VUE 基础 - 综合案例

目录 Contents

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

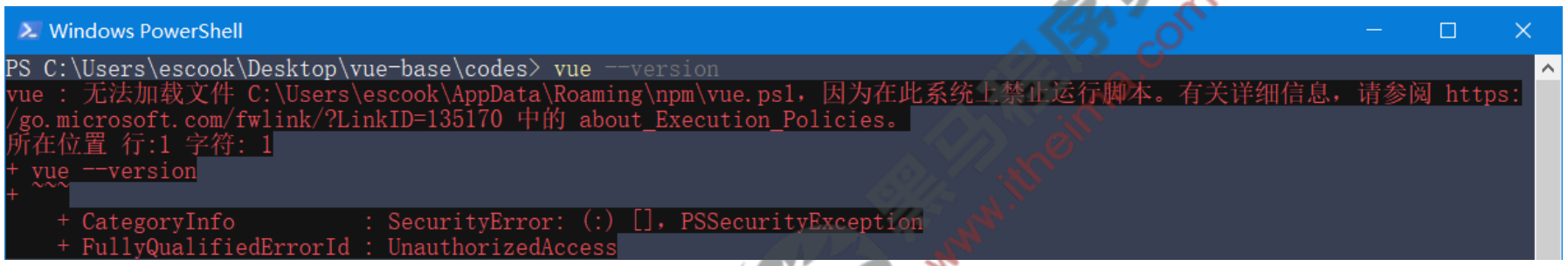
vue-cli 是基于 Node.js 开发出来的工具，因此需要使用 npm 将它安装为全局可用的工具：

```
# 全局安装 vue-cli
npm install -g @vue/cli

# 查看 vue-cli 的版本，检验 vue-cli 是否安装成功
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：

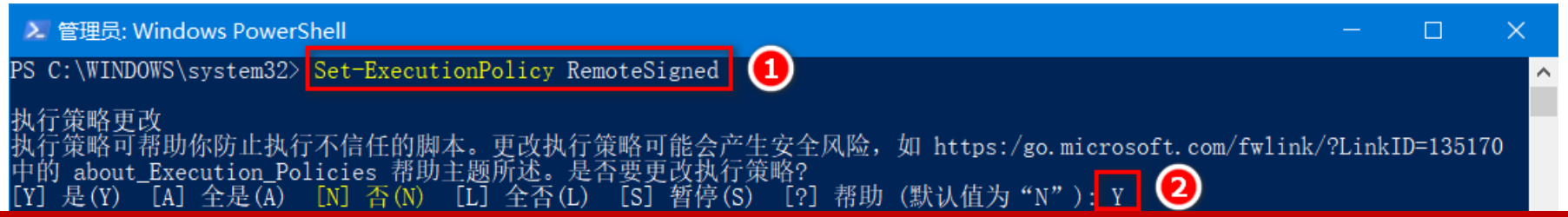


解决方案如下：

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



3. 创建项目

vue-cli 提供了创建项目的两种方式：

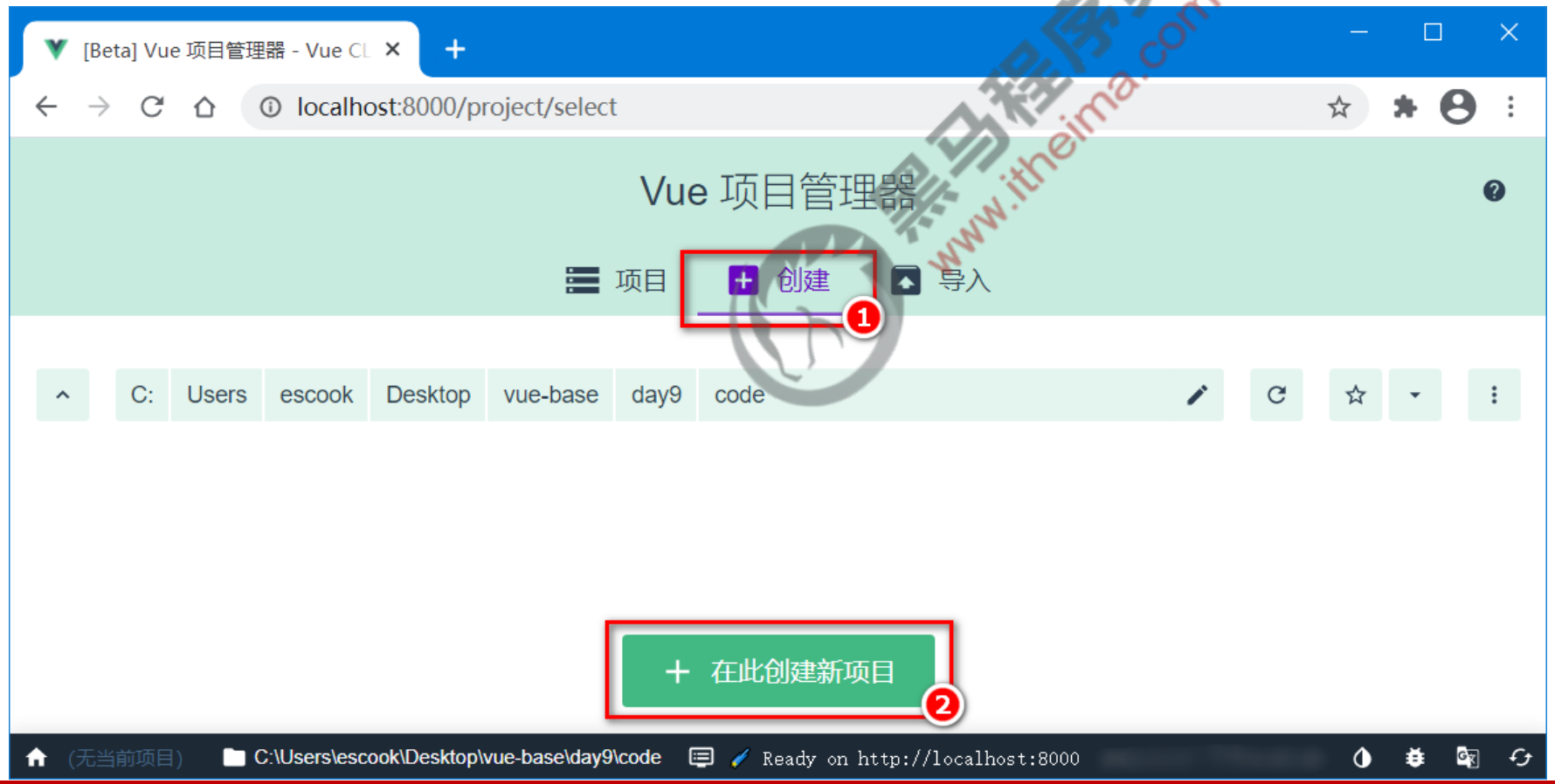
```
# 基于【命令行】的方式创建 vue 项目
vue create 项目名称

# OR

# 基于【可视化面板】创建 vue 项目
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：

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

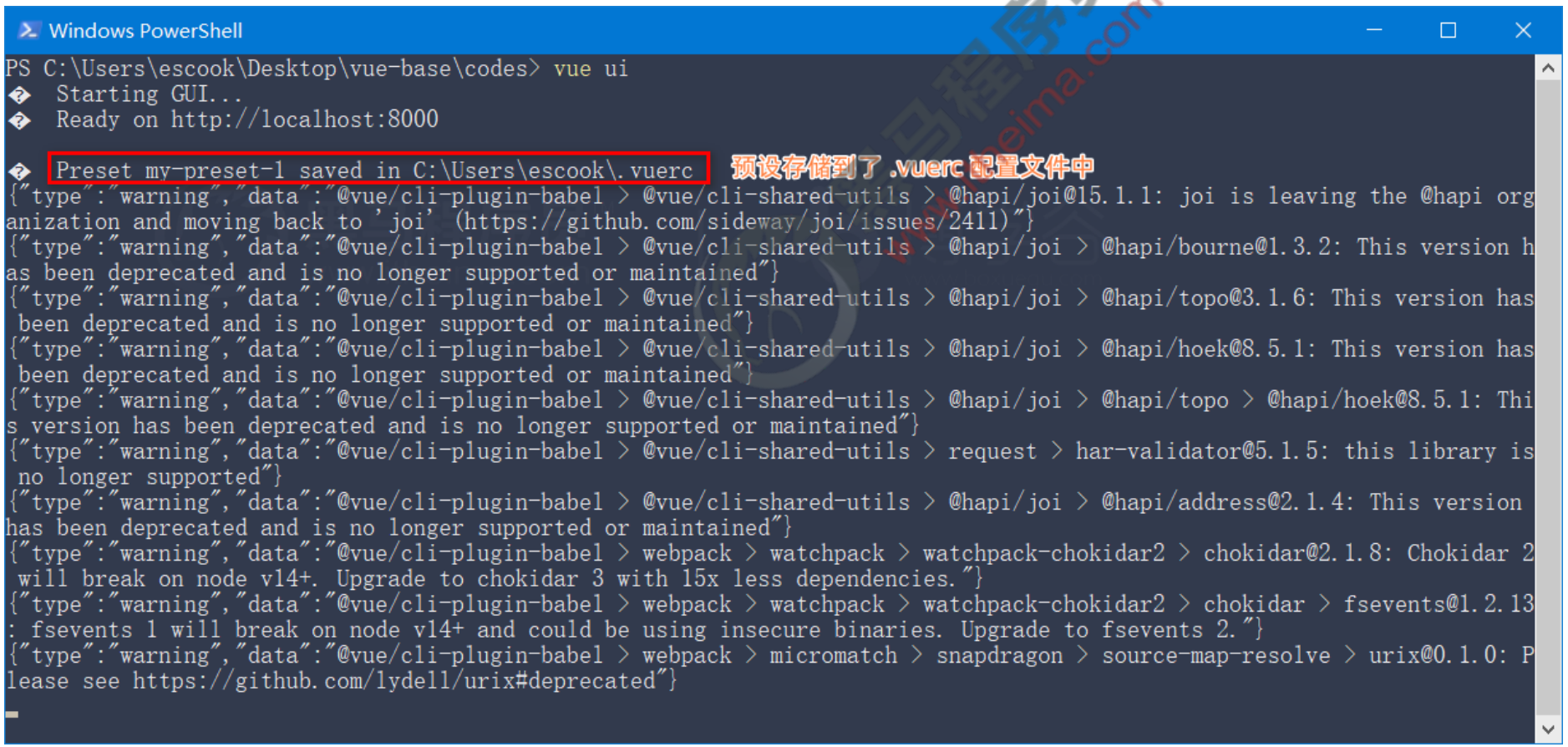
步骤6：将刚才所有的配置保存为预设（模板），方便下一次创建项目时直接复用之前的配置：

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：

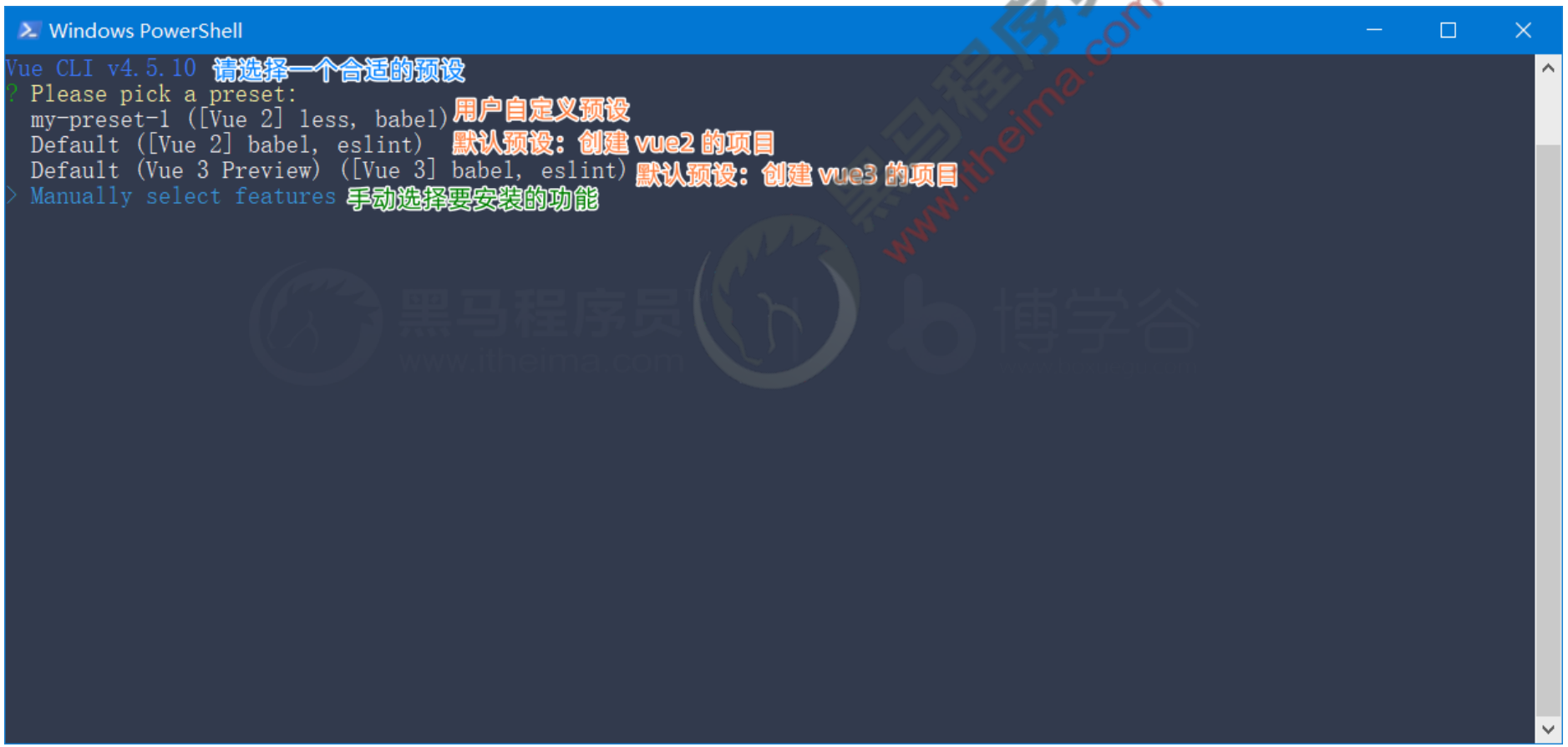


4. 基于 vue ui 创建 vue 项目

项目创建完成后，自动进入项目仪表盘：

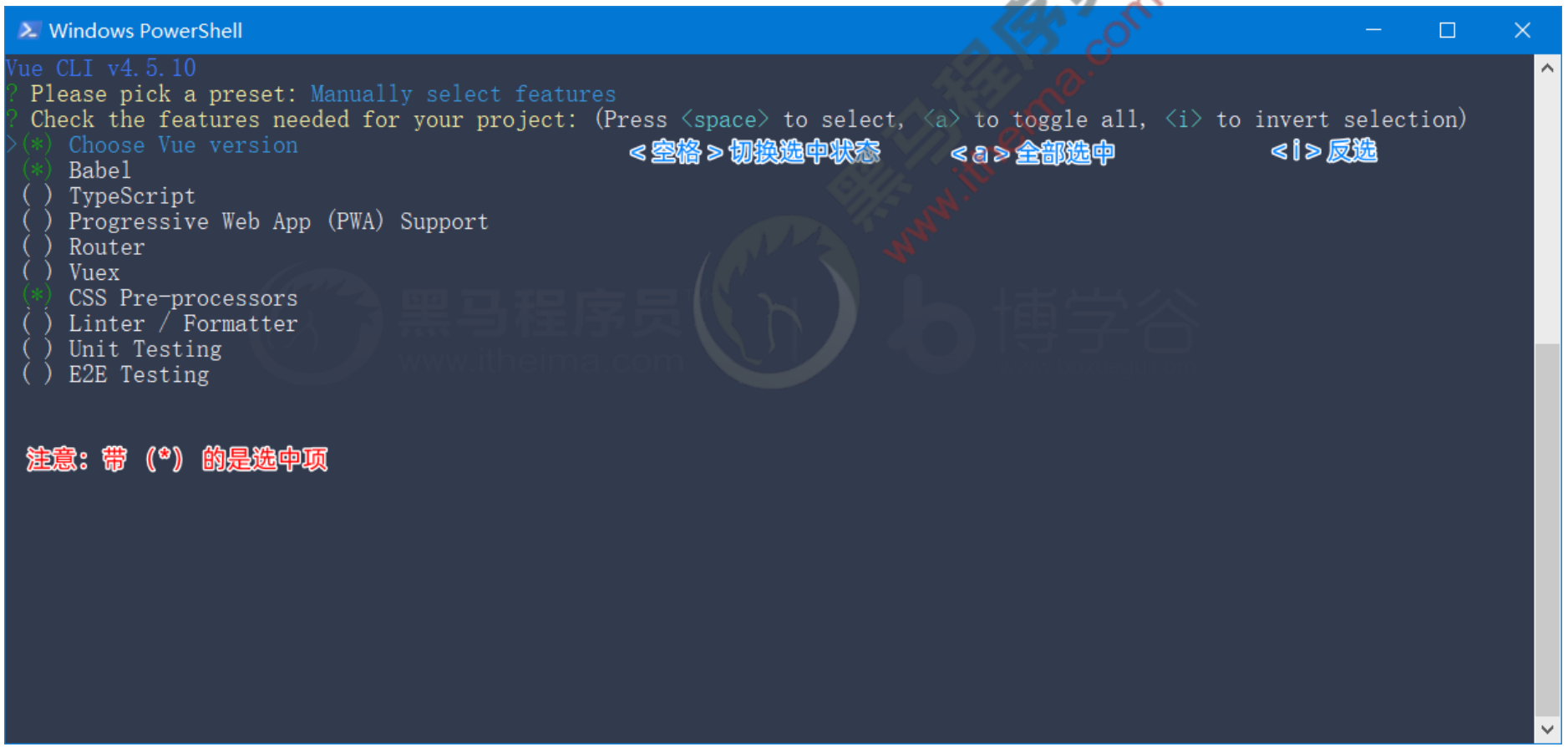
5. 基于命令行创建 vue 项目

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：



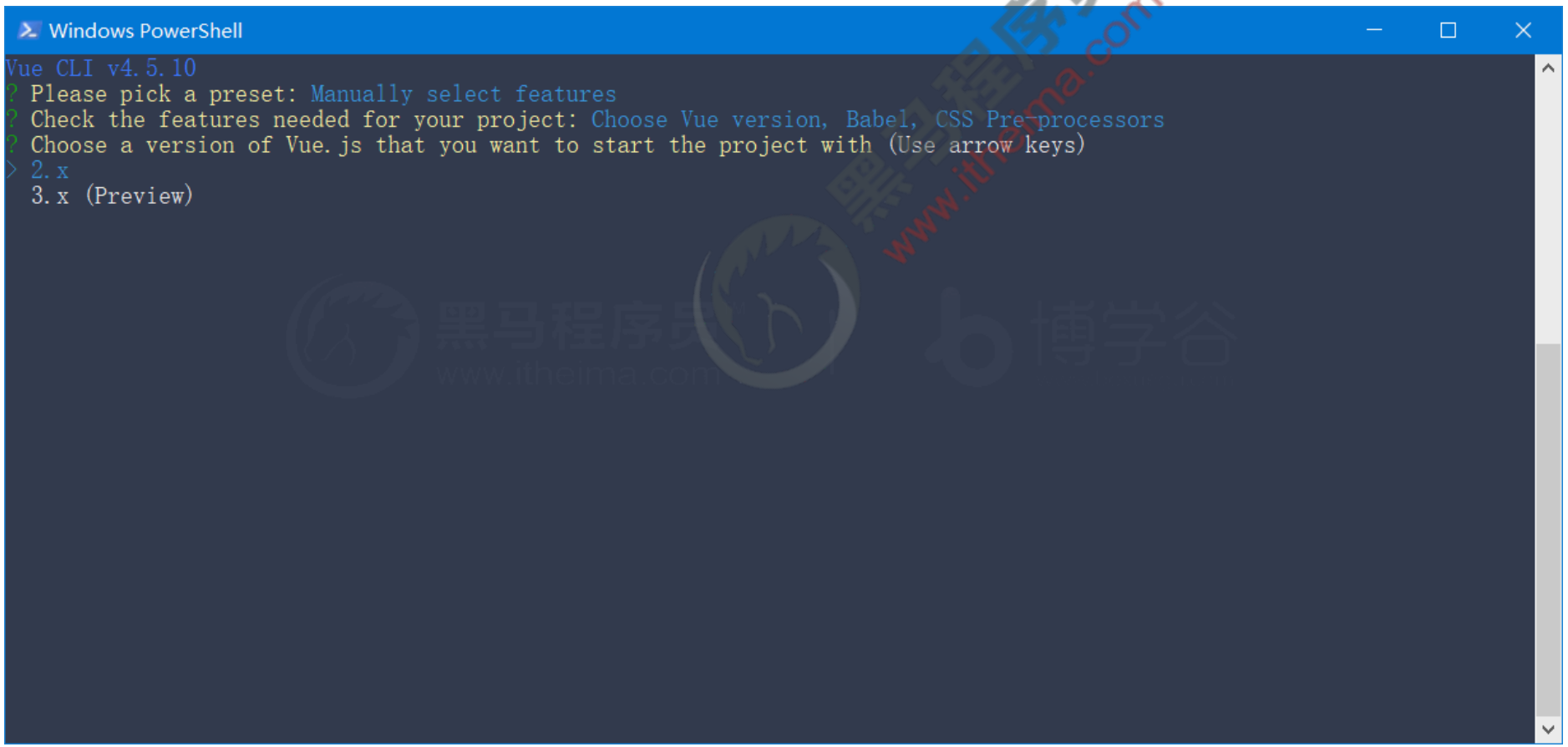
5. 基于命令行创建 vue 项目

步骤2：选择要安装的功能：



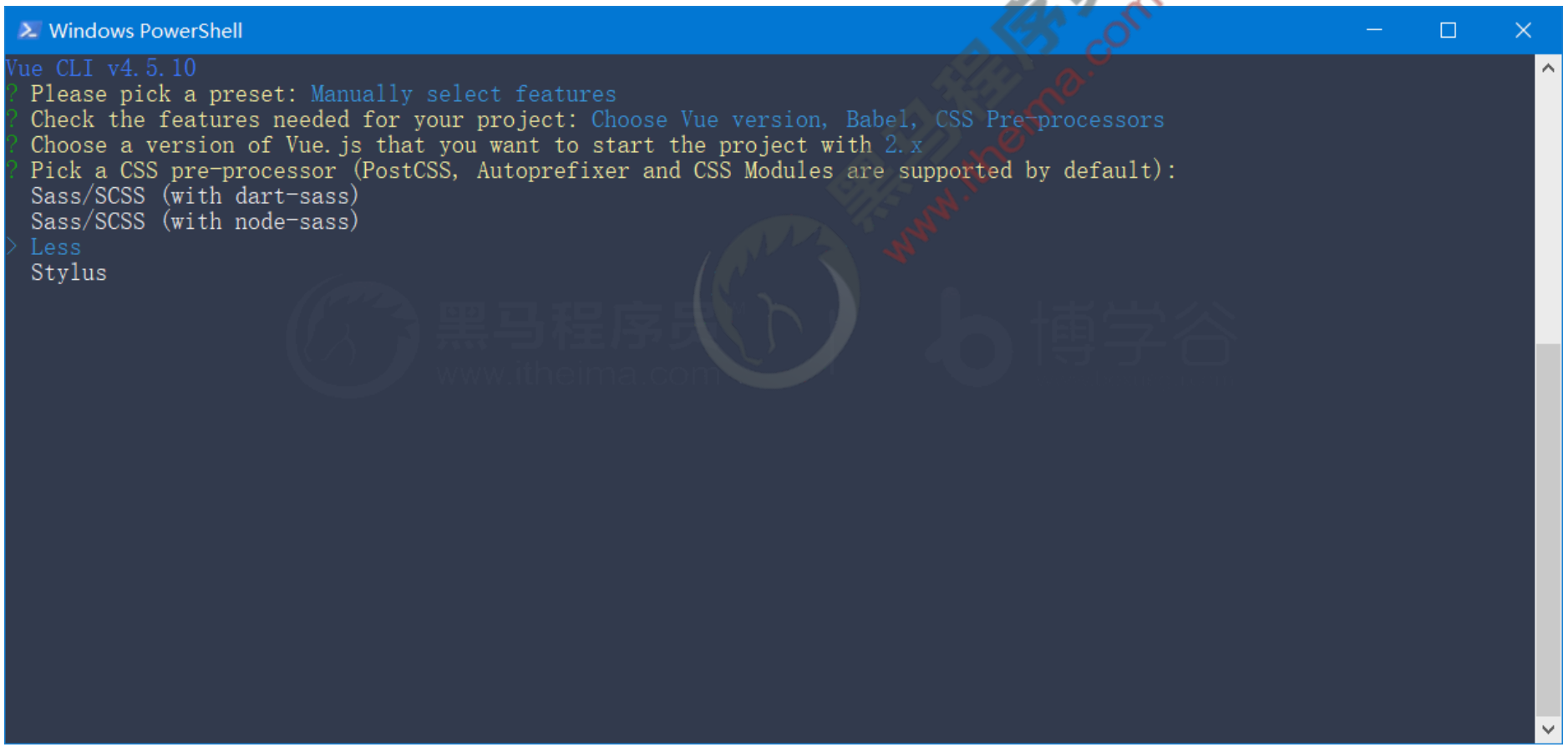
5. 基于命令行创建 vue 项目

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：



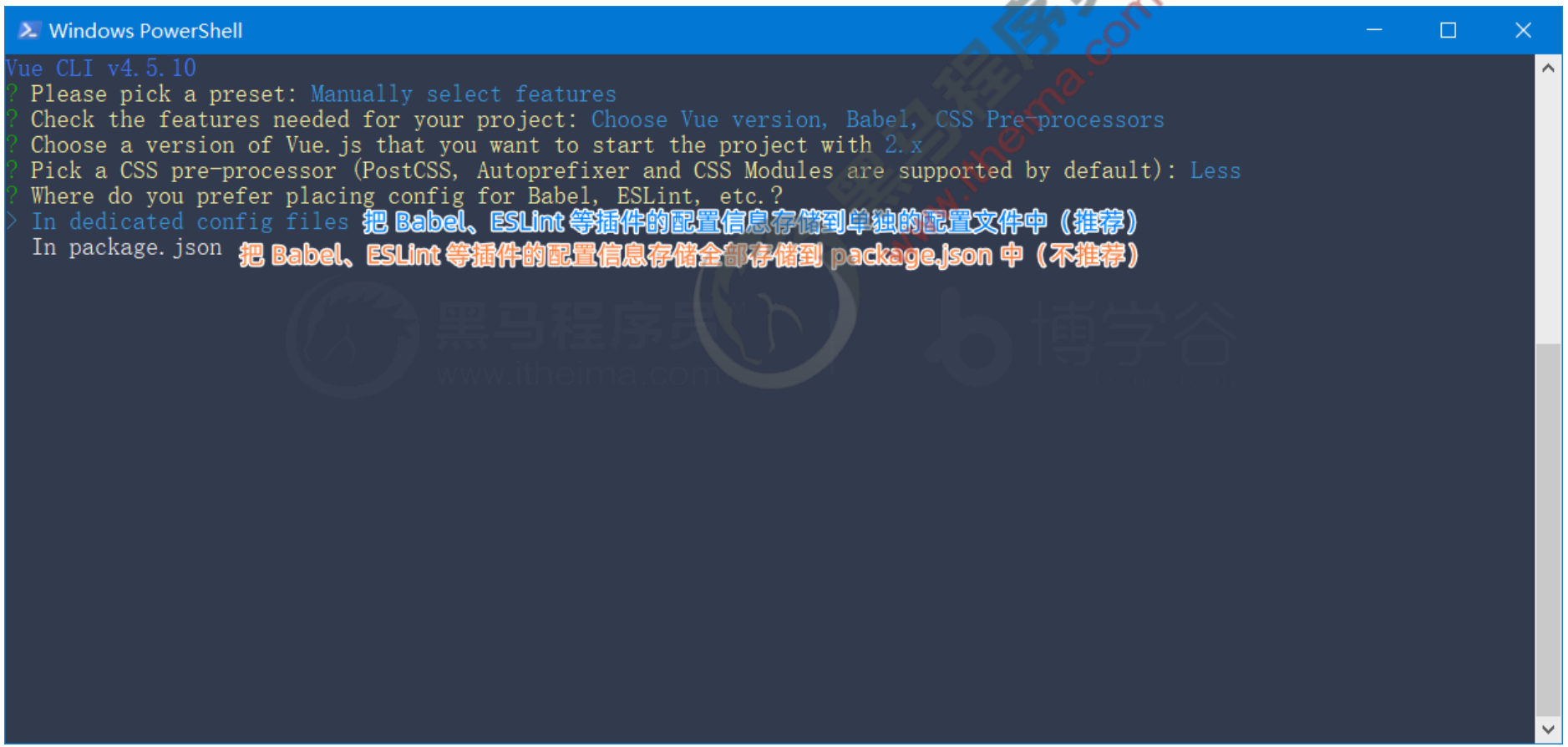
5. 基于命令行创建 vue 项目

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：



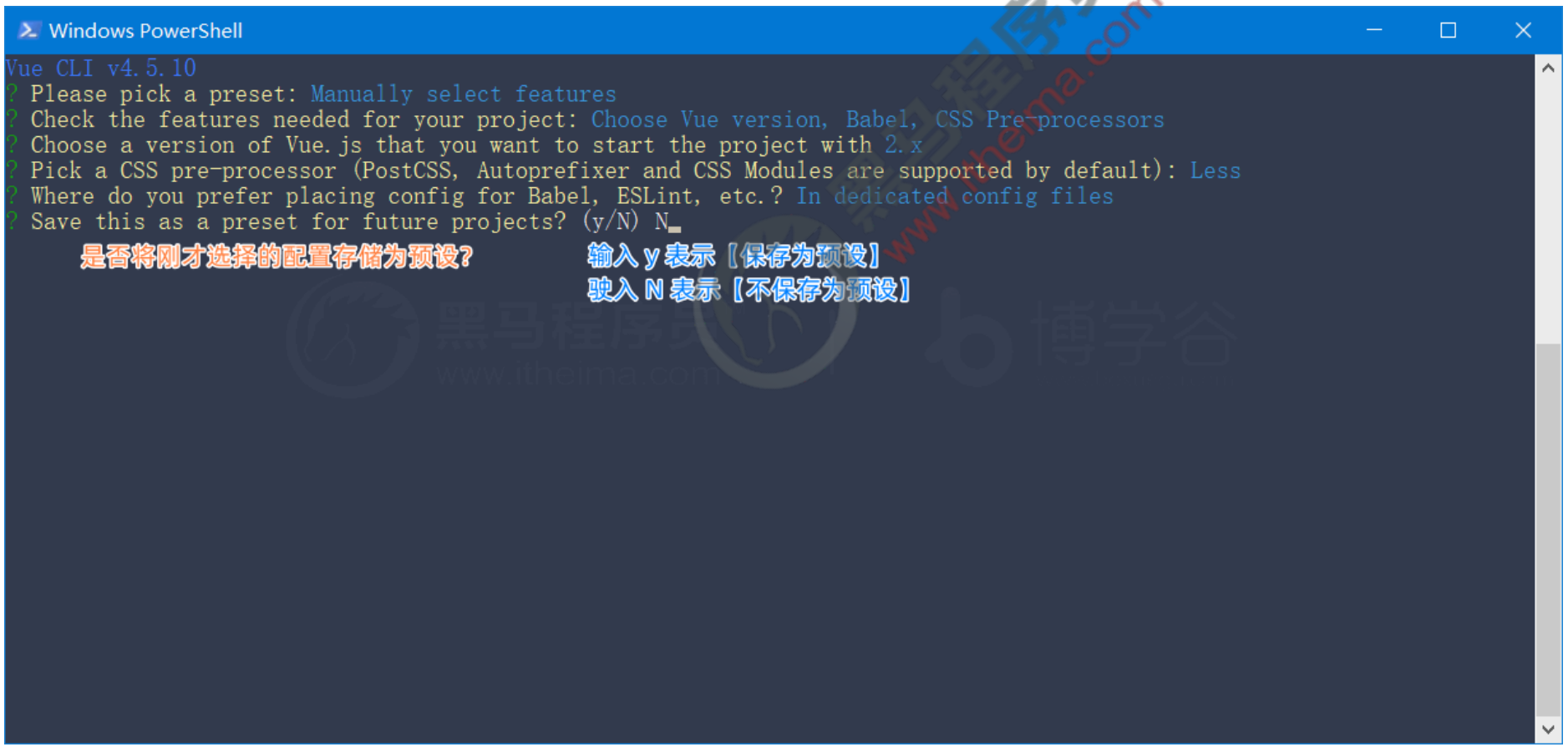
5. 基于命令行创建 vue 项目

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：



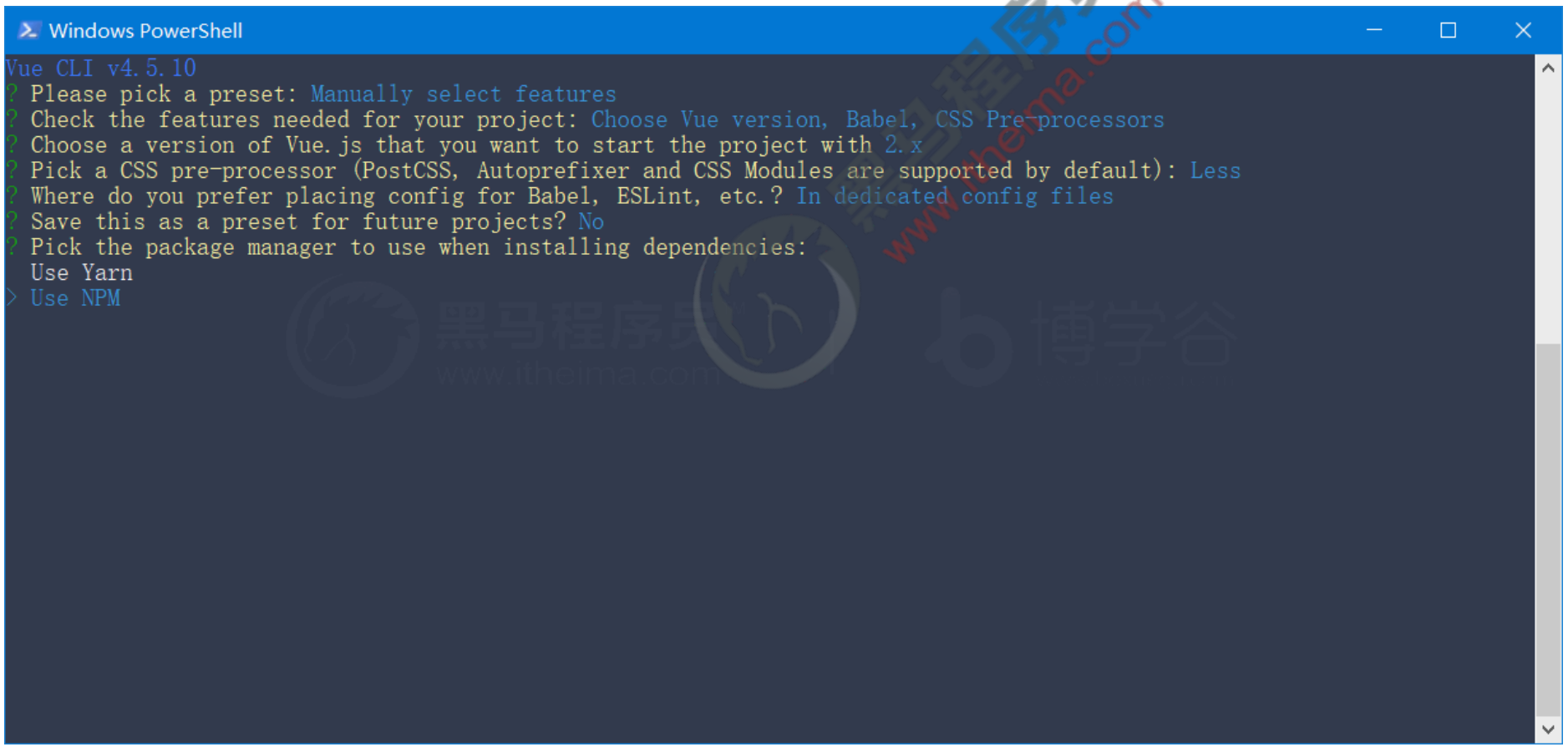
5. 基于命令行创建 vue 项目

步骤6：是否将刚才的配置保存为预设：



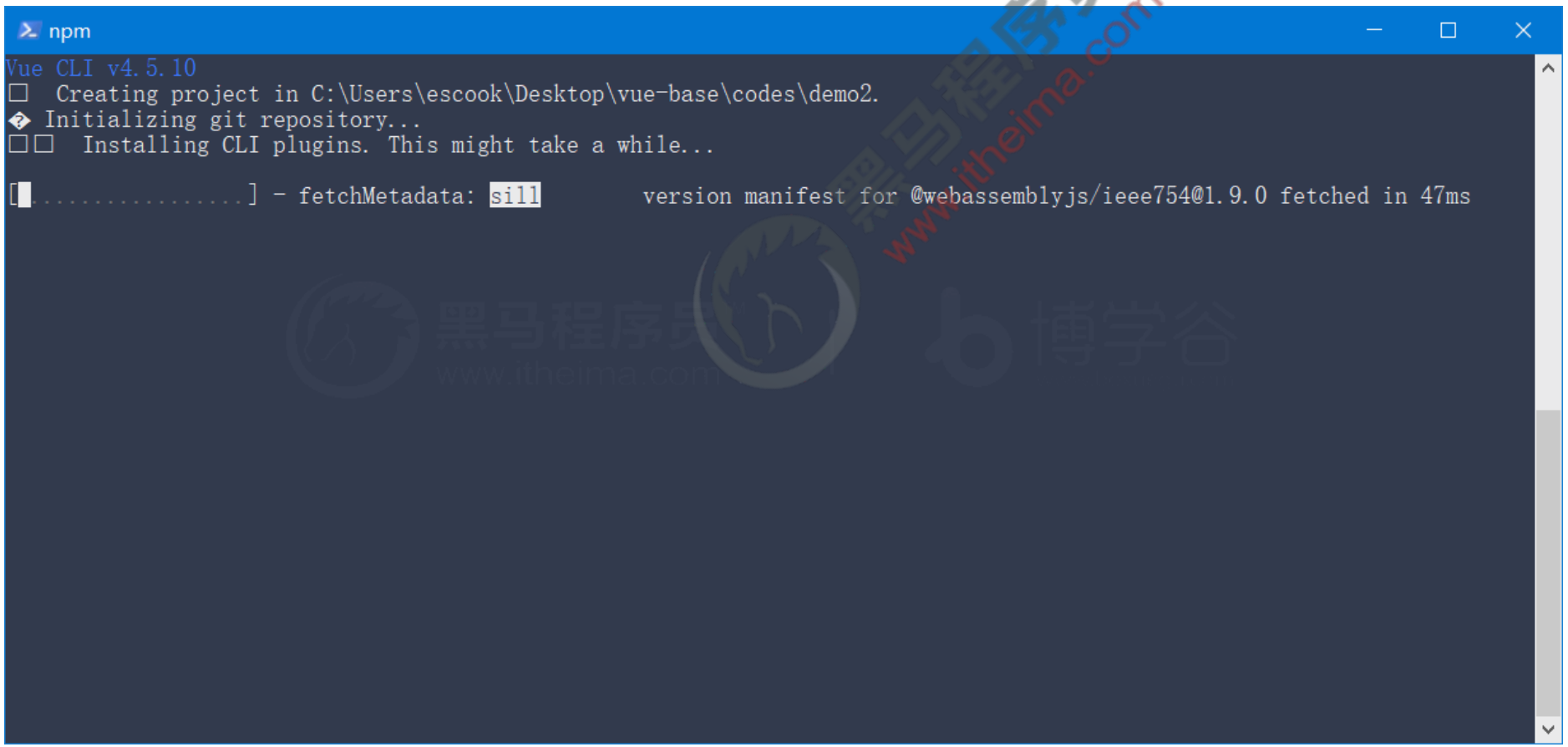
5. 基于命令行创建 vue 项目

步骤7：选择如何安装项目中的依赖包：



5. 基于命令行创建 vue 项目

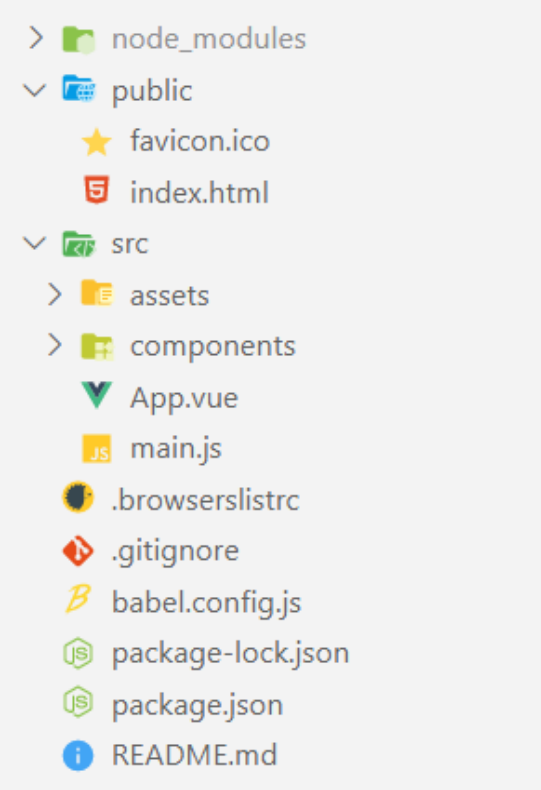
步骤8：开始创建项目并自动安装依赖包：



5. 基于命令行创建 vue 项目

步骤9：项目创建完成：

6. 梳理 vue2 项目的基本结构



主要的文件：

src -> App.vue

src -> main.js

7. 分析 main.js 中的主要代码

```
// 1. 导入 Vue 的构造函数
import Vue from 'vue'
// 2. 导入 App 根组件
import App from './App.vue'

// 屏蔽浏览器 console 面板的提示消息
Vue.config.productionTip = false

// 3. 创建 vue 实例对象
new Vue({
  render: h => h(App), // 3.1 使用 render 函数渲染 App 根组件
}).$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
```

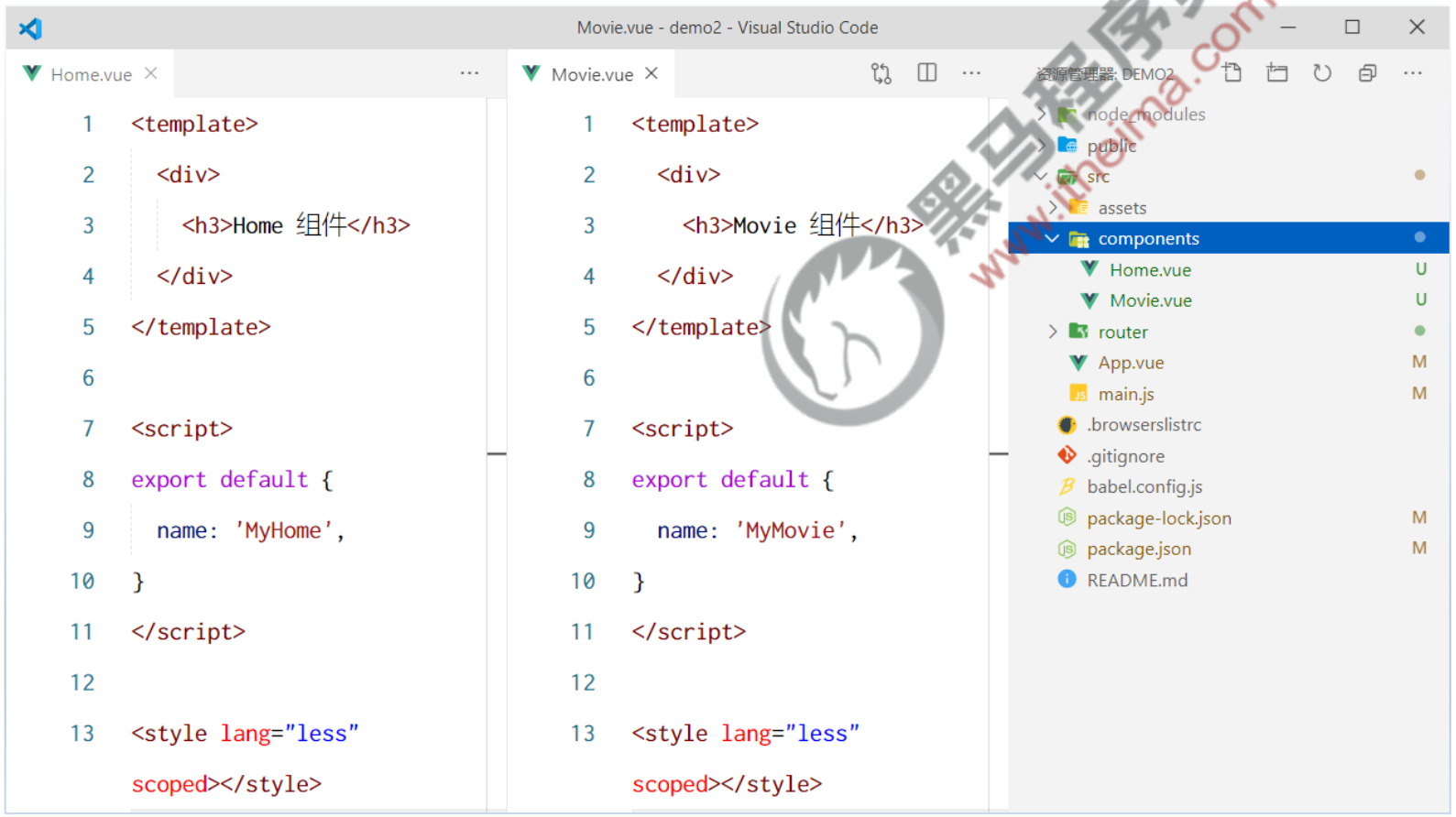
8.2 学习：3.x 版本的路由如何创建路由模块

步骤1：在 vue2 的项目中安装 3.x 版本的路由：

```
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：




```vue
<template>
  <div>
    <h3>Home 组件</h3>
  </div>
</template>

<script>
export default {
  name: 'MyHome',
}
</script>

<style lang="less"
scoped></style>
```

```vue
<template>
  <div>
    <h3>Movie 组件</h3>
  </div>
</template>

<script>
export default {
  name: 'MyMovie',
}
</script>

<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：

```
import Vue from 'vue'                    // 1. 导入 Vue2 的构造函数
import VueRouter from 'vue-router'       // 2. 导入 3.x 路由的构造函数

import Home from '@/components/Home.vue' // 3. 导入需要使用路由切换的组件
import Movie from '@/components/Movie.vue'

Vue.use(VueRouter)                       // 4. 调用 Vue.use() 函数，把路由配置为 Vue 的插件

const router = new VueRouter({           // 5. 创建路由对象
  routes: [                              // 5.1 声明路由规则
    { path: '/', redirect: '/home' },
    { path: '/home', component: Home },
    { path: '/movie', component: Movie },
  ],
})

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：

```
<template>
  <div>
    <h1>App 根组件</h1>
    <hr />
    <!-- 声明路由的占位符 -->
    <router-view></router-view>
  </div>
</template>
```

目录 Contents

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：

```
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：

```
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：

```
module.exports = {
  presets: ['@vue/cli-plugin-babel/preset'],
  // ↓ ↓ ↓
  plugins: [
    [
      'component',
      {
        libraryName: 'element-ui',
        styleLibraryName: 'theme-chalk',
      },
    ],
  ],
  // ↑ ↑ ↑
}
```

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：

```
import Vue from 'vue'
import App from './App.vue'
// 1. 按需导入 element ui 的组件
import { Button } from 'element-ui'

// 2. 注册需要的组件
Vue.component(Button.name, Button)
/* 或写为
 * Vue.use(Button)
 */

new Vue({
  render: h => h(App),
}).$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：

```
// → 模块路径 /src/element-ui/index.js
import Vue from 'vue'
// 按需导入 element ui 的组件
import { Button, Input } from 'element-ui'

// 注册需要的组件
Vue.use(Button)
Vue.use(Input)

// → 在 main.js 中导入
import './element-ui'
```

目录 Contents

axios 拦截器

1. 回顾：在 vue3 的项目中全局配置 axios

```
import { createApp } from 'vue'
import App from './App.vue'
// 1. 导入 axios
import axios from 'axios'

const app = createApp(App)
// 2. 配置请求根路径
axios.defaults.baseURL = 'https://www.escook.cn'
// 3. 全局配置 axios
app.config.globalProperties.$http = axios
app.mount('#app')
```

axios 拦截器

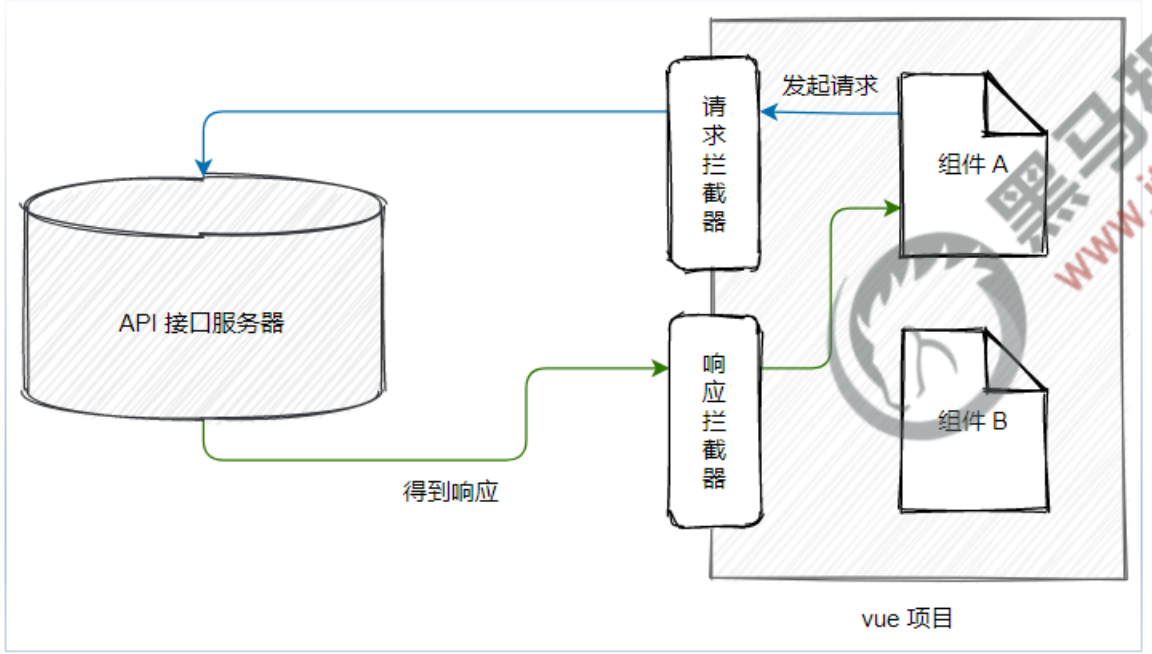
2. 在 vue2 的项目中全局配置 axios

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...

axios 拦截器

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：

```
axios.interceptors.request.use(function (config) {
    // Do something before request is sent
    return config;
  }, function (error) {
    // Do something with request error
    return Promise.reject(error);
})
```

注意：失败的回调函数可以被省略！

axios 拦截器

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. 按需导入 Loading 效果组件
import { Loading } from 'element-ui'

// 2. 声明变量，用来存储 Loading 组件的实例对象
let loadingInstance = null

// 配置请求的拦截器
axios.interceptors.request.use(config => {
  // 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
  loadingInstance = Loading.service({ fullscreen: true })
  return config
})
```

axios 拦截器

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：

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

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：

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

目录 Contents

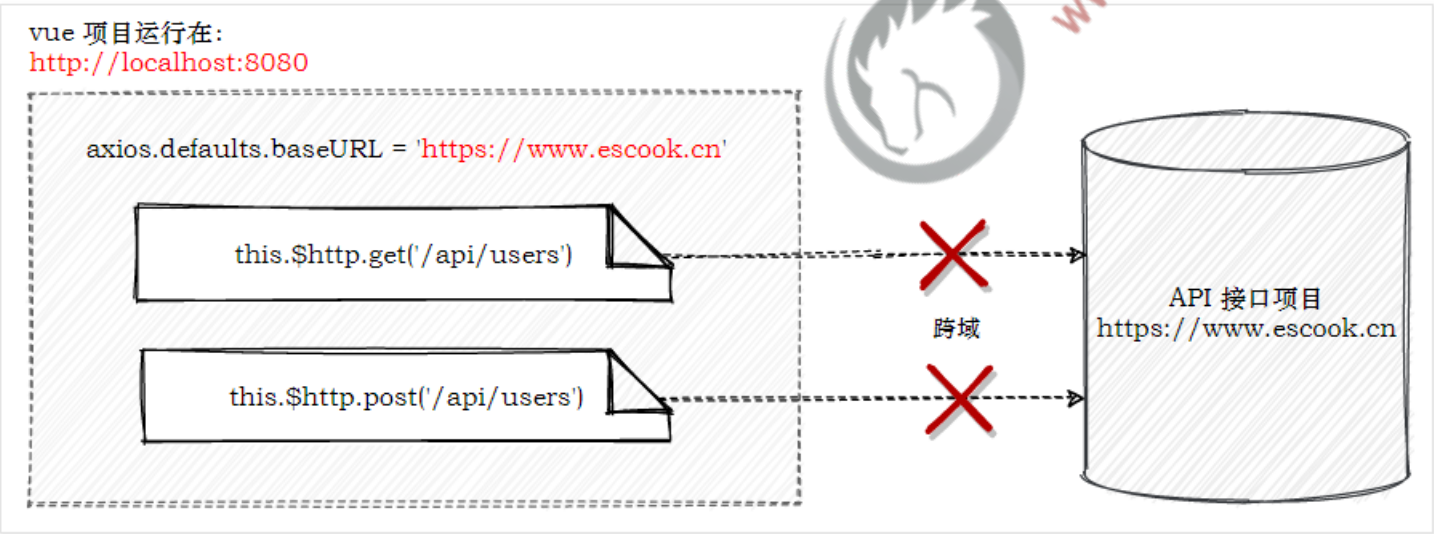
proxy 跨域代理

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

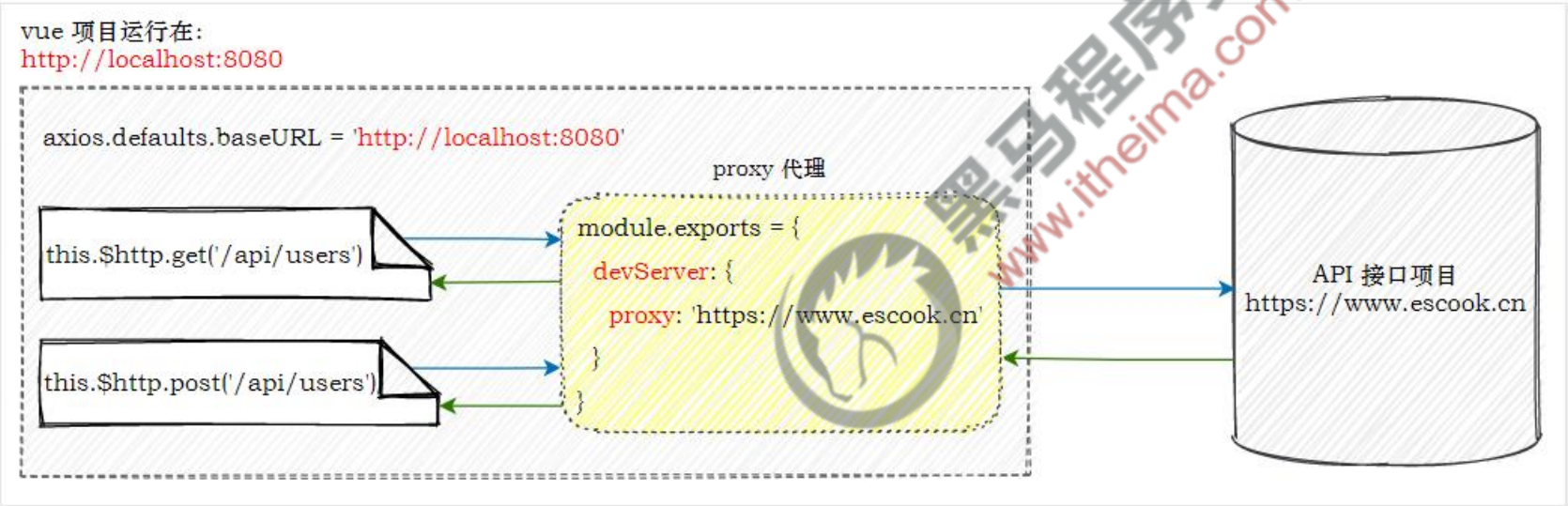
由于当前的 API 接口没有开启 CORS 跨域资源共享，因此默认情况下，上面的接口无法请求成功！



proxy 跨域代理

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

proxy 跨域代理

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：

```
// axios.defaults.baseURL = 'https://www.escook.cn'

// 配置请求根路径
axios.defaults.baseURL = 'http://localhost:8080'
```

proxy 跨域代理

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：

```
module.exports = {
  devServer: {
    // 当前项目在开发调试阶段，
    // 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
    proxy: 'https://www.escook.cn',
  },
}
```

注意：

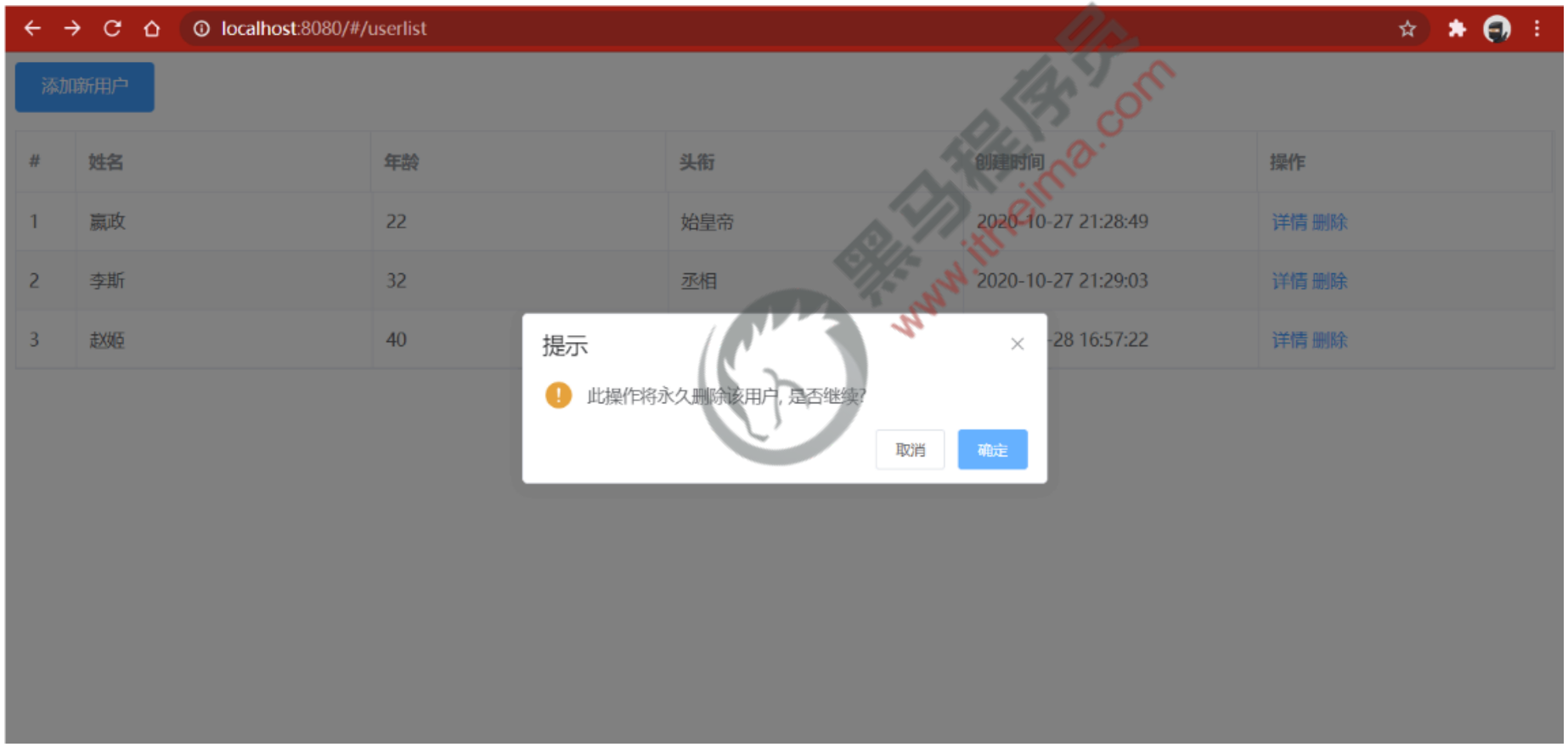
① devServer.proxy 提供的代理功能，仅在开发调试阶段生效

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

目录 Contents

用户列表案例

1. 案例效果



用户列表案例

1. 案例效果

用户列表案例

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
- vuer-router 路由

用户列表案例

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

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