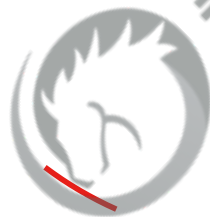




黑马程序员™
www.itheima.com

传智播客旗下
高端IT教育品牌

VUE 基础 - 综合案例



录 Contents

◆ vue-cli

◆ 组件库

◆ axios 拦截器

◆ proxy 跨域代理

◆ 用户列表案例

1. 什么是 vue-cli

vue-cli（俗称：vue 脚手架）是 vue 官方提供的、快速生成 vue 工程化项目的工具。

特点：

- ① 开箱即用
- ② 基于 webpack
- ③ 功能丰富且易于扩展
- ④ 支持创建 vue2 和 vue3 的项目

vue-cli 的中文官网首页：<https://cli.vuejs.org/zh/>

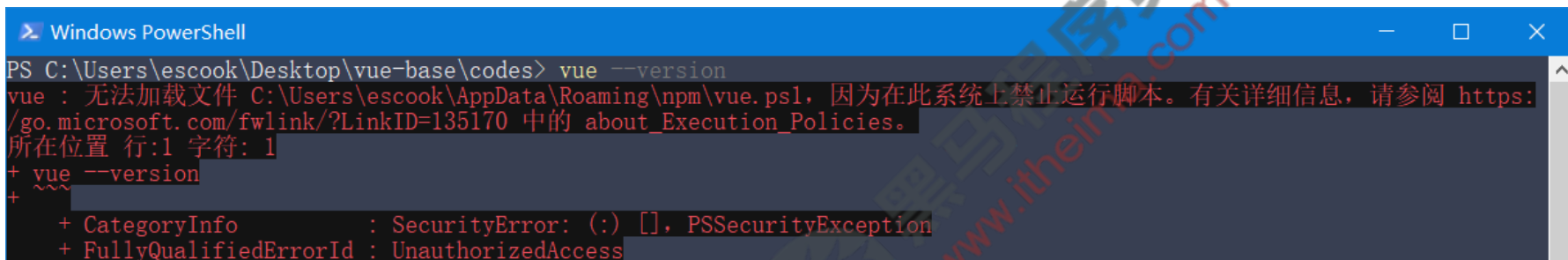
2. 安装 vue-cli

vue-cli 是基于 Node.js 开发出来的工具，因此需要使用 **npm** 将它安装为**全局可用的工具**：

```
1 # 全局安装 vue-cli
2 npm install -g @vue/cli
3
4 # 查看 vue-cli 的版本，检验 vue-cli 是否安装成功
5 vue --version
```

2.1 解决 Windows PowerShell 不识别 vue 命令的问题

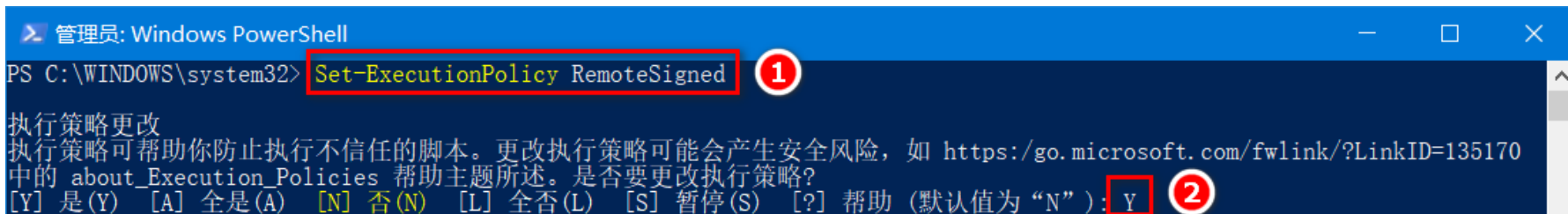
默认情况下，在PowerShell 中执行 `vue --version` 命令会提示如下的错误消息：



```
Windows PowerShell
PS C:\Users\escook\Desktop\vue-base\codes> vue --version
vue : 无法加载文件 C:\Users\escook\AppData\Roaming\npm\vue.ps1，因为在此系统上禁止运行脚本。有关详细信息，请参阅 https://go.microsoft.com/fwlink/?LinkID=135170 中的 about_Execution_Policies。
所在位置 行:1 字符: 1
+ vue --version
+ ~~~
+ CategoryInfo          : SecurityError: (:) [], PSSecurityException
+ FullyQualifiedErrorId : UnauthorizedAccess
```

解决方案如下：

- ① 以**管理员身份**运行 PowerShell
- ② 执行 `set-ExecutionPolicy RemoteSigned` 命令
- ③ 输入字符 **Y**，**回车**即可



```
管理员: Windows PowerShell
PS C:\WINDOWS\system32> Set-ExecutionPolicy RemoteSigned
执行策略更改
执行策略可帮助你防止执行不信任的脚本。更改执行策略可能会产生安全风险，如 https://go.microsoft.com/fwlink/?LinkID=135170
中的 about_Execution_Policies 帮助主题所述。是否要更改执行策略？
[Y] 是(Y) [A] 全是(A) [N] 否(N) [L] 全否(L) [S] 暂停(S) [?] 帮助 (默认值为“N”)： Y
```

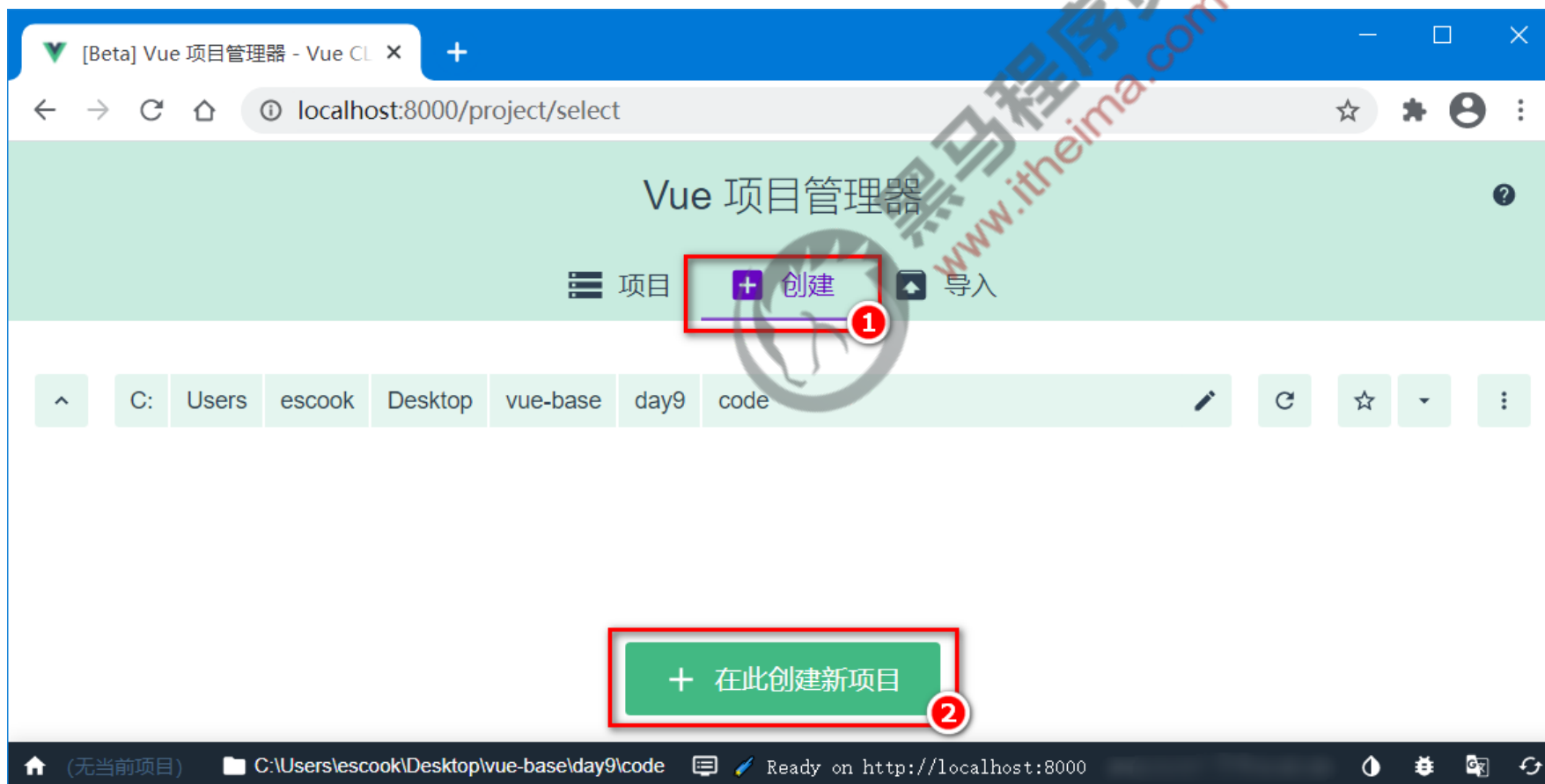
3. 创建项目

vue-cli 提供了创建项目的两种方式：

```
1 # 基于【命令行】的方式创建 vue 项目
2 vue create 项目名称
3
4 # OR
5
6 # 基于【可视化面板】创建 vue 项目
7 vue ui
```

4. 基于 vue ui 创建 vue 项目

步骤1：在终端下运行 **vue ui** 命令，自动在浏览器中打开**创建项目的可视化面板**：



4. 基于 vue ui 创建 vue 项目

步骤2：在详情页面填写项目名称：



创建新项目

详情 预设 功能 配置

项目文件夹

vue-ui-project-1 1 填写项目名称

C:/.../vue-base/day9/code/vue-ui-project-1

包管理器

默认

更多选项

若目标文件夹已存在则将其覆盖

无新手指引的脚手架项目

Git

初始化 git 仓库 (建议)

覆盖提交信息 (选填)

取消 下一步 2

4. 基于 vue ui 创建 vue 项目

步骤3：在预设页面选择手动配置项目：



4. 基于 vue ui 创建 vue 项目

步骤4：在功能页面勾选需要安装的功能（Choose Vue Version、Babel、CSS 预处理器、使用配置文件）：



4. 基于 vue ui 创建 vue 项目

步骤5：在配置页面勾选 vue 的版本和需要的预处理器：



4. 基于 vue ui 创建 vue 项目

步骤6：将刚才所有的配置**保存为预设**（模板），方便下一次创建项目时**直接复用之前的配置**：



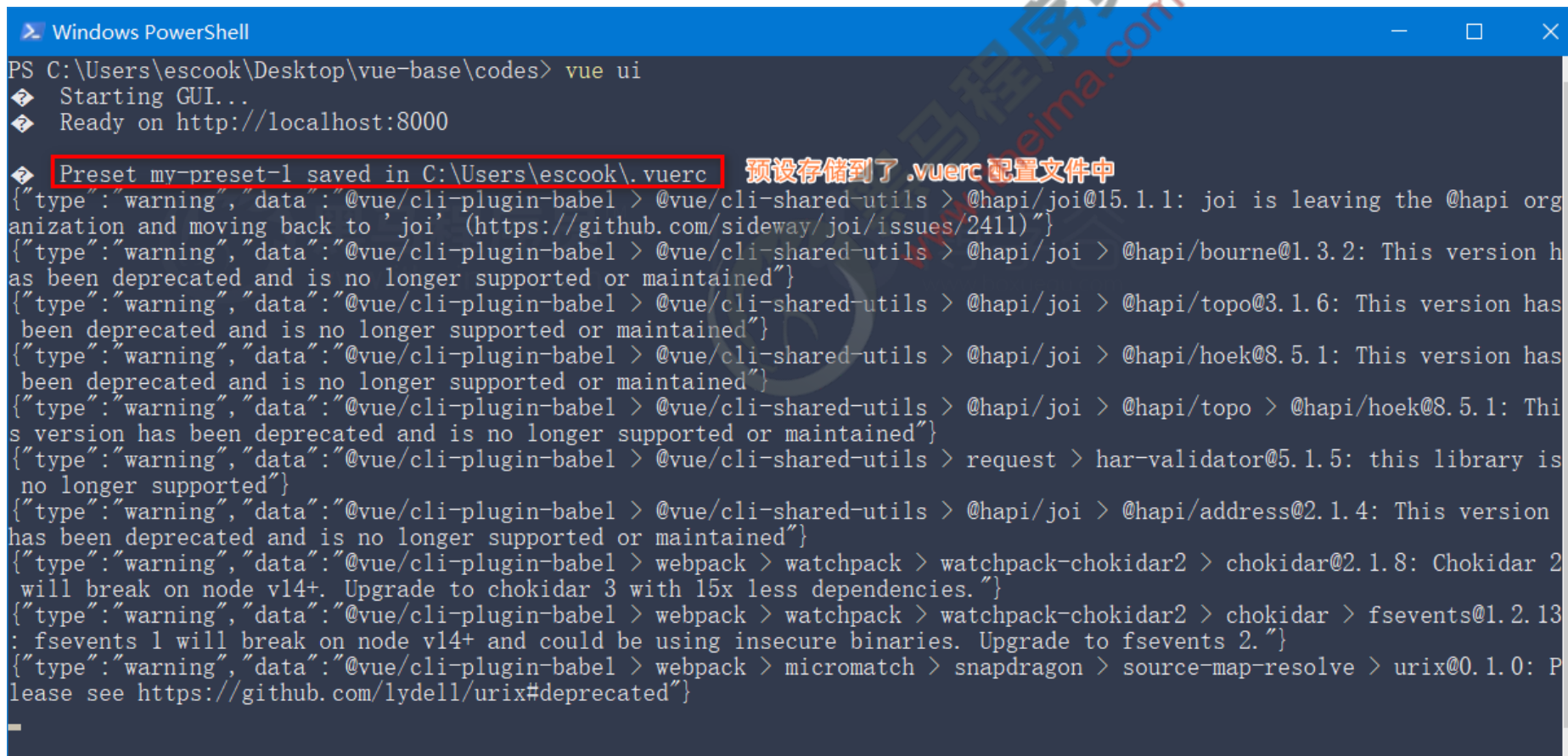
4. 基于 vue ui 创建 vue 项目

步骤7：创建项目并自动安装依赖包：



4. 基于 vue ui 创建 vue 项目

vue ui 的本质：通过可视化的面板采集到用户的配置信息后，在后台基于命令行的方式自动初始化项目：



```
Windows PowerShell
PS C:\Users\escook\Desktop\vue-base\codes> vue ui
Starting GUI...
Ready on http://localhost:8000

Preset my-preset-1 saved in C:\Users\escook\.vuerc 预设存储到了 .vuerc 配置文件中
{"type":"warning","data":"@vue/cli-plugin-babel > @vue/cli-shared-utils > @hapi/joi@15.1.1: joi is leaving the @hapi org
anization and moving back to 'joi' (https://github.com/sideway/joi/issues/2411)"}
{"type":"warning","data":"@vue/cli-plugin-babel > @vue/cli-shared-utils > @hapi/joi > @hapi/bourne@1.3.2: This version h
as been deprecated and is no longer supported or maintained"}
{"type":"warning","data":"@vue/cli-plugin-babel > @vue/cli-shared-utils > @hapi/joi > @hapi/topo@3.1.6: This version has
been deprecated and is no longer supported or maintained"}
{"type":"warning","data":"@vue/cli-plugin-babel > @vue/cli-shared-utils > @hapi/joi > @hapi/hoek@8.5.1: This version has
been deprecated and is no longer supported or maintained"}
{"type":"warning","data":"@vue/cli-plugin-babel > @vue/cli-shared-utils > @hapi/joi > @hapi/topo > @hapi/hoek@8.5.1: Thi
s version has been deprecated and is no longer supported or maintained"}
{"type":"warning","data":"@vue/cli-plugin-babel > @vue/cli-shared-utils > request > har-validator@5.1.5: this library is
no longer supported"}
{"type":"warning","data":"@vue/cli-plugin-babel > @vue/cli-shared-utils > @hapi/joi > @hapi/address@2.1.4: This version
has been deprecated and is no longer supported or maintained"}
{"type":"warning","data":"@vue/cli-plugin-babel > webpack > watchpack > watchpack-chokidar2 > chokidar@2.1.8: Chokidar 2
will break on node v14+. Upgrade to chokidar 3 with 15x less dependencies."}
{"type":"warning","data":"@vue/cli-plugin-babel > webpack > watchpack > watchpack-chokidar2 > chokidar > fsevents@1.2.13
: fsevents 1 will break on node v14+ and could be using insecure binaries. Upgrade to fsevents 2."}
{"type":"warning","data":"@vue/cli-plugin-babel > webpack > micromatch > snapdragon > source-map-resolve > urix@0.1.0: P
lease see https://github.com/lydell/urix#deprecated"}
```

4. 基于 vue ui 创建 vue 项目

项目创建完成后，自动进入**项目仪表盘**：



5. 基于**命令行**创建 vue 项目

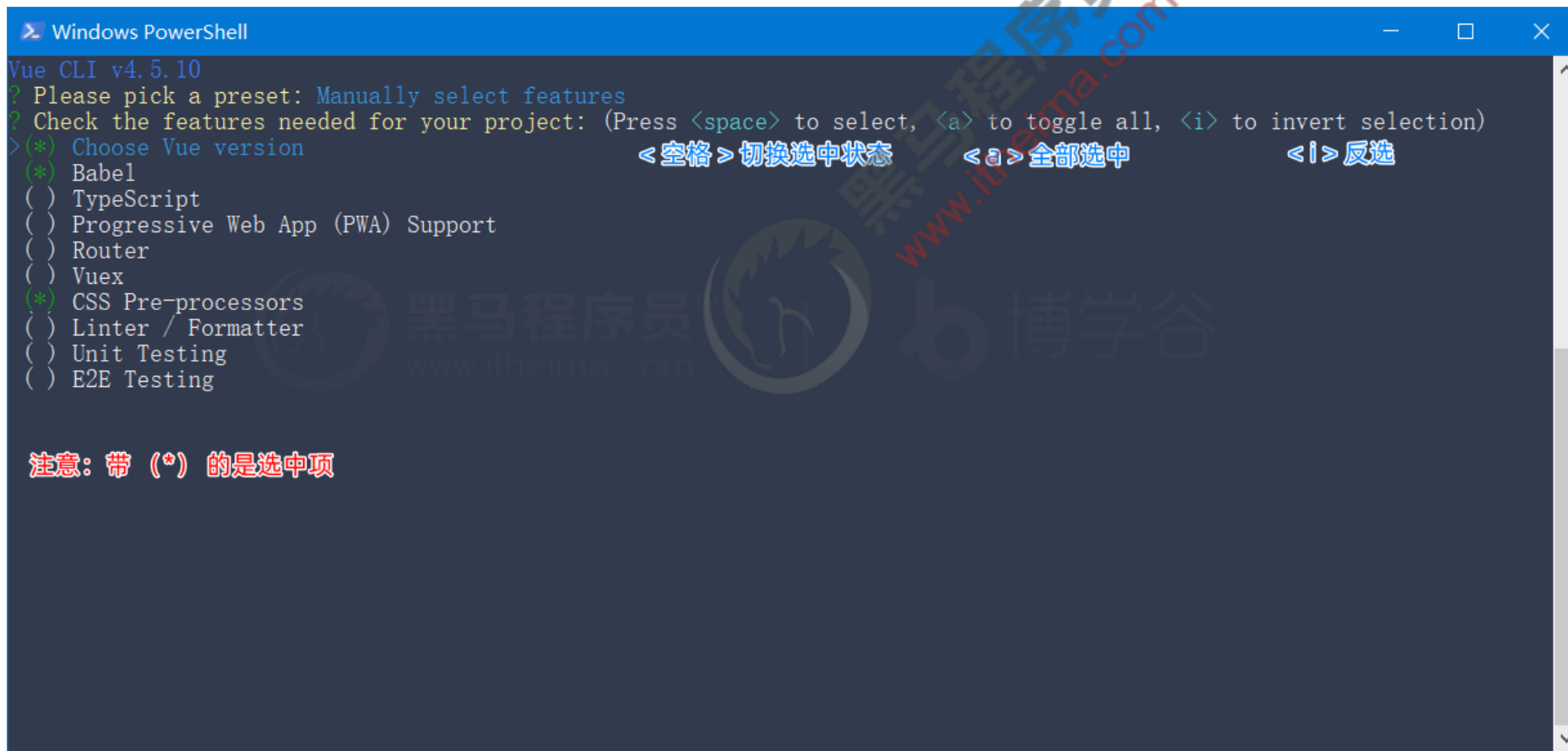
步骤1：在终端下运行 **vue create demo2** 命令，基于**交互式的命令行**创建 vue 的项目：



```
Windows PowerShell
Vue CLI v4.5.10 请选择一个合适的预设
? Please pick a preset:
  my-preset-1 ([Vue 2] less, babel) 用户自定义预设
  Default ([Vue 2] babel, eslint) 默认预设: 创建 vue2 的项目
  Default (Vue 3 Preview) ([Vue 3] babel, eslint) 默认预设: 创建 vue3 的项目
> Manually select features 手动选择要安装的功能
```

5. 基于命令行创建 vue 项目

步骤2：选择要安装的功能：

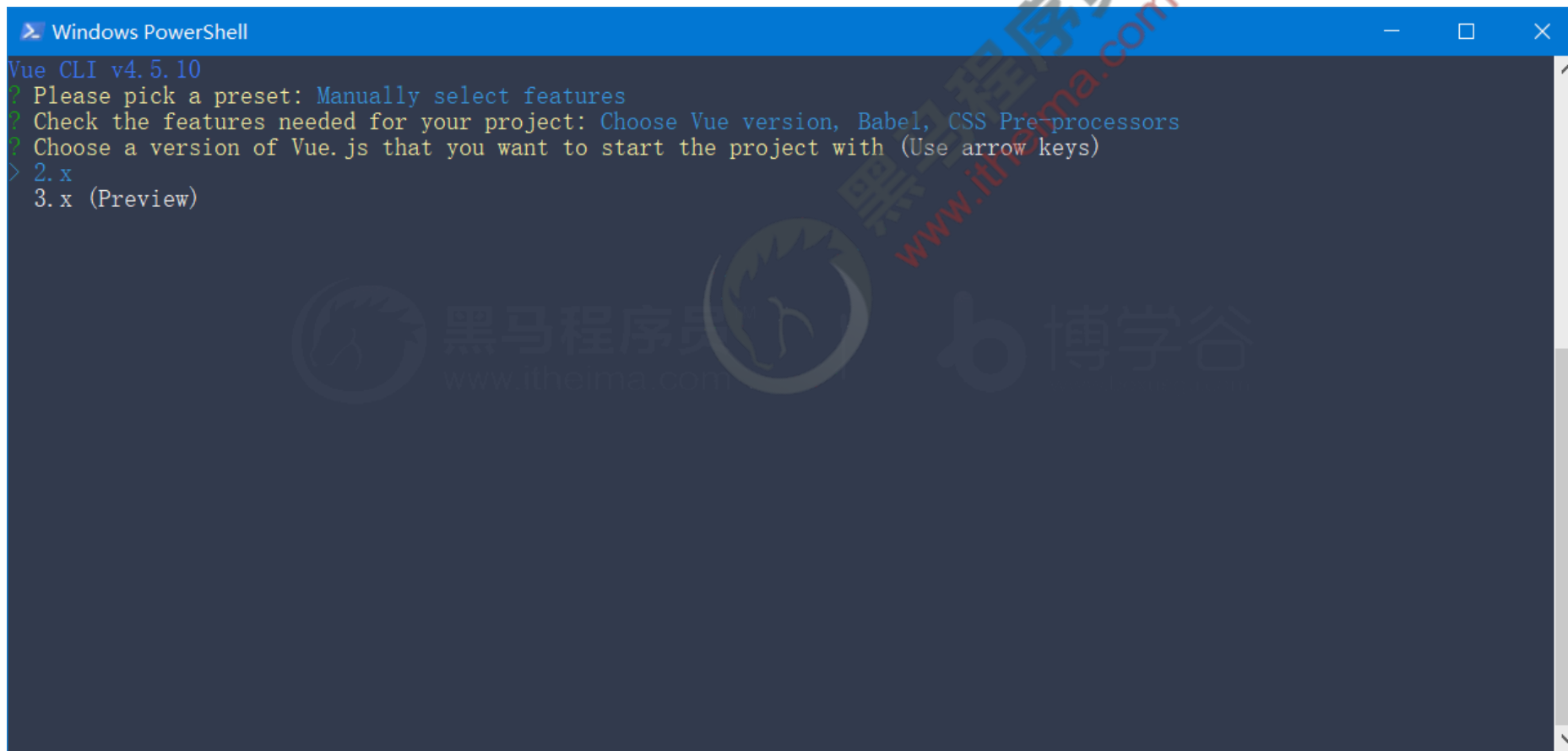


```
Windows PowerShell
Vue CLI v4.5.10
? Please pick a preset: Manually select features
? Check the features needed for your project: (Press <space> to select, <a> to toggle all, <i> to invert selection)
> (*) Choose Vue version          <空格> 切换选中状态    <a> 全部选中    <i> 反选
(*) Babel
() TypeScript
() Progressive Web App (PWA) Support
() Router
() Vuex
(*) CSS Pre-processors
() Linter / Formatter
() Unit Testing
() E2E Testing

注意：带 (*) 的是选中项
```

5. 基于**命令行**创建 vue 项目

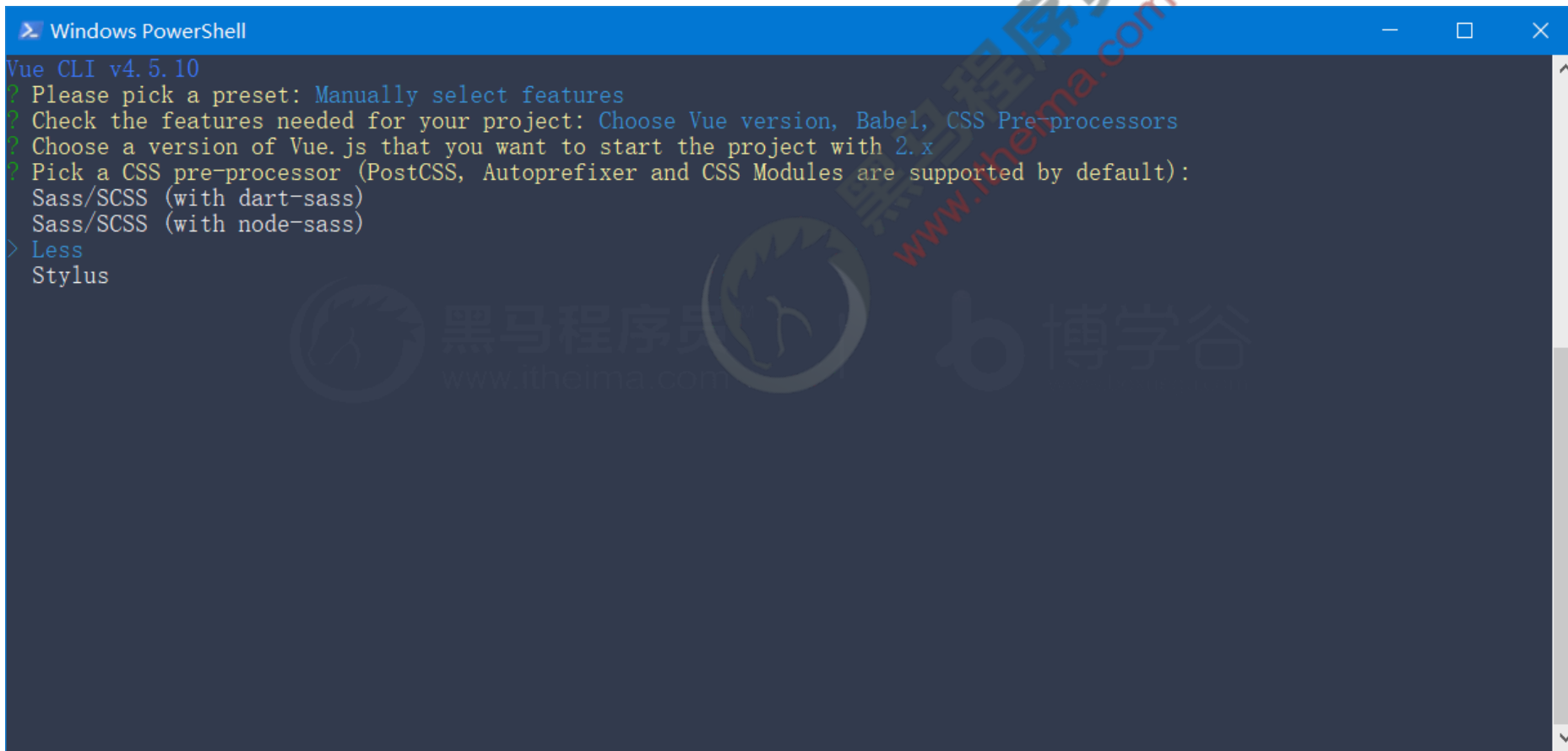
步骤3：使用**上下箭头**选择 vue 的版本，并使用**回车键**确认选择：



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

5. 基于**命令行**创建 vue 项目

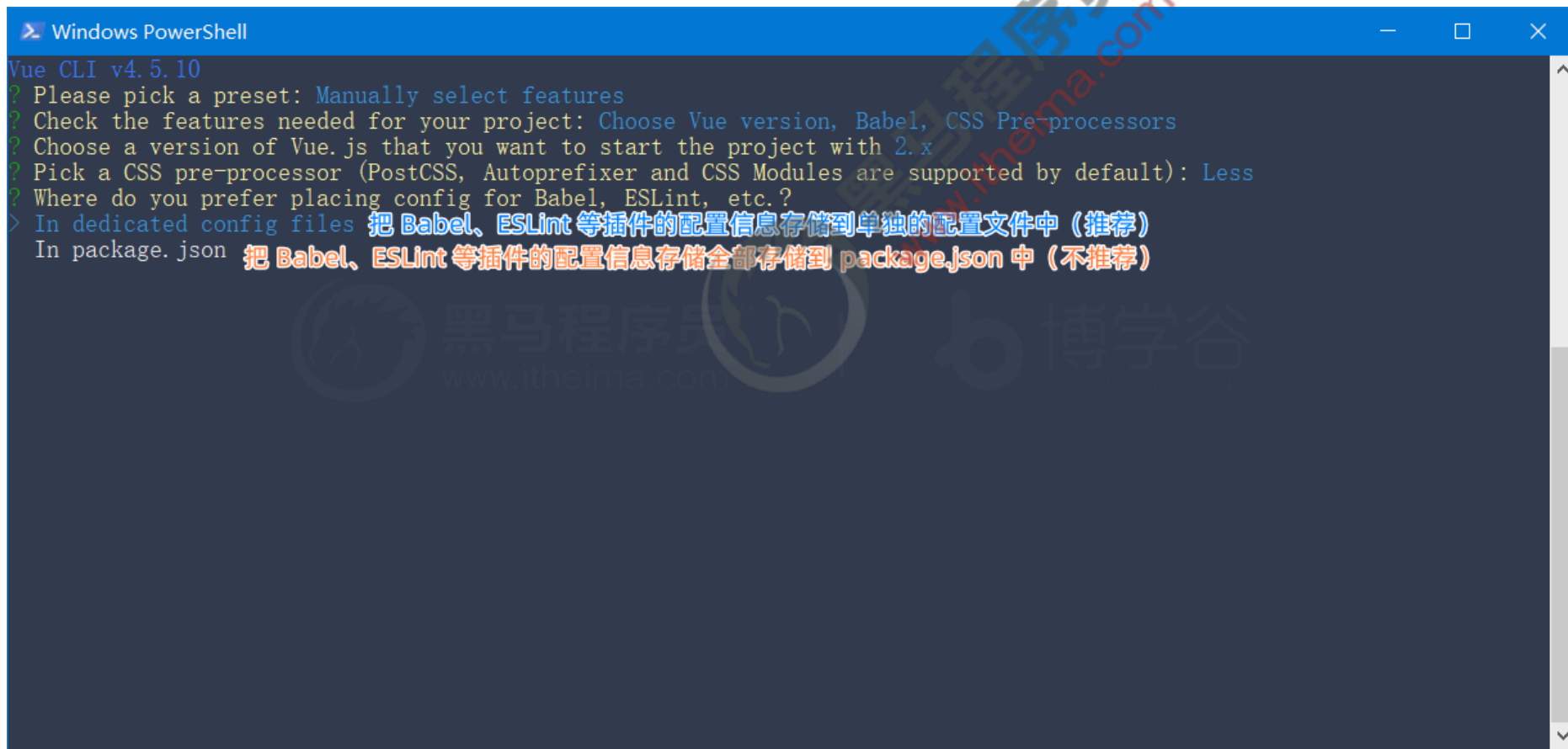
步骤4：使用**上下箭头**选择要使用的 css 预处理器，并使用**回车键**确认选择：



```
Windows PowerShell
Vue CLI v4.5.10
? Please pick a preset: Manually select features
? Check the features needed for your project: Choose Vue version, Babel, CSS Pre-processors
? Choose a version of Vue.js that you want to start the project with 2.x
? Pick a CSS pre-processor (PostCSS, Autoprefixer and CSS Modules are supported by default):
  Sass/SCSS (with dart-sass)
  Sass/SCSS (with node-sass)
> Less
  Stylus
```

5. 基于命令行创建 vue 项目

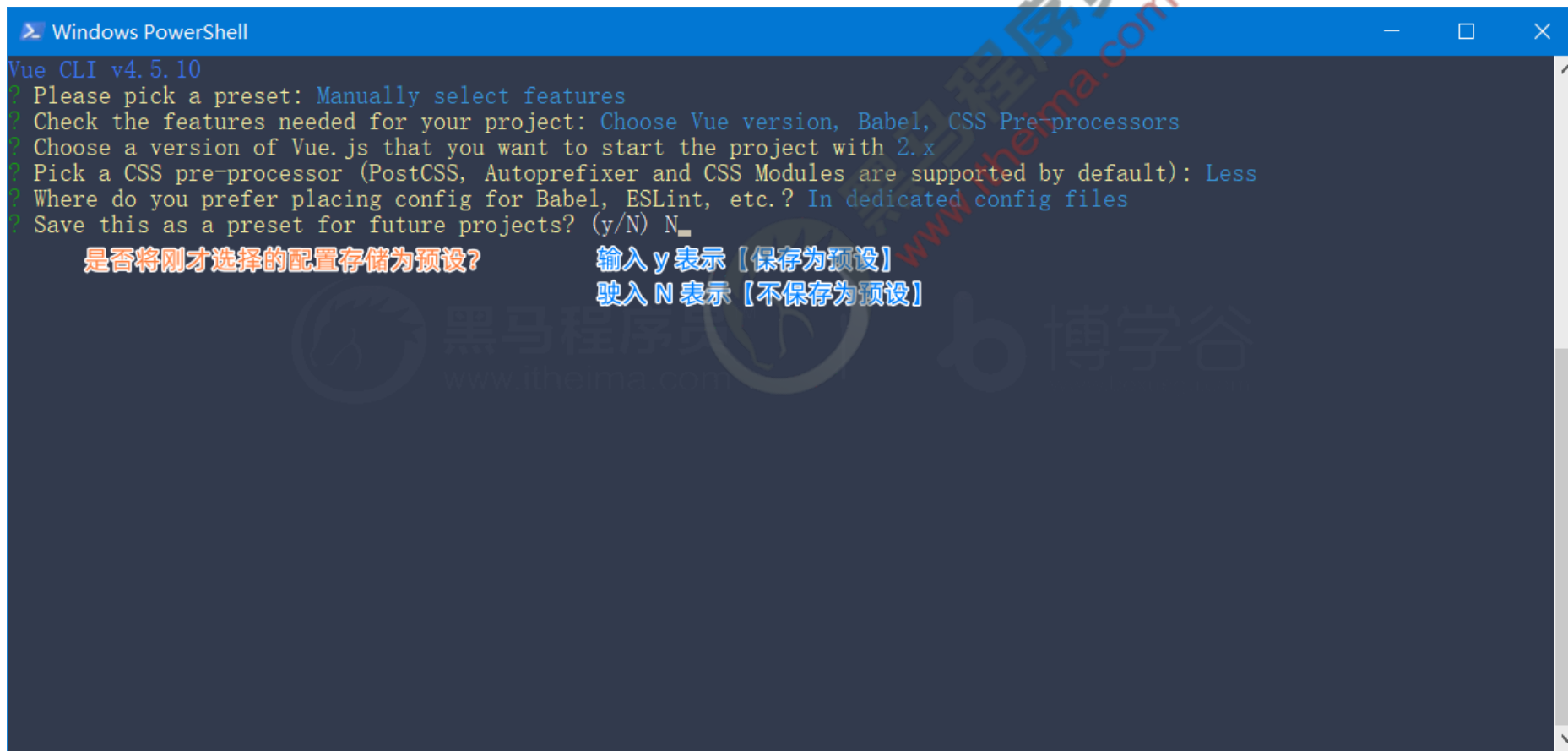
步骤5：使用上下箭头选择如何存储插件的配置信息，并使用回车键确认选择：



```
Windows PowerShell
Vue CLI v4.5.10
? Please pick a preset: Manually select features
? Check the features needed for your project: Choose Vue version, Babel, CSS Pre-processors
? Choose a version of Vue.js that you want to start the project with 2.x
? Pick a CSS pre-processor (PostCSS, Autoprefixer and CSS Modules are supported by default): Less
? Where do you prefer placing config for Babel, ESLint, etc.?
> In dedicated config files 把 Babel、ESLint 等插件的配置信息存储到单独的配置文件中（推荐）
  In package.json 把 Babel、ESLint 等插件的配置信息存储全部存储到 package.json 中（不推荐）
```

5. 基于命令行创建 vue 项目

步骤6：是否将刚才的配置保存为预设：

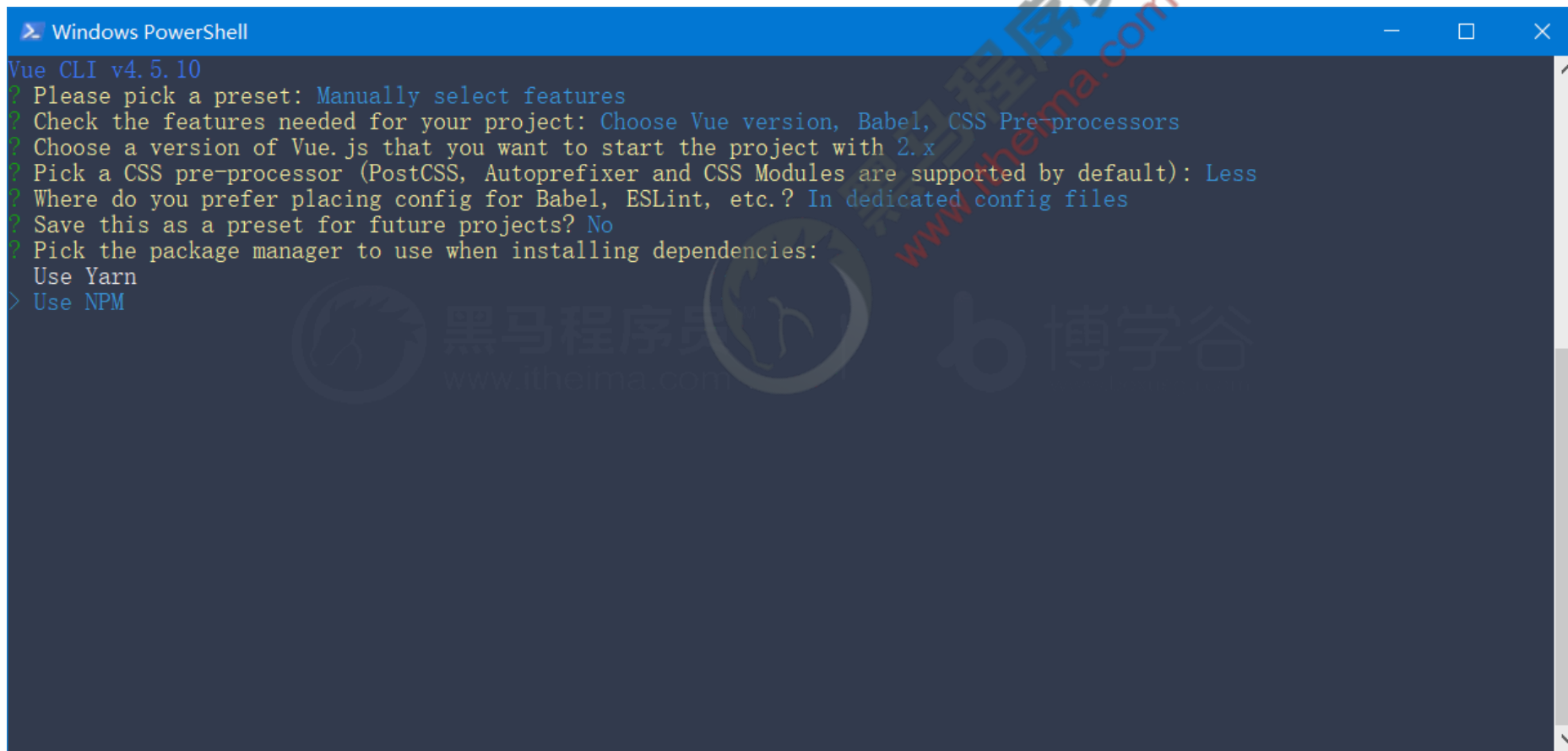


```
Windows PowerShell
Vue CLI v4.5.10
? Please pick a preset: Manually select features
? Check the features needed for your project: Choose Vue version, Babel, CSS Pre-processors
? Choose a version of Vue.js that you want to start the project with 2.x
? Pick a CSS pre-processor (PostCSS, Autoprefixer and CSS Modules are supported by default): Less
? Where do you prefer placing config for Babel, ESLint, etc.? In dedicated config files
? Save this as a preset for future projects? (y/N) N_

是否将刚才选择的配置存储为预设?      输入 y 表示【保存为预设】
                                      输入 N 表示【不保存为预设】
```

5. 基于**命令行**创建 vue 项目

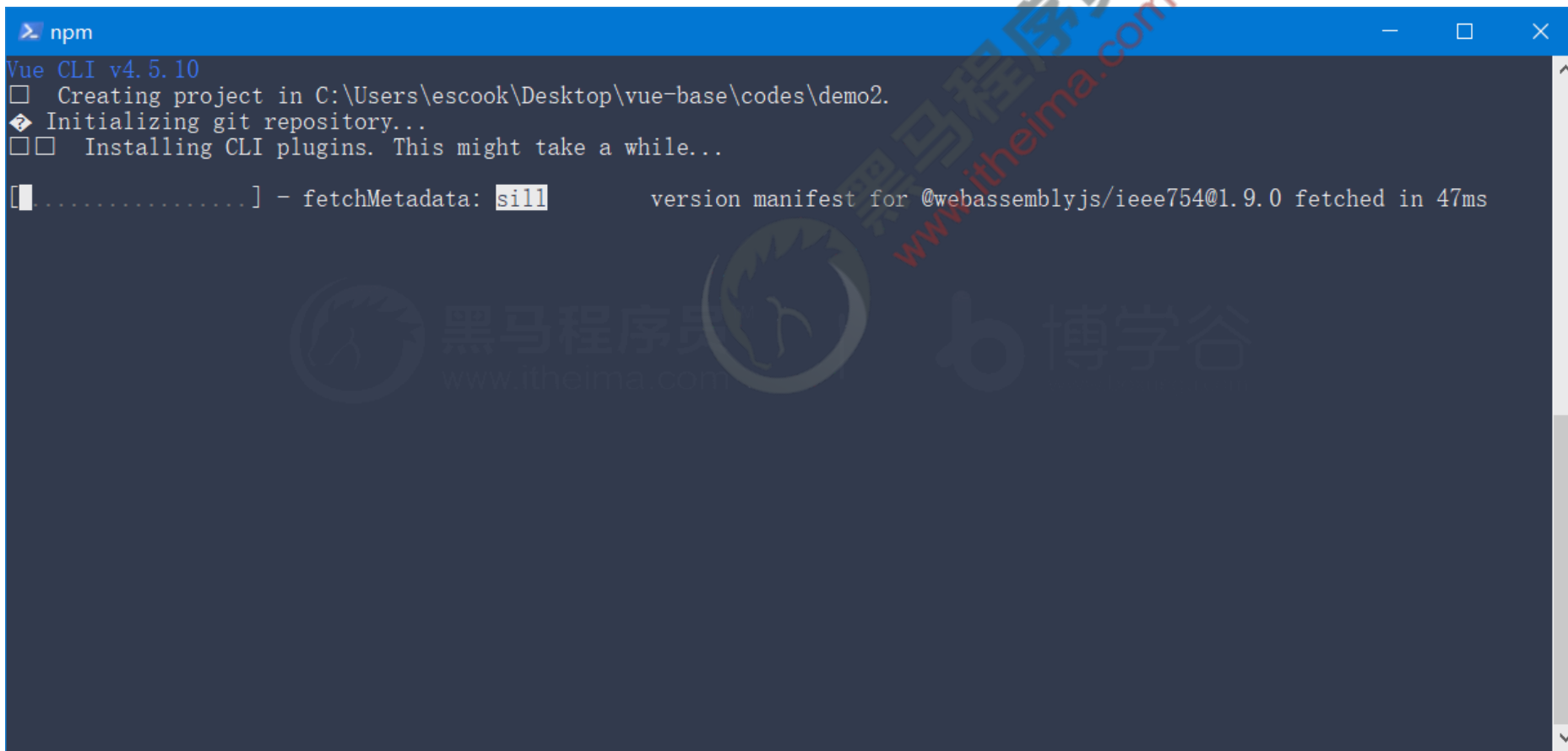
步骤7：选择如何安装项目中的依赖包：



```
Windows PowerShell
Vue CLI v4.5.10
? Please pick a preset: Manually select features
? Check the features needed for your project: Choose Vue version, Babel, CSS Pre-processors
? Choose a version of Vue.js that you want to start the project with 2.x
? Pick a CSS pre-processor (PostCSS, Autoprefixer and CSS Modules are supported by default): Less
? Where do you prefer placing config for Babel, ESLint, etc.? In dedicated config files
? Save this as a preset for future projects? No
? Pick the package manager to use when installing dependencies:
  Use Yarn
> Use NPM
```

5. 基于**命令行**创建 vue 项目

步骤8：开始创建项目并自动安装依赖包：



```
npm
Vue CLI v4.5.10
□ Creating project in C:\Users\escook\Desktop\vue-base\codes\demo2.
◆ Initializing git repository...
□□ Installing CLI plugins. This might take a while...

[ ] .....] - fetchMetadata: sill   version manifest for @webassemblyjs/ieee754@1.9.0 fetched in 47ms
```

5. 基于命令行创建 vue 项目

步骤9：项目创建完成：



```
Windows PowerShell
> ejs@2.7.4 postinstall C:\Users\escook\Desktop\vue-base\codes\demo2\node_modules\ejs
> node ./postinstall.js

added 1205 packages from 929 contributors in 36.772s

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

❖ Invoking generators...
❖ Installing additional dependencies...

added 13 packages from 68 contributors in 5.906s

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

❑ Running completion hooks...

❖ Generating README.md...

❖ Successfully created project demo2.
❖ Get started with the following commands:

$ cd demo2
$ npm run serve

PS C:\Users\escook\Desktop\vue-base\codes>
```

项目已完成创建，
执行如下的命令，即可把项目运行起来：

6. 梳理 vue2 项目的基本结构

```
> node_modules
└─ public
   ├── favicon.ico
   └─ index.html
└─ src
   ├── assets
   ├── components
   ├── App.vue
   ├── main.js
   ├── .browserslistrc
   ├── .gitignore
   ├── babel.config.js
   ├── package-lock.json
   ├── package.json
   └─ README.md
```

主要的文件：

src -> **App.vue**

src -> **main.js**



黑马程序员
www.itheima.com

7. 分析 main.js 中的主要代码

```
1 // 1. 导入 Vue 的构造函数
2 import Vue from 'vue'
3 // 2. 导入 App 根组件
4 import App from './App.vue'
5
6 // 屏蔽浏览器 console 面板的提示消息
7 Vue.config.productionTip = false
8
9 // 3. 创建 vue 实例对象
10 new Vue({
11   render: h => h(App), // 3.1 使用 render 函数渲染 App 根组件
12 }).$mount('#app')      // 3.2 把 App 根组件渲染到 $mount 函数指定的 el 区域中
```

8. 在 vue2 的项目中使用路由

在 vue2 的项目中，只能安装并使用 3.x 版本的 vue-router。

版本 3 和版本 4 的路由最主要的区别：创建路由模块的方式不同！



8.1 回顾：4.x 版本的路由如何创建路由模块

```
1 import { createRouter, createWebHashHistory } from 'vue-router' // 1. 按需导入需要的方法
2
3 import MyHome from './Home.vue' // 2. 导入需要使用路由进行切换的组件
4 import MyMovie from './Movie.vue'
5
6 const router = createRouter({ // 3. 创建路由对象
7   history: createWebHashHistory(), // 3.1 指定通过 hash 管理路由的切换
8   routes: [ // 3.2 创建路由规则
9     { path: '/home', component: MyHome },
10    { path: '/movie', component: MyMovie },
11  ],
12 })
13
14 export default router // 4. 向外共享路由对象
```

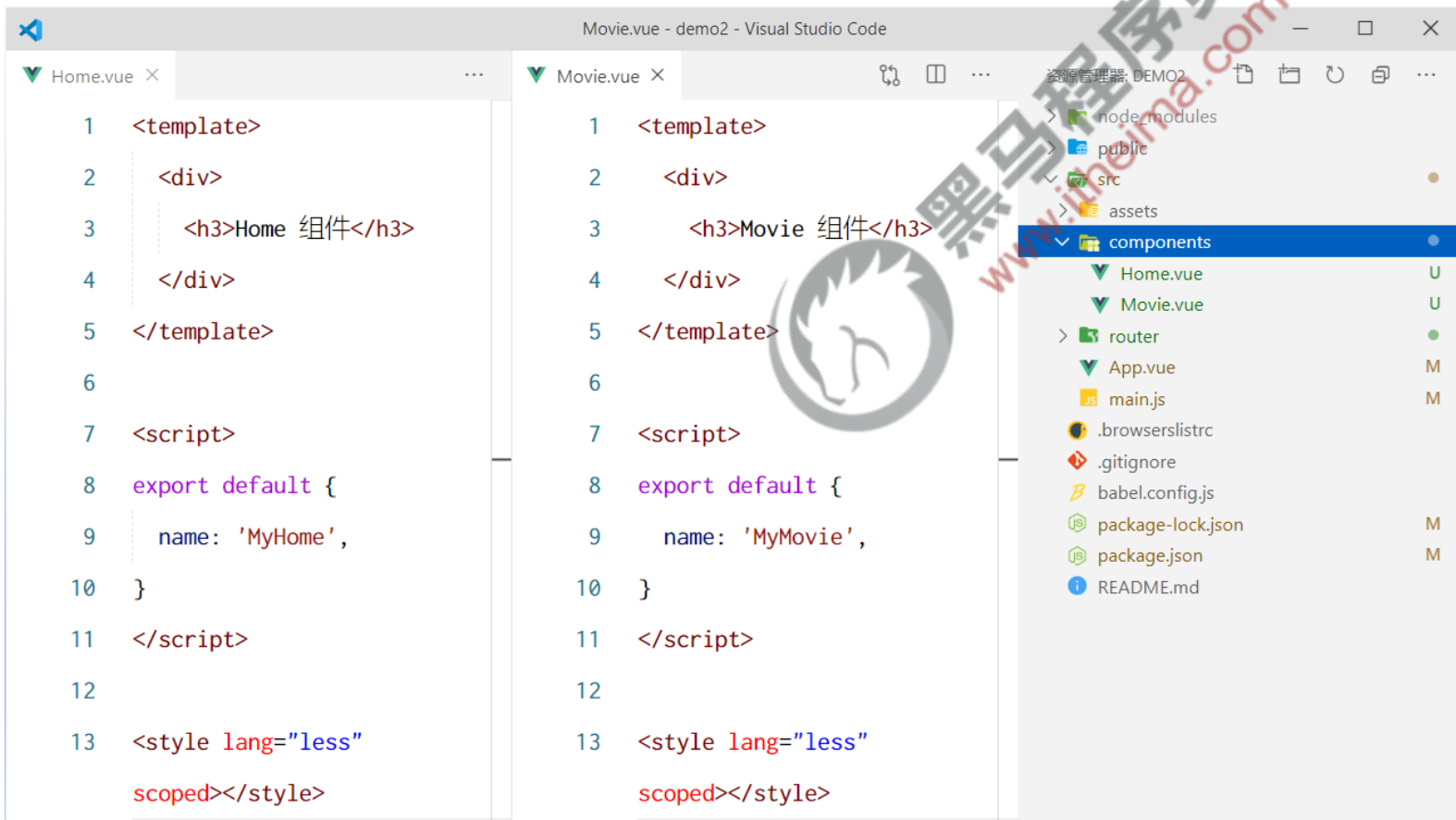
8.2 学习：3.x 版本的路由如何创建路由模块

步骤1：在 vue2 的项目中安装 3.x 版本的路由：

```
1 npm i vue-router@3.4.9 -S
```

8.2 学习：3.x 版本的路由如何创建路由模块

步骤2：在 src -> components 目录下，创建需要使用路由切换的组件：



The screenshot shows the Visual Studio Code interface with two files open: Home.vue and Movie.vue. The Movie.vue file is currently active and shows a template with a single h3 element and a script section with an export default object containing a name property set to 'MyMovie'. The Explorer sidebar on the right shows the project structure, with the 'components' directory selected under the 'src' folder. The 'components' directory contains Home.vue and Movie.vue. The 'router' directory is also visible, containing App.vue, main.js, and a README.md file.

```
1 <template>
2   <div>
3     <h3>Movie 组件</h3>
4   </div>
5 </template>
6
7 <script>
8   export default {
9     name: 'MyMovie',
10  }
11 </script>
12
13 <style lang="less"
  scoped></style>
```

8.2 学习：3.x 版本的路由如何创建路由模块

步骤3：在 src 目录下创建 router -> index.js 路由模块：

```
1 import Vue from 'vue' // 1. 导入 Vue2 的构造函数
2 import VueRouter from 'vue-router' // 2. 导入 3.x 路由的构造函数
3
4 import Home from '@/components/Home.vue' // 3. 导入需要使用路由切换的组件
5 import Movie from '@/components/Movie.vue'
6
7 Vue.use(VueRouter) // 4. 调用 Vue.use() 函数，把路由配置为 Vue 的插件
8
9 const router = new VueRouter({ // 5. 创建路由对象
10   routes: [ // 5.1 声明路由规则
11     { path: '/', redirect: '/home' },
12     { path: '/home', component: Home },
13     { path: '/movie', component: Movie },
14   ],
15 })
16
17 export default router // 6. 向外共享路由对象
```

8.2 学习：3.x 版本的路由如何创建路由模块

步骤4：在 `main.js` 中导入路由模块，并通过 `router` 属性进行挂载：

```
1 import Vue from 'vue'
2 import App from './App.vue'
3 // 1. 导入路由模块
4 import router from './router'
5
6 Vue.config.productionTip = false
7
8 new Vue({
9   render: h => h(App),
10   // 2. 挂载路由模块
11   // router: router,
12   router,
13 }).$mount('#app')
```

8.2 学习：3.x 版本的路由如何创建路由模块

步骤5：在 `App.vue` 根组件中，使用 `<router-view>` 声明路由的占位符：

```
1 <template>
2   <div>
3     <h1>App 根组件</h1>
4     <hr />
5     <!-- 声明路由的占位符 -->
6     <router-view></router-view>
7   </div>
8 </template>
```

录 Contents

◆ vue-cli

◆ 组件库

◆ axios 拦截器

◆ proxy 跨域代理

◆ 用户列表案例

1. 什么是 vue 组件库

在实际开发中，前端开发者可以把自己封装的 .vue 组件整理、打包、并发布为 npm 的包，从而供其他人下载和使用。这种可以直接下载并在项目中使用的现成组件，就叫做 vue 组件库。



2. vue 组件库和 bootstrap 的区别

二者之间存在本质的区别：

- bootstrap 只提供了**纯粹的原材料**（css 样式、HTML 结构以及JS 特效），需要由开发者做**进一步的组装和改造**
- vue 组件库是**遵循 vue 语法、高度定制**的现成组件，开箱即用



3. 最常用的 vue 组件库

① PC 端

- Element UI (<https://element.eleme.cn/#/zh-CN>)
- View UI (<http://v1.iviewui.com/>)

② 移动端

- Mint UI (<http://mint-ui.github.io/#!/zh-cn>)
- Vant (<https://vant-contrib.gitee.io/vant/#/zh-CN/>)

4. Element UI

Element UI 是饿了么前端团队开源的一套 PC 端 vue 组件库。支持在 vue2 和 vue3 的项目中使用：

- vue2 的项目使用旧版的 Element UI (<https://element.eleme.cn/#/zh-CN>)
- vue3 的项目使用新版的 Element Plus (<https://element-plus.gitee.io/#/zh-CN>)



4.2 在 vue2 的项目中**安装** element-ui

运行如下的终端命令：

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

4.2 引入 element-ui

开发者可以一次性完整引入所有的 element-ui 组件，或是根据需求，只按需引入用到的 element-ui 组件：

- 完整引入：操作简单，但是会额外引入一些用不到的组件，导致项目体积过大
- 按需引入：操作相对复杂一些，但是只会引入用到的组件，能起到优化项目体积的目的

4.3 完整引入

在 `main.js` 中写入以下内容：

```
1 import Vue from 'vue'
2 import App from './App.vue'
3 // 1. 完整引入 element ui 的组件
4 import ElementUI from 'element-ui'
5 // 2. 导入 element ui 组件的样式
6 import 'element-ui/lib/theme-chalk/index.css'
7
8 // 3. 把 ElementUI 注册为 vue 的插件【注册之后，即可在每个组件中直接使用每一个 element ui 的组件】
9 Vue.use(ElementUI)
10
11 new Vue({
12   render: h => h(App),
13 }).$mount('#app')
```

4.4 按需引入

借助 `babel-plugin-component`，我们可以只引入需要的组件，以达到减小项目体积的目的。

步骤1，安装 `babel-plugin-component`：

```
1 npm install babel-plugin-component -D
```

4.4 按需引入

步骤2，修改根目录下的 `babel.config.js` 配置文件，新增 `plugins` 节点如下：

```
1 module.exports = {
2   presets: ['@vue/cli-plugin-babel/preset'],
3   // ↓↓↓
4   plugins: [
5     [
6       'component',
7       {
8         libraryName: 'element-ui',
9         styleLibraryName: 'theme-chalk',
10      },
11    ],
12  ],
13   // ↑↑↑
14 }
```

4.4 按需引入

步骤3，如果你只希望引入部分组件，比如 Button，那么需要在 main.js 中写入以下内容：

```
1 import Vue from 'vue'
2 import App from './App.vue'
3 // 1. 按需导入 element ui 的组件
4 import { Button } from 'element-ui'
5
6 // 2. 注册需要的组件
7 Vue.component(Button.name, Button)
8 /* 或写为
9  * Vue.use(Button)
10  */
11
12 new Vue({
13   render: h => h(App),
14 }).$mount('#app')
```

4.5 把组件的导入和注册封装为独立的模块

在 src 目录下新建 `element-ui/index.js` 模块，并声明如下的代码：

```
1 // → 模块路径 /src/element-ui/index.js
2 import Vue from 'vue'
3 // 按需导入 element ui 的组件
4 import { Button, Input } from 'element-ui'
5
6 // 注册需要的组件
7 Vue.use(Button)
8 Vue.use(Input)
9
10 // → 在 main.js 中导入
11 import './element-ui'
```

录 Contents

- ◆ vue-cli
- ◆ 组件库
- ◆ axios 拦截器
- ◆ proxy 跨域代理
- ◆ 用户列表案例

1. 回顾：在 vue3 的项目中全局配置 axios

```
1 import { createApp } from 'vue'
2 import App from './App.vue'
3 // 1. 导入 axios
4 import axios from 'axios'
5
6 const app = createApp(App)
7 // 2. 配置请求根路径
8 axios.defaults.baseURL = 'https://www.escook.cn'
9 // 3. 全局配置 axios
10 app.config.globalProperties.$http = axios
11 app.mount('#app')
```

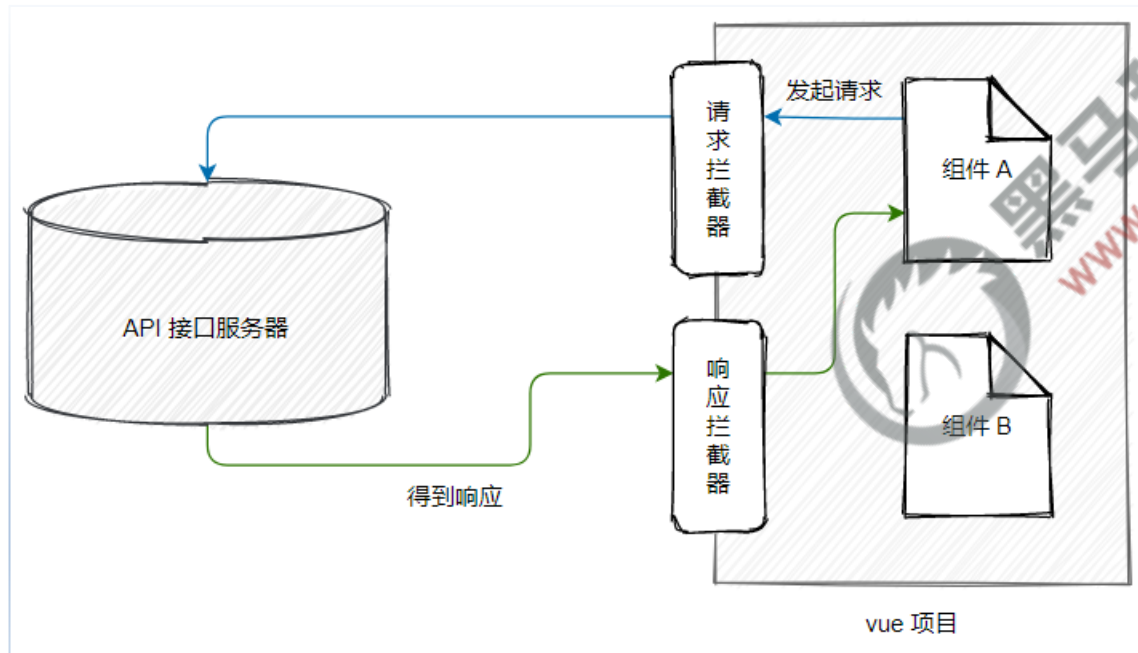
2. 在 vue2 的项目中全局配置 axios

需要在 `main.js` 入口文件中，通过 `Vue` 构造函数的 `prototype` 原型对象全局配置 axios：

```
1 import Vue from 'vue'
2 import App from './App.vue'
3 // 1. 导入 axios
4 import axios from 'axios'
5
6 // 2. 配置请求根路径
7 axios.defaults.baseURL = 'https://www.escook.cn'
8 // 3. 通过 Vue 构造函数的原型对象，全局配置 axios
9 Vue.prototype.$http = axios
10
11 new Vue({
12   render: h => h(App),
13 }).$mount('#app')
```

3. 什么是拦截器

拦截器（英文：Interceptors）会在每次发起 ajax 请求和得到响应的时候自动被触发。



应用场景：

- ① Token 身份认证
- ② Loading 效果
- ③ etc...

4. 配置请求拦截器

通过 `axios.interceptors.request.use`(成功的回调, 失败的回调) 可以配置请求拦截器。示例代码如下:

```
1 axios.interceptors.request.use(function (config) {  
2   // Do something before request is sent  
3   return config;  
4 }, function (error) {  
5   // Do something with request error  
6   return Promise.reject(error);  
7 })
```

注意: 失败的回调函数可以被省略!

4.1 请求拦截器 - Token 认证

```
1 import axios from 'axios'
2
3 axios.defaults.baseURL = 'https://www.escook.cn'
4 // 配置请求的拦截器
5 axios.interceptors.request.use(config => {
6   // 为当前请求配置 Token 认证字段
7   config.headers.Authorization = 'Bearer xxx'
8   return config
9 })
10
11 Vue.prototype.$http = axios
```

4.2 请求拦截器 - 展示 Loading 效果

借助于 element ui 提供的 **Loading 效果** 组件 (<https://element.eleme.cn/#/zh-CN/component/loading>) 可以方便的实现 Loading 效果的展示:

```
1 // 1. 按需导入 Loading 效果组件
2 import { Loading } from 'element-ui'
3
4 // 2. 声明变量, 用来存储 Loading 组件的实例对象
5 let loadingInstance = null
6
7 // 配置请求的拦截器
8 axios.interceptors.request.use(config => {
9   // 3. 调用 Loading 组件的 service() 方法, 创建 Loading 组件的实例, 并全屏展示 loading 效果
10   loadingInstance = Loading.service({ fullscreen: true })
11   return config
12 })
```

5. 配置响应拦截器

通过 `axios.interceptors.response.use`(成功的回调, 失败的回调) 可以配置响应拦截器。示例代码如下:

```
1 // Add a response interceptor
2 axios.interceptors.response.use(function (response) {
3     // Any status code that lie within the range of 2xx cause this function to trigger
4     // Do something with response data
5     return response;
6 }, function (error) {
7     // Any status codes that falls outside the range of 2xx cause this function to trigger
8     // Do something with response error
9     return Promise.reject(error);
10 });
```

注意: 失败的回调函数可以被省略!

5.1 响应拦截器 - 关闭 Loading 效果

调用 Loading 实例提供的 `close()` 方法即可关闭 Loading 效果，示例代码如下：

```
1 // 响应拦截器
2 axios.interceptors.response.use(response => {
3   // 调用 Loading 实例的 close 方法即可关闭 loading 效果
4   loadingInstance.close()
5   return response
6 })
```

录 Contents

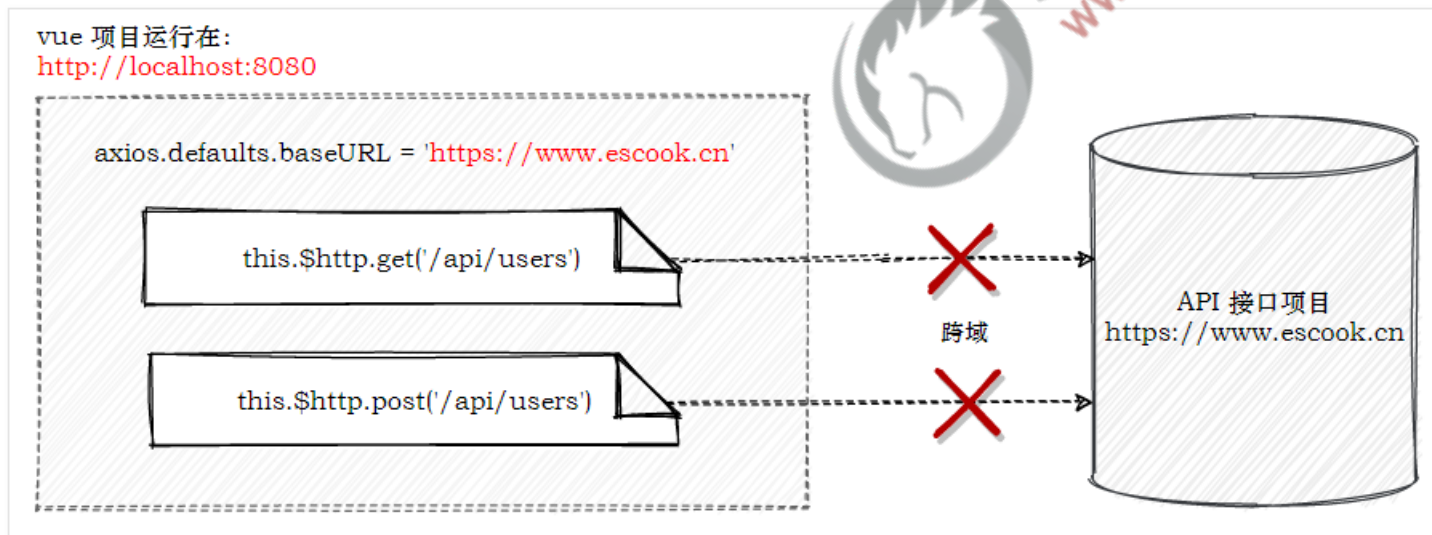
- ◆ vue-cli
- ◆ 组件库
- ◆ axios 拦截器
- ◆ proxy 跨域代理
- ◆ 用户列表案例

1. 回顾：接口的跨域问题

vue 项目运行的地址：<http://localhost:8080/>

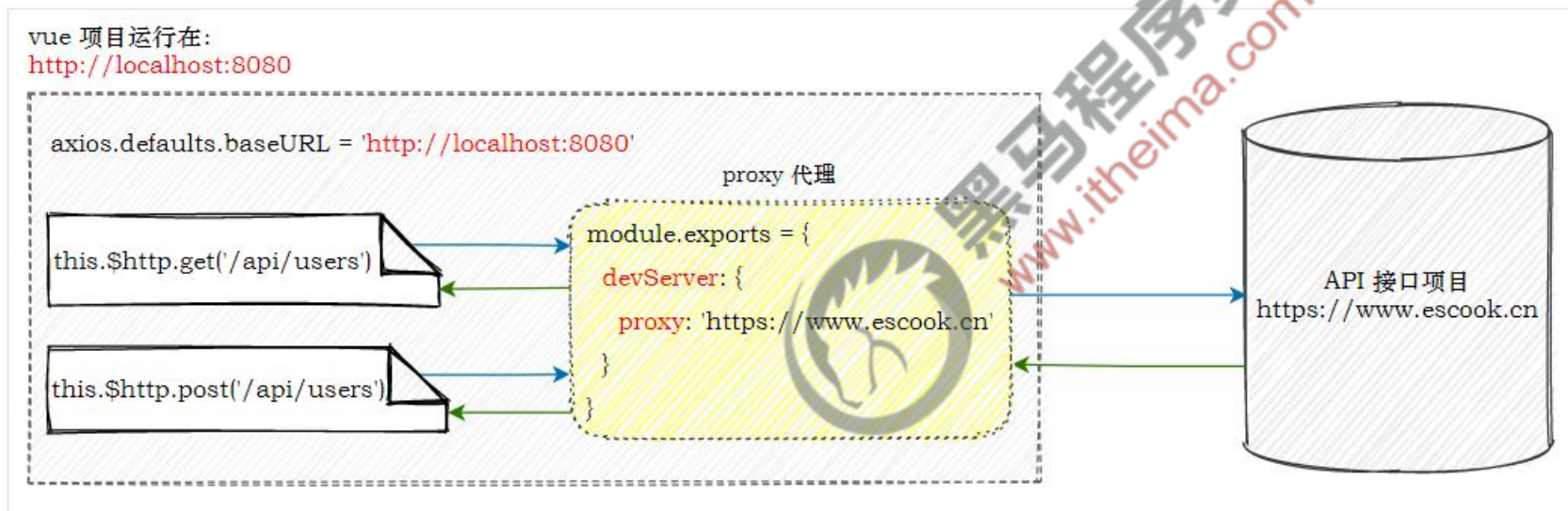
API 接口运行的地址：<https://www.escook.cn/api/users>

由于当前的 API 接口没有开启 CORS 跨域资源共享，因此默认情况下，上面的接口无法请求成功！



2. 通过代理解决接口的跨域问题

通过 vue-cli 创建的项目在遇到接口跨域问题时，可以通过代理的方式来解决：



- ① 把 axios 的请求根路径设置为 vue 项目的运行地址（接口请求不再跨域）
- ② vue 项目发现请求的接口不存在，把请求转交给 proxy 代理
- ③ 代理把请求根路径替换为 devServer.proxy 属性的值，发起真正的数据请求
- ④ 代理把请求到的数据，转发给 axios

3. 在项目中配置 proxy 代理

步骤1，在 `main.js` 入口文件中，把 `axios` 的请求根路径改造为当前 web 项目的根路径：

```
1 // axios.defaults.baseURL = 'https://www.escook.cn'
2
3 // 配置请求根路径
4 axios.defaults.baseURL = 'http://localhost:8080'
```

3. 在项目中配置 proxy 代理

步骤2，在**项目根目录**下创建 `vue.config.js` 的配置文件，并声明如下的配置：

```
1 module.exports = {  
2   devServer: {  
3     // 当前项目在开发调试阶段，  
4     // 会将任何未知请求（没有匹配到静态文件的请求）代理到 https://www.escook.cn  
5     proxy: 'https://www.escook.cn',  
6   },  
7 }
```

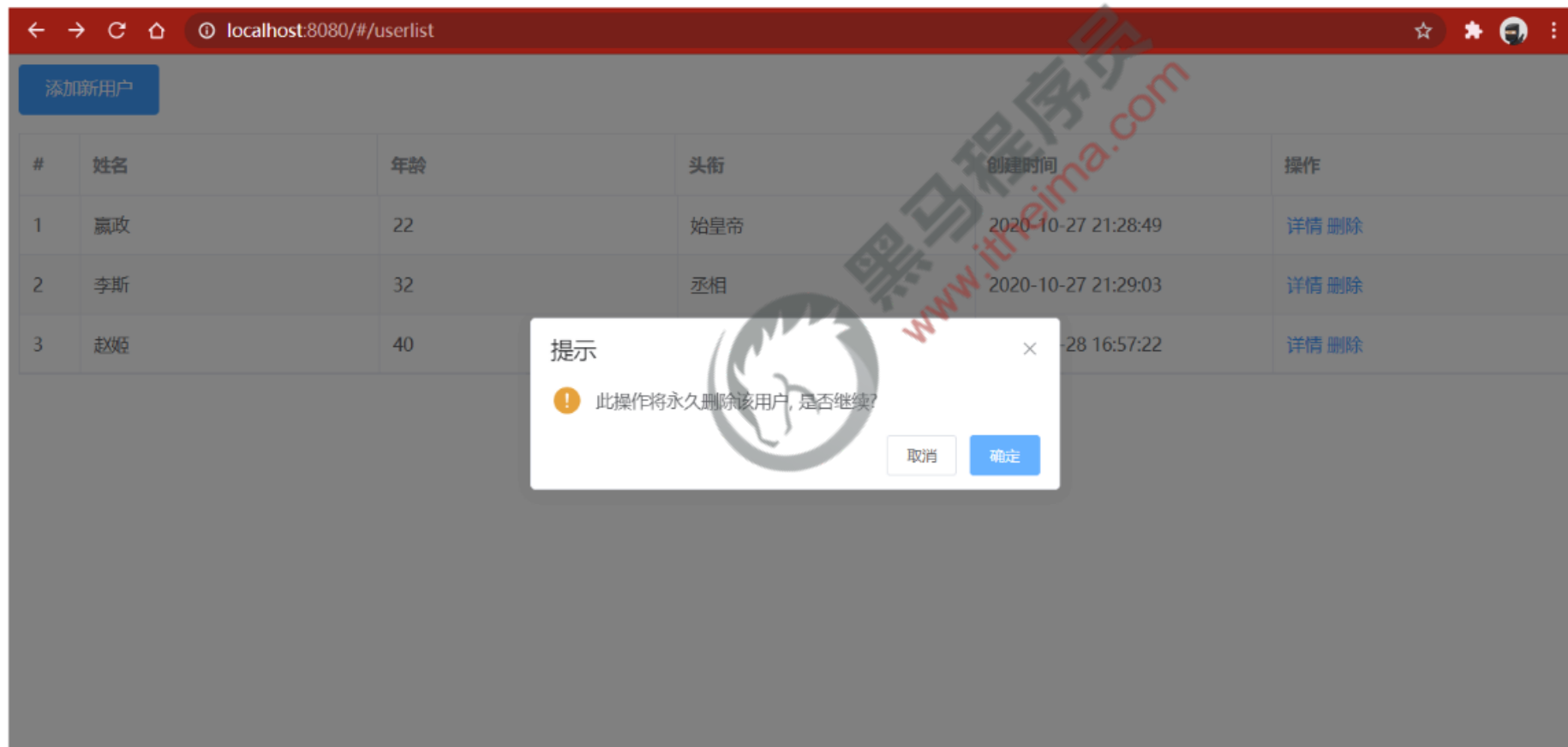
注意：

- ① `devServer.proxy` 提供的代理功能，**仅在开发调试阶段生效**
- ② 项目上线发布时，依旧需要 API 接口服务器**开启 CORS** 跨域资源共享

录 Contents

- ◆ vue-cli
- ◆ 组件库
- ◆ axios 拦截器
- ◆ proxy 跨域代理
- ◆ 用户列表案例

1. 案例效果



1. 案例效果



2. 用到的知识点

- vue-cli 创建 vue2 项目
- element ui 组件库
- axios 拦截器
- proxy 跨域接口代理
- vue-router 路由





3. 整体实现步骤

- ① 初始化项目
- ② 渲染用户表格的数据
- ③ 基于全局过滤器处理时间格式
- ④ 实现添加用户的操作
- ⑤ 实现删除用户的操作
- ⑥ 通过路由跳转到详情页



 总结

- ① 能够知道如何使用 vue-cli 创建项目
 - vue ui 、 vue create 项目名称
- ② 能够知道如何在项目中安装和配置 element-ui
 - 完整引入、按需引入 参考官方文档进行配置
- ③ 能够知道 element-ui 中常见组件的用法
 - Table 表格、Form 表单、Dialog 对话框、Message 消息、MessageBox 弹框
- ④ 能够知道如何使用 axios 拦截器
 - axios.interceptors.request.use()、axios.interceptors.response.use()
- ⑤ 能够知道如何配置 proxy 代理
 - 修改请求根路径、vue.config.js、devServer.proxy



黑马程序员

www.itheima.com

传智播客旗下高端IT教育品牌

