链滴

微信小程序 -mpvue

作者：Weixl

原文链接：https://ld246.com/article/1602155067617

来源网站：链滴

<img src="https://ld246.com/images/img-loading.svg" alt="" data-src="https://b3logfile
com/bing/20180214.jpg?imageView2/1/w/960/h/540/interlace/1/q/100">
<h2 id="微信小程序--mpvue">微信小程序——mpvue</h2>

<code>mpvue</code> 继承于 <code>Vue.js</code> 其技术规范和语法特点与 <code>Vu
.js</code> 保持一致。
官网地址：<a href="https://ld246.com/forward?goto=http%3A%2F%2Fmpvue.com%2Fm
vue%2F" target="_blank" rel="nofollow ugc">http://mpvue.com/mpvue/
开发者文档地址：<a href="https://ld246.com/forward?goto=https%3A%2F%2Fdevelopers
weixin.qq.com%2Fminiprogram%2Fdev%2Fcomponent%2Fbutton.html" target="_blank" rel=
nofollow ugc">https://developers.weixin.qq.com/miniprogram/dev/component/button.html<
a>
创建运行 <code>mpvue</code> 项目：
<pre><code class="highlight-chroma"><span class="highlight
cl"># 全局安装 vue-cli
$ npm install --gl
bal vue-cli

创建一个基于 m
vue-quickstart 模板的新项目
$ vue init mpvue
mpvue-quickstart my-project

# 安装依赖
$ cd my-project
$ npm install
# 启动构建
$ npm run dev
</code></pre>

启动 <code>微信开发者工具</code>， 引入刚启动搞得项目，就可以进行操作预览。

<h2 id="框架原理">框架原理</h2>

<code>mpvue</code> 保留了 vue 的核心方法，
<code>mpvue-template-compiler</code> 提供了将 vue 的模板语法转换到小程序的 wxml
语法的能力
修改了 vue 的建构配置，使之构建出符合小程序项目结构的代码格式： json/wxml/wxss/js 文件
/li>

<h2 id="生命周期函数">生命周期函数</h2>

同 vue， 不同的是会在小程序 <code>onReady</code> 后，再去触发 vue mounted 生命
期。
<pre><code class="highlight-chroma"><span class="highlight
cl">beforeCreate
created
beforeMount
mounted
beforeUpdate
updated
activated
deactivated
beforeDestroy

destroyed

除了 Vue 本身的命周期外，mpvue 还兼容了小程序生命周期，这部分生命周期钩子的来源于

</code></pre>

除了 Vue 本身的生命周期外，mpvue 还兼容了小程序生命周期，这部分生命周期钩子的来源于 微信小程序的 Page， 除特殊情况外，不建议使用小程序的生命周期钩子。
<pre><code class="highlight-chroma">app 部分：
onLaunch，初始化
onShow，当小程启动，或从后台进入前台显示
onHide，当小程从前台进入后台
onError起到错误听的作用。

page 部分：
onLoad，监听页加载
onShow，监听页显示
onReady，监听页初次渲染完成
onHide，监听页隐藏

onUnload，监听面卸载
onPullDownRefreh，监听用户下拉动作
onReachBottom 页面上拉触底事件的处理函数
onShareAppMessge，用户点击右上角分享
onPageScroll，页滚动
onTabItemTap，前是 tab 页时，点击 tab 时触发 （mpvue 0.0.16 支持）
</code></pre>

总体执行流程：
<pre><code class="highlight-chroma">beforeCreate—> created —> onLunch/onLoad --> onShow --> onReady -> beforeMounte --> mounted.....
</code></pre>

周期图：<img src="https://ld246.com/images/img-loading.svg" alt="img" data-src="http://b3logfile.com/file/2020/10/solofetchupload8575573403327853837-0141c9f3.jpeg?imageVi

w2/2/interlace/1/format/jpg">

<h2 id="注意事项-">注意事项：</h2>

几乎全支持 Vue 的模板语法。
不支持 <code>纯-HTML</code> ， <code>v-html</code> 指令不能用。
在插值表达式 <code>{{}}</code> 中不支持复杂的 JavaScript 渲染表达式：
<pre><code class="highlight-chroma"><span class="highlight
cl"><!-- 这种就不支持，建议写 computed -->
<p>{{ messa
e.split('').reverse().join('') }}</p>

<!-- 但写在 @ev
nt 里面的表达式是都支持的，因为这部分的计算放在了 vdom 里面 -->

     &lt
li v-for="item in list">

nbsp; <div @click="clickHandle(item, index, $event)">{{ item.value }}</p>
     &lt
/li>

</code></pre>

不支持 ElementUI 和 Vue-router
列表渲染： 只是需要注意一点，嵌套列表渲染，必须指定不同的索引！
<pre><code class="highlight-chroma"><span class="highlight
cl"><!-- 在这种嵌套循环的时候， index 和 itemIndex 这种索引是必须指定，且别名不能相同，
确的写法如下 -->
<template>
     &lt
ul v-for="(card, index) in list">

nbsp; <li v-for="(item, itemIndex) in card">

nbsp;   {{item.value}}

nbsp;
     &lt
/ul>
</template>
</code></pre>

原生组件上的事件绑定，需要以 <code>vue</code> 的事件绑定语法来绑定。

如 <code>bindchange="eventName"</code> 事件，需要写成 <code>@change="eventN
me"</code>

<h2 id="表单控件绑定-picker">表单控件绑定 picker</h2>

表单控件绑定 picker
picker 有多种选择形式，参考官网：<a href="https://ld246.com/forward?goto=https%3A%

F%2Fdevelopers.weixin.qq.com%2Fminiprogram%2Fdev%2Fcomponent%2Fpicker.html" targe
="_blank" rel="nofollow ugc">https://developers.weixin.qq.com/miniprogram/dev/compone
t/picker.html
select 组件用 picker 组件进行代替
默认 mode 格式：
<pre><code class="highlight-chroma"><span class="highlight
cl"><template>
  <div&gt

    &lt
picker @change="bindPickerChange" :value="index" :range="array">

nbsp;<view class="picker">

nbsp; 当前选择：{{array[index]}}

nbsp;</view>
    &lt
/picker>
  </div&g
;
</template>

<script>
export default {
  data () {
   return {
     i
dex: 0,
     a
ray: ['A', 'B', 'C']
   }
  },
  methods: {
   bindPick
rChange (e) {
     c
nsole.log(e)
     //
切换index位置
     t
is.index = e.mp.detail.value
   }
  }
}
</script>
</code></pre>

日期 mode=date 格式选择：
<pre><code class="highlight-chroma"><span class="highlight
cl"><template>
  <div&gt

    &lt

```
!-- value选中的日期， start有效开始日期， end有效结束的日期， 所选择的日期需要在start-end
间 --&gt;
</span></span><span class="highlight-line"><span class="highlight-cl">    &lt
picker mode="date" :value="date" start="2015-09-01" end="2017-09-01" @change="bindDa
eChange"&gt;
</span></span><span class="highlight-line"><span class="highlight-cl">    
nbsp;&lt;view class="picker"&gt;
</span></span><span class="highlight-line"><span class="highlight-cl">    
nbsp; 当前选择: {{date}}
</span></span><span class="highlight-line"><span class="highlight-cl">    
nbsp;&lt;/view&gt;
</span></span><span class="highlight-line"><span class="highlight-cl">    &lt
/picker&gt;
</span></span><span class="highlight-line"><span class="highlight-cl">  &lt;/div&g
;
</span></span><span class="highlight-line"><span class="highlight-cl">&lt;/template&gt;
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">&lt;script&gt;
</span></span><span class="highlight-line"><span class="highlight-cl">export default {
</span></span><span class="highlight-line"><span class="highlight-cl">  data () {
</span></span><span class="highlight-line"><span class="highlight-cl">   return {
</span></span><span class="highlight-line"><span class="highlight-cl">    
ate: '2020-05-01'
</span></span><span class="highlight-line"><span class="highlight-cl">   }
</span></span><span class="highlight-line"><span class="highlight-cl">  },
</span></span><span class="highlight-line"><span class="highlight-cl">  methods: {
</span></span><span class="highlight-line"><span class="highlight-cl">   bindDat
Change (e) {
</span></span><span class="highlight-line"><span class="highlight-cl">     c
nsole.log(e)
</span></span><span class="highlight-line"><span class="highlight-cl">     //
修改时间
</span></span><span class="highlight-line"><span class="highlight-cl">     t
is.date = e.mp.detail.value
</span></span><span class="highlight-line"><span class="highlight-cl">   }
</span></span><span class="highlight-line"><span class="highlight-cl">  }
</span></span><span class="highlight-line"><span class="highlight-cl">}
</span></span><span class="highlight-line"><span class="highlight-cl">&lt;/script&gt;
</span></span></code></pre>
</li>
</ul>
<h2 id="radio-group-组件">radio-group 组件</h2>
<ul>
<li>radio 使用 radio-group 组件代替：</li>
<li>代码如下：
<pre><code class="highlight-chroma"><span class="highlight-line"><span class="highlight
cl">&lt;template&gt;
</span></span><span class="highlight-line"><span class="highlight-cl">  &lt;div&gt

</span></span><span class="highlight-line"><span class="highlight-cl">    &lt
!-- radio-group 一组radio的集合， @change 为触发选中单选时调用的方法,会传入一个默认的 e
签对象 --&gt;
</span></span><span class="highlight-line"><span class="highlight-cl">    &lt
radio-group class="radio-group" @change="radioChange"&gt;
```

```
</span></span><span class="highlight-line"><span class="highlight-cl">     
nbsp;&lt;label v-for="(item, index) in items" :key="index"&gt;
</span></span><span class="highlight-line"><span class="highlight-cl">     
nbsp;  &lt;!-- value设置单选绑定搞得值，当选中后可以作为参数到method方法中 --&gt;
</span></span><span class="highlight-line"><span class="highlight-cl">     
nbsp;  &lt;!-- checked默认刚进来 选中的 --&gt;
</span></span><span class="highlight-line"><span class="highlight-cl">     
nbsp;  &lt;radio :value="item.name" :checked="item.checked"/&gt; {{item.value}}
</span></span><span class="highlight-line"><span class="highlight-cl">     
nbsp;&lt;/label&gt;
</span></span><span class="highlight-line"><span class="highlight-cl">     &lt
/radio-group&gt;
</span></span><span class="highlight-line"><span class="highlight-cl">   &lt;/div&g
;
</span></span><span class="highlight-line"><span class="highlight-cl">&lt;/template&gt;
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">&lt;script&gt;
</span></span><span class="highlight-line"><span class="highlight-cl">export default {
</span></span><span class="highlight-line"><span class="highlight-cl">  data () {
</span></span><span class="highlight-line"><span class="highlight-cl">     return {
</span></span><span class="highlight-line"><span class="highlight-cl">       i
ems: [
</span></span><span class="highlight-line"><span class="highlight-cl">     
nbsp;  {name: 'USA', value: '美国'},
</span></span><span class="highlight-line"><span class="highlight-cl">     
nbsp;  {name: 'CHN', value: '中国', checked: 'true'},
</span></span><span class="highlight-line"><span class="highlight-cl">     
nbsp;  {name: 'BRA', value: '巴西'},
</span></span><span class="highlight-line"><span class="highlight-cl">     
nbsp;  {name: 'JPN', value: '日本'},
</span></span><span class="highlight-line"><span class="highlight-cl">     
nbsp;  {name: 'ENG', value: '英国'},
</span></span><span class="highlight-line"><span class="highlight-cl">     
nbsp;  {name: 'TUR', value: '法国'}
</span></span><span class="highlight-line"><span class="highlight-cl">       ]
</span></span><span class="highlight-line"><span class="highlight-cl">     }
</span></span><span class="highlight-line"><span class="highlight-cl">  },
</span></span><span class="highlight-line"><span class="highlight-cl">  methods: {
</span></span><span class="highlight-line"><span class="highlight-cl">     radioCha
ge (e) {
</span></span><span class="highlight-line"><span class="highlight-cl">       c
nsole.log(e)
</span></span><span class="highlight-line"><span class="highlight-cl">       c
nsole.log('radio发生change事件，携带value值为：', e.target.value)
</span></span><span class="highlight-line"><span class="highlight-cl">     }
</span></span><span class="highlight-line"><span class="highlight-cl">  }
</span></span><span class="highlight-line"><span class="highlight-cl">}
</span></span><span class="highlight-line"><span class="highlight-cl">&lt;/script&gt;
</span></span></code></pre>
</li>
</ul>
<h2 id="目录介绍">目录介绍</h2>
<ul>
<li>vue 页面中的 属性介绍 主要包含 <code>&lt;template&gt;  &lt;script&gt;  &lt;style&gt;</c
```

de> 三个标签。
template 中可以使用 html 常用标签，无法使用小程序专属标签。但是在对页面进行编译后，在ist 中会生成小程序专属标签。
script 中，包含 config， data， components， methods 和 各种生命周期函数

config 是对小程序的一些配置，最后会生成为 json 文件
Data 是对数据的初始化，可以与页面进行绑定。Data 中使用 return 将变量名包起来，使其只当前组件中生效，不会造成变量污染。
methods 是方法和函数的集合，里面可以生成一些函数和方法。并向后台发送请求

src/main.js 文件（全局的配置文件）：
<pre><code class="highlight-chroma">import Vue from 'vue'
import App from '/App'

var Fly = require('fyio/dist/npm/wx')
var fly = new Fly()
// 添加全局配置、截器等
Vue.prototype.$http = fly
// 将fly实例挂在vu原型上

// 阻止启动生产消，阻止更多的消息产生，增加App的沉余体积
Vue.config.productionTip = false

// 设置 mpvue 的型
App.mpType = 'app'
// 进行注册App
const app = new ue(App)
// 挂机app实例 在sc根目录的 main.js中，设置的是整个 小程序App
app.$mount()
</code></pre>

src/app.json（全局的页面配置）
<pre><code class="highlight-chroma">
</code></pre>

<h2 id="页面跳转">页面跳转</h2>

小程序自带五中页面跳转方法，分别为 ，，

wx.navigateTo()： 保留当前页面，跳转到另一个页面（主要表现为左上角含有返回按钮）
wx.navigateBack()： 配合 navigateTo 使用，跳回到上 N 层界面
wx.redirectTo()： 关闭当前页面，跳转到某一个页面上（左上角没有返回按钮）
wx.switchTab()： 主要用于跳转到 tabBar 的操作，如果想要跳转 tabBar 只能使用这个。
wx.relaunch()： 关闭所有页面，跳转到某一页面

<h2 id="发送Http请求">发送 Http 请求</h2>

小程序中只有 Get， Post 两种请求方式。
使用 wx.request 发送 http 网络请求。具体参数有 url，data，header，method，success 等

列如代码：
<pre><code class="highlight-chroma"><span class="highlight
cl">wx.request({
    url
'http://47.95.199.43/abnormalFlight/ninfo',

nbsp; method: 'POST',

nbsp; header: {

nbsp;   'content-type': 'application/json'

nbsp; },

nbsp; data: {

nbsp;   flightdateid: this.flightdateid,

nbsp;   createdate: this.plandt

nbsp; },

nbsp; success: (res) => {

nbsp;   this.ninfos = res.data.data

nbsp; }

nbsp; Fail:(res)=>{

nbsp;   Console.log(“失败”)
   }

nbsp; Complete:(){

nbsp;Console.log(“回调函数”)
   }
})
</code></pre>

<h2 id="推送消息功能">推送消息功能</h2>

小程序推送消息给用户。

<hr>
<h2 id="小程序登录流程-">小程序登录流程：</h2>

详情图：

小程序前端通过 <code>wx.login()</code> 的方法。可以获取一个 code 值。
携带着 code 值，访问 java 后端。
java 后端，通过 appId 和 secret，+code 值。访问 微信 的 API 接口。获取到 用户登录的 openID 和 session_key。
将 openId 和 session_key 返回给前端。
每次进入页面，前端调用 <code>wx.checkSession</code> 方法，判断 session_key 是否过，过期的话，就 调用 <code>wx.login</code> 进行访问 java 后端登录。

<h2 id="注意点-">注意点：</h2>

session_key 默认的有效时间为 3 天
openId 为登录用户的 唯一 Id，一般存储在 数据表中
<code>wx.login</code> 中的 code 值，是根据用户的信息进行生成的。虽然每次都不一样，是均代表用户的信息。所以传递给后台，调用微信 API 接口后，不会生成新的 openID，还是和以前致，会刷新 session_key 会话 ID。
session_key 的作用为，可以解密出用户的 个人信息，手机号，所在地区..等等

<h2 id="获取用户信息">获取用户信息</h2>

通过 <code>wx.getUserInfo</code> 可以获取到用户的个人信息。 encryptedData 和 iv .. 等
<pre><code class="highlight-chroma">// 必须是在用户已经授权的情况下调用
   wx.getUserInfo({
   success(res) {
   const userInfo = res.userInfo //整体user对象
   const nickName = userInfo.nickName //用户昵称
   const avatarUrl = userInfo.avatarUrl //用户头像
   const gender = userInfo.gender // 性别 0：未知、1：男、2：女
   const province = userInfo.province //用户国家
   const city = userInfo.city //用户所在城市
   const country = userInfo.country

nbsp; console.log(res)
       }
     })
</code></pre>

<h2 id="引入Vant-Weapp样式UI">引入 Vant Weapp 样式 UI</h2>

官网地址：<a href="https://ld246.com/forward?goto=https%3A%2F%2Fyouzan.github.i
%2Fvant-weapp%2F%23%2Fquickstart" target="_blank" rel="nofollow ugc">https://youzan.gi
hub.io/vant-weapp/#/quickstart
引入方式（官网的不好用）：
<pre><code class="highlight-chroma"><span class="highlight
cl"># 通过 npm 安装
npm i @vant/wea
p -S --production

//进入node_modu
es\@vant\weapp包下，将dist 复制到 项目的 static 目录下

//在需要使用 Van
组件 的页面目录中创建 main.json 文件。
//需要什么组件就
入什么
{
     "n
vigationBarTitleText": "home首页",
     "u
ingComponents": {

nbsp;"van-button": "/static/vant/dist/button/index"
     }
}
</code></pre>

效果如下：<img src="https://ld246.com/images/img-loading.svg" alt="1589012081677.p
g" data-src="https://b3logfile.com/file/2020/10/1589012081677-b457db1d.png?imageView2
2/interlace/1/format/jpg">
使用组件：
<pre><code class="highlight-chroma"><span class="highlight
cl"><van-button type="primary">主要按钮</van-button>
</code></pre>

在微信开发者平台，需要在详情中设置 ES6 转 ES5 ， 不勾选无法使用<img src="https://ld246.
om/images/img-loading.svg" alt="1589012175200.png" data-src="https://b3logfile.com/file/
020/10/1589012175200-56fadeea.png?imageView2/2/interlace/1/format/jpg">

<h2 id="注意事项-踩坑点-">注意事项（踩坑点）</h2>

组件的函数调用属性 bind:change="xxx" 要写为 @change="xxx"，调用 methods 中的方法</l
>
上传组件 <code>Uploader</code> 的上传回调函数， after-read 不能用，要修改引入的 ind
x.js。
<pre><code class="highlight-chroma"><span class="highlight

cl">在 dist/uploader/index.js 文件中

将after-read 修改
afterread

然后在组件中 就以
@afterread="xxx" 的方式使用（貌似是不能中间有 - ）
</code></pre>

修改组件的样式，自定义样式：
<pre class="md-fences mock-cm md-end-block">custom-style="background-color=xxx"
</pre>

<h2 id="小程序原生组件">小程序原生组件</h2>
<h2 id="小程序事件">小程序事件</h2>
<pre class="md-fences mock-cm md-end-block">touchstart　手指触摸动作开始
touchmove　手指触摸后移动
touchcancel　手指触摸动作被打断，如来电提醒，弹窗
touchend　手指触摸动作结束
tap　手指触摸后马上离开
longpress　手指触摸后，超过350ms再离开，如果指定了事件回调函数并触发了这个事件，tap事
将不被触发　1.5.0
longtap　手指触摸后，超过350ms再离开（推荐使用longpress事件代替）
transitionend　会在 WXSS transition 或 wx.createAnimation 动画结束后触发
animationstart　会在一个 WXSS animation 动画开始时触发
animationiteration　会在一个 WXSS animation 一次迭代结束时触发
animationend　会在一个 WXSS animation 动画完成时触发
touchforcechange　在支持 3D Touch 的 iPhone 设备，重按时会触发
</pre>
<h2 id="页面-style-中-设置-page-全局样式-">页面 style 中 设置 page 全局样式：</h2>
<pre class="md-fences mock-cm md-end-block"></pre>
<h2 id="获取用户的个人信息">获取用户的个人信息</h2>

获取用户的信息 需要用户的授权。 通过 button 来进行
如果没有授权，就会弹出一个 是否确认授权的 按钮。
<pre class="md-fences mock-cm md-end-block">获取用户信息

// 获取用户的个人信息

getUser (event) {

console.log(event)

}

</pre>

自动获取用户的 信息。 需要进行过 授权处理。不然无法获取 会走 fail 方法
<pre class="md-fences mock-cm md-end-block"> // 需要写在 mounted 函数中
  mounted () {
    wx.getUserInfo({
      withCredentials: true,
      // 成功方法
      success: function (res) {

```
        console.log(res)
        var userInfo = res.userInfo
        console.log(userInfo)
      },
      fail () {
        // 失败方法
      }
    })
  },
</pre>
</li>
<li>进入页面 自动弹出 获取信息 授权窗口（暂时不能使用）：</li>
<li>授权网址：<a href="https://ld246.com/forward?goto=https%3A%2F%2Fdevelopers.weixin
qq.com%2Fminiprogram%2Fdev%2Fapi%2Fopen-api%2Fauthorize%2Fwx.authorize.html" tar
et="_blank" rel="nofollow ugc">https://developers.weixin.qq.com/miniprogram/dev/api/ope
-api/authorize/wx.authorize.html</a>
<pre class="md-fences mock-cm md-end-block">// 可以通过 wx.getSetting 先查询一下用户
否授权了 "scope.record" 这个 scope
    var self = this
    wx.getSetting({
      success (res) {
        if (!res.authSetting['scope.userInfo']) {
          wx.authorize({
            scope: 'scope.userInfo',
            success () {
              // 用户已经同意小程序使用录音功能，后续调用 wx.startRecord 接口不会弹窗询问
              console.log('授权成功')
              self.getUser()
            }
          })
        }
      }
    })
</pre>
</li>
</ul>
<h2 id="无需授权获取用户信息">无需授权获取用户信息</h2>
<ul>
<li>使用 open-data 可以获取用户的个人信息。但仅限展示效果，不能作为 js 中的数据进行交互操
：</li>
<li>文档地址：<a href="https://ld246.com/forward?goto=https%3A%2F%2Fdevelopers.weixin
qq.com%2Fminiprogram%2Fdev%2Fcomponent%2Fopen-data.html" target="_blank" rel="no
ollow ugc">https://developers.weixin.qq.com/miniprogram/dev/component/open-data.html
/a>
<pre class="md-fences mock-cm md-end-block">

// css样式， 圆形头像
.userinfo-avatar {
text-align: center;
margin: 0 auto;
overflow: hidden;
```

display: block;

width: 130rpx;

height: 130rpx;

border-radius: 50%;

border: 2px solid #ebebef;

}

</pre>

<h2 id="原生跳转页面">原生跳转页面</h2>

进行页面跳转的是可使用小程序提供的 API
在组件中 按钮 进行跳转：https://developers.weixin.qq.com/miniprogram/dev/component/navigator.html
<pre class="md-fences mock-cm md-end-block">跳转到新页面
在当前页打开
切换 Tab
打开绑定的小程序

</pre>

在 js 方法中 进行跳转：https://developers.weixin.qq.com/miniprogram/dev/api/route/wx.switchTab.html
<pre class="md-fences mock-cm md-end-block">wx.navigateTo() 保留当前页面，可回退
wx.redirectTo() 不保留，不能回退
wx.switchTab() 使用于tabBar页面

//代码示例

wx.navigateTo({

url: 'page/list/main'，// 路径要写成 对应的main.js文件

events: {

// 为指定事件添加一个监听器，获取被打开页面传送到当前页面的数据

acceptDataFromOpenedPage: function(data) {

console.log(data)

},

someEvent: function(data) {

console.log(data)

}

...

},

success: function(res) {
// 通过eventChannel向被打开页面传送数据
res.eventChannel.emit('acceptDataFromOpenerPage', { data: 'test' })
}
})

//test.js文件中，可以获取参数
const eventChannel = this.getOpenerEventChannel()
// 监听acceptDataFromOpenerPage事件，获取上一页面通过eventChannel传送到当前页面的数据
const eventChannel = this.getOpenerEventChannel()
eventChannel.emit('acceptDataFromOpenedPage', {data: 'test'});
eventChannel.emit('someEvent', {data: 'test'});
// 监听acceptDataFromOpenerPage事件，获取上一页面通过eventChannel传送到当前页面的数据
eventChannel.on('acceptDataFromOpenerPage', function(data) {
console.log(data)
})
</pre>

<h3 id="url请求中参数">url 请求中参数</h3>

在 url 中使用 ？ 参数需要。 encodeURIComponent 和 decodeURIComponent 编解码，否会被截断，导致?后面的数据无法传递后跳转页面。
对于多行字符串，可以使用反引号 `` 拼接，避免使用一大堆 和加号。
<pre class="md-fences mock-cm md-end-block">var encodeUrl = encodeURIComponent(param.url)

wx.navigateTo({ url: "../articleDetail/articleDetail?title=" + title + "&id=" + this.data.mid + "&id=" + id + "amp;url=" +encodeUrl });

跳转后页面js文件

...
onLoad:function(options){
     this.setData({
     nbsp; title:options.title,
     nbsp; id:options.id,
     nbsp; url: decodeURIComponent(options.url), // 使用 decode进行解码
      }),

</pre>

<h2 id="使用flyio发送HTTP请求">使用 flyio 发送 HTTP 请求</h2>

需要下载并且在 需要的组件 main.js 中引入使用
官方文档：https://github.com/wendux/fly
<pre class="md-fences mock-cm md-end-block">npm install flyio // 下载

// 在需要的组件 main.js中引入

// pages/logs/main.js中进行配置

// 配置网络请求 flyio

var Fly = require('flyio/dist/npm/wx')

var fly = new Fly()

// 添加全局配置、拦截器等

Vue.prototype.$http = fly

// 使用

this.$http.post('url' , { request playod中的请求数据 })

.then((d) => {

console.log(d) // 返回的数据

}).catch(err =>{

console.log(err.status , err.message) // 错误信息

})

</pre>

post 请求默认会在 request playod 中发送数据。但是可以设置为以 requestBody 中发送请求据：
<pre class="md-fences mock-cm md-end-block">// 第一种，通过 content-type设置

fly.post("../package.json",{aa:8,bb:9,tt:{xx:5}},{headers:{

"content-type":"application/x-www-form-urlencoded"

}})

.then(console.log)

// 第二种 使用 qs 库

var qs = require('qs');

fly.post('/foo', qs.stringify({ 'bar': 123 }));

</pre>

<h2 id="使用Vuex数据管理">使用 Vuex 数据管理</h2>

需要下载 vuex
<pre class="md-fences mock-cm md-end-block">// 下载 vuex
npm install vuex
</pre>

创建 vuex 声明文件
<pre class="md-fences mock-cm md-end-block">//在src下创建store/store.js文件
import Vue from 'vue'
import Vuex from 'vuex'

// 声明使用vuex

Vue.use(Vuex)

export default new Vuex.Store({

state: {

},
actions: {

},
getters: {

},
mutations: {

}

})
</pre>

在 src 根目录下的 main.js 文件中，配置 Vuex，使每一个实例组件都可以使用
<pre class="md-fences mock-cm md-end-block">import store from './store/store'
// 将store 对象防止Vue的原型上，每个实例都可以使用 Vuex
Vue.prototype.$store = store
</pre>

具体组件 (page/index/index.vue) 中使用 vuex 管理状态
<pre class="md-fences mock-cm md-end-block">import {mapState} from 'vuex'

// 在计算属性中，注册使用 vuex中的数据

computed: {

...mapState(['state中的数据名称'])

}

// 调用 vuex中 mutations 的方法

this.$store.commit("add")
</pre>

<h2 id="分包管理">分包管理</h2>

微信小程序规定，一个包的总大小不能超过 2M ， 所有包的总大小不能超过 10M
小程序分为 主包 和 分包。

主包：是 app.json 中 pages 属性定义的 路径内容，都是主包。
分包：subpackages 中定义的路径是分包。

当我们主包的大小超过 2M 就会报错。 所以我们要将 主包的 一些代码。可以放在分包中，当我进入分包页面，才会加载其里面的内容。
app.json 中分包设置：
<pre class="md-fences mock-cm md-end-block">{
  // 主包设置
"pages": [
    "pages/login/main",
    "pages/index/main",
    "pages/logs/main",
    "pages/counter/main"
  ],
    // 分包设置 , 分包页面不能设置 tabBar 导航标签
  "subpackages": [
    {   // {} 为一个分包
      "root": "pages/custom", // root设置 这个分包的 根目录 ， 不能与主包的相同
      "pages": [
        "main"   // 设置这个分包下几个 路径页面，写main.js
      ],
      "plugins": {   // 当前分包可以使用的 plugins 插件
        "myPlugin": {
          "version": "0.0.5",
          "provider": "wxfea216762bcf4ad9"
        }
      }
    }
  ],
}
</pre>

主包 跳转 到分包：
<pre class="md-fences mock-cm md-end-block">wx.navigateTo({ url: '../custom/main' })
</pre>

<h2 id="第三方插件使用-">第三方插件使用： </h2>
<h2 id="快递查询插件">快递查询插件</h2>

插件网址：https://mp.weixin.qq.com/wxamp/basicprofile/thirdauth?toke=595757880&lang=zh_CN
根据 添加的 第三方插件的 开发者文档，可以导入使用：使用网址：<a href="https://ld246.co/forward?goto=https%3A%2F%2Fdevelopers.weixin.qq.com%2Fminiprogram%2Fdev%2Ffra

ework%2Fplugin%2Fusing.html" target="_blank" rel="nofollow ugc">https://developers.weixn.qq.com/miniprogram/dev/framework/plugin/using.html
添加完小程序后，在需要使用的 分包中，导入：
列如：<code>快递单查询</code> 插件的使用
<pre class="md-fences mock-cm md-end-block">{
  "subpackages": [
    {
      "root": "packageA",
      "pages": [
        "pages/cat",
        "pages/dog"
      ],
      "plugins": {
        "myPlugin": { // 自定义插件的名称
          "version": "1.0.0", // 设置使用哪个版本
          "provider": "wxidxxxxxxxxxxxxxxxx" // 插件的Appid
        }
      }
    }
  ]
}
</pre>

列如： 跳转到 插件的操作页面：
<pre class="md-fences mock-cm md-end-block">wx.navigateTo({url: 'plugin://myPlugin/logitics'})
</pre>

<h2 id="地图导航插件">地图导航插件</h2>
<h2 id="反编译出微信小程序">反编译出微信小程序</h2>

可以反编译出，微信上已经上线的小程序源码。
是 微信小程序原生代码。 可以 扒下来，参考 css 样式编写。
教程地址：https://zhuanlan.zhihu.com/p/138936448

<h2 id="需要工具">需要工具</h2>

夜神模拟器，node 环境，反编译脚本代码：
反编译代码：M:\wxapp\wxappUnpacker-master

<h2 id="反编译过程">反编译过程</h2>

在夜神模拟器上下载 微信 软件。进行登录。
安装 PE 文件管理器： E:\百度云下载\Root_Explorer-v4.2.4(3.0)-Origin_icon_group-Patched by_Alphaeva.apk . （直接拉进夜神中即可安装）
在微信程序中，打开一个小程序。
然后通过 PE 找到 /data/data/com.tencent.mm/MicroMsg/{一串 16 进制字符}/appbrand/pg/ 目录。（通过事件判断，哪个是刚打开的小程序）
一般文件大小不会超过 2M .
长按文件，进行压缩。 然后通过微信发送给一个好友。
电脑端也能查看这个发送的文件 （<code>.wxapkg</code> 结尾）。

通过 反编译 脚本。进行反编译：
<pre class="md-fences mock-cm md-end-block">cmd 进入到 反编译文件夹中： M:\wxapp\wxppUnpacker-master
安装依赖：
1、npm install esprima
2、npm install css-tree
3、npm install cssbeautify
4、npm install vm2
5、npm install uglify-es
6、npm install js-beautify
7、npm install escodegen -g

//进行反编译命令 C:_163200311_32.wxapkg 为要编译的 小程序代码

node wuWxapkg.js C:_163200311_32.wxapkg

</pre>

编译出的代码文件夹，直接在 微信小程序开发者工具中，导入打开即可。
出现 _vd_version_info_is not defined 错误:： 解决：https://www.cnblogs.com/wukong8/p/11612470.html

<h2 id="Git拉取代码强制覆盖">Git 拉取代码强制覆盖</h2>

拉取代码覆盖本地的：
在项目 跟目录下
<pre class="md-fences mock-cm md-end-block">$ git fetch --all
$ git reset --hard origin/master
$ git pull
</pre>

