

项目构建

10.1 准备项目开发环境

本项目的开发环境以及安装方法如下：

1. 安装 WebStorm 或 Visual Studio Code 平台

本项目的开发平台可以选用 WebStorm 或 Visual Studio Code 平台，WebStorm 可以从其官网（<https://www.jetbrains.com/webstorm/>）中下载并安装。本项目开发使用的 WebStorm 平台 Version 信息如图所示。



图 10-1 本项目开发使用的 WebStorm 平台 Version 信息

2. 安装 Node.js

由于安装开发 Vue 项目必要的 webpack 工具包首先需要安装 Node.js，因此从其官网（<https://nodejs.org/en/>）下载并安装 Node.js，本项目需要的安装包可在本项目提供的安装文件夹查看。Node.js 自带软件包管理工具 npm（Node Package Manager），便于后期项目的开发和设计，可以使用 npm 安装开发过程中的依赖包。Node.js 安装成功之后，在 WebStorm 的终端应用以下命令查看 Node.js 的版本信息，如图所示。若未正常安装 Node.js，则无法显示下图版本信息。

```
node -v
```

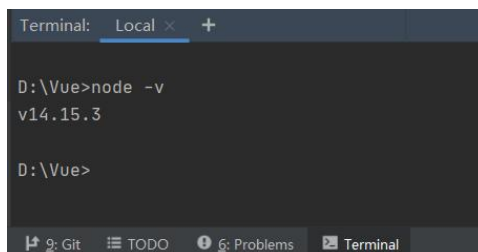


图 10-2 本项目安装的 Node.js 版本信息

由于直接使用 npm 的官方镜像安装依赖包一般较慢，因此可以使用淘宝 npm 镜像代替默认的 npm，在 WebStorm 的终端或 Windows 命令行中应用以下命令，全局化安装 cnpm 命令如

下:

```
npm install -g cnpm --registry=https://registry.npm.taobao.org
```

或者

```
npm config set registry https://registry.npm.taobao.org
```

全局化安装 cnpm 之后就可以使用 cnpm 命令安装各类模块,如下代码中[name]表示模块名。

```
cnpm install [name]
```

3. 安装 webpack

本项目借助于工具 webpack 进行模块化开发,处理模块间的各种依赖关系,处理模块化的代码,并将各种资源模块进行打包合并成一个或多个包(Bundle)。使用 webpack 处理模块之间的关系后,将多个 js 打包到一个 js 文件后,只引用一个 js 文件就非常方便。在打包的过程中,webpack 可以对资源进行处理,如压缩图片、将 TypeScript 转成 JavaScript 等操作。

在 WebStorm 的终端或 Windows 命令行中应用以下命令全局安装 webpack,其中参数-g 是指全局安装。

```
npm install -g webpack
```

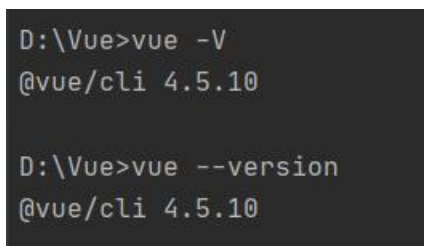
4. 安装 Vue CLI

本项目使用 Vue.js 开发设计,需要考虑代码目录结构、项目结构和部署、代码单元测试等,因此使用 Vue 推出的可以进行工程化管理的工具 Vue CLI (Command-Line Interface, 命令行界面, 俗称脚手架),应用 Vue CLI 可以快速搭建 Vue 开发环境以及对应的 webpack 配置,快速构建一个项目的基本的雏形结构。Vue.js 官方脚手架工具使用 webpack 模板,对资源进行压缩等优化操作,提高项目开发效率。

使用如下命令全局安装 Vue CLI:

```
cnpm install -g @vue/cli
```

成功安装 Vue CLI 之后,使用命令 `vue -V` 或者 `vue --version` 查看 Vue CLI 的版本信息,如图所示。



```
D:\Vue>vue -V
@vue/cli 4.5.10

D:\Vue>vue --version
@vue/cli 4.5.10
```

图 10-3 查看 Vue CLI 的版本信息

10.2 项目创建

10.2.1 项目创建

1. 创建项目

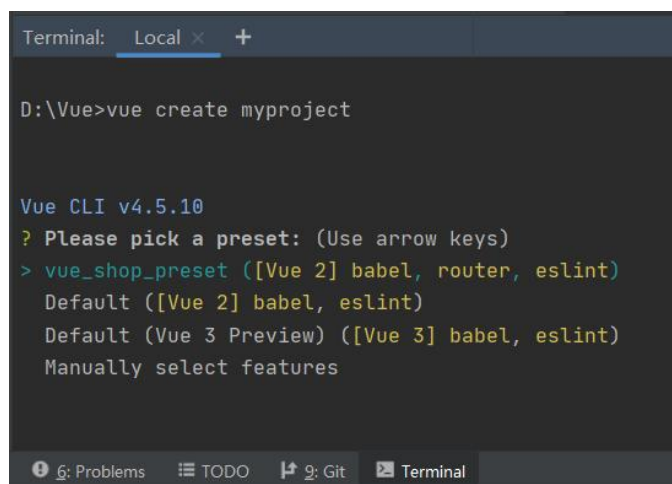
在 WebStorm 的终端利用 cd 指令进入需要保存项目的文件夹下,输入以下命令创建项目,其中“myproject”可以改为需要创建的项目名,注意项目名称不能使用中文,且不能使用大写字母。

```
vue create myproject
```

2. 选择项目的配置方式

选择项目的配置方式如图所示,本文选择自定义配置项目(Manually select features),

使用键盘的上下按键选择配置项目，确定选项后按下回车键即可进入下一步。



```
Terminal: Local x +  
  
D:\Vue>vue create myproject  
  
Vue CLI v4.5.10  
? Please pick a preset: (Use arrow keys)  
> vue_shop_preset ([Vue 2] babel, router, eslint)  
  Default ([Vue 2] babel, eslint)  
  Default (Vue 3 Preview) ([Vue 3] babel, eslint)  
  Manually select features  
  
6: Problems  TODO  Git  Terminal
```

图 10-4 终端创建 Vue 项目

3. 手动选择项目需要的特性

设置项目名称、项目描述、作者、打包方式、是否使用 ESLint 规范代码等，手动选择项目需要选择的特性如图所示，每一项的配置说明如下。选择需要安装的插件，使用键盘的上下按键选择配置项目，按空格键切换是否选中（*为选中状态），本项目选择 Choose Vue version、Babel、Router、Vuex 四个选项，按下回车键后转到下一步。

Babel: 帮助解析 ES6 代码（ECMAScript 6），对于一些低版本浏览器不能识别 ES6 代码，该插件将 ES6 代码适配成低版本浏览器能够识别的代码（必须安装）；

TypeScript: TypeScript 是 JavaScript 的一个超集，支持 ECMAScript 6 标准；

Progressive Web App (PWA) Support: 渐进式 Web 应用，专门应对手机 web 开发；

Router: Vue 路由，在项目中一般都需要使用 Vue 路由，但是本项目以从零创建项目为例，暂时不选择路由选项，需要路由时再进行创建；

Vuex: Vue 状态管理，本项暂不选择，后期需要状态管理时再进行创建；

CSS Pre-processors: CSS 预编译器（包括：SCSS/Sass、Less、Stylus），本项可以生成项目后根据个人 CSS 编写习惯选择安装；

Linters / Formatters: 代码规范标准（刚入门学习情况不建议安装）；

Unit Testing: 单元测试；

E2E Testing: 端到端测试。

```
Terminal: Local x +
D:\Vue>cd Vuejs

D:\Vue\Vuejs>vue create myproject

Vue CLI v4.5.10
? Please pick a preset: Manually select features
? Check the features needed for your project:
  (*) Choose Vue version
  (*) Babel
  ( ) TypeScript
  ( ) Progressive Web App (PWA) Support
  ( ) Router
  ( ) Vuex
  ( ) CSS Pre-processors
  ( ) Linter / Formatter
  ( ) Unit Testing
> ( ) E2E Testing
```

图 10-5 选择项目需要的特性

4.选择 Vue 项目的版本，本文选择是 2.x 版本，如图所示。

```
D:\Vue\Vuejs>vue create myproject

Vue CLI v4.5.10
? Please pick a preset: Manually select features
? Check the features needed for your project: Choose Vue version, Babel
? Choose a version of Vue.js that you want to start the project with (Use arrow keys)
> 2.x
  3.x (Preview)
```

图 10-6 选择 Vue 项目的版本

5.配置保存位置

选择配置文件是单独放在一个 package.json 中还是分开存放，这里选择第一项分开存放 (In dedicated config files)，按下回车键进入下一步，如图所示。说明：

In dedicated config files: 单独保存在各自的配置文件中

In package.json: 保存在 package.json 文件中

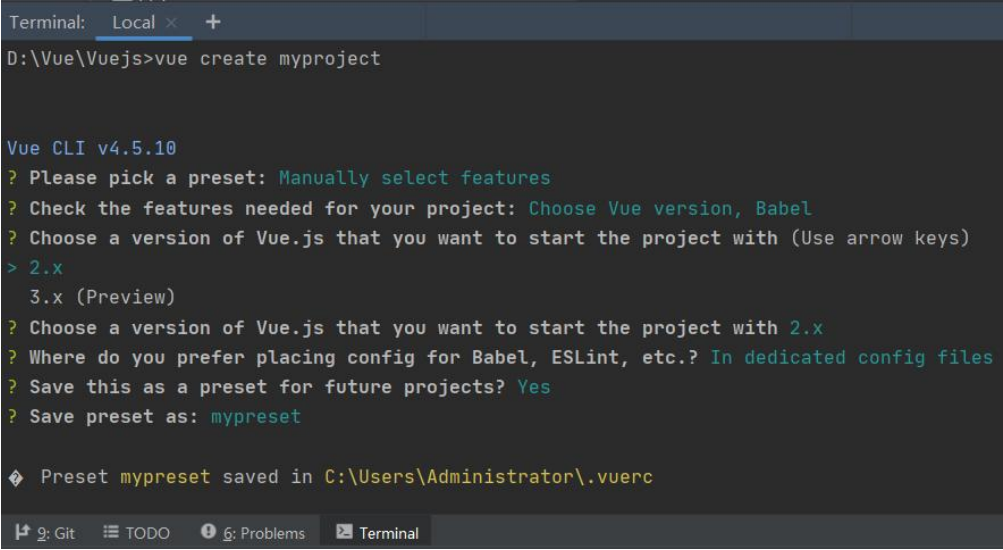
```
Terminal: Local x +

Vue CLI v4.5.10
? Please pick a preset: Manually select features
? Check the features needed for your project: Choose Vue version, Babel
? Choose a version of Vue.js that you want to start the project with (Use arrow keys)
> 2.x
  3.x (Preview)
? Choose a version of Vue.js that you want to start the project with 2.x
? Where do you prefer placing config for Babel, ESLint, etc.? (Use arrow keys)
> In dedicated config files
  In package.json
```

图 10-7 配置保存位置

6.是否需要保存预设并应用到后续的项目

一般创建项目可以不用保存预设，直接选择 no，按下回车键后等待生成项目；使用 Yes 保存预设是为了方便下次创建相同需求的配置过程项目。若选择 Yes，用英文自定义过程预设命名，例如“mypreset”，如图所示。若设置保存预设，则系统自动生成一个保存在 C 盘的.vuerc 文件，后面创建配置相同的项目时，即可直接选该方式创建。



```
Terminal: Local x +
D:\Vue\Vuejs>vue create myproject

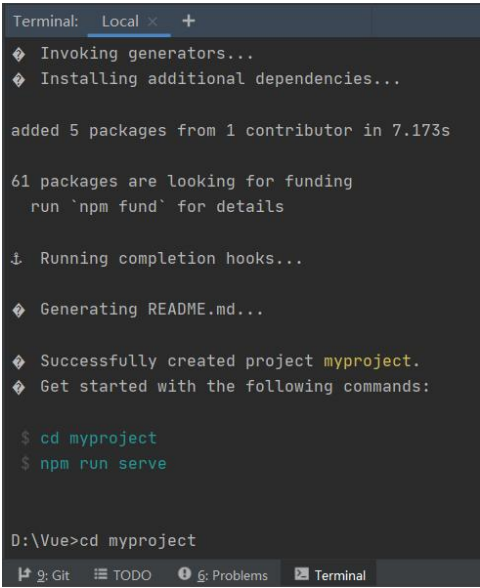
Vue CLI v4.5.10
? Please pick a preset: Manually select features
? Check the features needed for your project: Choose Vue version, Babel
? Choose a version of Vue.js that you want to start the project with (Use arrow keys)
> 2.x
  3.x (Preview)
? Choose a version of Vue.js that you want to start the project with 2.x
? Where do you prefer placing config for Babel, ESLint, etc.? In dedicated config files
? Save this as a preset for future projects? Yes
? Save preset as: mypreset

💎 Preset mypreset saved in C:\Users\Administrator\.vuerc
```

图 10-8 保存预设配置

7.项目生成

项目将生成在 cmd 设置的当前路径目录下，如图所示。



```
Terminal: Local x +
💎 Invoking generators...
💎 Installing additional dependencies...

added 5 packages from 1 contributor in 7.173s

61 packages are looking for funding
  run `npm fund` for details

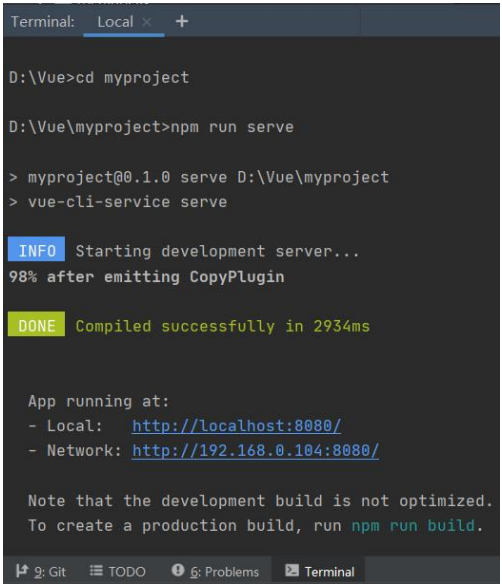
↓ Running completion hooks...

💎 Generating README.md...
💎 Successfully created project myproject.
💎 Get started with the following commands:

$ cd myproject
$ npm run serve

D:\Vue>cd myproject
```

图 10-9 项目生成



```
Terminal: Local x +
D:\Vue>cd myproject

D:\Vue\myproject>npm run serve

> myproject@0.1.0 serve D:\Vue\myproject
> vue-cli-service serve

INFO Starting development server...
98% after emitting CopyPlugin

DONE Compiled successfully in 2934ms

App running at:
- Local: http://localhost:8080/
- Network: http://192.168.0.104:8080/

Note that the development build is not optimized.
To create a production build, run npm run build.
```

图 10-10 启动项目 myproject

应用 `cd myproject` 命令进入项目 myproject 目录，输入命令：`npm run serve` 启动 vue 项目，如图所示，则 myproject 项目创建成功。

在浏览器中输入地址 `http://localhost:8080/`，或者单击该地址在浏览器中预览项目运行效果，如图所示。

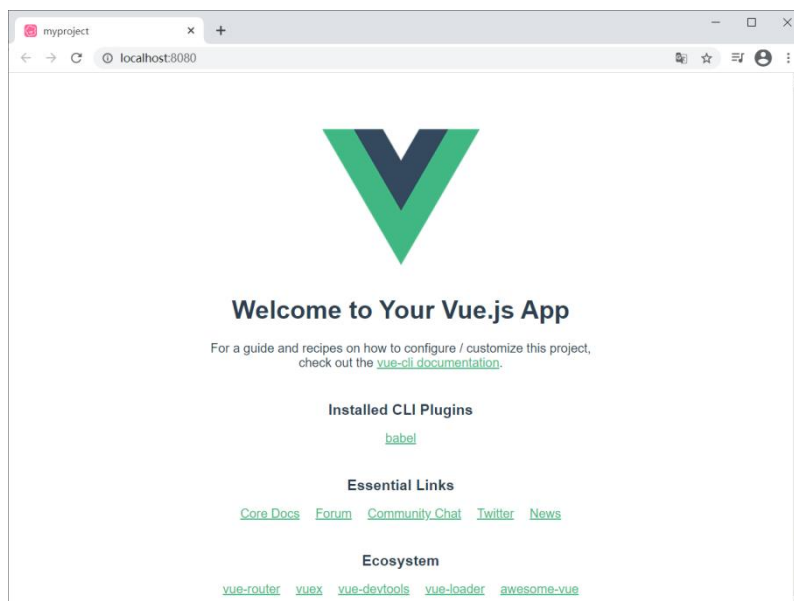


图 10-11 在浏览器测试生成的项目 myproject

10.2.2 安装 Element UI

Element UI 是基于 vue 实现的一套不依赖业务的 UI 组件库，提供了丰富的 PC 端组件，减少用户对常用组件的封装，降低了开发的难易程度。

Vue 与 Element Ui 的关系如下：

(1) Element Ui 是基于 Vue 封装的组件库，简化了常用组件的封装，提高了重用性原则；

(2) Vue 是一个渐进式框架，Element Ui 是组件库。

Element-ui 官网地址 <https://element.eleme.cn/#/zh-CN/component/installation>。

1. 安装 Element-ui

使用 npm 安装 Element-ui 依赖包，代码如下，安装成功后可以在项目的 node_modules 路径下找到已安装的依赖包。

```
npm i element-ui -S
```

2. 引入 Element

可以引入整个 Element，或是根据需要仅引入部分组件。我们先介绍如何引入完整的 Element。

引入依赖的方式有多种，在 src→main.js 中引入的插件、组件和样式都是全局的；若你页面中的 script 中引入则只作用于当前的页面。若整个项目所有页面都需要使用到该插件，则在 main.js 中导入并注册需要使用的组件，代码如下。

```
import ElementUi from 'element-ui'
import 'element-ui/lib/theme-chalk/index.css'
Vue.use(ElementUi)
```

以上代码便完成了 Element 的引入。需要注意的是，样式文件需要单独引入。当然也可以按需导入，具体参考 Element 官网。

10.2.3 项目 GitHub 托管

为了方便管理，可以将项目托管到 Github (<https://github.com/>) 或者 Gitee

(<https://gitee.com/>) 中托管，本文以将项目托管到 Github 为例。

在 Github 中注册用户并登录后，选择新建 repository 后，在图中 Repository name 处输入项目名称，如 “myproject”，其他保持默认选项，单击 “Create repository” 创建托管项目，如图所示。

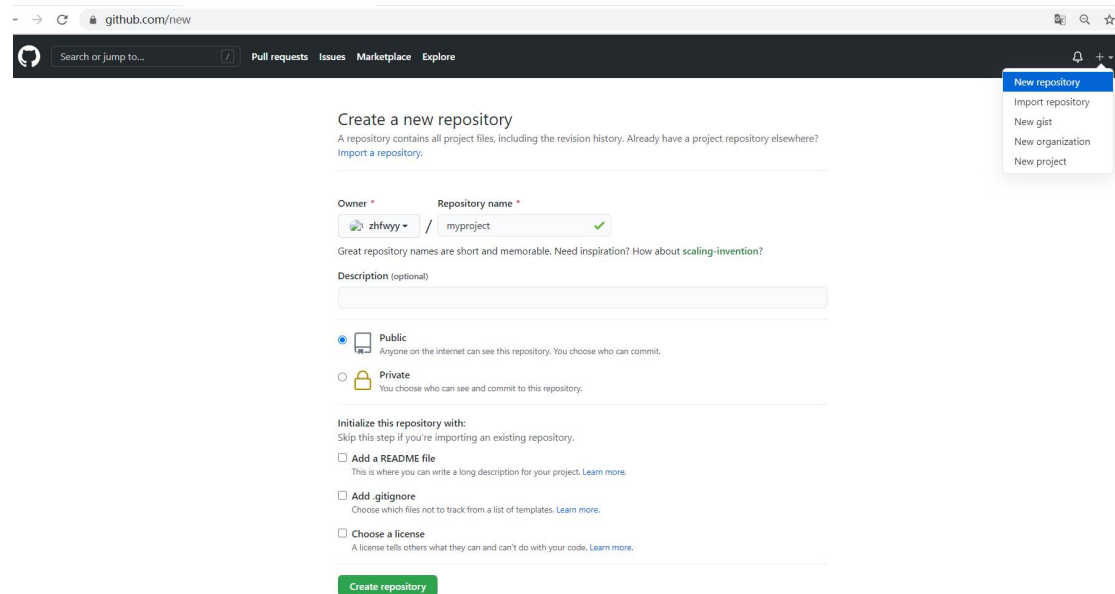


图 10-12 创建托管项目 myproject

进入项目 myproject，在 WebStorm 项目路径下的终端，分别执行图中选中的两条命令将本地项目 push 到 Github，如图所示。

```
git remote add origin https://github.com/zhfwyy/myproject.git
git push -u origin master
```

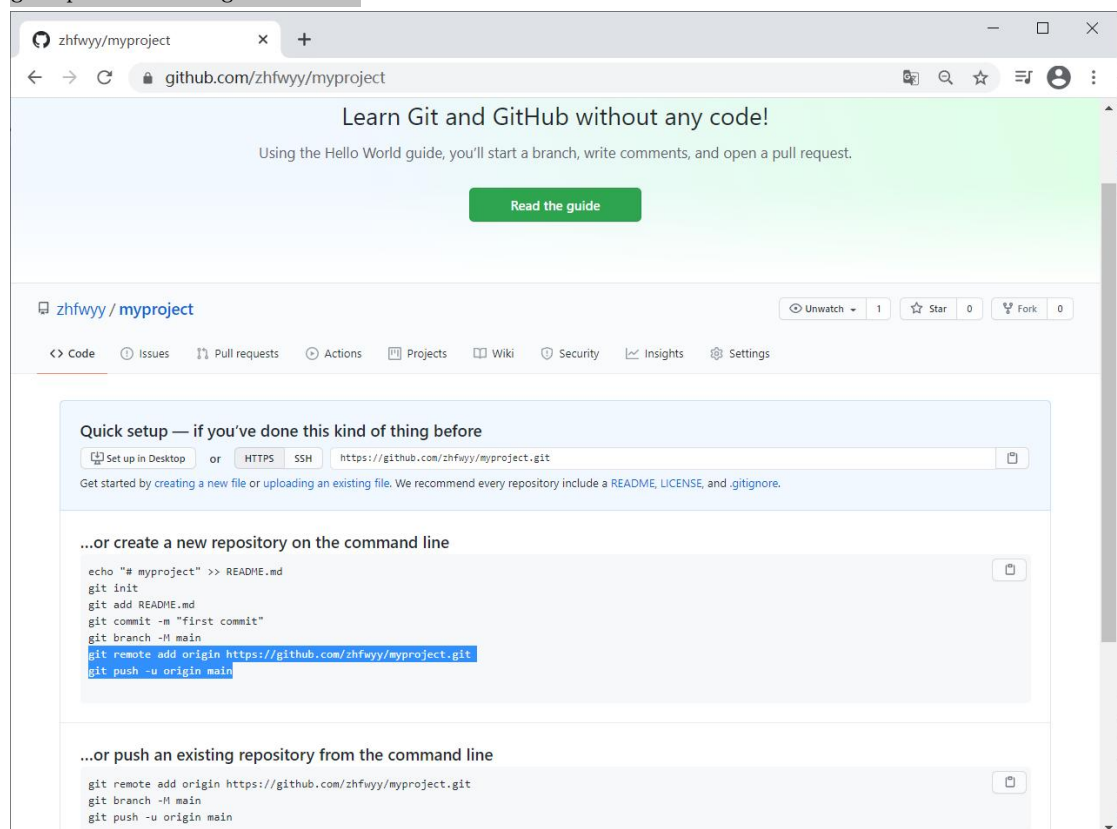


图 10-13 进入托管项目 myproject

在执行第二条语句 `git push -u origin master` 之后，将提示在浏览器之后授权 Github 代码托管上传，如图所示；同时输入注册 Github 用户的密码验证，如图所示。

```
D:\Vue\myproject>git push -u origin master
info: please complete authentication in your browser...
```

图 10-14 执行 git push 界面

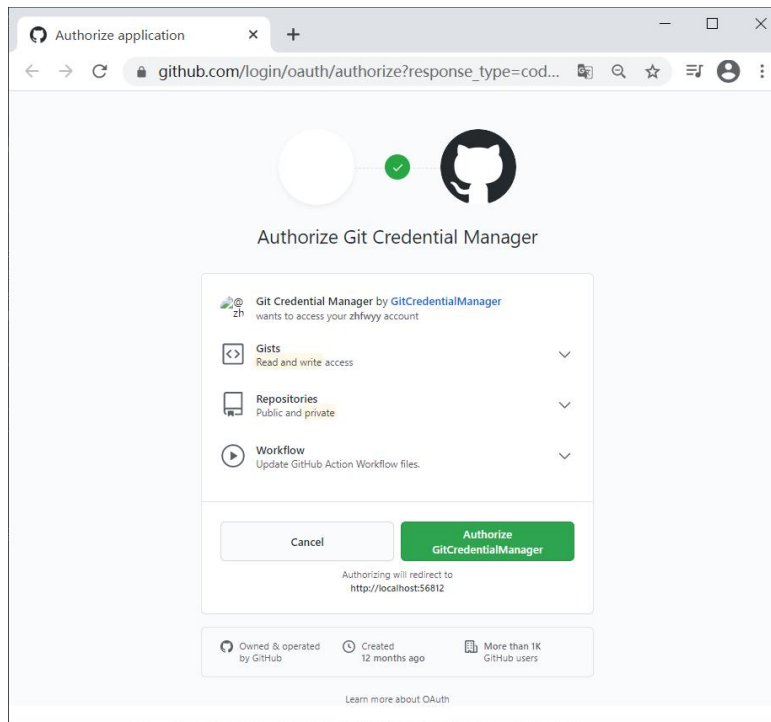


图 10-15 Github 授权代码托管页面

授权登录成功之后，终端的显示如图所示，同时刷新浏览器的 Github 页面，本地的项目已经成功托管到 Github，如图所示。

```
Terminal: Local x +
D:\Vue\myproject>git push -u origin master
info: please complete authentication in your browser...
Enumerating objects: 25, done.
Counting objects: 100% (25/25), done.
Delta compression using up to 8 threads
Compressing objects: 100% (20/20), done.
Writing objects: 100% (25/25), 86.13 KiB | 5.74 MiB/s, done.
Total 25 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com:zhfwy/myproject.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.

D:\Vue\myproject>
D:\Vue\myproject>
D:\Vue\myproject>
```

图 10-16 授权登录之后终端显示

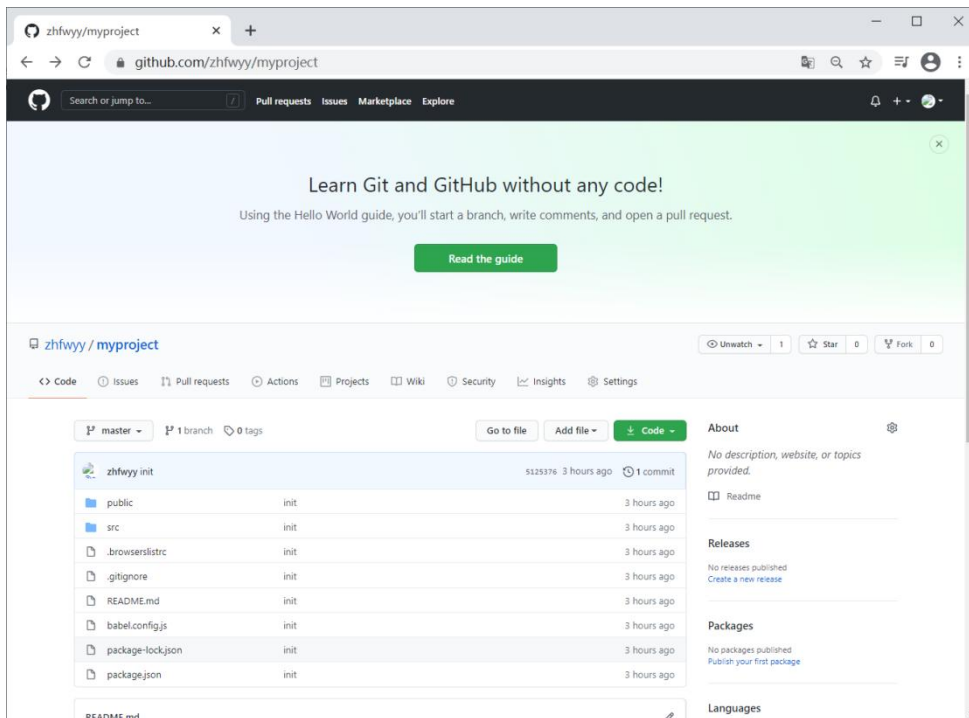


图 10-17 成功将本地项目托管到 Github

10.3 划分目录结构

10.3.1 项目文件说明

项目成功创建之后，目录结构如图所示，各个项目文件说明如下。

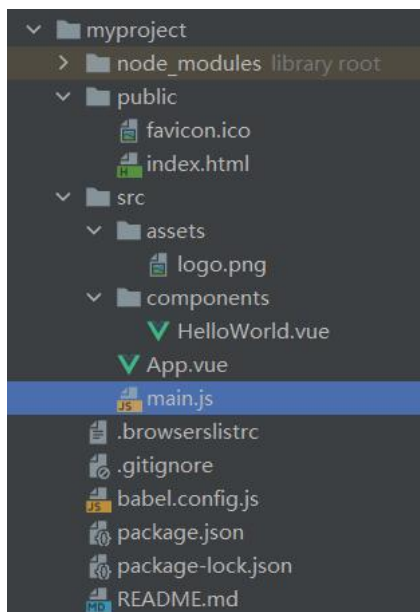


图 10-18 myproject 项目创建初始目录结构

1. **node_modules**: 用于存放使用 **npm** 命令下载项目开发环境和生成环境的各种依赖包，其中包括很多基础依赖，也可以根据需要安装其他依赖；

安装方法为打开终端，进入项目目录，输入以下命令：**npm install [依赖包名称]**，按下回车键即可。一般在两种情况下，需要自行安装依赖：

(1) 项目运行缺少某种依赖包：例如项目加载外部 css 会用到的 `css-loader`，路由跳转 `vue-loader` 等（安装方法示例：`npm install css-loader`）；

(2) 安装插件：如 `vux`（基于 WEUI 的移动端组件库），`vue-swiper`（轮播插件）等。

注意：若需要安装指定依赖版本，需在依赖包名称后加上版本号信息，如安装 11.1.4 版本的 `vue-loader`，输入 `npm install vue-loader@11.1.4`。

2. `public`：用于存放公共静态资源，其中的文件不会被 `webpack` 处理，而是会直接被复制到最终的打包目录（默认是 `dist` 目录）下。`dist` 文件夹存放应用 `npm run build` 命令打包生成的静态资源文件，用于生产部署。

3. `public→index.html`：是一个模板文件，作用是生成项目的入口文件。

4. `public→favicon.ico`：可以通过替换 `favicon.ico` 图片文件，从而替换项目运行后浏览器上方的图标。

5. `src`：存放各种 `vue` 文件，是编写开发项目源码的主要位置。

6. `src→assets`：用于存在着各种静态文件，例如图片、图标、字体，该文件夹与 `public` 的区别是，`assets` 目录中的文件会被 `webpack` 处理解析为模块依赖。在实际的开发中，总的来说 `public` 存放不会变动的文件，`assets` 存放可能会变动的文件。

7. `src→components`：用于存放公共组件，如 `header`、`footer` 等可以复用的小组件。

8. `src→App.vue`：最主要的 `vue` 模块，主要是使用 `router-link` 引入其他操作，`App.vue` 是项目的主组件，是页面的入口文件，所有页面在 `App.vue` 下切换，是路由组件的顶层路由。

9. `src→main.js`：是 `vue-cli` 工程的入口文件，主要作用是初始化 `vue` 实例，加载各种公共组件，同时可以在此文件中引入全局组件库或者全局挂载一些变量。

10. `.browserslistrc`：用于设置浏览器的兼容，例如，部分配置参数：“>1%”表示全球超过 1% 人使用的浏览器；“last 2 versions”表示所有浏览器兼容到最后两个版本。

11. `.gitignore`：配置 `git` 上传时需要忽略的文件格式设置。

12. `babel.config.js`：是一个工具链，主要用于在当前和较旧的浏览器或者环境中将 ES6 的代码转换向后兼容（低版本 ES）。

13. `package.json`：用于 `node-modules` 资源部署和启动、打包项目的 `npm` 命令管理，存放项目开发需要的模块版本、项目名称。

14. `package-lock.json`：是执行命令 `npm install` 时生成的文件，用于记录当前状态下实际安装的各个 `npm package` 的具体来源和版本号。

15. `README.md`：项目说明文件，`markdown` 格式。

10.3.2 优化目录结构

1. 为了优化目录结构，首先删除默认添加的、与本项目无关的文件和代码。

删除项目 `myproject` 中 `src→components` 目录下的 `HelloWorld.vue` 文件，删除 `src→assets` 目录下的 `logo.png` 文件，同时修改入口主文件 `App.vue`，删除创建项目时默认添加的代码如下，便于后期开发项目，执行命令 `npm run serve` 启动 `myproject` 项目，如图所示，此时浏览器显示为空。

```
<template>
  <div id="app">
  </div>
</template>
<script>
  export default {
    name: 'app',
```

```
components:{  
  }  
}  
</script>  
<style>  
</style>
```

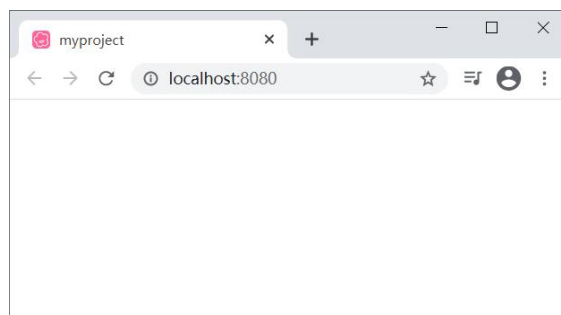


图 10-19 删除默认代码的项目运行效果

2. 根据项目开发需要优化项目的目录结构，优化后的项目文件目录。

(1) 在 `src→assets` 目录下分别新建两个存放样式和图片资源文件的目录 `css` 和 `img`。

(2) 在 `src` 目录下新建 `views` 目录，用于存放页面级、大的组件，也是路由对应的文件，例如 `Home` 组件。`views` 和 `components` 文件夹都用于存放 `Vue` 组件，一般 `components` 中存放从 `views` 中抽离的可以在多个项目或多个页面使用的公共小组件，例如 `Tabbar`、`Navbar` 组件。

(3) 在 `src→components` 目录下分别新建两个目录 `common` 和 `content`，`common` 文件夹一般用于存放可以同时用于本项目以及其他项目的公共组件；`content` 文件夹一般用于存放与本项目业务相关的组件。

(4) 在 `src` 目录下新建 `router` 目录，用于管理和配置路由。

(5) 在 `src` 目录下新建 `store` 目录，用于存放 `vue` 中的状态数据，使用 `vuex` 集中管理，是 `vue.js` 应用项目开发的状态管理器，主要用于维护 `vue` 组件间公用的一些变量和方法。

(6) 在 `src` 目录下新建 `network` 目录，用于存放与项目网络相关的封装。

(7) 在 `src` 目录下新建 `common` 目录，用于存放抽取的公共的 `js` 文件，例如公共常量等。

优化后的项目文件目录如图所示。

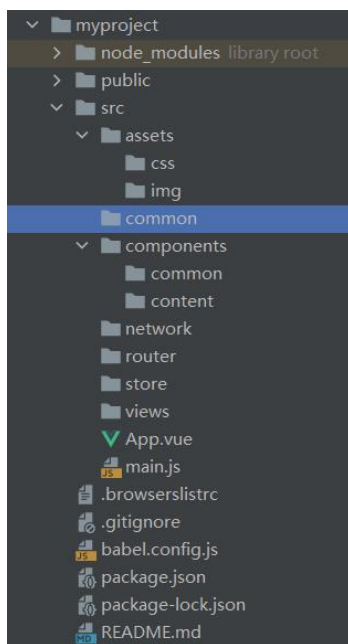


图 10-20 优化后的项目目录结构

3. 引入跨浏览器统一网页元素的样式文件 Normalize.css

不同浏览器在对于 CSS 没有定义的一些样式属性是不同的，例如，若未在 CSS 中设置超链接是否有下划线时，有些浏览器显示有下划线，而有些浏览器没有下划线；有一些浏览器规定的超链接默认颜色是蓝色，另外一些浏览器设置是黑色。

样式文件 Normalize.css 的功能是对默认样式进行重置，能够使所有浏览器对于未定义的样式浏览效果达到一致。Normalize.css 是一个可以定制的 CSS 文件，使不同浏览器在渲染网页元素的时候形式更统一，在 HTML 元素样式上提供了跨浏览器的高度一致性。

Normalize.css 的项目地址：<http://necolas.github.io/normalize.css/>

Normalize.css 在 GitHub 上的源码地址：<https://github.com/necolas/normalize.css>

将样式文件 Normalize.css 下载并保存至目录 src→assets→css 目录下。

4. 新建项目的 base.css 样式文件

新建项目的 base.css 样式文件用于设置本项目相关的通用样式设计，具体的样式文件代码见项目附件。在 App.vue 文件的<style></style>部分添加对 base.css 文件的引入，代码如下：

```
<style>
  @import "../assets/css/base.css";
</style>
```

10.3.3 配置路径别名 alias

在 Vue 项目开发中，经常需要引入不同文件目录的组件，通常是通过“import 组件名 from ‘组件路径’”的结构来实现对组件的引用，而当文件路径较深或者引用的组件跨越的较远时很容易引用出错，这里引入 alias（别名）的概念。因此，为了方便引用需要配置路径别名。

在引入模块的时候经常会用到@符号，用@符号指代根目录下的 src 文件夹路径，@+ 可以获取到 src 文件夹下的文件，这是 webpack 的默认配置。

vue-cli3 是零配置环境，因此手动配置 webpack 需要在项目 myproject 的根目录新建一个 vue.config.js 文件，这是个可选文件，项目创建时默认是没有的，不过@vue/cli-service 会自动识别加载。在 vue.config.js 文件中添加如下代码进行路径别名配置，这样下次再引用的

时候不需要寻找路径，直接使用别名即可。

```
module.exports = {
  configureWebpack: {
    resolve: {
      alias: {
        'views': '@/views',
        'components': '@/components',
        'network': '@/network',
        'common': '@/common',
        'assets': '@/assets',
      }
    }
  }
}
```

注意：由于路由 `router` 与 `store` 目录只需要使用 `this.$router` 或者 `this.$store` 即可获取其路径，因此不需要对 `router` 与 `store` 目录设置别名。

设置别名后，2.3.2 节在 `App.vue` 文件的 `<style></style>` 部分添加对 `base.css` 文件引入的代码可以删除 `assets` 前面的路径表示，修改后的代码如下：

```
<style>
  @import "assets/css/base.css";
</style>
```

在 DOM 中使用别名时的别名引用代码如下：

```

```

10.3.4 设置定义代码格式

为了规范团队开发项目的代码格式风格统一问题，设置定义代码格式 `.editorconfig`。在项目的根目录新建一个 `.editorconfig` 文件，代码如下：

```
root = true

[*]
charset = utf-8
indent_style = space
indent_size = 2
end_of_line = lf
insert_final_newline = true
trim_trailing_whitespace = true
```