一、进入项目根目录（root） npm init，生成packpage.json 全部默认即可

二、接着在devDependencies环境下安装babel-cli、bebel-loader、babel-preset-es2015

命令：npm install --save-dev babel-cli、bebel-loader babel-preset-es2015

在根目录下新建.babelrc文件，文件内容如下：

{

"presets": [

"es2015"

],

"plugins": []

}

使node环境下可以支持es6语法，并且webpack编译后，可以将js中的es6转成es5。

三、

1、新建文件夹 build 用来存放webpack编译文件。 root/build

2、新建文件webpack.dev.config.js开发环境的webpack配置文件。 build/webpack.dev.config.js

内容如下：

export default {

entry: 'index.js'

};

3、新建文件 build-dev.js 开发环境的编译文件。 build/build-dev.js

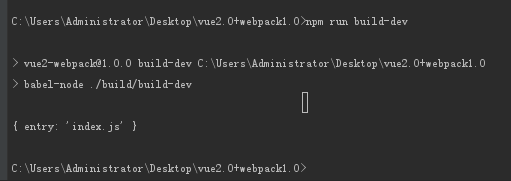
内容如下：

import webpackConfig from './webpack.dev.config';

console.log(webpackConfig);

4、package.json文件的script参数配置命令："build-dev": "babel-node ./build/build-dev"

5、执行命令：npm run build-dev 输出如下：



四、

1、新建文件夹src，存放源文件、源代码。 root/src

src下新建index.html文件、index.js

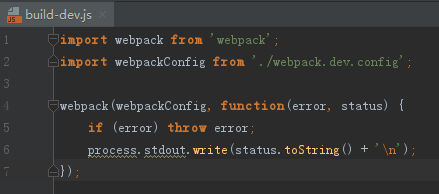
index.js文件内容如下：

console.log(‘hello world’);

webpack.dev.config.js内容改成：



build-dev.js内容改成如下：

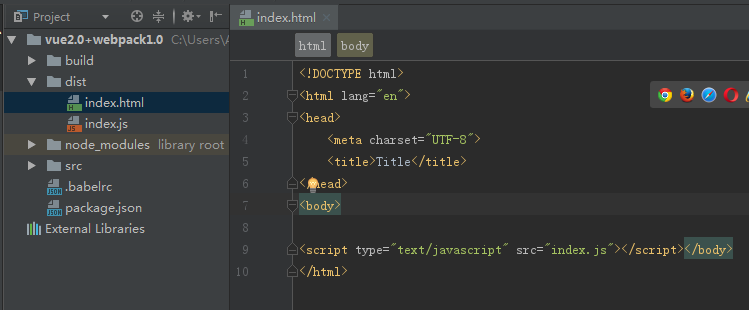


安装html-webpack-plugin插件npm install --save-dev html-webpack-plugin

安装webpack1.13.1 npm install --save-dev webpack@1.13.1

命令 npm run dev 后生成dist目录，目录中有index.html文件、index.js

index.html文件内容如下：



1. 引入vue、vue-router

1、安装vue、vue-router

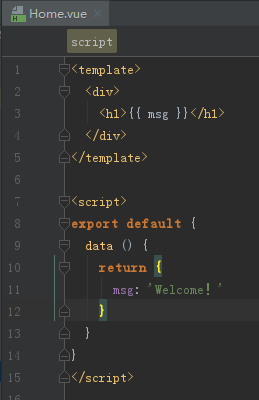
命令 npm install --save vue vue-router

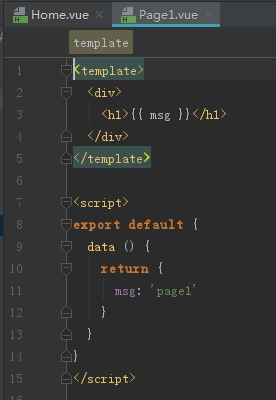
2、新建文件夹assets，保存字体、图片。 src/assets

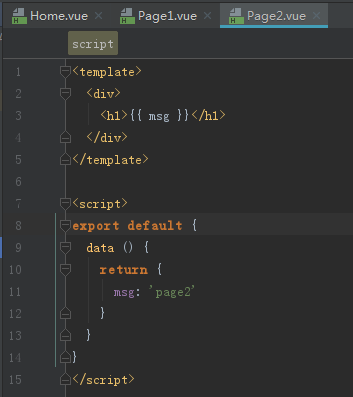
新建文件夹components，下面新建文件Home.vue、Page1.vue、Page2.vue。

src/components

内容分别如下：

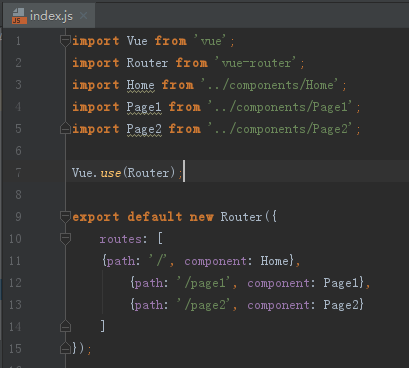




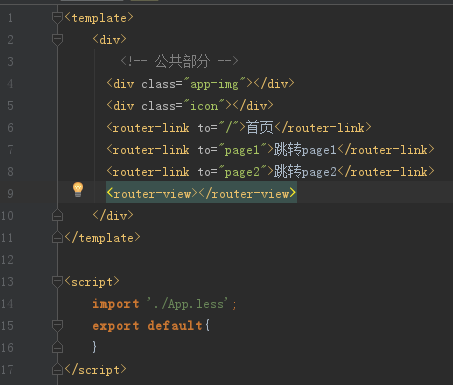


3、新建文件夹router，下面新建index.js文件，保存路由配置。src/router

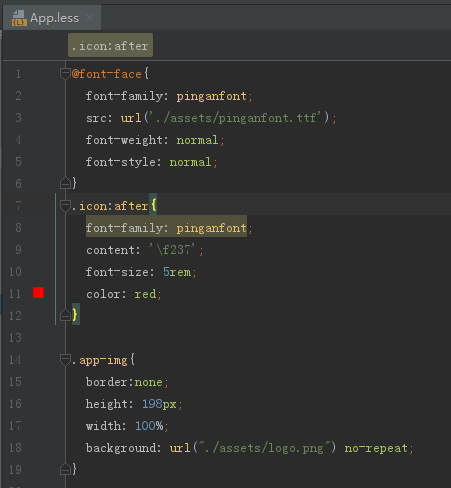
内容如下：



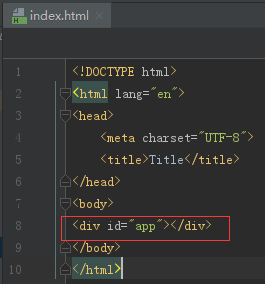
1. 新建App.vue文件，src/App.vue 内容如下：



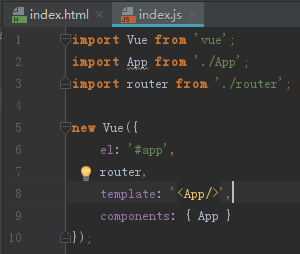
1. 新建App.less，src/App.less，内容如下：



1. src/index.html，新增红框部分



7、src/index.js内容如下：



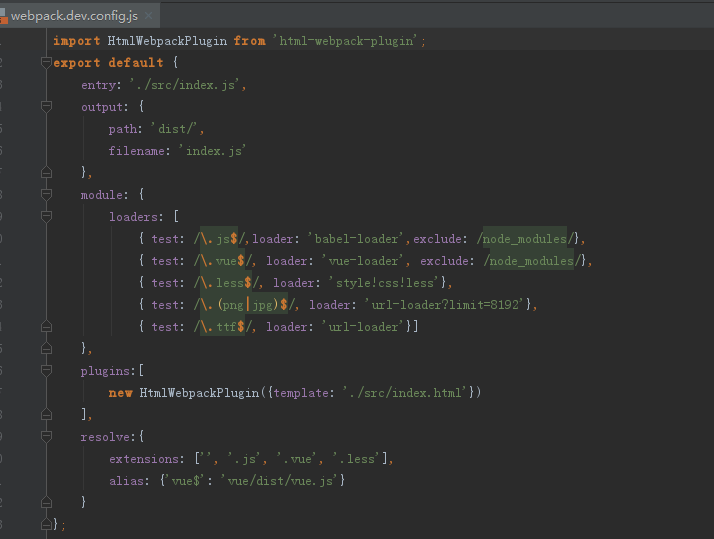
1. 安装url-loader、vue-loader、vue-template-compiler、

安装style-loader、css-loader、less-loader、less

命令npm install --save-dev url-loader vue-loader vue-template-compiler

命令npm install --save-dev style-loader css-loader less-loader less

1. Webpack.dev.config.js内容修改如下：

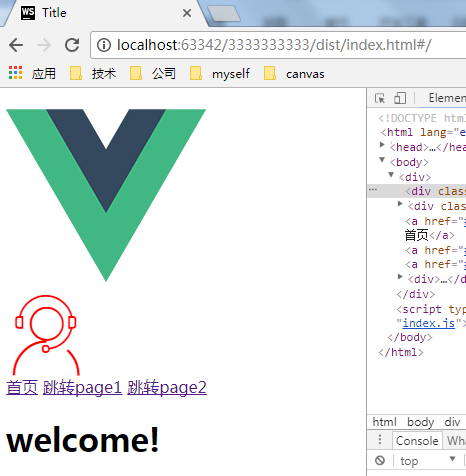


10、

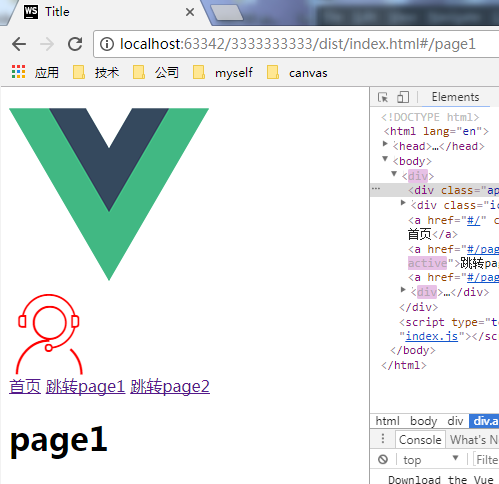
执行命令 npm run build-dev

打开dist/index.html

效果图如下：



链接可以实现路由跳转

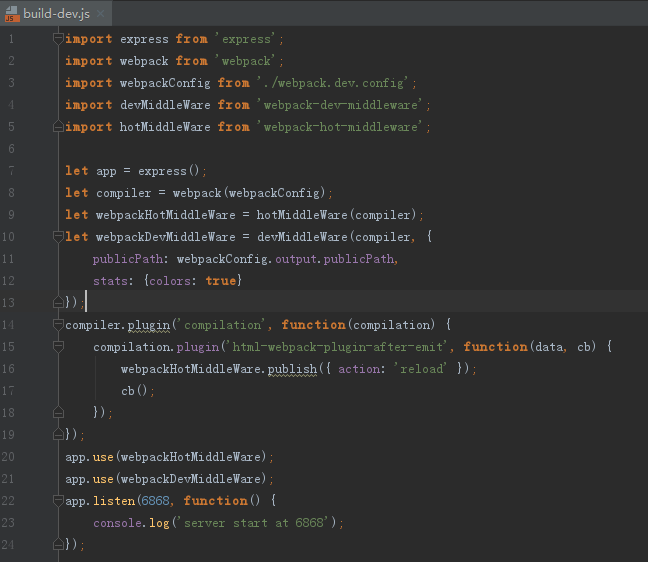


11、使用 webpack-dev-middleware、webpack-hot-middleware中间件，实现开启本地服务以及热加载。

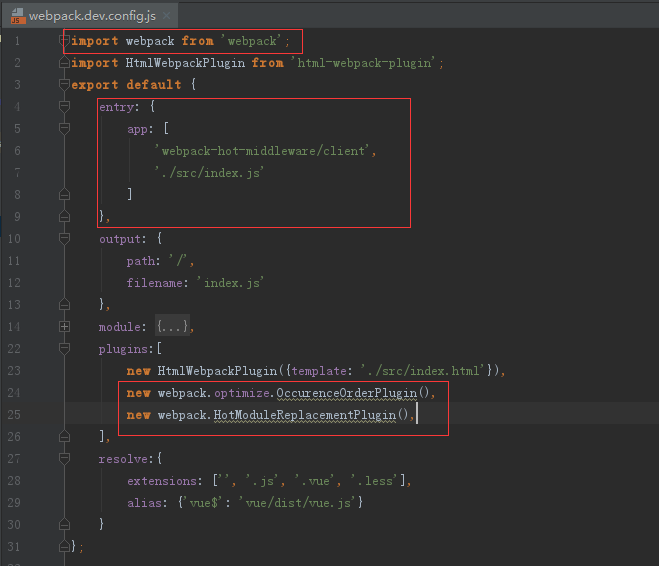
安装 webpack-dev-middleware、webpack-hot-middleware、express

命令 npm install --save-dev webpack-dev-middleware webpack-hot-middleware express

build.dev.js 内容修改如下：

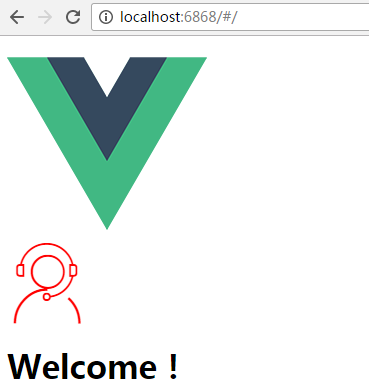


Webpack.dev.config.js 内容修改如下：



执行命令 npm run build-dev

浏览器，地址栏输入 localhost:6868 效果图如下：修改Home.vue内容可以实时刷新浏览器。



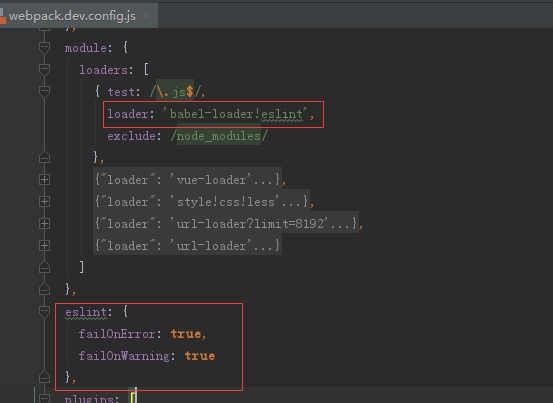
1. 加上代码校验eslint

安装 eslint、eslint-loader eslint-plugin-standard eslint-plugin-promise babel-eslint

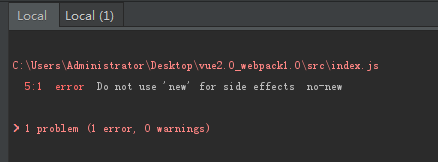
命令 npm install --save-dev eslint eslint-loader eslint-plugin-standard eslint-plugin-promise babel-eslint

从eslint官网拷贝一份.eslintrc到根目录

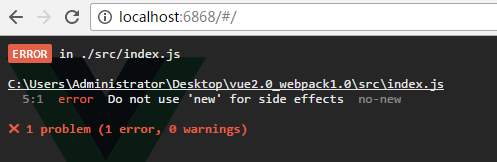
webpack.dev.config.js 内容修改如下：



执行命令npm run build-dev 即有如下信息报错，根据eslint规则写就不会报错。



浏览器也显示报错信息



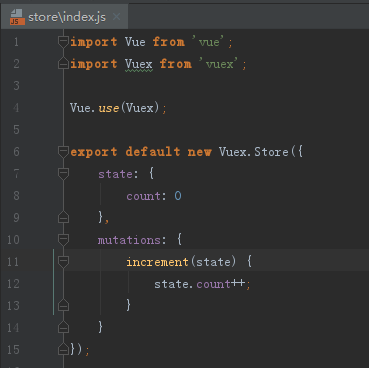
1. 引入vuex ，实现组件间的传值。

安装 vuex

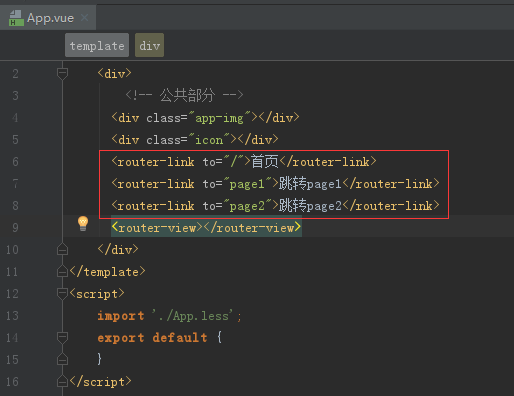
命令 npm install --save vuex

src目录下，新建store文件夹

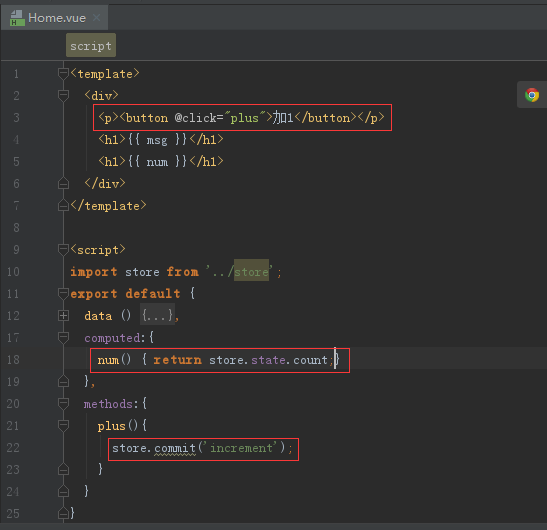
src/store 新建文件index.js，内容如下：



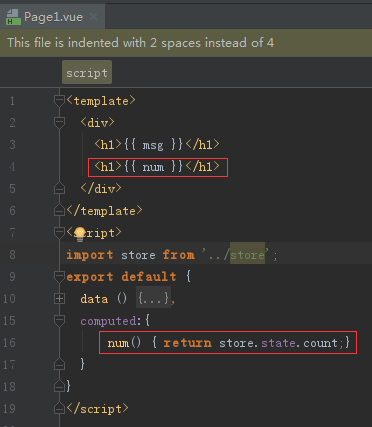
App.vue 内容修改如下：



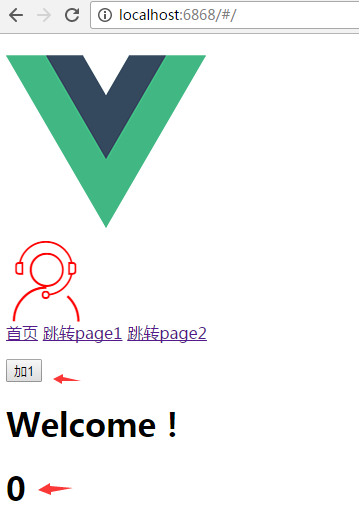
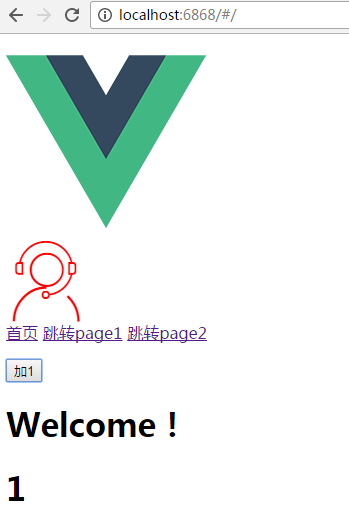
Home.vue 内容修改如下：



Page1.vue 内容修改如下：



首页的值count通过按钮增加1之后，在page1页面可以获取这个值。

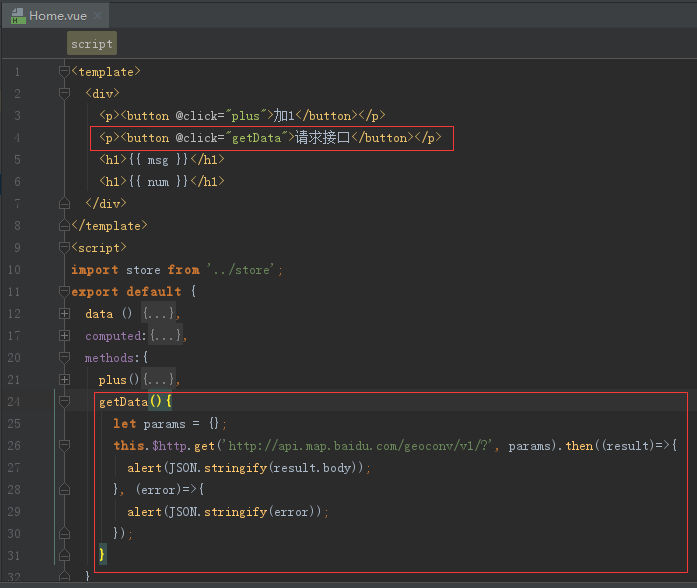


1. 引入vue-resouce，实现http请求

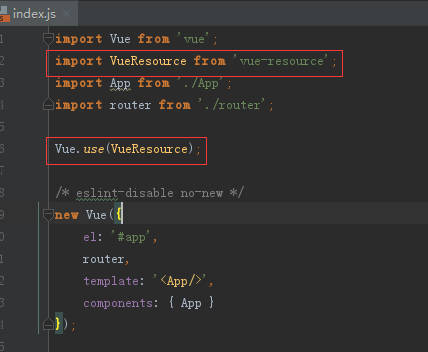
安装vue-resource

命令 npm install --save vue-resource

Home.vue 内容修改如下：



src/index.js 内容修改如下：



效果图如下：

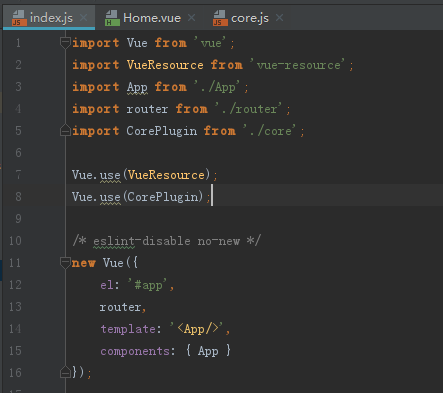


1. 定义公有方法，使每个vue组件可以调用：

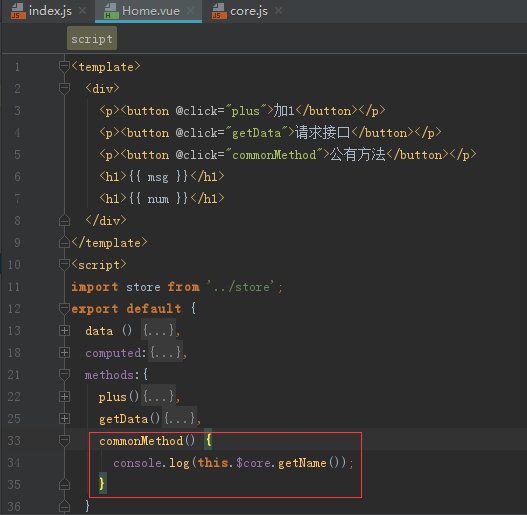
新建core.js 内容如下： src/core.js



Src/index.js 内容修改如下：

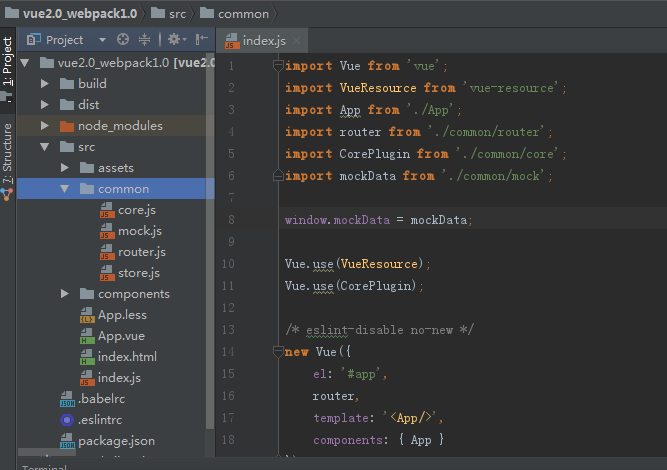


Home.vue 内容修改如下：

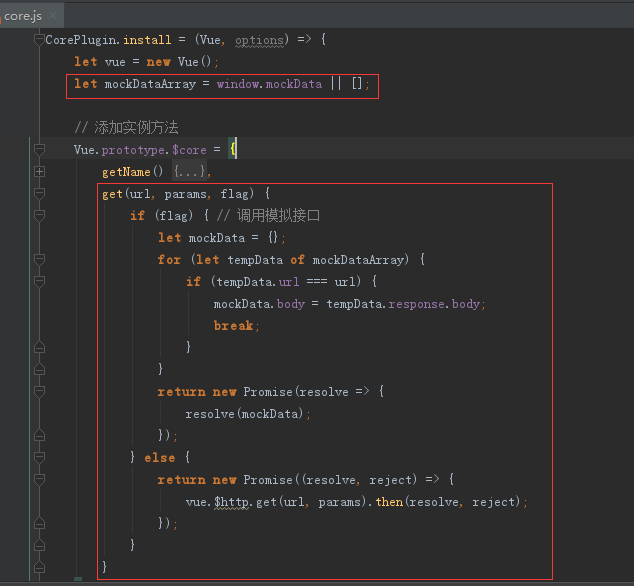


1. 模拟数据、真实接口之间的切换，弄个开关进行切换。

先调整目录结构，并且src/index.js 修改如下：



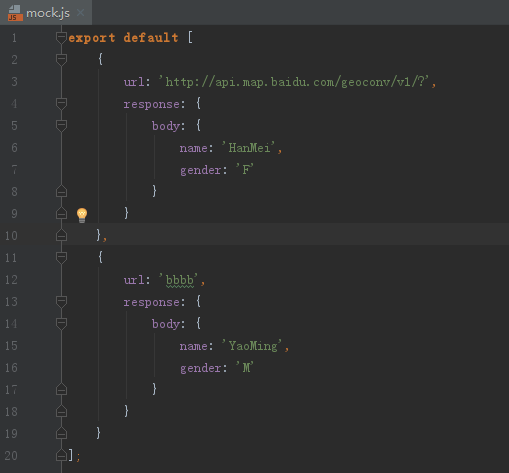
Core.js 修改内容如下：



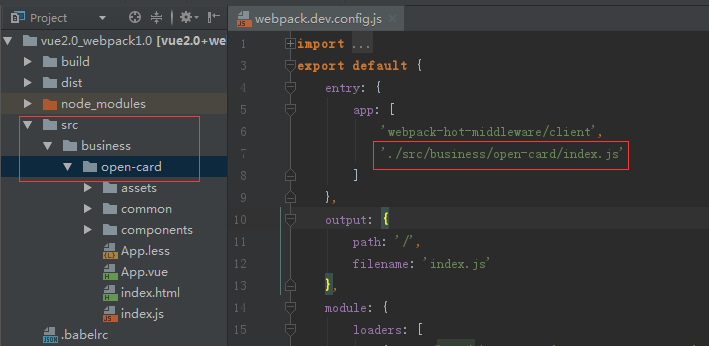
Home.vue 内容修改如下：



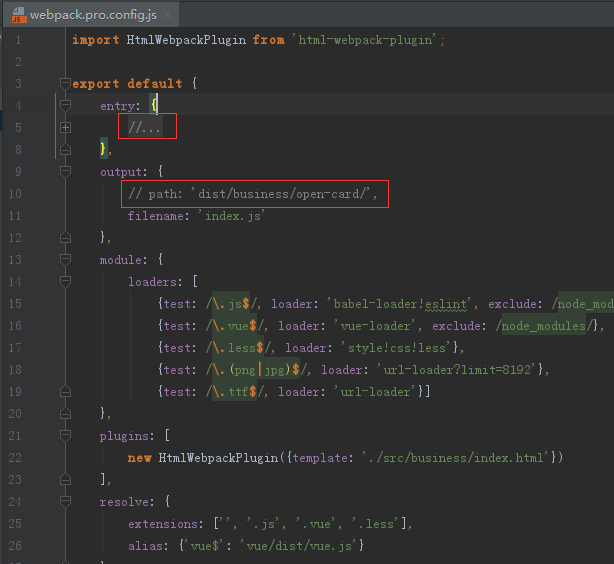
mock.js 内容如下：



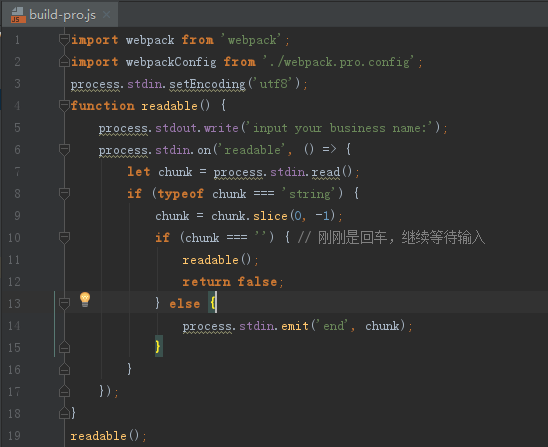
1. 实现不同的业务，新增debug配置、production配置，先调整目录结构

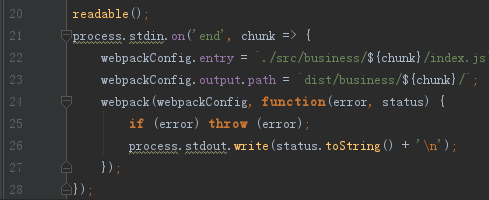


Webpack.pro.config.js 修改如下：



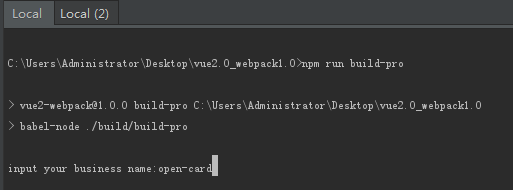
build-pro.js 内容修改如下：





执行命令：npm run build-pro

控制台可以根据用户输入字符串，控制编译输出文件

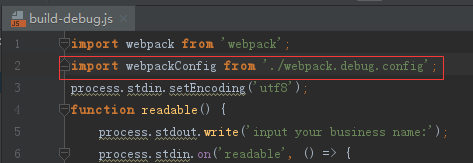


其实当前编译后的dist/business/open-card/index.js 是没有压缩的，所以其实是当前实际的debug模式。

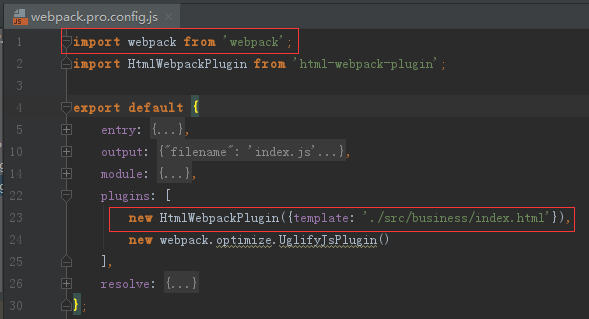
复制一份build-pro.js 取名为build-debug.js、

复制一份webpack.pro.config.js取名为webpack.debug.config.js

build-debug.js 内容修改如下：



Webpack.pro.config.js 内容修改如下：



现在就有了，开发模式、生产模式、debug模式

开发模式和生产模式都还有问题，先暂时如此，先实现开发模式。后面会再优化。

1. 写ui-components项目

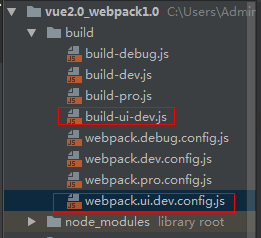
（这个项目用来展示组件的样式，包括公有组件、FB项目组件）

新建文件夹ui-components。src/business/ui-components

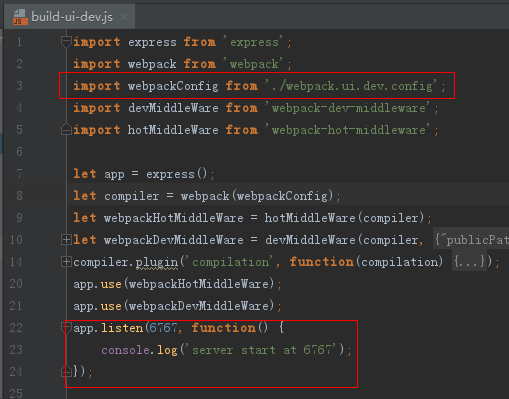
将open-card项目里的所有文件拷贝到ui-components下

复制一份build/build-dev.js取名为 build/build-ui-dev.js。

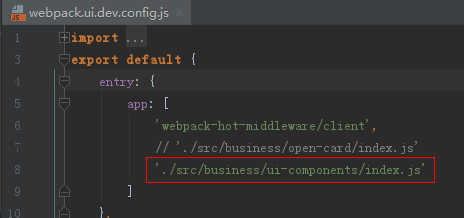
复制一份build/webpack.dev.config.js取名为 build/webpack.ui.dev.config.js。



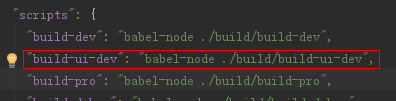
build.ui.dev.js内容修改如下：



webpack.ui.dev.config.js内容修改如下：



package.json 的script参数新增命令：



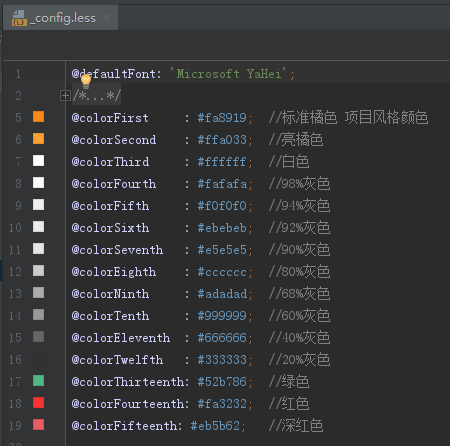
接下来要画如下页面：

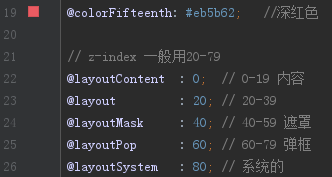


新建文件\_config.less，路径src/styles/config/\_config.less

business下所有业务的样式，颜色变量都引用这里的，为了统一。

\_config.less内容如下：



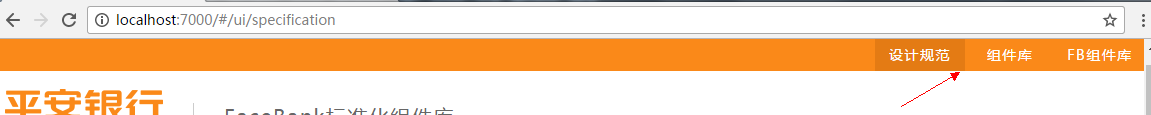




将pingan-bank/src/business/ui-components/assets 拷贝到当前目录对应的文件夹下。

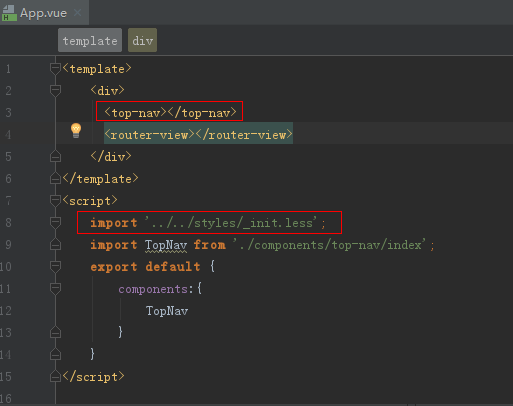
将pingan-bank/src/styles/\_init.less 拷贝到当前目录对应的文件夹下。

下面这块使用组件完成，组件名top-nav，放在ui-components/components下

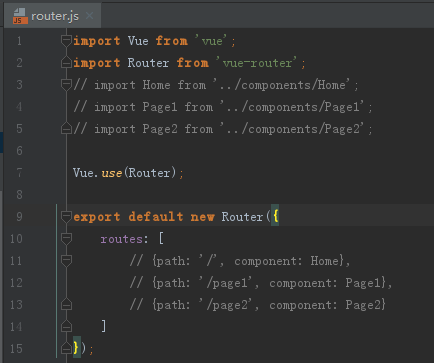


直接从pingan-bank/src/business/ui-components/components/top-nav拷贝过来即可。

App.vue内容修改如下：

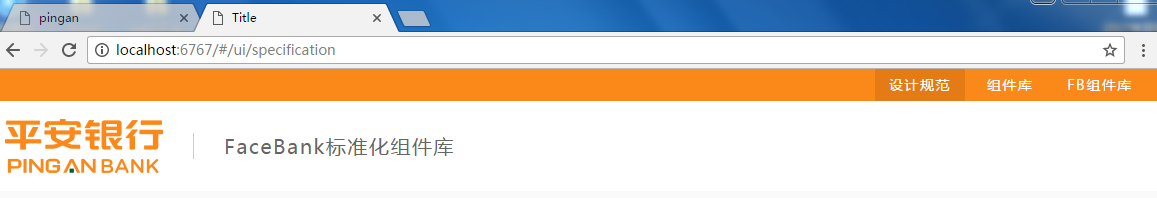


Src/business/ui-components/components/router.js 内容修改如下：



执行命令 npm run build-ui-dev

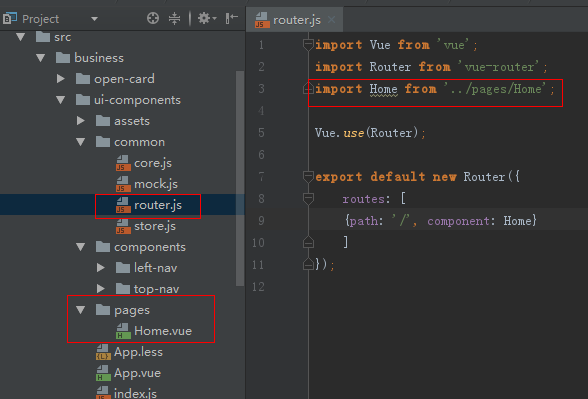
第一个组件迁移成功！



接着引入左侧导航



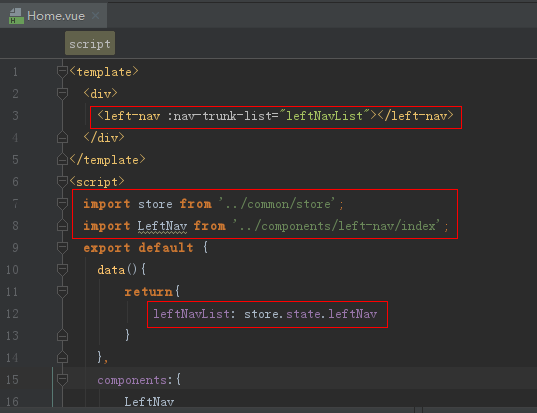
将目录结构调整为如下，router.js内容修改如下：



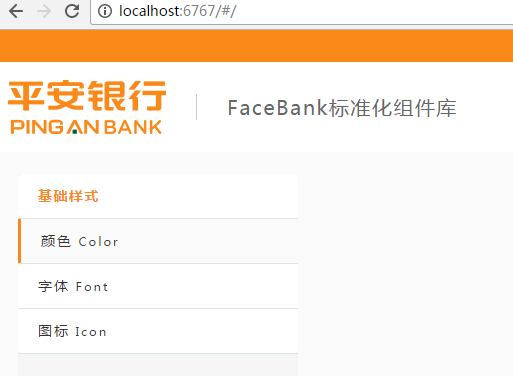
将pingan-bank/src/business/ui-components/modules/specification.js的

state参数的leftNav值拷贝到ui-components/common/store.js的state参数下

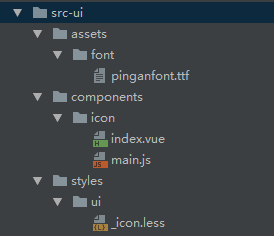
Home.vue修改如下：



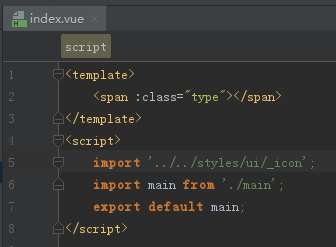
刷新页面，左侧导航出现：



将pingan-bank/src-ui下的文件，拷贝到自己项目对应文件夹中：

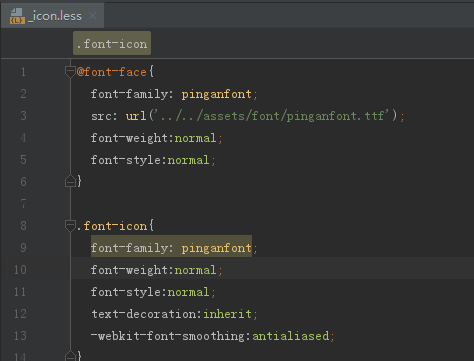


Src-ui/components/icon/indev.vue内容修改如下：



Src-ui/components/icon/main.js里面不再引用样式。

Src-ui/styles/ui/\_icon.less内容修改如下：



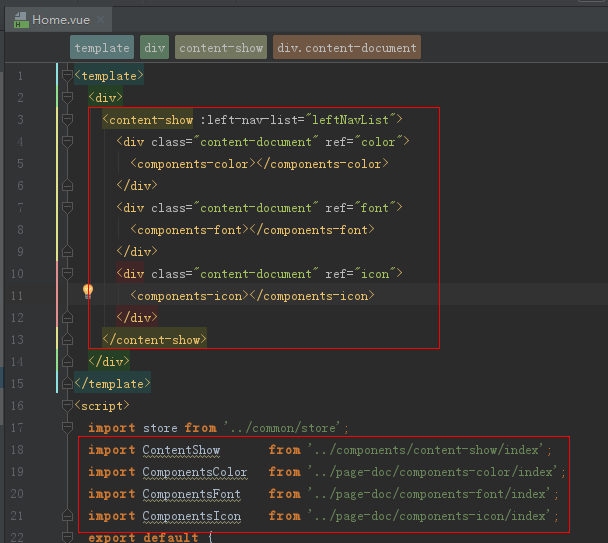
src/styles/\_init.less 中的样式修改如下：

html，body{

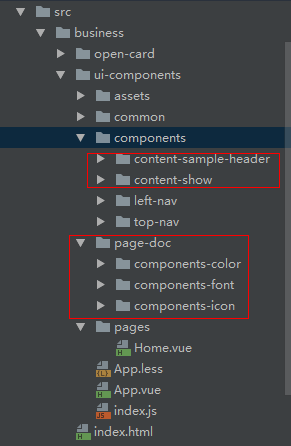
// Overflow:hidden 注释

}

home.vue内容修改如下：

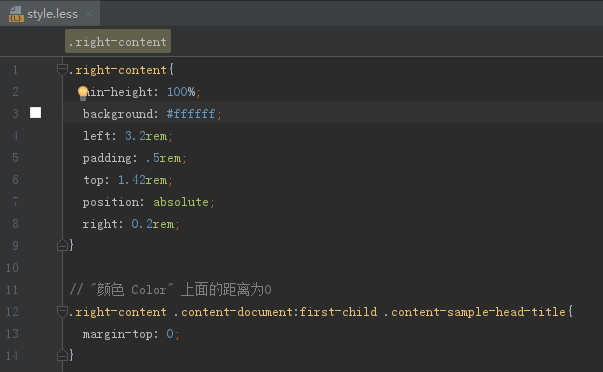


将下面红框的组件从pingan-bank项目里拷贝过来，如下：

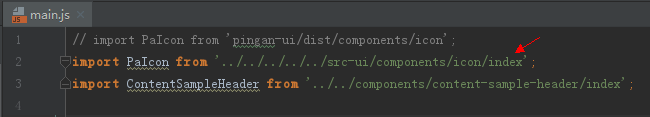


新建文件style.less，路径content-show/style.less。

并在content-show/index.vue中引入它。内容如下：



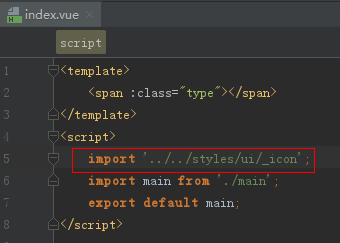
src/business/ui-components/page-doc/components-icon/main.js内容修改如下：



要清楚哪个组件是哪个项目下才有的。

比如ContentSampleHeader组件只在ui-components项目里才有，所以，这个组件放在这个项目的components下。

Src-ui/components/icon/index.vue 内容修改如下：



刷新页面

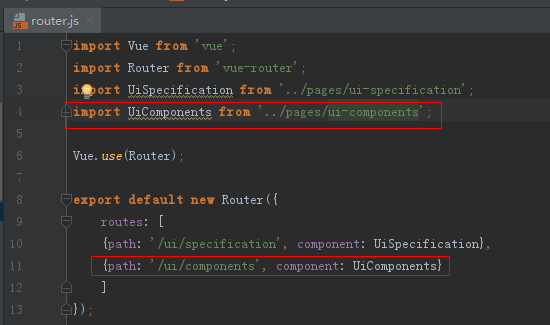
第一个模块--设计规范的页面就做好了。



接来下开始实现第二个模块：

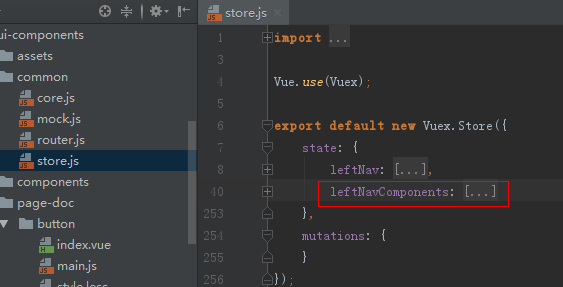
将pages/Home.vue改成ui-specification.vue。新建一个vue文件，取名ui-components.vue。

Common/router.js内容修改如下：

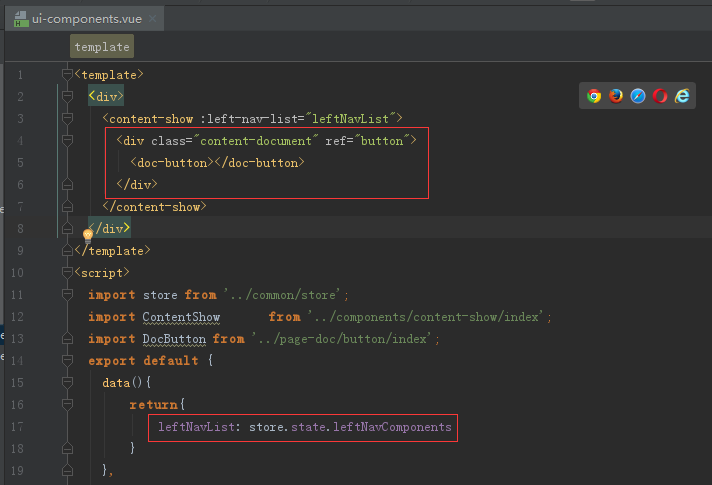


将pingan-bank/src/business/ui-components/modules/components.js 的

参数state的leftNav的值，拷贝到自己项目的store下，key为leftNavComponents



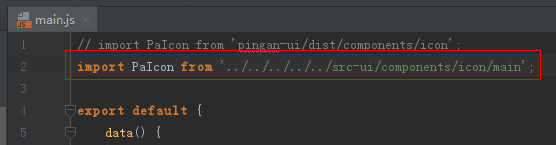
ui-components.vue 内容如下：



从pingan-bank/src/business/ui-components

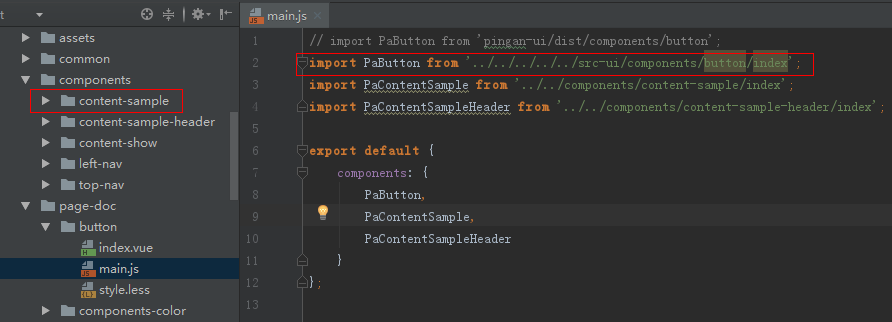
里拷贝content-sample组件到src/business/ui-components/components下

content-sample/main.js 内容修改如下：



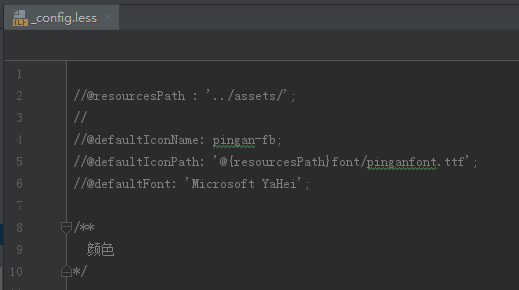
从pingan-bank/src/business/ui-components/page-doc下拷贝button组件到

相应目录下，button/main.js内容修改如下：



从pingan-bank的src-ui/components下拷贝button组件到自己项目对应目录下。

将pingan-bank/src-ui/styles/\_config.less拷贝到自己项目对应目录下，内容修改如下：



因为ui-components可以看成是一个独立的项目，所以它应该有自己的样式。

并且，公有组件，有自己的样式。比如：src-ui/styles下的。

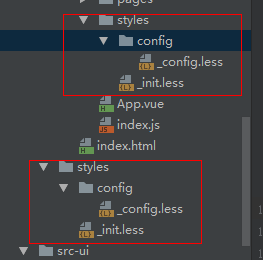
FB项目所有的业务，也有自己的样式，所有业务公共的样式放在src/styles下。

所有业务私有的样式放在自己业务里（open-card下）

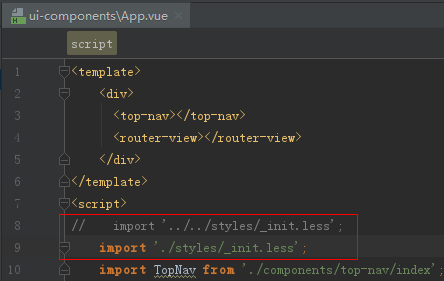
因为ui-components是独立的项目，所以应该有一套自己独立的样式。

新建文件夹styles，删掉App.less。

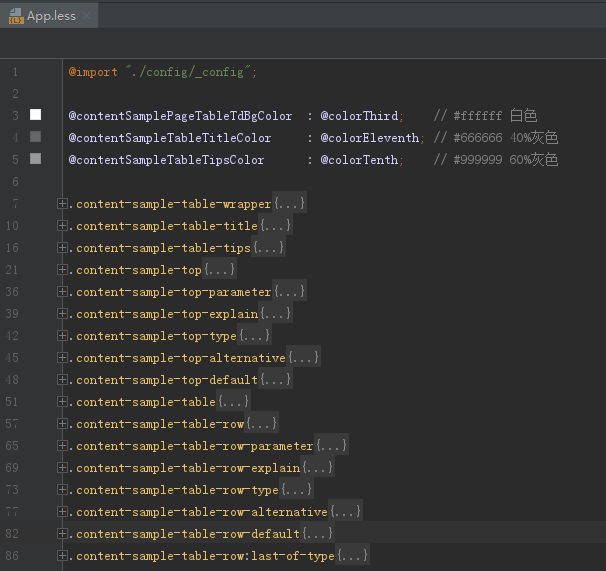
复制src/styles下两个样式到src/business/ui-componens/styles下。



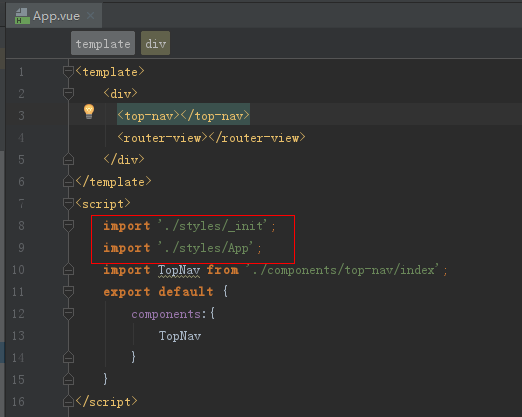
src/business/ui-componens/App.vue内容修改如下：



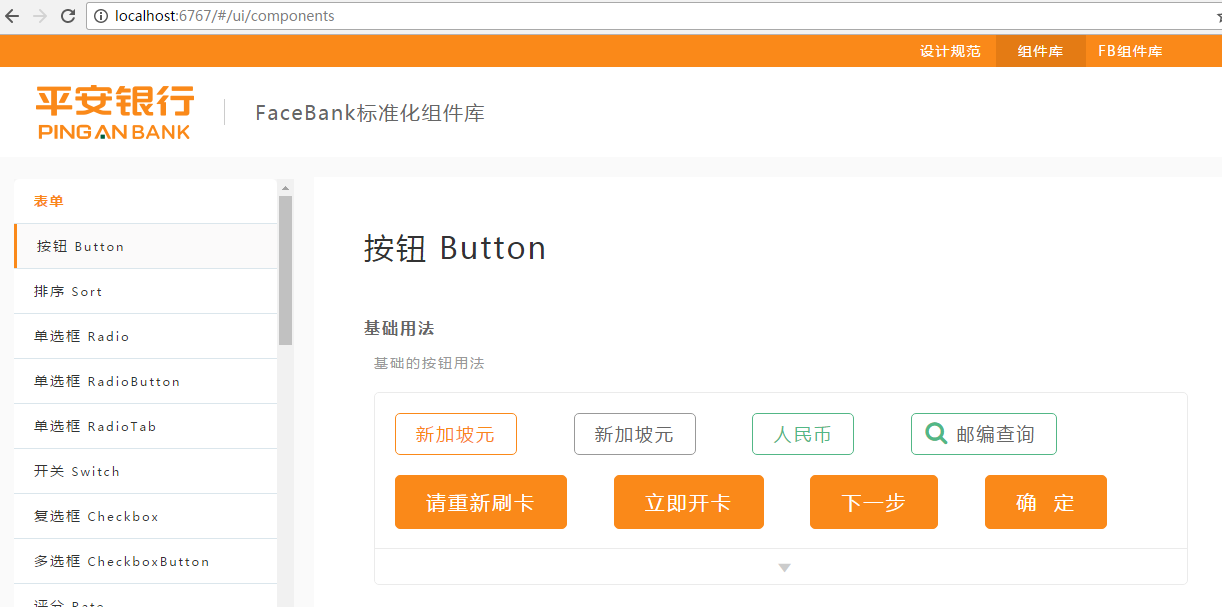
ui-components/styles下新增文件App.less内容如下：



其中黄色部分样式和变量从pingan-bank/src/business/ui-componnts/style/style.less中复制



到现在就实现了一个组件。

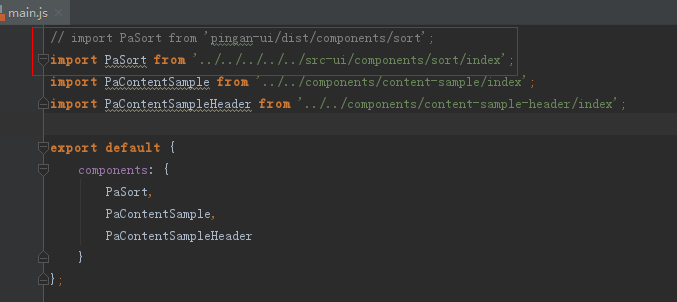


接下来迁移sort组件

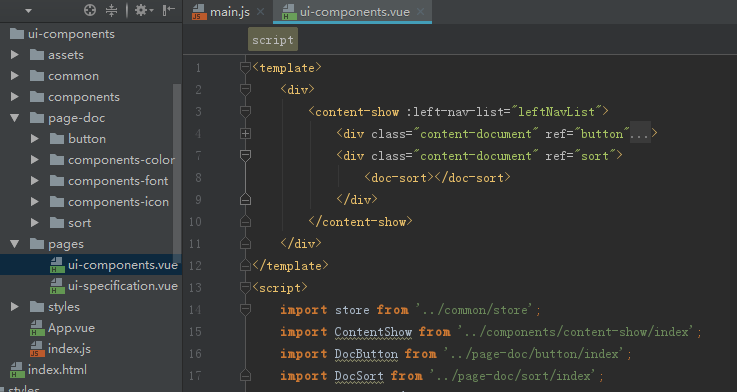
将pingan-bank/src-ui/components/sort拷贝到自己项目的相应目录下。

将pingan-bank/src/business/ui-components/page-doc/sort拷贝到自己项目的相应目录。

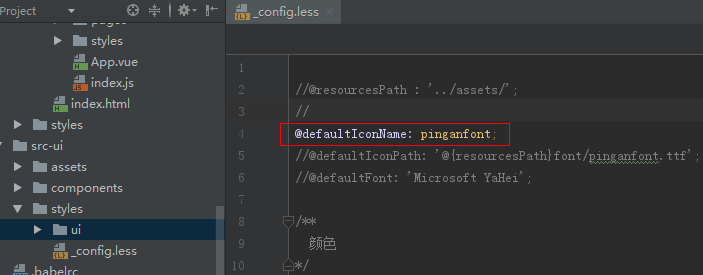
ui-components/page-doc/sort/main.js 修改如下：



pages/ui-components.vue 内容修改如下：



src-ui/styles/\_config.less内容修改如下：



刷新页面，发现sort组件迁移成功。



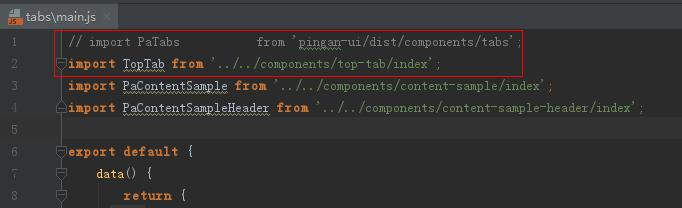
相同的方法，实现迁移Radio组件。

相同的方法，实现迁移表单其他组件。

迁移tabs组件时，

**注意：**page-doc/tabs组件引用的是ui-components/components/top-tab的组件

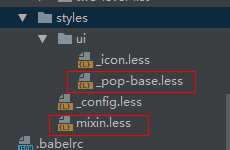
这个组件应该不属于公有组件，只是临时性存放在此的组件。**后面应该放到正确的地方**。



ui-componenst/page-doc/slider-toggle/main.js内容修改如下：

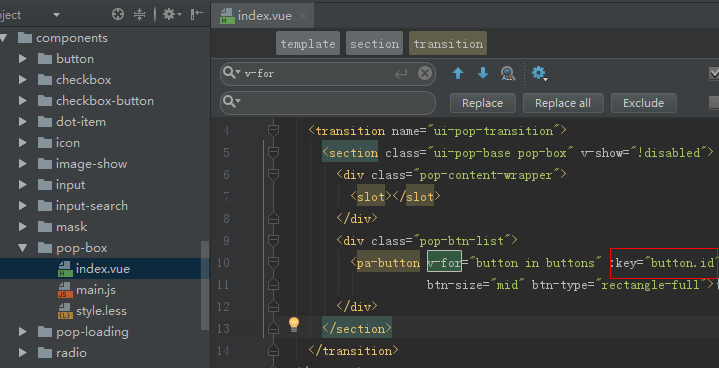


迁移弹框loading组件时，要从pingan-bank拷贝两个样式到对应位置。



Src-ui/components/pop-box/index.vue加上 :key = “button.id” 否则会报警告。

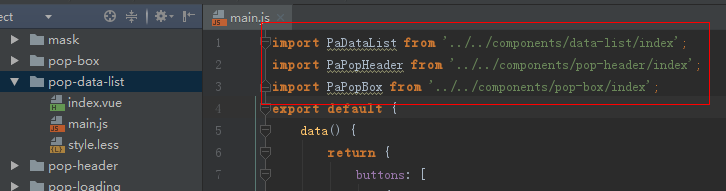
这个警告是vue2.2.0以上才有的，pingan-bank是vue2.1.0。



**注意**：pop-data-list组件要引用三个组件，以后还是不要组件嵌套组件，嵌套太多不好。

1. data-list
2. pop-header
3. Pop-box

如下所示：

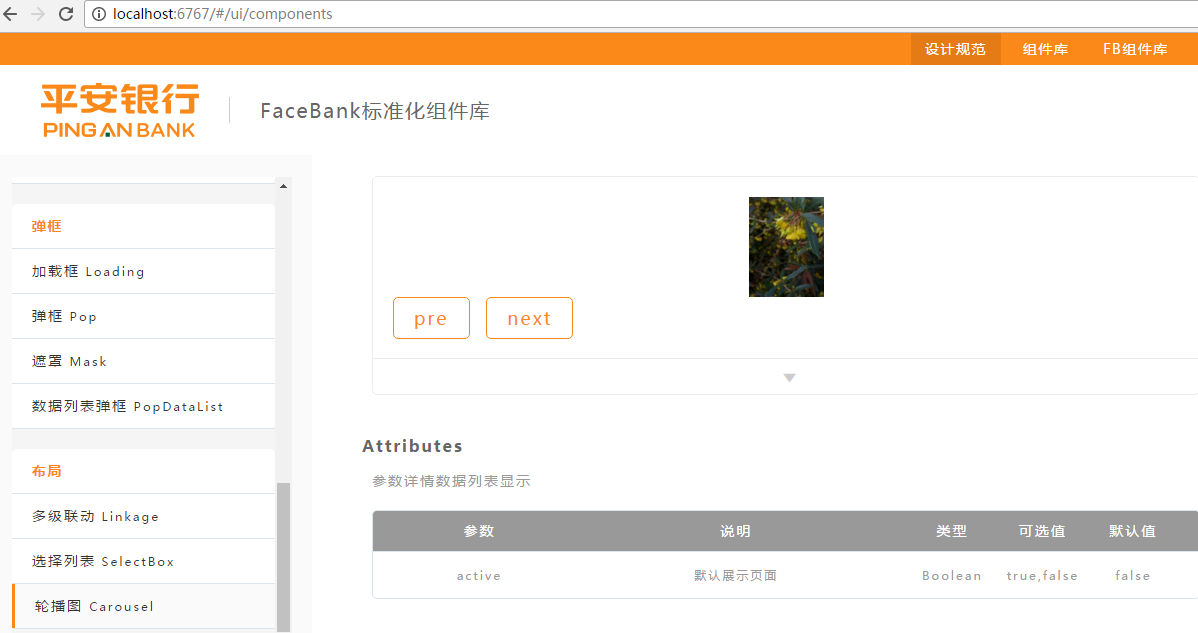


**注意**：select-box组件要引用三个组件

Ui-components/page-doc/select-box/main.js 内容中引用如下：



**注意**：carousel组件要引用一个组件：carousel-item组件。



到此第二个模块就实现了。

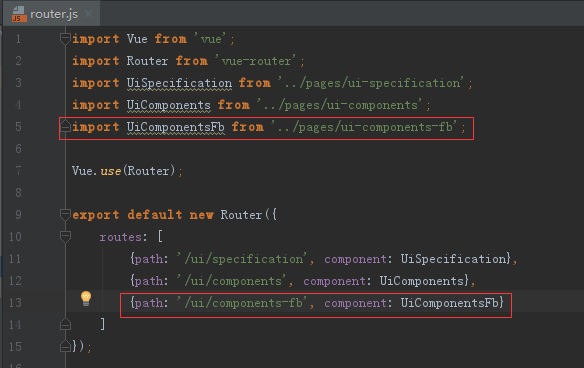
有一个东西要问杰哥 ui-components/components/top-tab到底是不是公共组件？

//-----------------------------------------------------------------------------------------------------------------

接来下开始实现第三个模块：

Src/business/ui-components/pages下新建ui-components-fb.vue文件

Common/router.js内容修改如下：

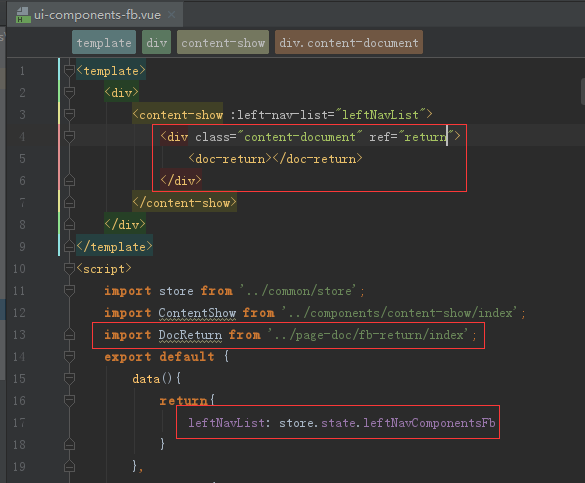


将pingan-bank/src/business/ui-components/modules/components-fb.js 的

参数state的leftNav的值，拷贝到自己项目的store下，key为leftNavComponentsFb



ui-components-fb.vue 内容如下：



因为return组件是fb项目才有的组件，并且是fb项目下的所有业务都要用到，所以该组件应该放到src/components 下面。

将pnigan-bank/src/business/ui-components/page-doc下的fb-return组件拷贝到相应目录

将pnigan-bank/src/components下的return组件拷贝到相应目录

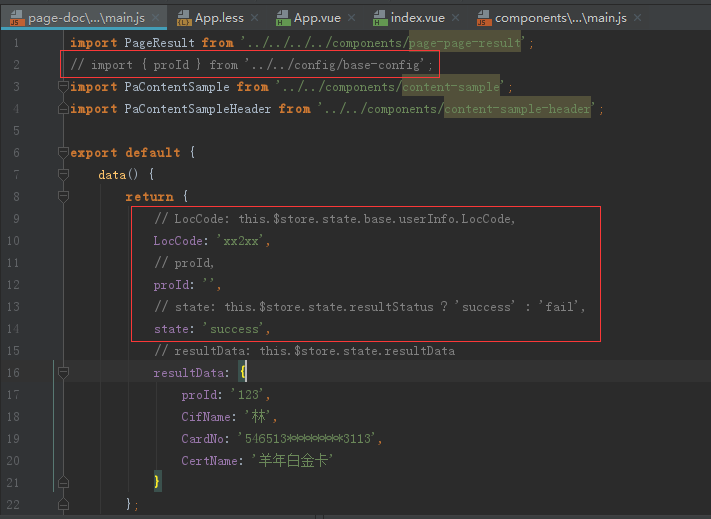
Return组件迁移成功，显示如下：



同样的方法迁移PageResult组件

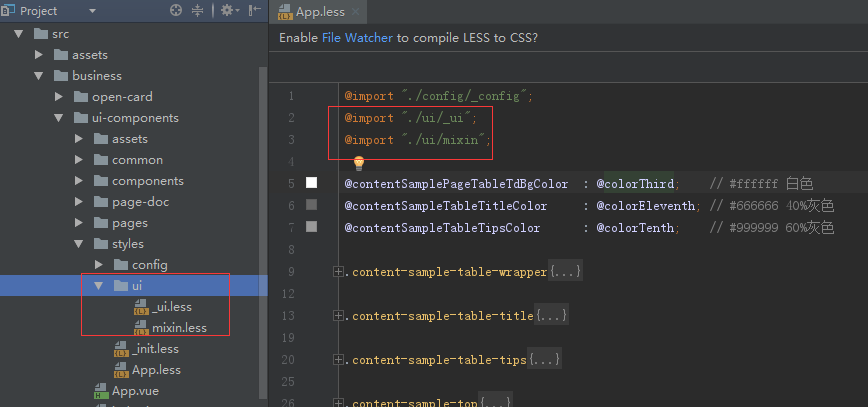
Src/business/ui-components/page-doc/

page-result/page-page-resul/main.js修改如下：



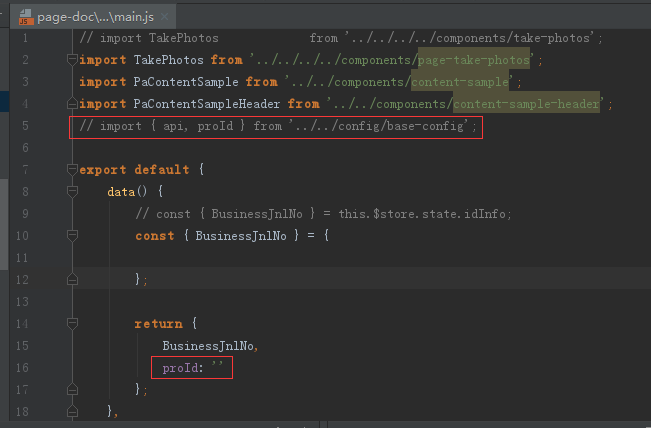
Src/business/ui-components/styles下新增目录ui，ui下新增文件\_ui.less和mixin.less

这两个文件从pingan-bank/src/styles/目录下拷贝过来。

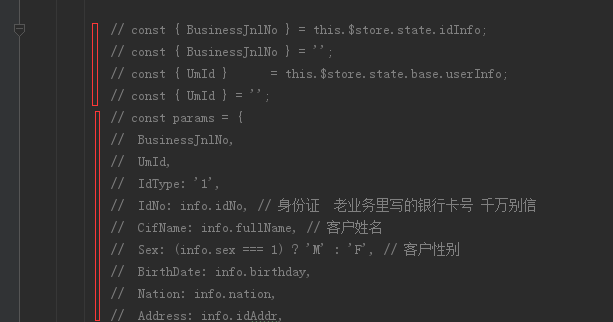


相同方法迁移 navigator

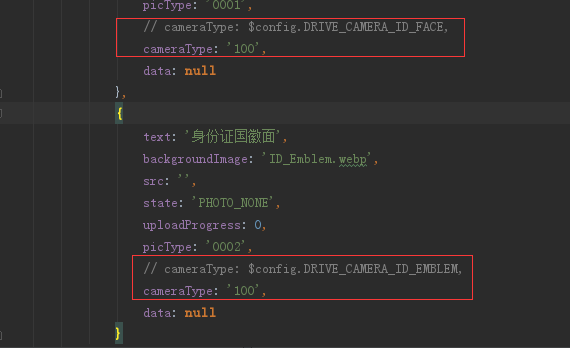
迁移swipe-card时，src/business/ui-components/page-doc/page-swipe-card/main.js



还要注释一些不要的代码，不然校验过不去



迁移take-photo组件时，Src/components/page-take-photos/main.js修改如下：

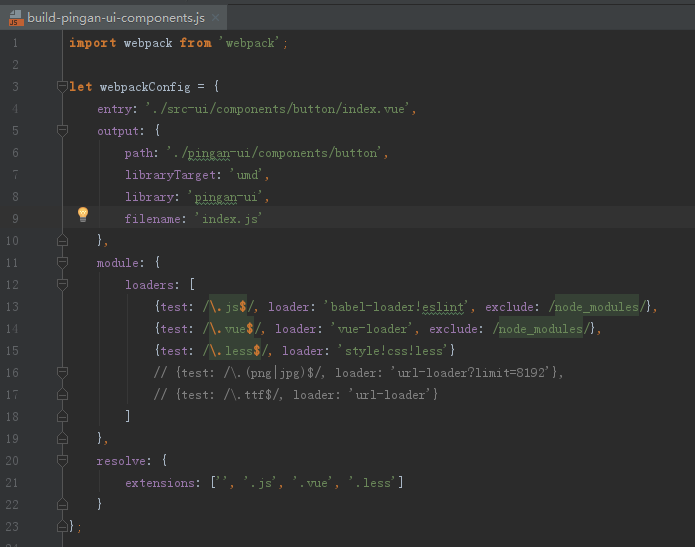


相同方法，迁移readvideo组件，有些代码需要注释掉，后面要调用android/ios的方法，再回来优化。

//---------------------------------------------------------------------

接下来，将src-ui目录下的所有组件进行编译，fb项目在使用时，直接使用编译之后的组件。

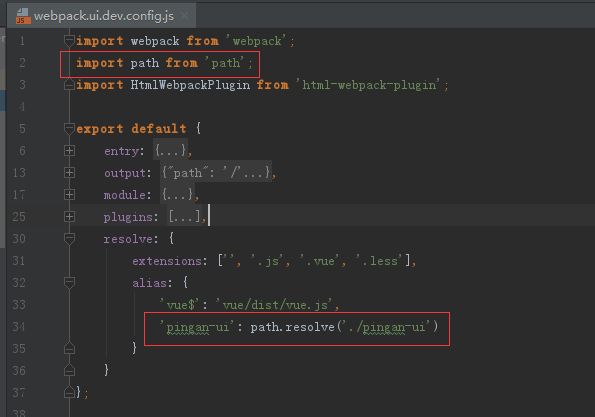
新建build-pingan-ui-components.js （build / build-pingan-ui-components.js ）内容如下：



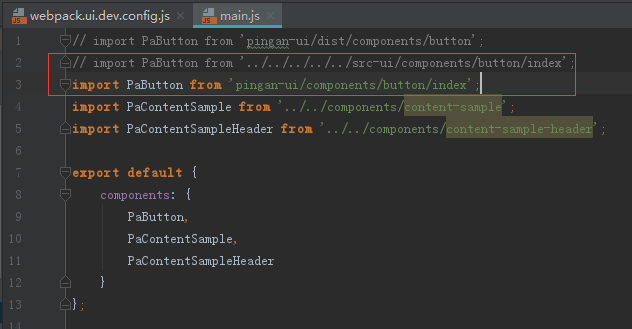
注意啊：libraryTarget 与 library 表示pingan-ui目录下生成的js可以被umd模式加载。

Js库文件可以分AMD模式、CMD模式。

webpack.ui.dev.config.js 修改如下：



Src/business/ui-components/page-doc/button/main.js 修改如下：



编译之后，button显示正常则没有问题：



相同的方法，编译其他全部pingan-ui组件，结束之后，修改src-ui文件夹名字。

再run build-ui-dev 看是否有报错。修改没有引用编译之后ping-ui组件的目录路径。

没有报错之后，再将src-ui文件夹名称改回来。

// ----------------------------------------------------------------------------------------------------------------------

1. 尝试迁移业务

以远程监测（longDistanceCheck）为例，迁移中，完善与ios/android对接的公有方法；完善生产模式、debug模式的配置。

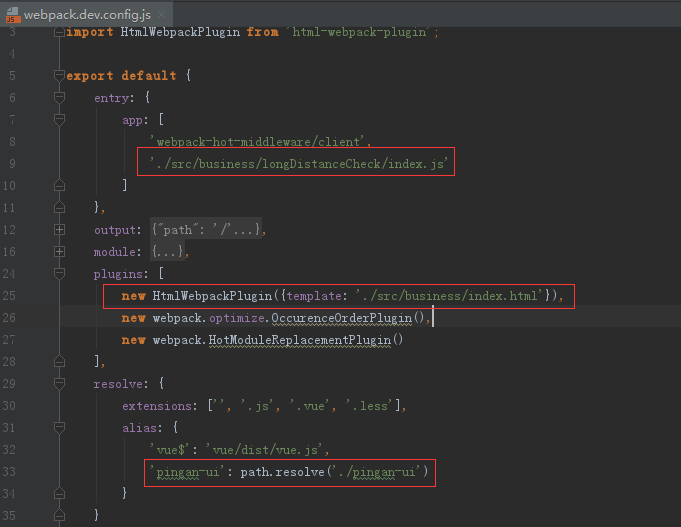
1. src/business下新建文件夹longDistanceCheck。

Src/business 下新建文件夹 assets，存放所有业务都要用到的资源。

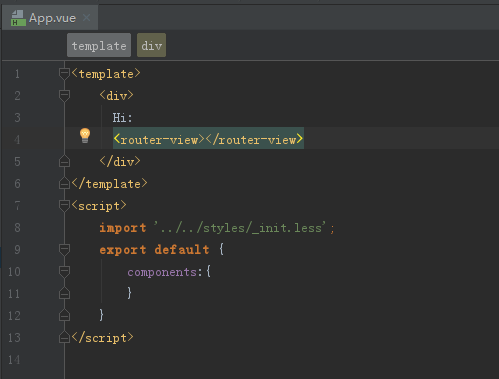
Src/business/assets/font 下放pingan现在正在使用的字体。

Src/business/common 下从ui-components/common下拷贝core.js过来

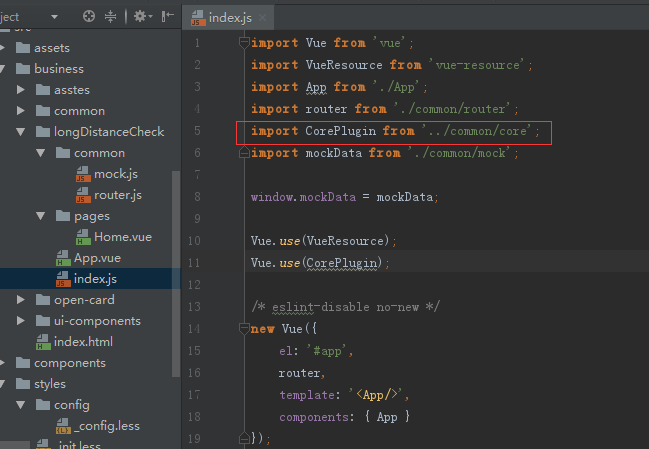
webpack.dev.config.js 文件修改如下：



1. longDistanceCheck目录下，新建App.vue内容如下：



longDistanceCheck目录下，新建index.js内容如下：



将longDistanceCheck文件夹复制一份，取名demo，以后要改业务时直接复制它，再改名，再在上面修改即可。

// -----------------------------------------------

将这个项目名称和子目录名称稍微改下：

项目名称改成：pingAnFaceBank

src-ui 改成 pa-ui-src

pingan-ui改成 pa-ui

src改成pa-facebank-src

dist改成 pa-facebank