



下载APP



## 15 | 实战痛点 1：复杂 Vue 项目的规范和基础库封装

2021-11-19 大圣

《玩转Vue 3全家桶》

课程介绍 &gt;



讲述：大圣

时长 09:44 大小 8.92M



你好，我是大圣，欢迎进入课程的第 15 讲。

在全家桶实战篇的前几讲里，我们学习了 Vue 3 本身的进阶内容。从今天开始，我们尝试着把这些技能都用在实际项目中，聊一下实战中常见的痛点。不过，既然是实际项目，那还是有很多库需要引入的，比如网络请求时用到的 axios、时间处理时用到的 Dayjs 等等。今天我要跟你聊的，则是复杂 Vue 项目的规范和基础库的封装。



### 组件库



在项目开发中，我们首先需要有一个组件库帮助我们快速搭建项目，组件库提供了各式各样的封装完备的组件。现在社区可选择的组件库有 element-plus、antdvue、Naive-UI、

Element3 等，我们选择 Element3 来搭建项目，首先我们来到项目目录下，执行下面的代码安装 Element3。

[复制代码](#)

```
1 npm install element3 --save
```

然后，我们在 src/main.js 中使用一下 Element3。看下面的代码，我们在其中引入了 Element3 和主体对应的 CSS，并使用 use(Element3) 加载组件库。

[复制代码](#)

```
1 import { createApp } from 'vue'
2 import Element3 from 'element3'
3 import 'element3/lib/theme-chalk/index.css'
4 import store from './store/index'
5 import App from './App.vue'
6 import router from './router/index'
7
8 const app = createApp(App)
9
10 app.use(store)
11   .use(router)
12   .use(Element3)
13   .mount('#app')
```

这样，项目的入口页面就注册好了 Element3 内置的组件。关于 Element3 的组件列表，你可以到 [Element3 官网](#) 查看。接下来，我们就可以使用组件库去搭建我们的页面了。

首先，我们打开项目目录下的 src/App.vue 文件，把之前的学习清单应用时的测试代码移除，然后新增下面的代码。你能看到，我们在代码中使用 Element3 的 [Container 布局组件](#) 实现管理系统的整体布局。

[复制代码](#)

```
1 <template>
2
3 <el-container>
4   <el-header>Header</el-header>
5   <el-container>
6     <el-aside width="200px">
7       <div>
8         <router-link to="/"> Home</router-link>
```

```
9     </div>
10     <div>
11         <router-link to="/about">About</router-link>
12     </div>
13 </el-aside>
14 <el-container>
15     <el-main>
16         <router-view></router-view>
17     </el-main>
18 </el-container>
19 </el-container>
20 </el-container>
21 </template>
22
23 <script setup>
24
25 </script>
26 <style>
27   .el-header,
28   .el-footer {
29     background-color: #b3c0d1;
30     color: #333;
31     text-align: center;
32   }
33
34   .el-aside {
35     background-color: #d3dce6;
36     color: #333;
37   }
38
39   .el-main {
40     background-color: #e9eef3;
41     color: #333;
42   }
43
44   body > .el-container {
45     margin-bottom: 40px;
46   }
47 </style>
```

上面代码对应在前端的显示格局如下，代码上方的 Header 组件，承载着页面的头部信息，包括项目左上角的名字、右上角的用户信息、消息等等。代码中的 aside 对应了前端页面左侧的侧边栏，承载着页面主要的导航信息；main 组件内部使用 router-view 渲染路由对应的组件，然后我们继续使用 Element3 逐渐丰富 Header 和导航信息。



在 Element3 中，我们也可以很方便地找出我们需要的组件，就像 Menu 等等。在下面的代码中，我们使用 el-menu 组件渲染 header 组件，el-menu 内部使用 el-menu-item 渲染导航组件。

[复制代码](#)

```
1 <el-header>
2 <el-menu
3   :default-active="1"
4   class="el-menu-demo"
5   mode="horizontal"
6   background-color="#545c64"
7   text-color="#fff"
8   active-text-color="#ffd04b"
9 >
10 <el-menu-item index="1">处理中心</el-menu-item>
11 <el-submenu index="2">
12   <template v-slot:title>我的工作台</template>
13   <el-menu-item index="2-1">选项1</el-menu-item>
14   <el-menu-item index="2-2">选项2</el-menu-item>
15   <el-menu-item index="2-3">选项3</el-menu-item>
16   <el-submenu index="2-4">
17     <template v-slot:title>选项4</template>
18     <el-menu-item index="2-4-1">选项1</el-menu-item>
19     <el-menu-item index="2-4-2">选项2</el-menu-item>
20     <el-menu-item index="2-4-3">选项3</el-menu-item>
21   </el-submenu>
22 </el-submenu>
23 <el-menu-item index="3" disabled>消息中心</el-menu-item>
24 <el-menu-item index="4"
25   ><a href="https://element3-ui.com" target="_blank"
26     >订单管理</a>
27   </el-menu-item>
28 >
29 </el-menu>
30
31 </el-header>
```

使用 menu 组件渲染 header 和 aslide 组件后，页面布局示意图如下，这样页面的基本结构就搭建完毕了。



## 工具库

完成页面基本结构的搭建后，在我们获取后端数据时，需要使用 axios 发起网络请求。在项目的根目录下，打开命令行，执行下面的命令，这样我们就可以安装 axios 了（axios 跟 Vue 版本没有直接关系，安装最新即可）。

```
1 npm i axios -D
```

[复制代码](#)

axios 作为现在最流行的网络请求库，可以直接使用 axios.get 或者 axios.post 去获取数据。但是在项目开发中，业务逻辑有很多配置需要进行统一设置，**所以安装完 axios 之后，我们需要做的就是封装项目中的业务逻辑。**


首先，在项目在登录成功之后，后端会返回一个 token，用来存储用户的加密信息，我们把 token 放在每一次的 http 请求的 header 中，后端在收到请求之后，会对请求 header 中的 token 进行认证，然后解密出用户的信息，过期时间，并且查询用户的权限后，校验完毕才会返回对应的数据。

所以我们要对所有的 http 请求进行统一拦截，确保在请求发出之前，从本地存储中获取 token，这样就不需要在每个发起请求的组件内去读取本地存储。后端数据如果出错的话，



接口还要进行统一拦截，比如接口返回的错误是登录状态过期，那么就需要提示用户跳转到登录页面重新登录。

这样，我们就把网络接口中需要统一处理的内容都放在了拦截器中统一处理了。在下面的代码中，所有接口在请求发出之前，都会使用 `getToken` 获取 token，然后放在 header 中。在接口返回报错信息的时候，会在调试窗口统一打印报错信息。在项目的组件中，我们只需要直接使用封装好的 `axios` 即可。

 复制代码

```
1 import axios from 'axios'
2 import { useMsgbox, Message } from 'element3'
3 import store from '@/store'
4 import { getToken } from '@/utils/auth'
5
6 const service = axios.create({
7   baseURL: process.env.VUE_APP_BASE_API, // url = base url + request url
8   timeout: 5000, // request timeout
9 })
10
11 service.interceptors.request.use(
12   config => {
13     if (store.getters.token) {
14       config.headers['X-Token'] = getToken()
15     }
16     return config
17   },
18   error => {
19     console.log(error) // for debug
20     return Promise.reject(error)
21   },
22 )
23
24 service.interceptors.response.use(
25   response => {
26     const res = response.data
27     if (res.code !== 20000) {
28       console.log('接口信息报错', res.message)
29       return Promise.reject(new Error(res.message || 'Error'))
30     } else {
31       return res
32     }
33   },
34   error => {
35     console.log('接口信息报错' + error)
36     return Promise.reject(error)
37   },
38 )
```

```
39 export default service
40
41
```

然后，我们在项目里集成 CSS 预编译器，CSS 预编译器可以帮我们更快、更高效地管理和编写 CSS 代码。在这里，我们选择 Sass 作为 CSS 预处理语言，然后我们就进入项目根目录下执行下面代码安装 Sass。

```
1 npm install -D sass
```

[📄 复制代码](#)

然后，我们进入 `src/components/Todolist.vue` 文件中。看下面的代码，我们直接在 `style` 标签上新增 `lang="scss"`，这样就可以使用 Sass 的语法了。有了 Sass 之后，我们在 CSS 里使用了变量、嵌套、继承等逻辑，并定义了 *padding* 和 *white* 这两个变量。这样我们就可以嵌套书写 CSS 选择器，也就极大地提高我们写 CSS 的效率。

```
1 <style lang="scss" scoped>
2 $padding:10px;
3 $white:#fff;
4 ul {
5   width:500px;
6   margin:0 auto;
7   padding: 0;
8   li {
9     &:hover {
10       cursor: pointer;
11     }
12     list-style-type: none;
13     margin-bottom: $padding;
14     padding: $padding;
15     background: $white;
16     box-shadow: 1px 3px 5px rgba(0, 0, 0, 0.1);
17   }
18 }
19 </style>
```

[📄 复制代码](#)

Sass 让我们在 CSS 的世界里也拥有了编程的概念，在实际项目中可以使用变量和函数等概念优化 CSS 代码，这个你在 Element3 组件库的实现中也能看到。

在 Element3 的 [GitHub 项目](#) 中，我们可以看到所有的 Sass 代码。以 common/var.scss 文件为例，在这个文件中，我们可以看到 Element3 中所有的变量，并且这个文件中的代码也对颜色、动画函数、边框，字体大小等等都做了统一的设置。

我们也可以修改这些变量，从而获得一个定制风格的 Element3。所以项目在开始之初，我们就可以想一下整体设计风格，最好能够预先定义好整体的颜色，边框，字体大小等等，这能极大降低后续的 css 维护成本。

至此，一个基于 Vite + Vue 3 + Vue Router + Vuex + Element3 + Axios + Sass 的前端项目开发环境搭建完毕。下面，我们来打磨一下这个项目。**简单来说，就是给项目增加代码规范约束、提交规范约束等，让其更完善、更健壮。**

## 代码规范和提交规范

由于个人习惯的不同，每个人写代码的风格也略有不同。比如在写 JavaScript 代码中，有些人习惯在每行代码之后都写分号，有些人习惯不写分号。但是团队产出的项目就需要有一致的风格，这样代码在团队之间阅读起来时，也会更加流畅。ESLint 就是专门用来做规范团队代码的一个库。

首先我们安装 ESLint，进入到项目文件夹，使用下面的命令，我们就可以在全局或者本地安装 ESLint 了。

```
1 npm i eslint -D
```

[复制代码](#)

ESLint 安装成功后，在项目根目录下执行 `npx eslint --init`，然后按照终端操作的提示完成一系列设置来创建配置文件。你可以按照下图所示的选择来始化 ESLint。



```
→ difu git:(master) x npx eslint --init
✓ How would you like to use ESLint? · problems
✓ What type of modules does your project use? · esm
✓ Which framework does your project use? · vue
✓ Does your project use TypeScript? · No / Yes
✓ Where does your code run? · browser
✓ What format do you want your config file to be in? · JSON
The config that you've selected requires the following dependencies:
eslint-plugin-vue@latest
✓ Would you like to install them now with npm? · No / Yes
Installing eslint-plugin-vue@latest
```

我们设置的是比较松散的校验规则，你可以根据团队风格去额外配置 ESLint 的插件。我们进入到项目目录下的 `eslinttrc.json` 中，在 `rules` 中新增下面代码，也就是强制要求 JavaScript 的行尾不写分号。

```
1  "rules": {
2    "semi": ["warn", "never"]
3  }
```

[复制代码](#)

然后，我们在命令行中执行 `npx eslint src`，接着你就会看到下图所示的报错信息，其中详细告诉你了哪里的代码不合规范。根据报错信息的提示，我们进入到 `router/index.js` 文件后，删掉第 15 行代码结束的分号就可以解除这个警告。

```
/Users/woniuppp/github/geektime-vue/geektime-vue-course/01-vue3-intro/src/router/index.js
15:39  warning  Extra semicolon  semi

* 1 problem (0 errors, 1 warning)
0 errors and 1 warning potentially fixable with the `--fix` option.
```

前面我们已经统一了代码规范，并且在提交代码时进行强约束来保证仓库代码的质量。多人协作的项目中，在提交代码这个环节，也存在一种情况：不能保证每个人对提交信息的准确描述，因此会出现提交信息紊乱、风格不一致的情况。

对于这种情况，一种比较好的解决方案是，在执行 `git commit` 命令的时候，同时执行 ESLint。我们使用 `husky` 管理 `git` 的钩子函数，在每次代码提交至 `git` 之前去执行 ESLint，只有 ESLint 的校验通过，`commit` 才能执行成功。后面的进阶开发篇中，单元测试也会放在 `git` 的钩子函数中执行，确保提交到 `git` 中的代码都是测试通过的。

项目代码符合规范后，我们就可以把代码提交到代码仓库中，git 允许我们在每次提交时，附带一个提交信息作为说明。我们在项目根目录执行下面的命令，提交了一个附带信息是 commit 的代码。

[复制代码](#)

```
1 git add .
2 git commit -m 'init commit'
```

然后我们需要再定义一下 git 的提交规范，描述信息精准的 git 提交日志，会让我们在后  
期维护和 处理 Bug 时有据可查。在项目开发周期内，我们还可以根据规范的提交信息，  
快速生成开发日志，从而方便我们追踪项目和把控进度。如下图所示，我们可以看到 Vue  
3 的代码提交日志。

yyx990803 workflow: separate unit and e2e tests <span>7c11c58 yesterday</span> <span>🕒 4,051 commits</span>		
📁 .github	ci: update checkout version (#4881)	14 days ago
📁 .vscode	workflow: cross platform vscode jest debugging (#414)	17 months ago
📁 packages	workflow: separate unit and e2e tests	yesterday
📁 scripts	workflow: separate unit and e2e tests	yesterday
📁 test-dts	fix(types/sfc): fix withDefaults type inference when using union types (	yesterday
📄 .eslintrc.js	feat(experimental): standalone ref transform	3 months ago
📄 .gitignore	chore: ignore .idea folder (#4838) [ci skip]	14 days ago
📄 .prettierrc	feat(types): map declared emits to onXXX props in inferred prop typ...	4 months ago
📄 CHANGELOG.md	release: v3.2.22	yesterday
📄 LICENSE	chore: license	2 years ago
📄 README.md	docs: update vite init command (#4176) [ci skip]	4 months ago
📄 SECURITY.md	chore: improve security.md [ci skip]	last month
📄 api-extractor.json	feat(types): adjust type exports for manual render function and tooli...	17 months ago
📄 jest.config.js	refactor: ensure ssr branches are included in esm-bundler build	2 months ago

看了 Vue 3 代码日志提交的格式，初次接触的你可能会觉得复杂。其实不然，Vue 3 在代  
码日志中，使用的是【类别: 信息】的格式，我们可以通过类别清晰地知道这次提交是代码  
修复，还是功能开发 feat。冒号后面的信息是用来解释此次提交的内容，在修复 bug 时，  
还会带上 issue 中的编号。在现在的项目开发中，我们也会强制要求使用和 Vue 3 一样的  
git 日志格式。

## 总结

今天这一讲的内容到这就结束了，我们来复习一下今天学到的内容。首先我们引入了 Element3 组件库，在项目入口注册 Element3 后，你可以在项目的任意地方直接使用 [🔗 Element3 首页的组件列表](#)。

这样，我们就可以很方便地使用 layou 和 container 布局实现页面的搭建，然后引入 axios 作为网络请求库，并且对接口统一做了全局拦截。下一讲中，项目权限管理也是在 axios 拦截函数里实现的。

当然，复杂的 Vue 项目更需要良好的规范，毕竟没有规矩不成方圆，为此，我们进一步规范了代码格式，使用 ESLint 统一 JavaScript 的代码风格，husky 管理 git 的钩子函数，并且规定了 git 的提交日志格式，确保代码的可维护性。

## 思考题

关于 Element3 组件库布局和导航组件的使用，你有什么其他布局的建议呢？

欢迎在留言区发表你的看法，也欢迎你把这一讲的内容推荐给你的同事和朋友们，我们下一讲再见。

分享给需要的人，Ta 订阅后你可得 **20 元现金** 奖励

 生成海报并分享

 赞 5     提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇    14 | TypeScript：Vue 3 中如何使用 TypeScript？

更多课程推荐

# 跟月影学可视化

## 系统掌握图形学与可视化核心原理

月影

奇虎 360 奇舞团团长

可视化 UI 框架 SpriteJS 核心开发者



新版升级：点击「 请朋友读」，20位好友免费读，邀请订阅更有**现金**奖励。

### 精选留言 (5)

[写留言](#)

轻度

2021-11-19

为了避免出现歧义等情况，不应该是强制javascript写分号吗

共 2 条评论 >

2



海阔天空

2021-11-19

项目前期的规范和基础库封装是非常重要的。统一的代码规范和提交规范，为项目的后期扩展和维护项目带来便利。在老师的课程中又学到了。

- 1、简单来说，就是给项目增加代码规范约束、提交规范约束等，让其更完善、更健壮。
- 2、git 的提交规范，描述信息精准的 git 提交日志，会让我们在后期维护和 处理 Bug 时有据可查。

展开 ∨



1



魔魔

2021-11-19

提纲挈领，总结加复习，跟着打一遍代码，然后再默写一遍。受益匪浅发，赞！



椰\_\_季

2021-11-19

老师，能发一下从实战到这节课的源代码么，我看自己的代码，缺胳膊少腿。配置了eslint好多创建了没有使用的一些

展开



peterpc

2021-11-19

大圣，husky如何管理git的钩子函数？一笔带过？

展开

