# 文档说明

## 第一天创建框架结构

此项目为移动端（手机端）项目，使用js框架推荐使用vue

## 1.使用vue

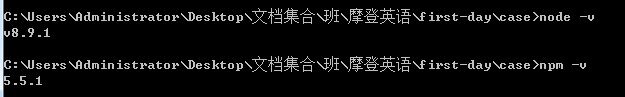
第一步：

搭建vue项目

（注意）：

先验证本地电脑是否安装node 指令：node -v

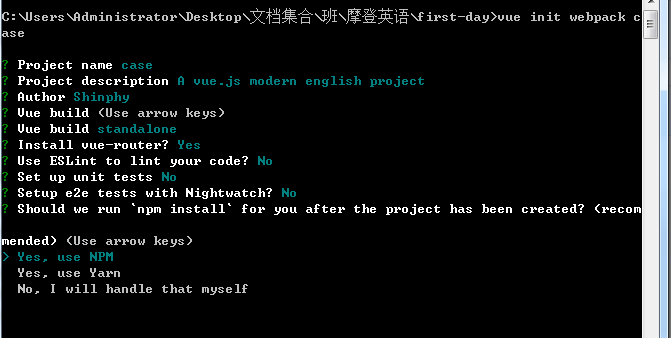


再验证本地是否安装npm依赖 指令： npm -v  
   
 最后验证本地是否安装vue-cli 指令： vue -V --->（V为大写）  
 

如果没有安装vue-cli 创建vue项目之前要先依赖vue-cli 指令：npm install --global vue-cli --->（--global指全局安装）

验证完毕以后创建一个基于webpack的vue项目

指令：vue init webpack case(指自己的文件夹名)



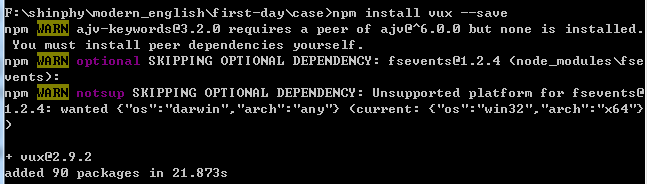
指令说明不明白，请看上个vue项目说明

项目创建完毕，可以启动项目

## 安装vux ---->UI框架

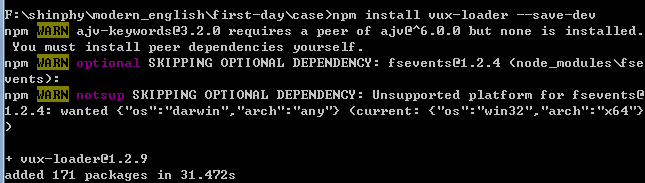
此项目是用vux-ui框架，vux官网网址：<https://doc.vux.li/zh-CN/>

安装指令：npm install vux --save



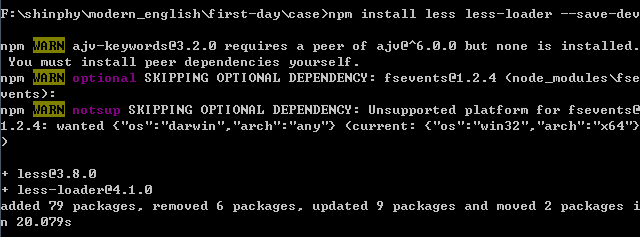
安装vux-loader 必须结合vux-loader使用

指令：npm install vux-loader --save-dev



安装less-loader

指令：npm install less less-loader --save-dev



需要在build里面的webpack.base.conf.js里面的extensions里加入less配置

resolve: {

extensions: ['.js', '.vue', '.json', 'less'],

alias: { 'vue$': 'vue/dist/vue.esm.js',

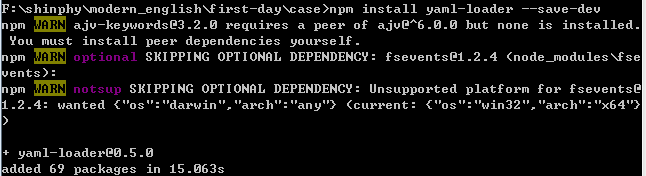
'@': resolve('src'),

}

}

安装yaml-loader 语言工具

指令：npm install yaml-loader --save-dev



最后在build/webpack.base.conf.js文件里将原来的module.exports的代码赋值给一个新的变量webpackConfig

const vuxLoader = require('vux-loader')

module.exports 换成 const webpackConfig

最后加上

module.exports = vuxLoader.merge(webpackConfig, {

plugins: ['vux-ui']

})

这个vux因为应用于移动端，所以是按需引入 最好不要直接全部引入 ，那样会导致手机很卡，加载量比较大，PC端因为用的网线 可以直接引入 没问题的

import { XHeader, Tabbar } from 'vux'

components: {

XHeader,

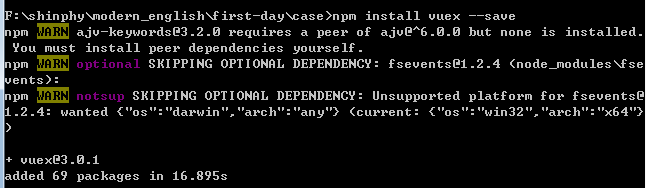
Tabbar

}

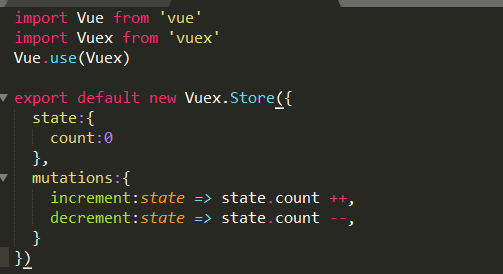
需要什么就引入什么就好了

## 安装vuex

安装指令：npm install vuex --save



1. 在src下创建一个store文件夹，里面创建一个store.js



做一个简单的计数

1. 在 main.js 中导入

import store from './store/store'

new Vue({

el: '#app',

store,

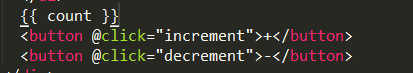
router,

template: '<App/>',

components: { App }

})

1. 在自带的helloWord页面中加入两个事件

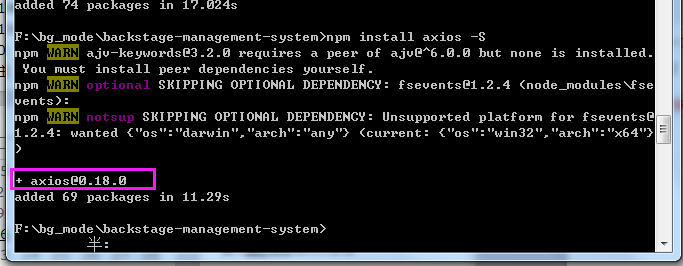
  
代指的是加减方法

然后声明这两个方法 和 计算count



## 4.安装axios

* 1. npm install axios -S



默认引入



默认调用方法

this.$axios.get(‘/user’)

.then((response)=>{

})

.catch((error)=>{

})

this.$axios.post(‘/user’,{id:1,age:20})

.then((response)=>{

})

.catch((error)=>{

})

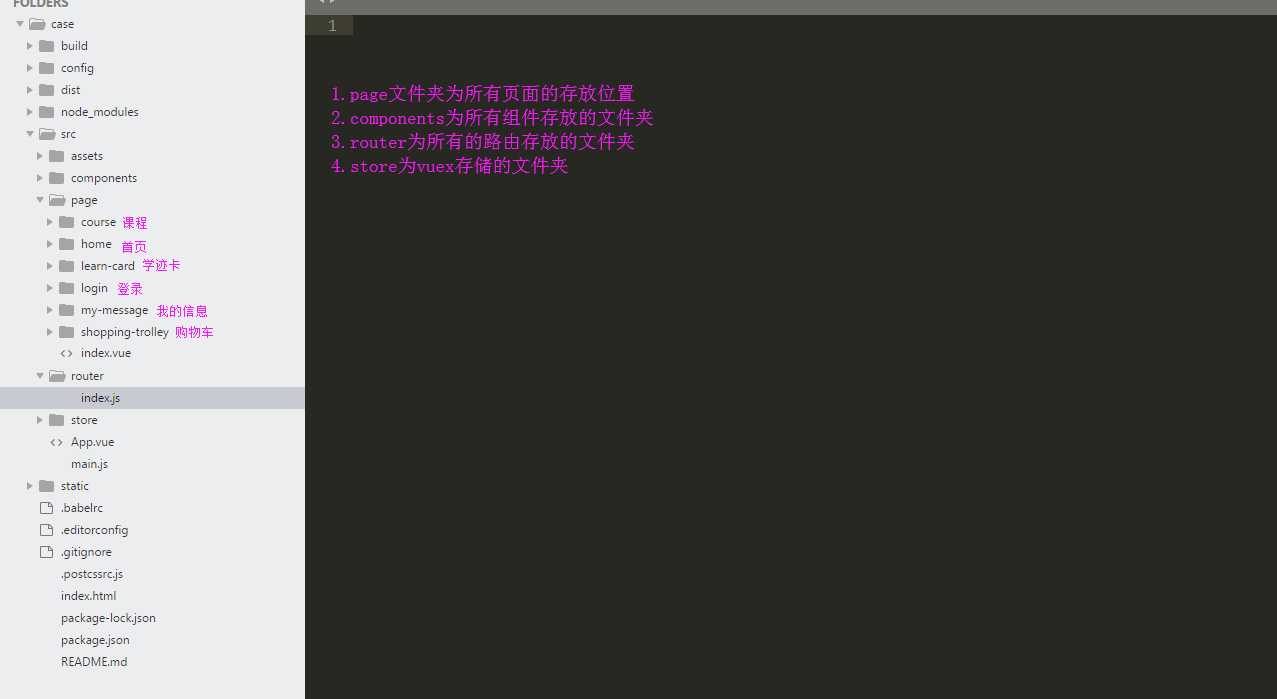
## 5.创建页面结构

1.页面分析

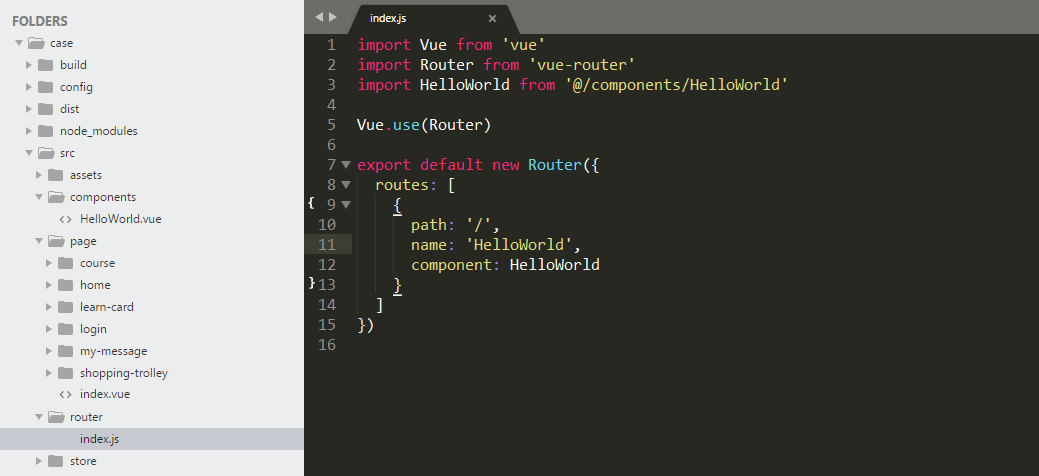


由此可以分析，页面共分为六大块：登录、首页、课程、学迹卡、购物车、我的

对应的在项目中创建六个文件夹与其对应



1. 编写router.js



此上为原始写法，在外部声明路径，在routes内部声明跳转路径



根据vue-router中的路由懒加载问题，就是页面第一个加载首页缓慢的问题，进行解决方案，有时候需要懒加载，提升首页加载速度，一般是页面层级较为复杂的时候

const tvProgram = resolve => require(['@/page/tvProgram'], resolve);

const game = resolve => require(['@/page/gameIndex'], resolve);

const gameitem = resolve => require(['@/page/gameItem'], resolve);

此上就是vue懒加载写法规范

resolve => require([路径], resolve) 语法



此上两种写法是一样的 而下面的写法还有解决懒加载的问题，结果不言而喻，用下面的写法吧

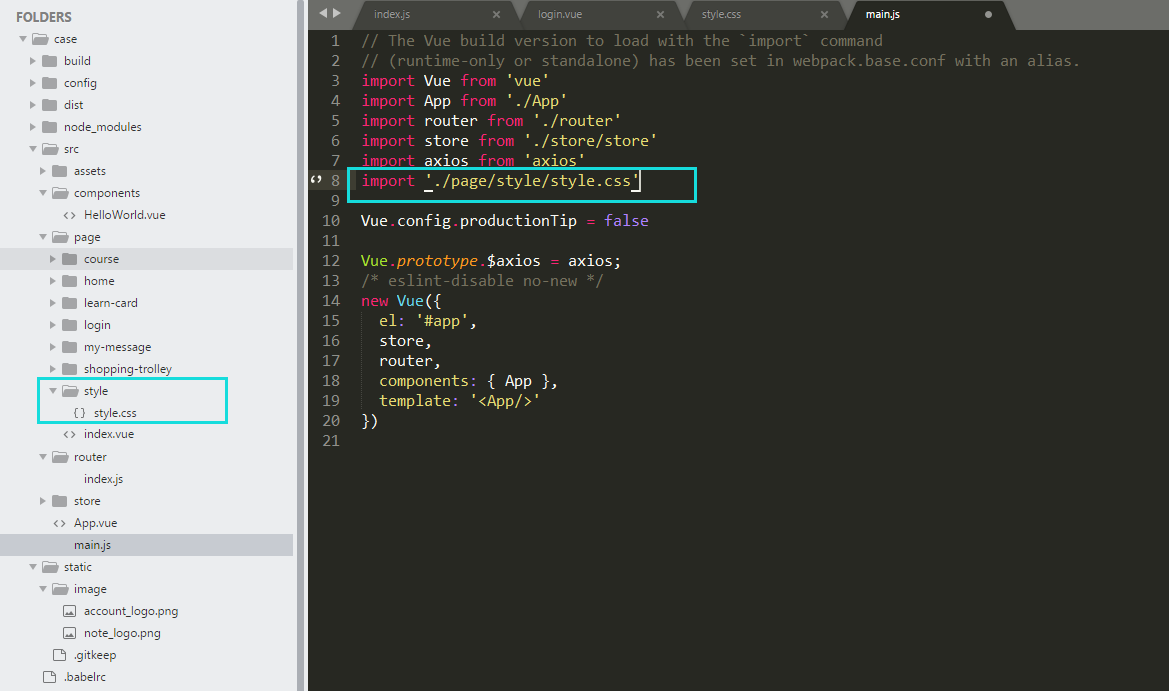


这里就是默认跳转至登录页

## 第二天编写登录页面及相关页面

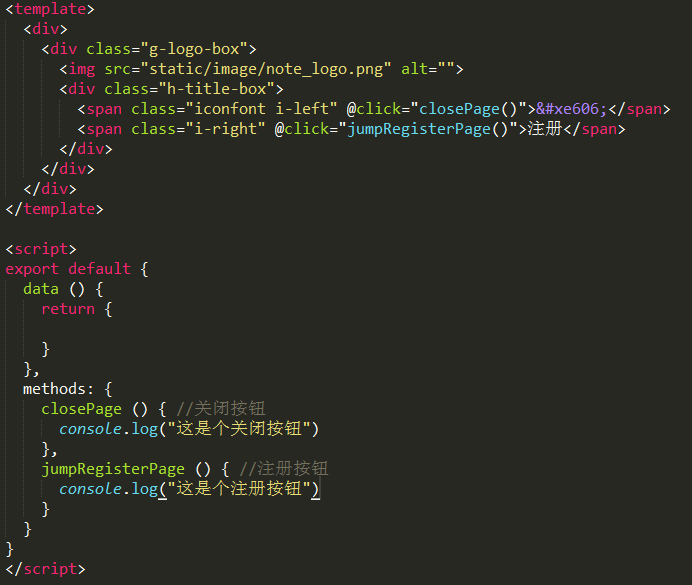
### 开发登录页面

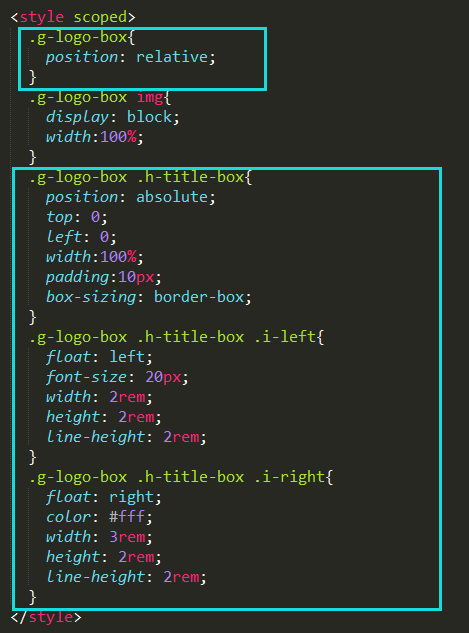


1. .创建一个公共的css页面 并在main.js引用  
    
2. 制作login上面的logo
   1. 先把app.vue中的margin-top去掉，使图片是紧贴移动端上边框显示
   2. 设置公共样式 body 使margin、padding都为0
   3. 为img设置display为块级，并且给一个宽度为100%，使图片按照宽度为100%来自适应，得到结果如下：



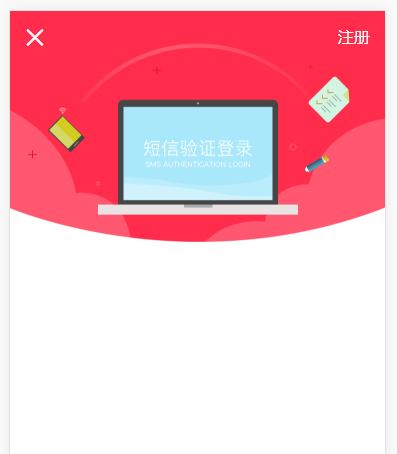
1. 制作头部的关闭和注册按钮



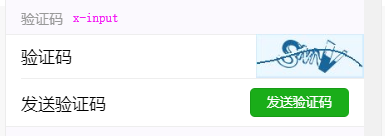


* 1. 考虑到关闭为图标，用矢量图标，引入iconfont
  2. 给g-logo-box设置定位，给子元素h-title-box设置相对定位，位于头部，并给定宽度，设置好padding，做一个盒子模型
  3. 然后给h-title-box里面的i-left与i-right分别做一个float一个左浮动一个右浮动
  4. 可以对应的添加click事件，也可以适当给i-left、i-right定义宽度

结果如下：

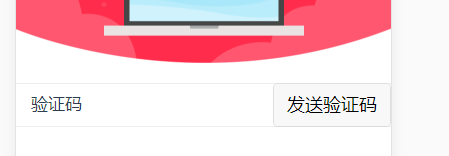


1. 制作主体form表单
   1. 根据原图形可以看到第一个手机号的样式与vux里面的x-input很类试





* 1. 模拟第一个 把图片换为x-button 出现一下效果，然后在做对应的样式修改

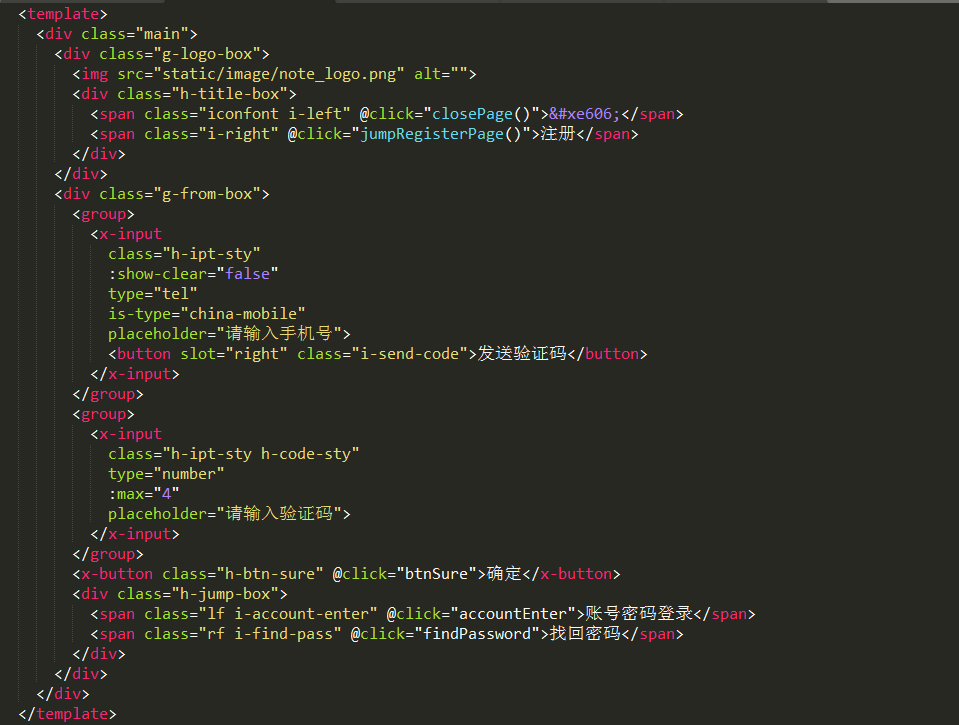


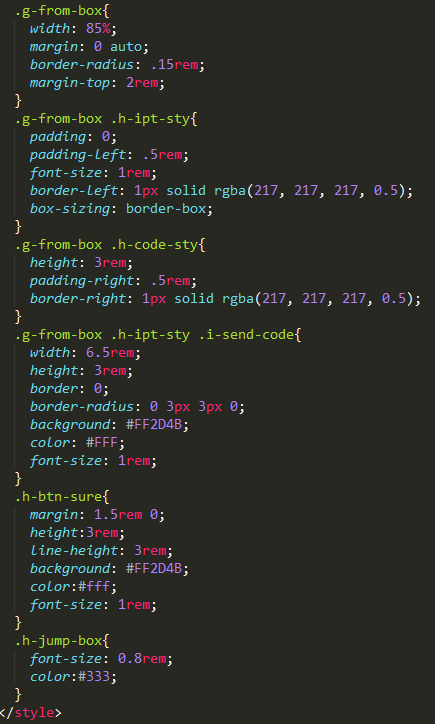
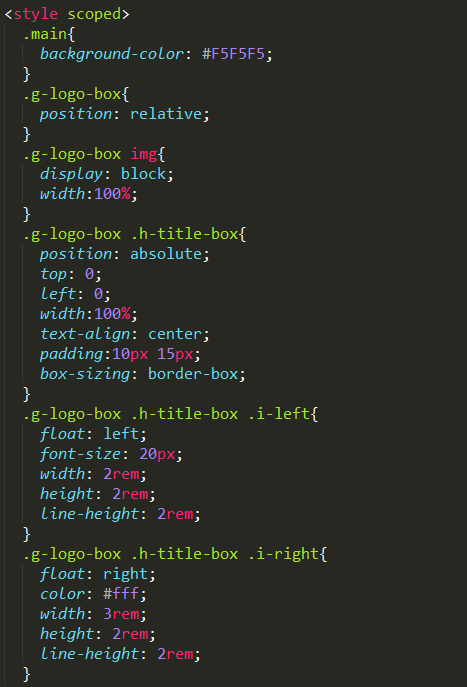


* 1. 验证码输入x-input和确定x-button,以及调整样式

注意：login页面页面默认高度要更改为100%，在公共css里面添加html，body，#app，.main高度为100%

* 1. 最后定义下方账号密码登录和找回密码







## 账号密码登录页面

* 1. 根据图形显示来看 我们会发现这个账号密码登录页和手机号验证码登录页 大同小异，我们可以进行下面比对：



比对来看的话，有三点不同地方，

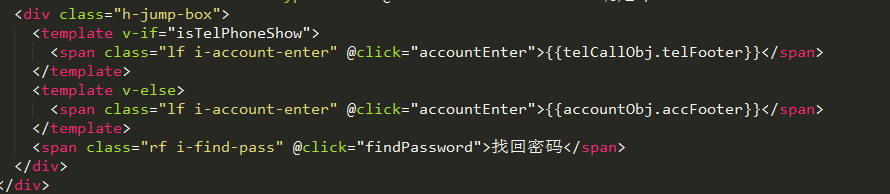
1. 图片显示不同，这个前端来考虑也就是一个url的路径不同，只需要动态显示控制即可

1. 输入状态两个框不同，前端可以考虑写两短代码，根据某个状态值来进行控制显示某一个输入框即可



1. 显示切换名称不一样，手机验证码登录时，要显示账号密码登录，这样可以直观看到要跳转的页面状态，可以根据上部的显示状态值进行判断



* 1. 两个图模板类型一样，我们就采取用一个页面来进行显示，只是控制页面代码段的显示隐藏
     1. 由上图可以看出咱们是由isTelPhoneShow这个属性值来进行控制，我们可以发现，切换的时候可以写同一个方法，也就是说无论是切换至账号密码还是切换至手机验证码，都是这个方法，我们可以在这个方法中做个判断来控制img的src值



* + 1. 但是注意 开发阶段 可以先根据isTelPhoneShow的值来进行开发某一个显示代码段来书写样式，样式改好以后再来切换，具体情况需要具体分析
    2. 好了，这个登录页面两个页面就开发完毕了 是不是比写两个页面简便呢

## 3.开发注册页面和找回密码页面

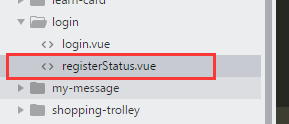
(1).根据图形展示来看 这两个页面属于同一个模板，区别一是头部信息字体不一样， 区别二是密码的placeholder值不一样，区别三是法律声明块显示的问题



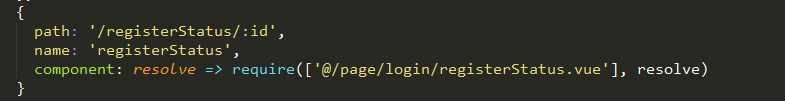
根据上面描述 这两个模块可以作为同一个模块，从登录页面进来使这两个页面显示不同状态即可，但是问题来了，我点击注册和找回密码都是跳转到这个公共页面，怎么来判断是从哪个点击进来的呢？

这个我们通常来讲都会在路由后面跟上参数，根据不同值代表不同的从何处点击

我们分为三步来在这个公共页面拿到是从哪里点击进来的（前提创建好这个公共vue页面）



①在路由配置router.js页面

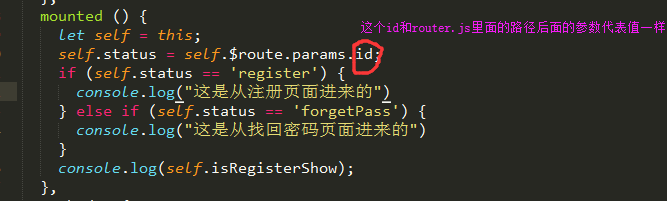


②在点击注册和找回密码的事件中写上路由跳转



我这里是根据register和forgetPass来进行区分的

③在registerStatus.vue(公共页面)的mounted方法中拿到这个传过来的参数

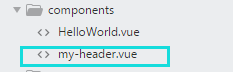


这样我们就能知道是从哪个页面进来的了

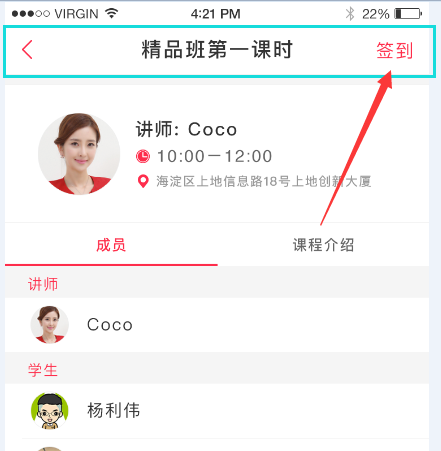
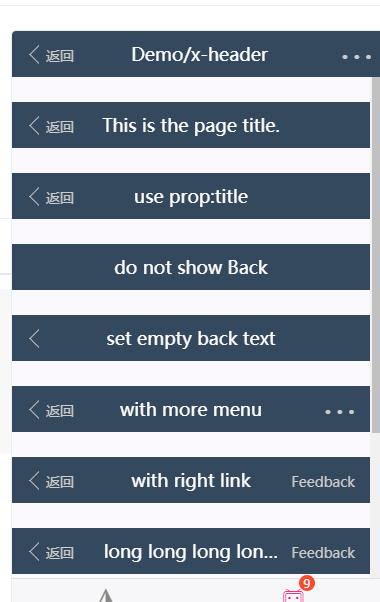
(2). 开发页面头部

虽然在vux已经给封装好了header这个组件，但是如果我们后面很多页面都用到了这个组件的话不就是要写很多遍吗，所以我们就想到在做一个公共组件

首先在components里面建立一个my-header.vue代指header公共组件

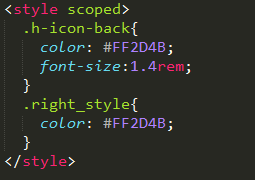
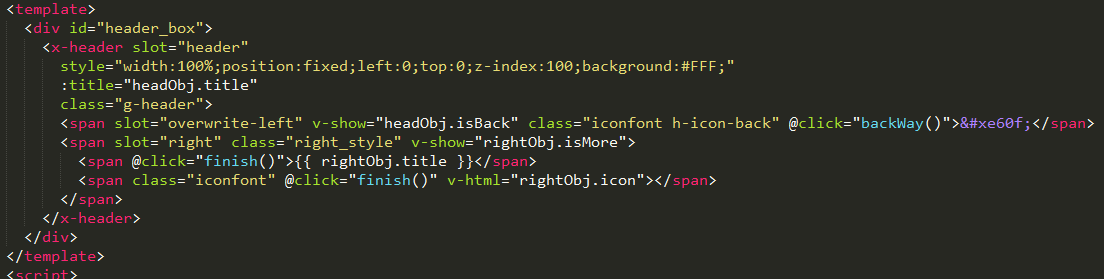


在vux中XHeader这个组件是定义好的组件库



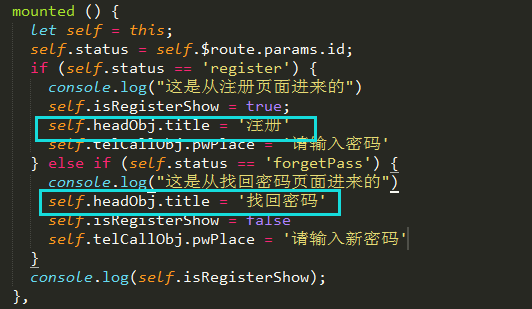
我们头部信息和vux很类似但是呢 我们的没有背景 并且返回按钮是红色的 title字体是黑色的 右面的字体也是红色的 但是在咱们这个页面是没有的 所以就用一个值来进行控制显示隐藏

子组件封装：

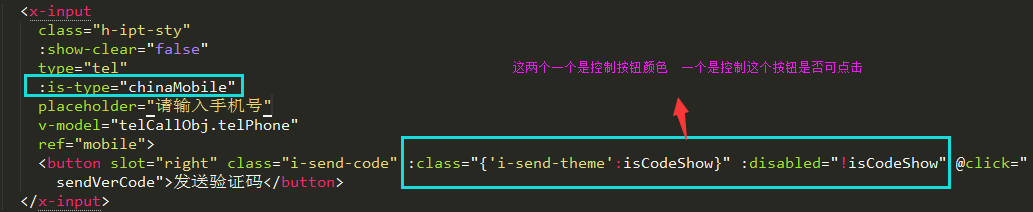
父组件调用：

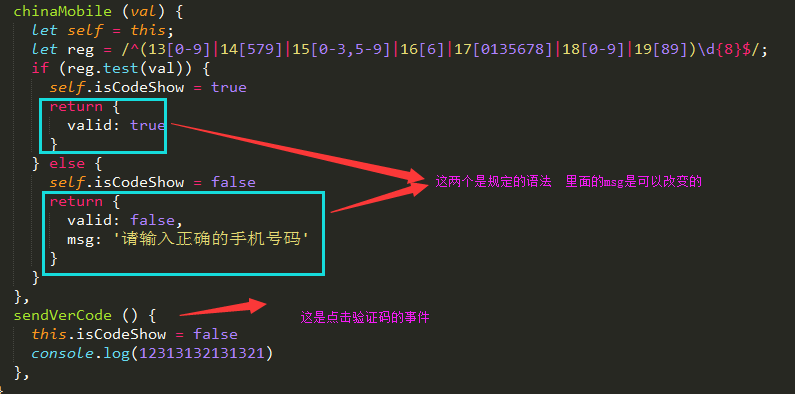




这样咱们的头部就是动态的了 它会动态显示注册和找回密码

* 1. 开发主题内容部分
     1. 大体内容和上个页面雷同 就是一个密码输入语不一样，上面父组件代码就有直接显示的
     2. 手机号码的验证码是灰色的，我想了一个特效 就是验证手机号成功的时候显示红色，否则就是灰色，这个需要动态绑定class 然后在x-input的is-type属性动态绑定一个函数 在函数内部加上控制显示隐藏这个class的值就好





* + 1. 剩余的开发 看一下我的源代码就好了
    2. 好了 这两个页面就开发完成了 还有一个注册的法律声明