

## 目录

S70 自定义组件介绍 .....	1
1. 启动 Vue 项目管理器 .....	1
2. 创建自定义前端组件项目 .....	1
3. 项目依赖配置 .....	4
4. 安装项目依赖 .....	5
5. 创建自定义组件 .....	5
6. 组件前端调测 .....	6
7. 启动调测 .....	8
8. 编译自定义组件 .....	9
9. 后端项目引用自定义组件 .....	9

# S70 自定义组件介绍

## 1. 启动 Vue 项目管理器

CMD 中输入 `vue ui` 启动 Vue 项目管理器



## 2. 创建自定义前端组件项目

^

E:

github

S70

✎

↺

☆

▼

⋮

📁

.git

📁

.vs

📁

Toowell.Common

📁

Toowell.Factory

📁

Toowell.Manage

📁

Toowell.S70

📁

Toowell.S70.Apps

📁

Toowell.S70.Generator

📁

Toowell.S70.Migrations

📁

Toowell.S70.Models

📁

toowell.s70.comp ▼

📁

toowell.s70.designer ▼

+

在此创建新项目

## 创建新项目

详情

预设

功能

配置

项目文件夹

toowell.s70.cust

E:/github/S70/toowell.s70.cust

包管理器

npm

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)

取消

下一步 →

创建新项目

详情 预设 功能 配置

预设就是一套定义好的插件和配置。你也可以将自己的配置保存成预设，方便以后创建项目使用。

选择一套预设

☐ 默认  
[Vue 2] babel, eslint

☒ Default preset (Vue 3 preview)  
[Vue 3] babel, eslint

☐ 手动  
手动配置项目

☐ 远程预设  
从 git 仓库拉取预设

← 上一步

✓ 创建项目

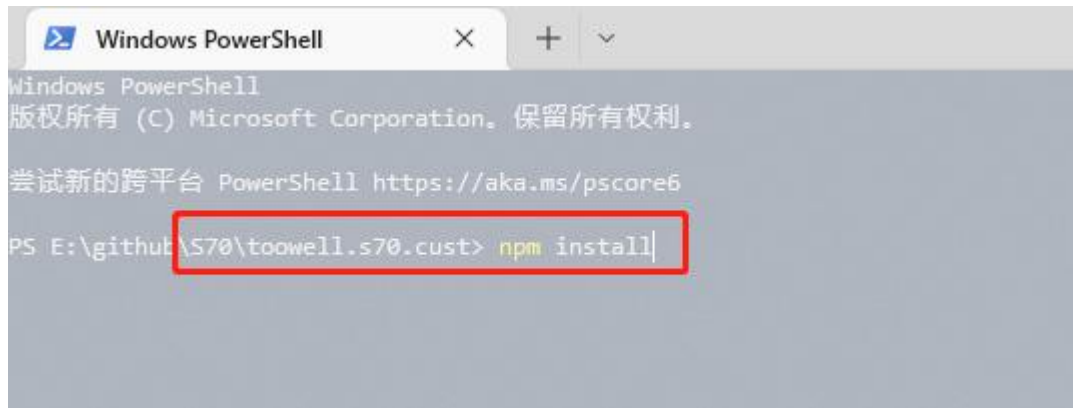
### 3. 项目依赖配置

修改 package.json，增加 lib 编译配置及依赖保持和 S70Comp 项目一致，配置如下

```
package.json X
toowell.s70.cust > package.json > {} devDependencies > eslint-plugin-vue
1 {
2   "name": "toowell.s70.cust",
3   "version": "0.1.0",
4   "private": true,
5   "scripts": {
6     "serve": "vue-cli-service serve",
7     "build": "vue-cli-service build",
8     "lint": "vue-cli-service lint",
9     "lib": "vue-cli-service build --target lib --name s70Cust --dest ../Toowell.S70/wwwroot/js/cust src/components/index.js"
10  },
11  "dependencies": {
12    "axios": "^0.21.1",
13    "core-js": "3.9.1",
14    "element-plus": "1.0.2-beta.69",
15    "vue": "3.0.9"
16  },
17  "devDependencies": {
18    "@vue/cli-plugin-babel": "4.5.0",
19    "@vue/cli-plugin-eslint": "4.5.0",
20    "@vue/cli-service": "4.5.0",
21    "@vue/compiler-sfc": "3.0.0",
22    "babel-eslint": "10.1.0",
23    "eslint": "6.7.2",
24    "eslint-plugin-vue": "7.0.0"
25  },
26  "eslintConfig": {
27    "root": true,
28    "env": {
29      "node": true
30    },
31    "extends": [
32      "plugin:vue/vue3-essential",
33      "eslint:recommended"
34    ],
35    "parserOptions": {
36      "parser": "babel-eslint"
37    },
38    "rules": {}
39  },
}
```

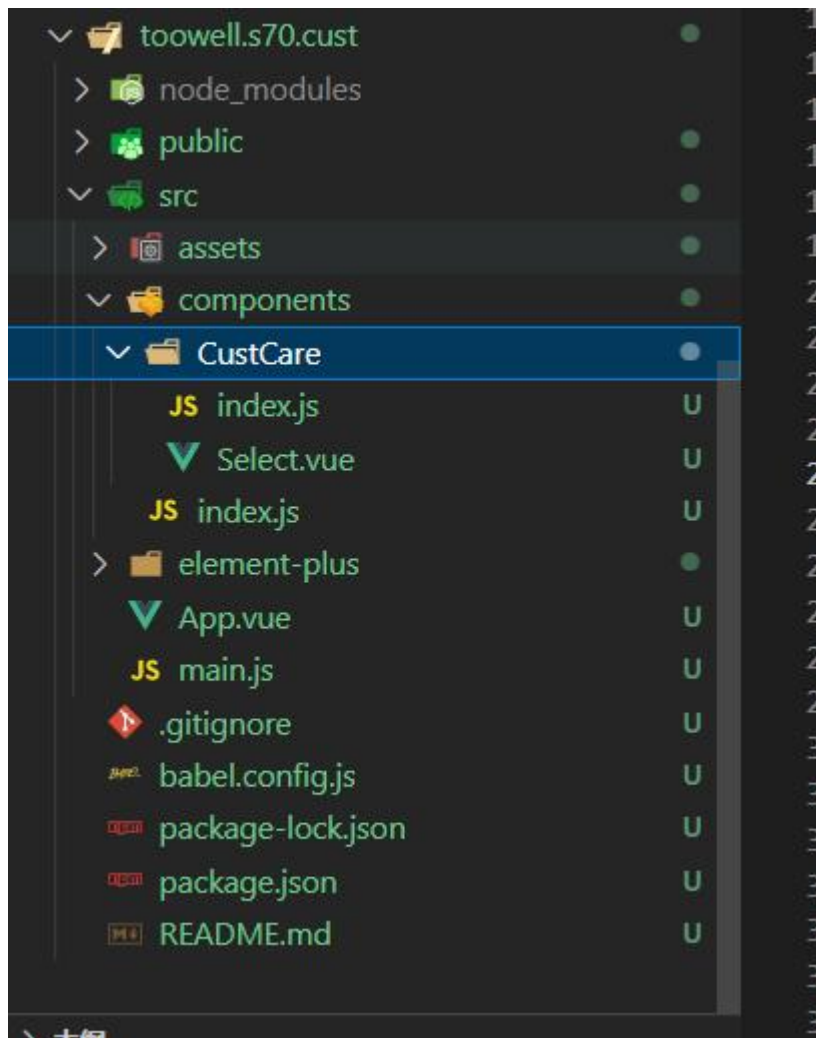
#### 4. 安装项目依赖

CMD 进入项目目录，执行 `npm install` 安装项目依赖



#### 5. 创建自定义组件

使用 VS Code 打开自定义组件项目创建自定义组件



在 `components` 目录中创建 `index.js` 和 `CustCare` 目录，在 `CustCare` 目录创建 `.vue` 和 `index.js` 文件：

`index.js` 为组件注册和导出方法，参考

<https://v3.cn.vuejs.org/guide/component-registration.html>

.vue 文件为单文件组件， 参考

<https://v3.cn.vuejs.org/guide/single-file-component.html#%E4%BB%8B%E7%BB%8D>

CustCare 中 index.js 为编写插件公开方法， 参考

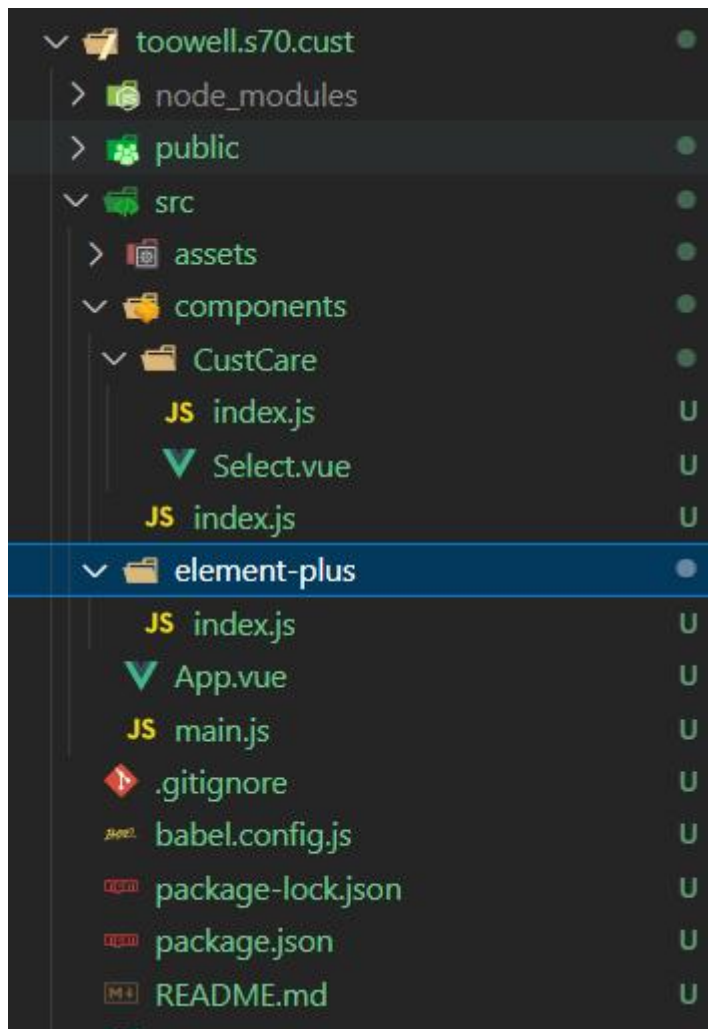
<https://v3.cn.vuejs.org/guide/plugins.html#%E7%BC%96%E5%86%99%E6%8F%92%E4%BB%B6>

Element-plus js 版文档示例， 参考

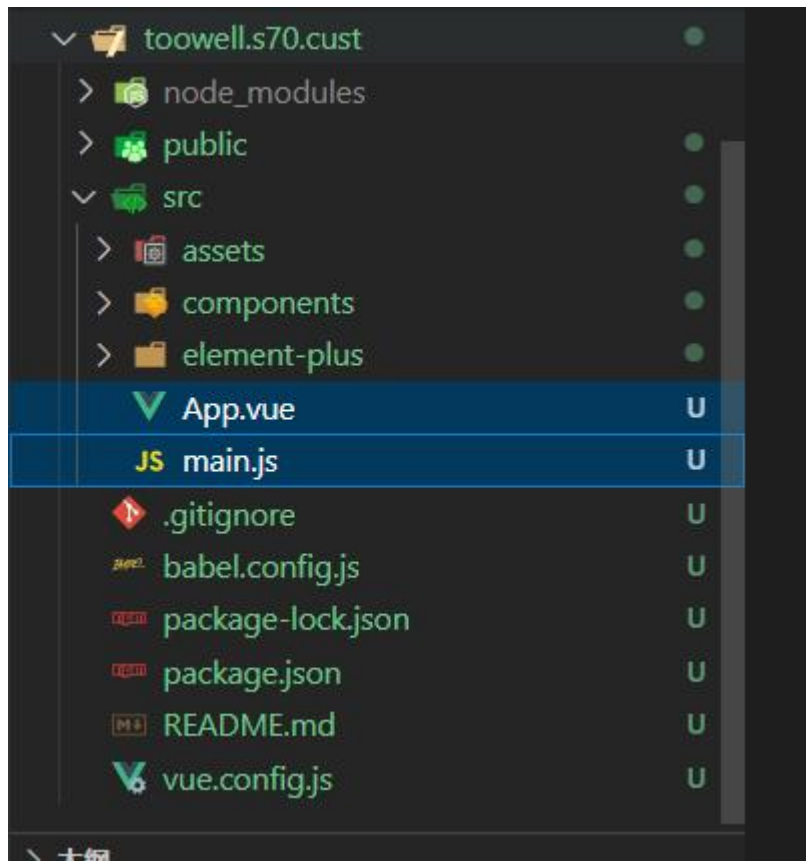
<https://github.com/element-plus/element-plus/tree/1.0.2-beta.69/website/docs/zh-CN>

## 6. 组件前端调测

创建 element-plus 目录和 index.js， index.js 为按需引入 element-plus 组件内容



修改 App.vue,main.js 文件测试组件



```
package.json U App.vue U X
toowell.s70.cust > src > App.vue > {} "App.vue" > script > default > data > custcare
1 <template>
2 <mx-cust-care-select v-model="custcare" placeholder="请选择"></mx-cust-care-select>
3 </template>
4
5 <script>
6 export default {
7   name: "App",
8
9   data: function () {
10     return {
11       custcare: "",
12     };
13   },
14 };
15 </script>
16
17 <style>
18 #app {
19   font-family: Avenir, Helvetica, Arial, sans-serif;
20   -webkit-font-smoothing: antialiased;
21   -moz-osx-font-smoothing: grayscale;
22   text-align: center;
23   color: #2c3e50;
24   margin-top: 60px;
25 }
26 </style>
27
```

```
App.vue U JS main.js U X
toowell.s70.cust > src > JS main.js > ...
1 import {
2   |   createApp
3   | } from 'vue'
4   | import ElementPlus from '@element-plus'
5   | import 'element-plus/lib/theme-chalk/index.css'
6   | import S70Cust from '@components'
7   | import App from './App.vue'
8
9  /* 前后端联调时需要token认证
10  S70Cust.axios.post(`http://localhost:5000/api/v1/token`, {
11    |   "UserName": "admin",
12    |   "Password": "123456"
13  }).then((res) => {
14    |   let token = res.data.token;
15    |   console.log(token);
16    |   const app = createApp(App)
17    |   app.use(ElementPlus)
18    |   app.use(S70Cust, {
19    |     |   baseUrl: "http://localhost:5000/api/comp/data/",
20    |     |   token: token
21    |   })
22    |   app.mount('#app')
23  })
24  */
25
26  const app = createApp(App)
27  app.use(ElementPlus)
28  app.use(S70Cust, {})
29  app.mount('#app')
```

## 7. 启动调测





## 8. 编译自定义组件

package.json 的 lib 指定组件输出目录

```
{
  "name": "toowell.s70.cust",
  "version": "0.1.0",
  "private": true,
  "scripts": {
    "serve": "vue-cli-service serve",
    "build": "vue-cli-service build",
    "lint": "vue-cli-service lint",
    "lib": "vue-cli-service build --target lib --name s70Cust --dest ../Toowell.S70/wwwroot/js/cust src/components/index.js"
  },
  "devDependencies": {}
}
```

src 目录增加 vue.config.js 文件，内容如下：

```
package.json U App.vue U vue.config.js U X
toowell.s70.cust > vue.config.js > ...
1 function getProdExternals() {
2   return {
3     // axios: "axios",
4     vue: "Vue"
5   };
6 }
7 module.exports = {
8   productionSourceMap: false,
9   configureWebpack: {
10    externals: process.env.NODE_ENV === 'production' ?
11      getProdExternals() : {}
12  },
13};
```

The screenshot shows the Vue CLI interface with the 'lib' task highlighted in the left sidebar. The main panel displays the build output for the 'lib' target, showing the compilation of the 's70Cust' library. The output includes a table of files and their sizes, and a summary of the build process.

File	Size	Gzipped
..\Toowell.S70\wwwroot\js\cust\s70Cust.umd.min.js	3.19 KiB	1.46 KiB
..\Toowell.S70\wwwroot\js\cust\s70Cust.umd.js	10.15 KiB	3.14 KiB
..\Toowell.S70\wwwroot\js\cust\s70Cust.common.js	9.57 KiB	2.95 KiB

Total task duration: 6.16s  
\$ vue-cli-service build --target lib --name s70Cust --dest ../Toowell.S70/wwwroot/js/cust src/components/index.js

- Building for production as library (commonjs,umd,umd-min)...

Compiled successfully in 238ms at 10:43 PM

Compiled successfully in 361ms at 10:44 PM

Compiled successfully in 405ms at 10:44 PM

## 9. 后端项目引用自定义组件

```
MainLogin.cshhtml  X
10 <link rel="stylesheet" href="/css/login.css" />
11 <script src="/lib/vue.global.prod.js" type="text/javascript"></script>
12 <script src="/lib/element-plus/index.full.js" type="text/javascript"></script>
13 <script src="/js/und/s70CustComponent.umd.min.js?v=1.0" type="text/javascript"></script>
14 <script src="/js/cust/s70Cust.umd.min.js?v=1.0" type="text/javascript"></script>
15 <script>
16   const { defineAsyncComponent, createApp, h, render } = Vue;
17 </script>
18 </head>
19 <body>
20   <div id="login">
21     <div class="login-wrap">
22       <div class="ms-login">
23         <div class="ms-title">登陆</div>
24         <el-form :model="param" :rules="rules" ref="login" label-width="0px" class="ms-content">
25           <el-form-item>...</el-form-item>
26           <el-form-item>...</el-form-item>
27           <el-form-item>...</el-form-item>
28           <el-form-item>...</el-form-item>
29           <mx-cust-care-select v-model="custcare" placeholder="请选择"></mx-cust-care-select>
30           </el-form-item>
31           <div class="login-btn">...</div>
32           <p class="login-tips">{{errMsg}}</p>
33         </el-form>
34       </div>
35     </div>
36   </div>
37
38   <script>
39     const myApp = {
40       data() {
41         return {
42           errMsg: '',
43           param: {},
44           rules: {
45             custcare: ''
46           }
47         }
48       },
49       methods: {},
50       mounted: function () {}
51     }
52
53     let app = createApp(myApp)
54     app.use(ElementPlus)
55     app.use(s70Cust, {})
56     app.mount('#login')
57   </script>
58 </body>
59 </html>
```