# 项目优化任务指导书

[项目优化任务指导书 1](#_Toc1693524483)

[一． 实验效果 2](#_Toc317728443)

[二． 实验目的 3](#_Toc1417595059)

[三． 实验原理 4](#_Toc1336576795)

[1. 组件封装 4](#_Toc1167245945)

[2. 请求库 Axios 二次封装 6](#_Toc639482270)

[四． 实验环境 7](#_Toc1770342302)

[1． 开发语言 7](#_Toc757140529)

[2． 开发环境 7](#_Toc1420262428)

[3． 调试环境 7](#_Toc1069890991)

[4． Vue-cli配置 7](#_Toc777309406)

[5． Nginx服务器 7](#_Toc1096161941)

[五． 实验内容 7](#_Toc2079018421)

[1. 页面结构搭建 7](#_Toc356181410)

[2.接口连调 8](#_Toc1304033681)

[3.页面适配 8](#_Toc1823458932)

[4. 实训任务工作量对标公司工作量 8](#_Toc135143787)

[六． 功能思路分析 9](#_Toc1471413230)

[七． 关键技术点解决方案 9](#_Toc1767961405)

[1、组件封装思路 9](#_Toc1543593943)

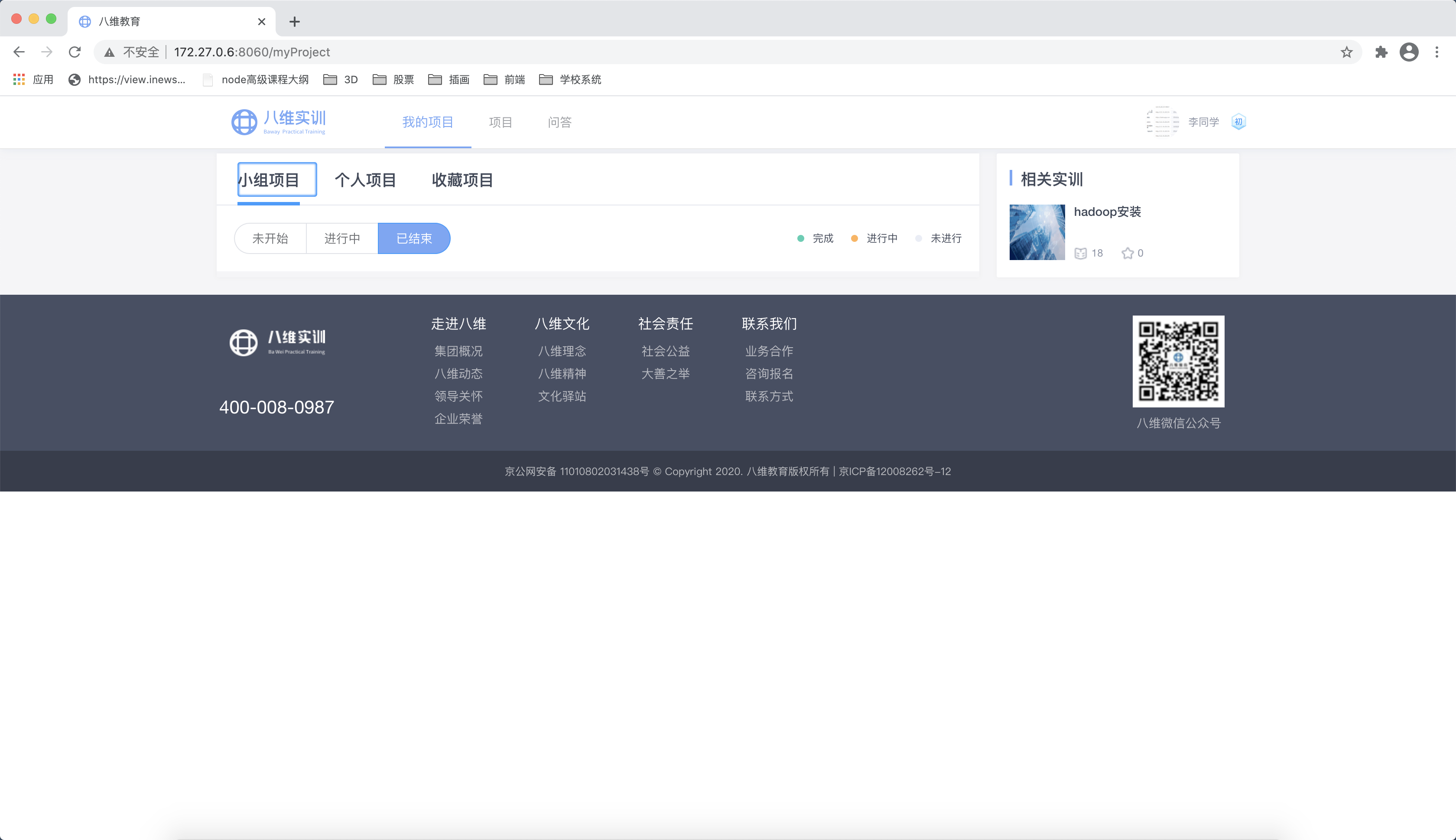
[八． 实验步骤 10](#_Toc1580944241)

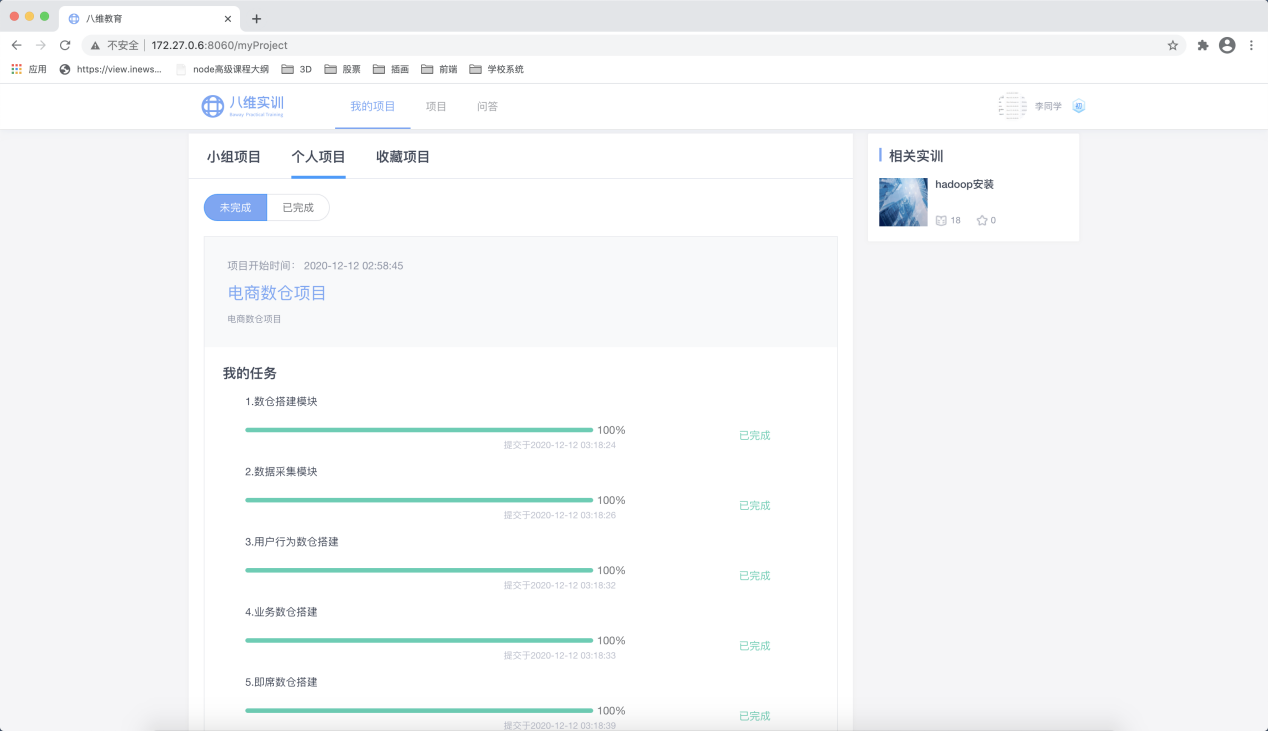
[1． 组件模版开发 10](#_Toc114694156)

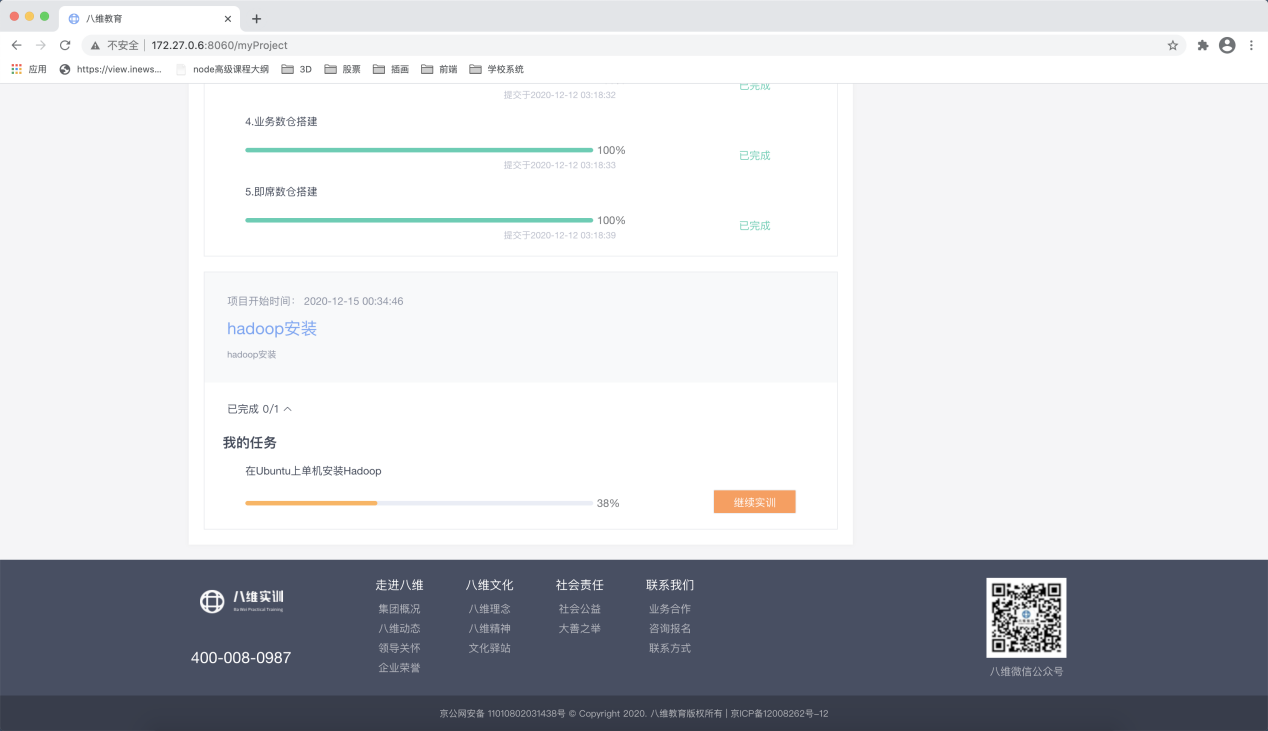
[2． 页面接口调用 11](#_Toc1371848533)

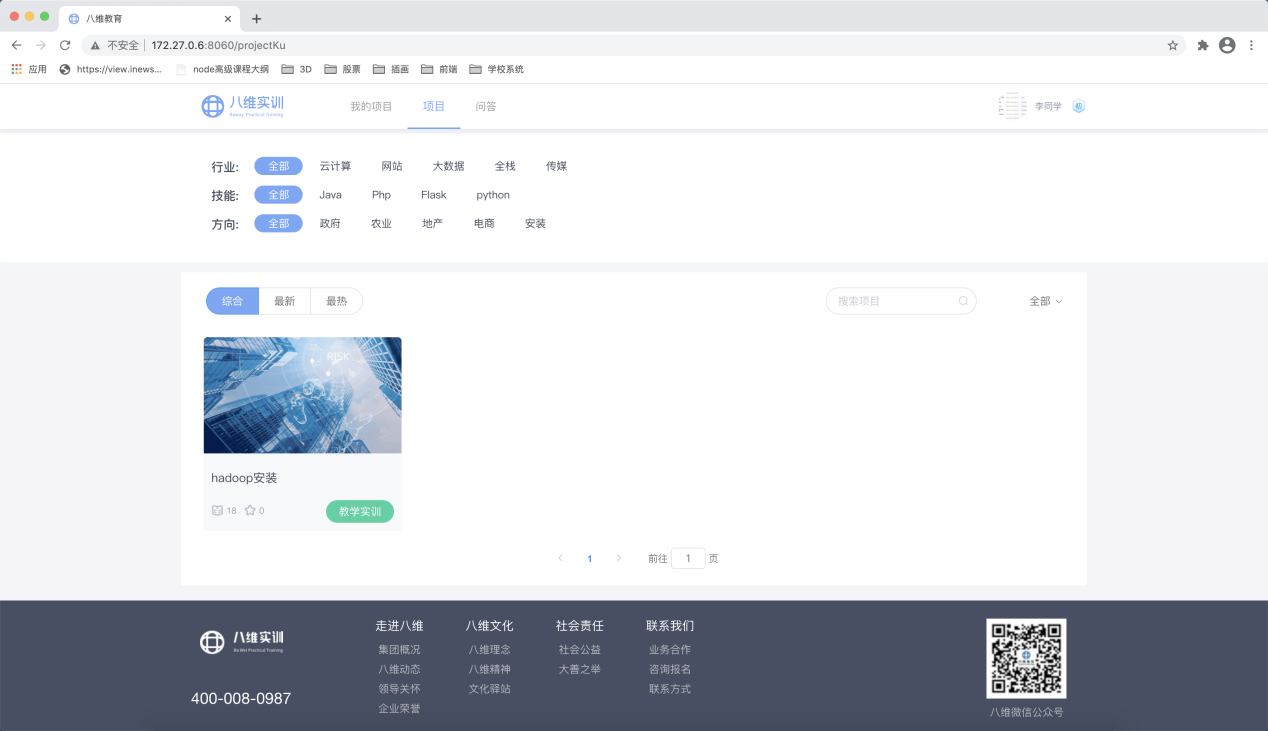
[3． 页面样式开发 12](#_Toc1273859939)

## 实验效果









## 实验目的

项目库模块是整个学生端最重要的功能模块，通过项目库模块合理划分不同类型的项目（小组项目，个人项目，收藏项目），并且更具线上IDE的在线开发，自动识别当前项目进度，进行可视化进度标识，可以进行项目的存档，下一次可直接通过任务进入到当前开发， 避免了学生个人电脑上开发，目录划分不清晰导致项目混乱的问题。

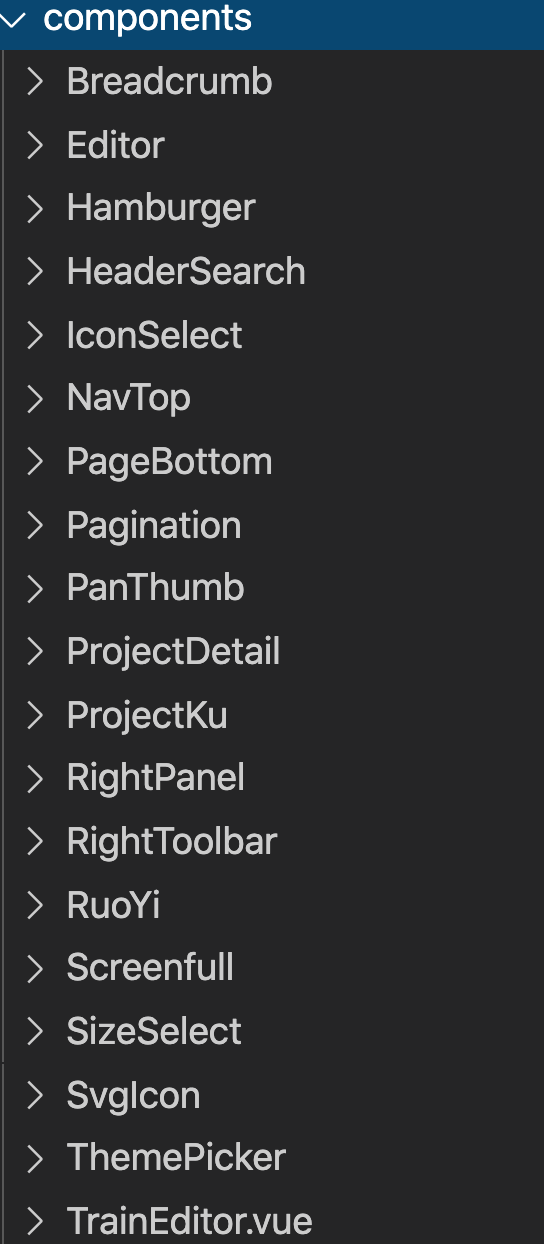
技术上项目库模块没有特别难的技术点实现，但都是网站常见功能开发，通过项目库模块的开发保证学生的基础技术开发工作

## 实验原理

依据在企业开发中如何提高项目开发效率为核心进行业务功能开发时候的封装

### 1. 组件封装

在项目中使用的饿了么出品的ElementUI组件库，但组件库提供的都是基础UI组件，为了提高项目的开发效率我们需要在UI组件库的基础之上针对于当前项目进行更加贴合当前项目的业务组件



### 2. 请求库 Axios 二次封装

请求库的封装在一个前后端分离的项目中尤为关键，为什么这么说呢？api 可以统一管理，不管接口数量增加多少，所有的接口都可以统一经过一个地方做处理，逻辑会比较清晰。当项目小且接口数不多的情况下，直接使用 axios 在页面里请求，问题不大。但一旦页面多了，接口多了，问题就被无限放大，错误的返回，接口的维护都变得不可控。 首先要安装 axios：



## 实验环境

### 开发语言

Vue+Element-UI+Axios+Webpack+Nginx

### 开发环境

Node+npm

### 调试环境

Chrome Devtools中network审查，缓存测试

### Vue-cli配置

webpack配置调优，webpack构建脚本

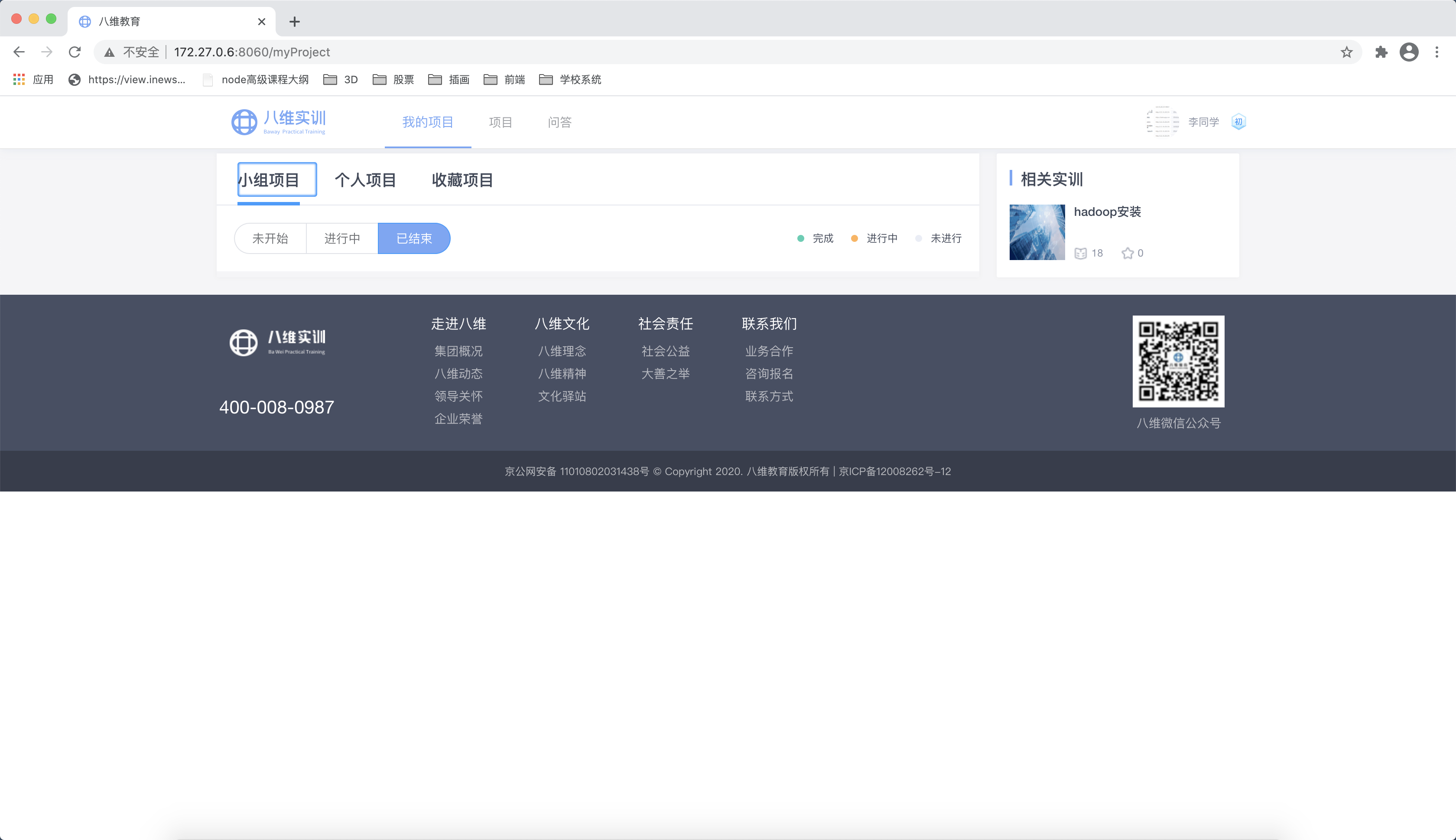
### Nginx服务器

gzip压缩配置，服务器缓存配置

## 实验内容

## 1. 页面结构搭建

根据UI设计稿进行页面结构搭建，保证百分比还原设计



## 2.接口连调

根据Swagger提供的接口文档，进行接口的调用，并且渲染数据，根据业务逻辑展示正确的数据结果

## 3.页面适配

如何适配移动端和PC端进行适配，我们采用栅栏布局进行系统布局实现响应式网站开发,让用户通过各种尺寸的设备浏览网站获得良好的视觉效果的方法

响应式 Web 设计工作原理

为了应用响应式 Web 设计，您需要创建一个包含适应各种设备尺寸样式的 CSS。一旦页面在特定的设备上加载，该页面上使用了各种字体和 Web 开发技术，比如媒体查询（Media Queries），此时，会先检测设备的视口大小，然后加载特定于设备的样式。

## 实训任务工作量对标公司工作量

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 组件封装 | 架构搭建 | 业务对接 | 接口联调 | 页面布局 | 总计时 |
| 涉及的知识点 | 组件开发 | Vue-cli | 联调 | axios | css |  |
| 公司工作量 | 1周 | 2天 | 3天 | 1天 | 2天 | 13天 |
| 实训工作量 | 2天 | 1天 | 1天 | .5天 | 1天 | 5.5天 |

## 

## 功能思路分析

* + 1. 理解模块需求，完成功能拆分
    2. 抽象功能，拆分组件，封装组件
    3. 设计组件参数和数据流向
    4. 完成页面响应式布局
    5. 对接后端接口进行数据渲染

## 关键技术点解决方案

## 1、组件封装思路

Vue.js 组件主要分成三类：

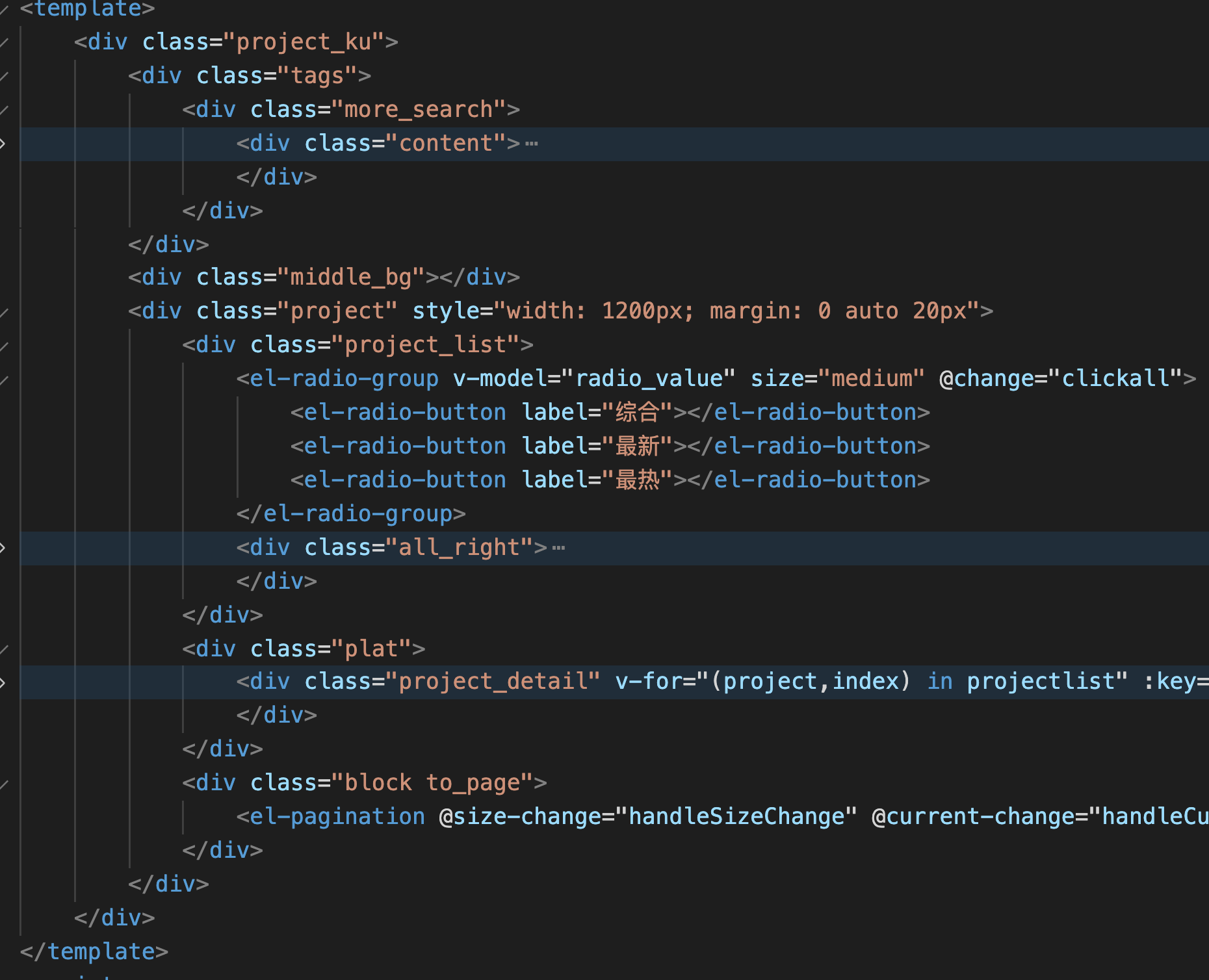
由 vue-router 产生的每个页面，它本质上也是一个组件（.vue），主要承载当前页面的 HTML 结构，会包含数据获取、数据整理、数据可视化等常规业务。整个文件相对较大，但一般不会有 props 选项和 自定义事件，因为它作为路由的渲染，不会被复用，因此也不会对外提供接口。在项目开发中，我们写的大部分代码都是这类的组件（页面），协同开发时，每人维护自己的页面，很少有交集。这类组件相对是最好写的，因为主要是还原设计稿，完成需求，不需要太多模块和架构设计上的考虑。

不包含业务，独立、具体功能的基础组件，比如日期选择器、模态框等。这类组件作为项目的基础控件，会被大量使用，因此组件的 API 进行过高强度的抽象，可以通过不同配置实现不同的功能。比如笔者开源的 iView，就是包含了 50 多个这样基础组件的 UI 组件库。每个公司都有自己的组件使用规范或组件库，但要开发和维护一套像 iView 这样的组件库，投入的人力和精力还是很重的，所以出于成本考虑，很多项目都会使用已有的开源组件库。独立组件的开发难度要高于第一类组件，因为它的侧重点是 API 的设计、兼容性、性能、以及复杂的功能。这类组件对 JavaScript 的编程能力有一定要求，也会包含非常多的技巧，比如在不依赖 Vuex 和 Bus（因为独立组件，无法依赖其它库）的情况下，各组件间的通信，还会涉及很多脑壳疼的逻辑，比如日期选择器要考虑不同时区、国家的日历习惯，支持多种日期格式。

业务组件。它不像第二类独立组件只包含某个功能，而是在业务中被多个页面复用的，它与独立组件的区别是，业务组件只在当前项目中会用到，不具有通用性，而且会包含一些业务，比如数据请求；而独立组件不含业务，在任何项目中都可以使用，功能单一，比如一个具有数据校验功能的输入框。业务组件更像是介于第一类和第二类之间，在开发上也与独立组件类似，但寄托于项目，你可以使用项目中的技术栈，比如 Vuex、axios、echarts 等，所以它的开发难度相对独立组件要容易点，但也有必要考虑组件的可维护性和复用性。

### 实验步骤

### 组件模版开发



### 页面接口调用



### 页面样式开发

