...}



# 3.2 网络接口简介：
# 如何在小程序中封装 wx.request 请求？（二）


中国联通
14:04
100%
微信小程序全栈开发实践
wx.request
网络请求
3.2 在登陆之后调用接口
先登陆后请求1
先登陆后请求2

复用原代码，在登陆后直接调用接口

如果已经登录，则不需要重复登陆

再加一个接口调用，将登录逻辑剥离为一个方法

模块化，在其它页面中也可以使用

# 复用原代码，在登陆后直接调用接口

如果已经登录，则不需要重复登陆

# 再加一个接口调用，将登录逻辑剥离为一个方法

# 模块化，在其它页面中也可以使用

```
const login = () => {
  wx.login({
    success(res0) {
      if (res0.code) {
        requestLoginApi(res0.code)
      ...
    }
  })
}
```

# 3.3 网络接口简介：
# 如何在小程序中封装 wx.request 请求？（三）

```
function func1(){
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      //reject('hi')
      reject('error')
    }, 200);
  })
}
let promise1 = func1()
promise1.then((res) => {
  console.log(res)
},(err)=>{
  console.log(err)
});
```

输出:

```
error
```

Promise.prototype.then()

Promise.prototype.catch()

Promise.prototype.finally()

Promise.all()

Promise.any()

Promise.race()

Promise.allSettled()

promise.then(onResolved[, onRejected])

# promise.catch(onRejected)

# promise.finally(onFinally)

Promise.all()

Promise.any()

Promise.race()

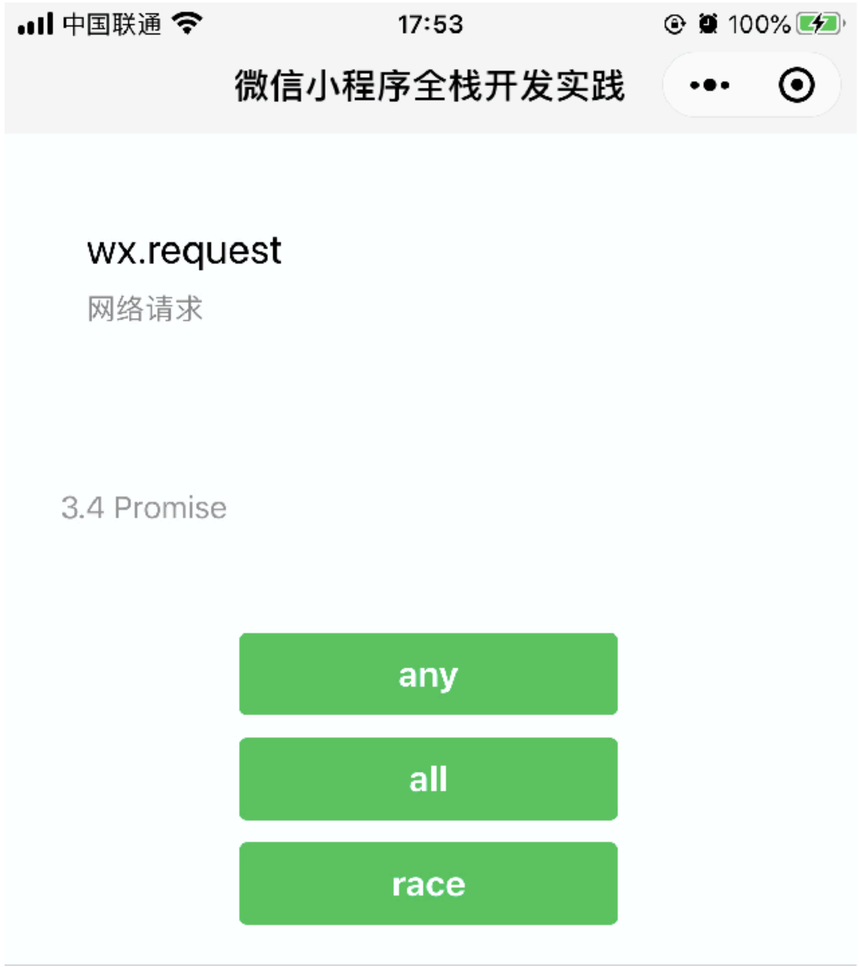
Promise.allSettled()

# 3.4 网络接口简介：
# 如何在小程序中封装 wx.request 请求？（四）

Promise.any()

Promise.all()

Promise.race()



# any

API Promise化链接：
https://developers.weixin.qq.com/miniprogram/dev/extended/utils/api-promise.html

安装：
npm install --save miniprogram-api-promise

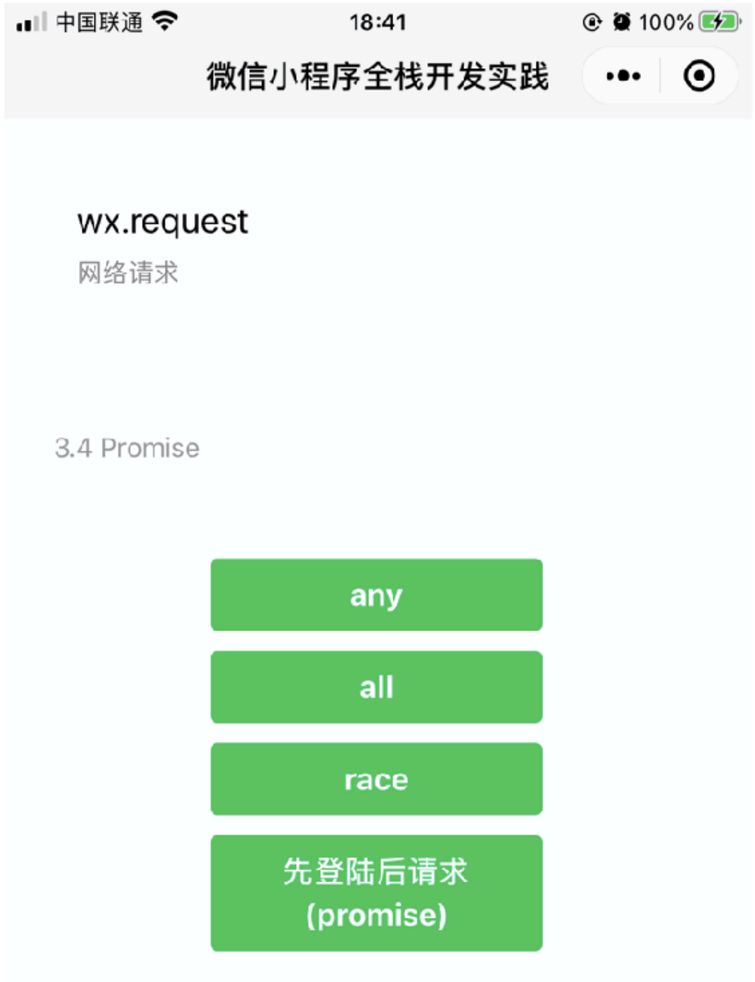
```
import { promisifyAll } from 'miniprogram-api-promise'
const wxp = {}
promisifyAll(wx, wxp)
//示例
wxp.getSystemInfo().then(console.log)
...
App({
  wxp:wxp
```

all

# race

# 3.5 网络接口简介：
# 如何在小程序中封装 wx.request 请求？（五）

```
getApp().wxp.checkSession().finally(res=>{
  let token = wx.getStorageSync('token')
  if (token) {
    onUserLogin(token)
  } else {
    login()
  }
})
```



```
function func1(x) {
  return new Promise(resolve => {
    setTimeout(() => {
      resolve(x)
    }, 100);
  });
}


async function f1() {
  var x = await func1(10)
  console.log(x)
}
f1()
```

输出:

```
10
```

```
async function func1() {
  try {
    var z = await Promise.reject("error")
    console.log('ok')
  } catch (e) {
    console.log(e)
  }
}
func1()
```

输出:

```
error
```

```
async function func1() {
  var y = await 20
  console.log(y)
}
func1()
```

输出：

```
20
```

# 3.6 网络接口简介：
# 如何在小程序中封装 we.request 请求？（六）

- 代码只要在页面中定义，就会被隐含执行
- 关于 Promise 的 catch

# 代码只要在页面中定义，就会被隐含执行

```
// 模拟一个 any 方法
Promise.any = (arr)=>{
  let numTotal = arr.length
  let numSettled = 0
  let resolved = false
  return new Promise((resolve, reject)=>{
    for(let j=0;j<numTotal;j++){
      let p = arr[j]
      p.then(res=>{
        resolved = true
        resolve(res)
      },err=>{
        console.log('any err', err);
      }).finally((res)=>{
        numSettled++
        if (numSettled >= numTotal && !resolved) reject('all failed')
      })
    }
  })
}
```

miniprogram/lib/any.js:

```
Promise.any = (arr)=>{
  ...
}
```

miniprogram/app.js:

```
import "./lib/any
```

# 关于 Promise 的 catch

```
any(e){
  const app = getApp()
  let promise1 = app.wxp.request({url:...'}).catch(console.log),
      promise2 = app.wxp.request({url:...'}).catch(console.log),
      promise3 = app.wxp.request({url:...'}).catch(console.log)
  let promise = Promise.any([promise1,promise2,promise3])
  ...
}


Promise.any = (arr)=>{
  ...
      p.then(res=>{
        resolved = true
        resolve(res)
      },err=>{
        console.log('any err', err);
      }).finally((res)=>{
        ...
      })
    }
  })
```

any promise res undefined

```
any(e){
  const app = getApp()
  let promise1 = app.wxp.request({url:...'}).catch(err=>{
        console.log(err)
        throw err
      }),
      ...
}
Promise.any = (arr)=>{
  ...
      },err=>{
        console.log('any err', err);
        reject(err)
      }).finally((res)=>{
        ...
    }
  })
}
```

# wxp 模块化

```
import wxp from './lib/wxp'
```

# 3.7 网络接口简介：
# 如何在小程序中封装 we.request 请求？（七）

# EventChannel

EventChannel.emit(string eventName, any args)
EventChannel.on(string eventName, EventCallback fn)
EventChannel.once(string eventName, EventCallback fn)
EventChannel.off(string eventName, EventCallback fn)

```
wx.navigateTo({
  url: 'test?id=1',
  events: {
    acceptDataFromOpenedPage: function(data) {
      console.log(data)
    }
    ...
  },
  success: function(res) {
    res.eventChannel.emit('acceptDataFromOpenerPage', { data: 't' })
  }
})
test:
Page({
  onLoad: function(option){
    const eventChannel = this.getOpenerEventChannel()
    eventChannel.emit('acceptDataFromOpenedPage', {data: 'test'});
    eventChannel.on('acceptDataFromOpenerPage', function(data) {
      console.log(data)
    })
  }
})
```

1. 使用 pop-up 组件，实现一个底部滑出的登陆面板

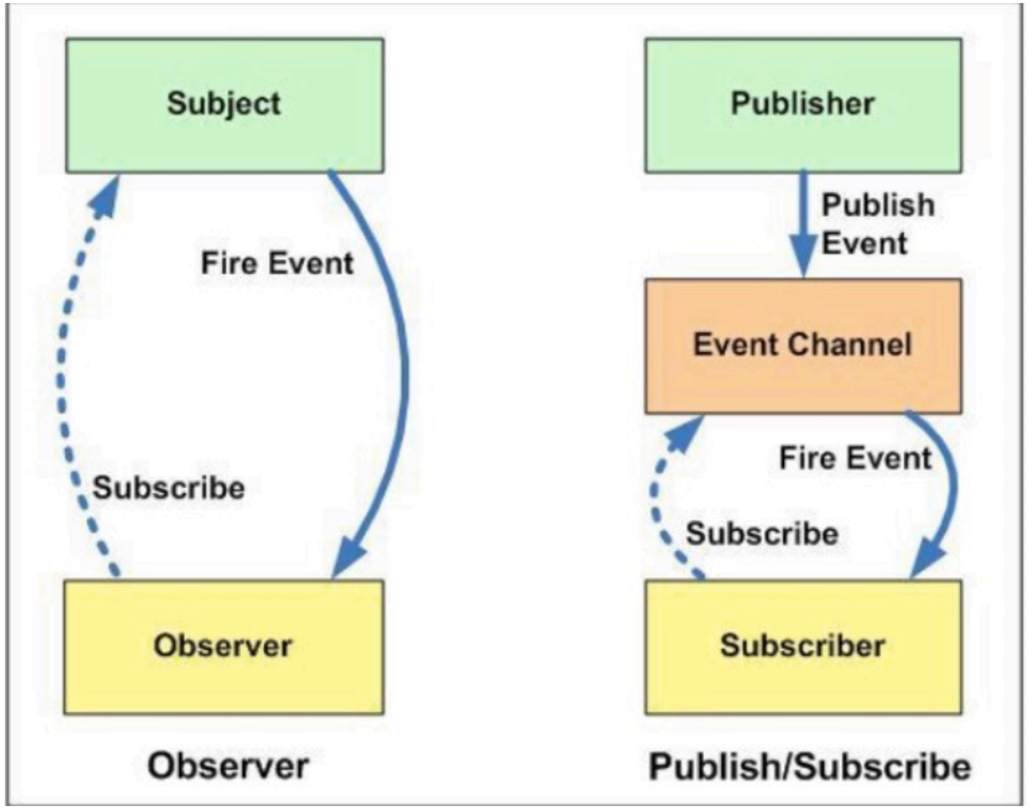
2. 自定义实现一个观察者模式对象 Event

3. 在 wxp 组件中扩展实现 request3 接口

# 使用 pop-up 组件，实现一个底部滑出的登陆面板

# 3.8 网络接口简介：
# 如何在小程序中封装 we.request 请求？（八）

# 重新截图


Subject
Fire Event
Subscribe
Observer
Observer
Publisher
Publish
Event
Event Channel
Fire Event
Subscribe
Subscriber
Publish/Subscribe

# 3.9 网络接口简介：
# 如何在小程序中封装 we.request 请求？（九）

# 在 wxp 组件中扩展实现 request3 接口

# wx.request 经 Promise 封装后如何拿到 RequestTask 实例？

```
class Request {
  constructor(parms) {
    ...
    this.requestTask = null;
  }
  request(method, url, data) {
    const vm = this;
    return new Promise((resolve, reject) => {
      this.requestTask = wx.request({
        ...
        success(res) {
          resolve(res);
        },
        fail() {
          reject({...});
        }
      });
    });
  }
  abort() {
    if (this.requestTask) {
      this.requestTask.abort();
    }
  }
}
module.exports = Request;
```

miniprogram/node_modules/miniprogram-api-promise/src/promise.js:

```
function _promisify(func) {
  if (typeof func !== 'function') return fn
  return (args = {}) =>{
    return new Promise((resolve, reject)=> {
      let rtnObj = func(
        Object.assign(args, {
          success: resolve,
          fail: reject
        })
      )
      if (args.onReturnObject) args.onReturnObject(rtnObj)
    })
  }
}
```

# 3.10 tabBar 组件：如何自定义实现一个 tabBar？（一）

1. 使用系统默认的 tabBar
2. 使用系统自定义的方式
3. 使用 weui 组件库
4. 基于组件自定义

# 使用默认的系统 tabBar

```
"tabBar": {
  "list": [
    {
      "pagePath": "pages/3-10/index",
      "iconPath":"components/tab-bar/component.png",
      "selectedIconPath":"components/tab-bar/component-on.png",
      "text": "首页"
    },
    {
      "pagePath": "pages/3-10/custom/index",
      "iconPath":"components/tab-bar/component.png",
      "selectedIconPath":"components/tab-bar/component-on.png",
      "text": "自定义"
    }
  ]
}
```

无效：<navigator open-type="navigate" url="/pages/3-10/index2">navigate to inex2</navigator>

有效：<navigator open-type="switchTab" url="/pages/3-10/index2">switchtab to inex2</navigator>

重新截图

```
var data = wx.getMenuButtonBoundingClientRect()
console.log('胶囊按钮', data);
console.log('胶囊按钮高度：', data.height) //32
console.log('上边界坐标：', data.top) //24
console.log('下边界坐标：', data.bottom) //56


let res = wx.getSystemInfoSync()
console.log("screenHeight", res.screenHeight);
console.log("statusBarHeight", res.statusBarHeight);
console.log("windowHeight", res.windowHeight)
```

# 小程序原生 tabbar 选中状态背有背景色，为什么?


【阎良甜瓜】果肉细腻，自...
48小时发货
销量：166箱
¥29.9 原价¥59.9
马上抢
【洛川苹果】限量活动
72小时发货 遇下雨天顺延
马上抢
【泰国山竹】秒杀活动
首页
购物车
会员中心

# 3.11 tabBar 组件：如何自定义实现一个 tabBar？（二）


中国联通
11:13
44%
微信小程序全栈开发实践
tabBar
页面导航
3.10 原生tabBar
navigate to inex2
switchtab to inex2
navigate to inex3
index0
index1
自定义2

重新截图

```
"tabBar": {
    "custom":true
...
custom-tab-bar/index
```

```
list: [{
  pagePath: "/pages/3-10/index",
  iconPath: "/components/tab-bar/component.png",
  selectedIconPath: "/components/tab-bar/component-on.png",
  text: "index",
  iconClass:"icon-homefill",
  iconTopClass:""
}, {
  pagePath: "/pages/3-10/index2",
  iconPath: "/components/tab-bar/component.png",
  selectedIconPath: "/components/tab-bar/component-on.png",
  text: "index",
  iconClass:"cu-btn icon-add bg-green shadow",
  iconTopClass:"add-action"
},{
  pagePath: "/pages/3-10/index3",
  iconPath: "/components/tab-bar/component.png",
  selectedIconPath: "/components/tab-bar/component-on.png",
  text: "自定义",
  iconClass:"icon-my",
  iconTopClass:""
}]
```

```
onShow: function () {
  if (typeof this.getTabBar === 'function' &&
  this.getTabBar()) {
  this.getTabBar().setData({
    selected: 0
  })
  }
},
```

```
.cu-bar.tabbar {
  padding: 0;
  z-index: 0;
  height: calc(130rpx + env(safe-area-inset-bottom) / 2);
  padding-bottom: calc(env(safe-area-inset-bottom) / 2);
}
```

# 3.12 tabBar 组件：如何自定义实现一个 tabBar？（三）

跳转方面

使用灵活

```
wxp: (wx.wxp = wxp),
globalData: (wx.globalData = {}),
globalEvent: (wx.globalEvent = new Event())
...
```

https://developers.weixin.qq.com/miniprogram/dev/extended/weui/tabbar.html

# 3.13 开放接口：在小程序中实现用户一键登陆？（一）

取消

微信授权登录

小程序将申请获取以下权限

- 获得你的公开信息（昵称、头像、地区及性别）

登陆

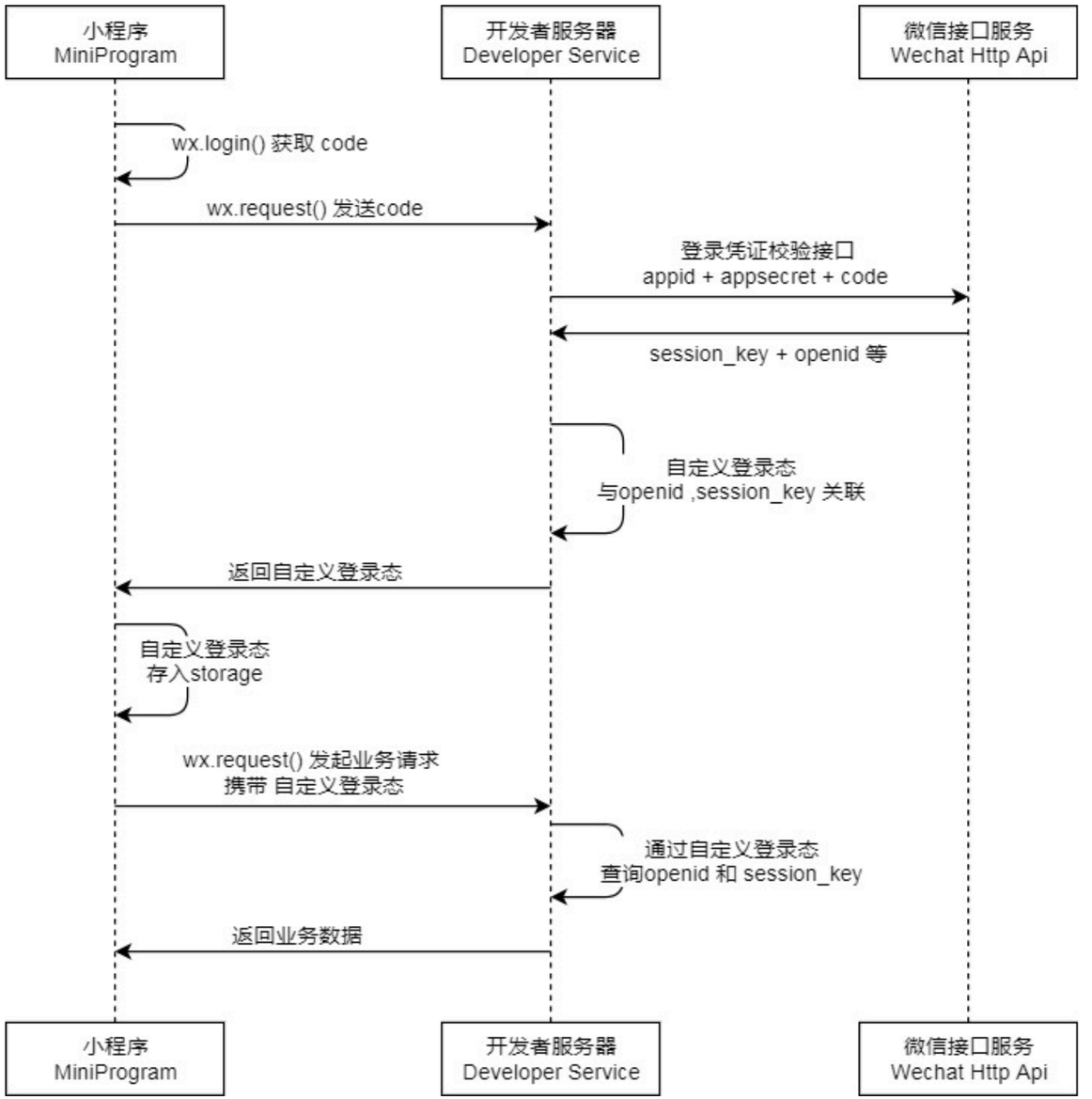
UI 还需要调整

有时候会服务器端出现解密错误

```
err Error: error:06065064:digital envelope
routines:EVP_DecryptFinal_ex:bad decrypt
    ...
  library: 'digital envelope routines',
  function: 'EVP_DecryptFinal_ex',
  reason: 'bad decrypt',
  code: 'ERR_OSSL_EVP_BAD_DECRYPT'
}
```

部分讨论 bad decrypt 的帖子：

- https://forum.leancloud.cn/t/aes/12944/14
- https://developers.weixin.qq.com/community/develop/doc/000a4e620e0040e75fb6772985b800
- https://developers.weixin.qq.com/community/develop/doc/bd11f741bdc90138ccdb51fa4d4b9896?_at=1559291397004


小程序
MiniProgram
开发者服务器
Developer Service
微信接口服务
Wechat Http Api
wx.login() 获取 code
wx.request() 发送code
登录凭证校验接口
appid + appsecret + code
session_key + openid 等
自定义登录态
与openid ,session_key 关联
返回自定义登录态
自定义登录态
存入storage
wx.request() 发起业务请求
携带 自定义登录态
通过自定义登录态
查询openid 和 session_key
返回业务数据
小程序
MiniProgram
开发者服务器
Developer Service
微信接口服务
Wechat Http Api

# 3.14 开放接口：在小程序中实现用户一键登陆？（二）

```
async login(e){
  ...
  try {
    await getApp().wxp.checkSession()
    token = wx.getStorageSync('token')
    if (token) {
      tokenIsValid = true
    }
  } catch (error) {}
  if (!tokenIsValid) {
    let res1 = await getApp().wxp.login()
    let code = res1.code
    let res = await getApp().wxp.request({
      url: 'http://localhost:3000/user/wexin-login2',
      method: 'POST',
      header: {
        'content-type': 'application/json'
      },
      data: {
        code,
        userInfo,
        encryptedData,
        iv
      }
    })
    console.log('登录接口请求成功', res.data)
    token = res.data.data.authorizationToken
    wx.setStorageSync('token', token)
  }
  getApp().globalData.token = token
  wx.showToast({
    title: '登陆成功了',
  })
  ...
}
```

复用旧的尚且有效的 sessionKey

在 token 保存当前用户信息

添加重试功能

token 持久化

# 复用旧的尚且有效的 sessionKey

# 在 token 保存当前用户信息

# 添加重试功能

# token 持久化

极客时间

扫码试看/订阅

《微信小程序全栈开发实战》视频课程