

👉 VUE入门手册 👉

一、安装

VUE框架的安装主要有两种方式:

1. 一种是将VUE当作普通的JS插件使用, 直接用 `<script>` 引入;

```
<script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
```

这样的引用方法可以很方便的使用vue的基本功能, 缺点是没法在IE浏览器上预览, 如果对自己CSS兼容写法有信心的同学可以用这个方法; 😊

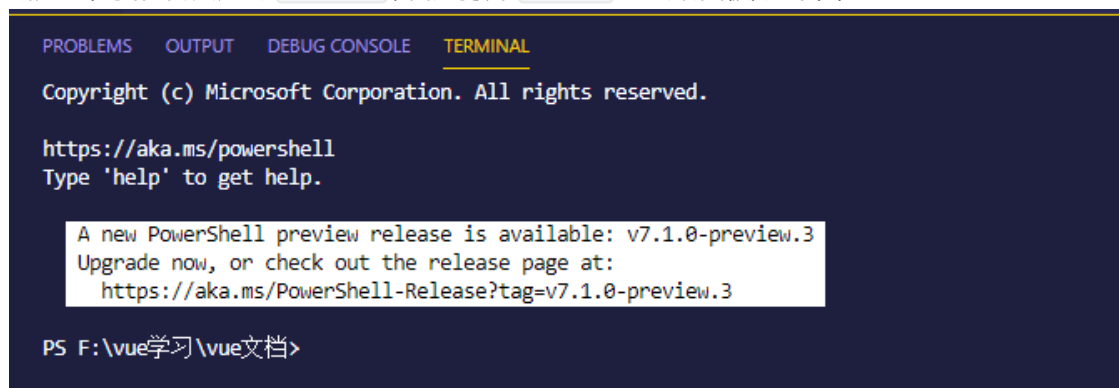
2. 下面这种是我比较推荐的使用VUE框架的方法, **NPM 安装**;

- 在使用 **NPM** 安装VUE之前, 我们首先需要安装 [Node.js](#), 直接按照默认的设置一路安装完就行. 安装完后需要检查下是否正确安装 [Node.js](#), 打开 **Powershell** 输入 `node -v`, 如果输出版本号的话, 就说明安装成功了;
- VUE开发我推荐使用 [VS CODE](#), 大家自个下载下并且安装好, 安装好后按 `CTRL + K + CTRL + O` 打开一个文件夹, 然后按 `CTRL + ~` 打开终端面板, 以后凡是有关 **NPM** 的命令都可以直接在这里输入;
- 安装VUE脚手架 -- [Vue CLI](#), 很简单, 在终端面板里输入 `npm install -g @vue/cli` 即可;

上面三步完成后, 准备工作就做好了, 下面我们来讲讲如何新建一个工程. 📁

二、新建工程

1. 新建工程我们需要用到 **VUE CLI**, 首先打开 **VSCODE** 的终端面板, 如下图



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Copyright (c) Microsoft Corporation. All rights reserved.

https://aka.ms/powershell
Type 'help' to get help.

A new PowerShell preview release is available: v7.1.0-preview.3
Upgrade now, or check out the release page at:
https://aka.ms/PowerShell-Release?tag=v7.1.0-preview.3

PS F:\vue学习\vue文档>
```

然后输入

```
vue create hello-world
```

这样就会新建一个名叫 `hello-world` 的工程, 接下来我们要设置这个工程的各项参数和插件, 之后终端面板会出现如下信息

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: node

Vue CLI v4.3.1
? Please pick a preset: (Use arrow keys)
> normal_test (dart-sass, babel, router, vuex, eslint)
  default (babel, eslint)
  Manually select features
```

意思是让我们选择一个预设, 我建议大家第一次选择 **Manually select features**, 之后如下

```
PROBLEMS OUTPUT TERMINAL ... 1: node

Vue CLI v4.3.1
? Please pick a preset: Manually select features
? Check the features needed for your project: (Press <space> to select, <a> to toggle
election)
>(*) Babel
election)
>(*) Babel
  ( ) TypeScript
  ( ) Progressive Web App (PWA) Support
  ( ) Router
  ( ) Vuex
  ( ) CSS Pre-processors
  (*) Linter / Formatter
  ( ) Unit Testing
  ( ) E2E Testing
```

大家选择下面4个就行,然后按 `Enter` 确定即可;

```
(*) Babel
( ) TypeScript
( ) Progressive Web App (PWA) Support
(*) Router
>(*) Vuex
( ) CSS Pre-processors
(*) Linter / Formatter
( ) Unit Testing
( ) E2E Testing
```

之后又会有几个选项, 分别按照下面我的截图输入或选择,

```
PROBLEMS OUTPUT TERMINAL ... 1: node

Vue CLI v4.3.1
? Please pick a preset: Manually select features
? Check the features needed for your project: Babel, Router, Vuex, Linter
? Use history mode for router? (Requires proper server setup for index fallback in produc
tion) (Y/n) Y
```

```
PROBLEMS OUTPUT TERMINAL ... 1: node

Vue CLI v4.3.1
? Please pick a preset: Manually select features
? Check the features needed for your project: Babel, Router, Vuex, Linter
? Use history mode for router? (Requires proper server setup for index fallback in production) Yes
? Pick a linter / formatter config: (Use arrow keys)
> ESLint with error prevention only
  ESLint + Airbnb config
  ESLint + Standard config
  ESLint + Prettier
```

```
PROBLEMS OUTPUT TERMINAL ... 1: node

Vue CLI v4.3.1
? Please pick a preset: Manually select features
? Check the features needed for your project: Babel, Router, Vuex, Linter
? Use history mode for router? (Requires proper server setup for index fallback in production) Yes
? Pick a linter / formatter config: Basic
? Pick additional lint features: (Press <space> to select, <a> to toggle all, <i> to invert selection)
>(*) Lint on save
  ( ) Lint and fix on commit
```

```
PROBLEMS OUTPUT TERMINAL ... 1: node

Vue CLI v4.3.1
? Please pick a preset: Manually select features
? Check the features needed for your project: Babel, Router, Vuex, Linter
? Use history mode for router? (Requires proper server setup for index fallback in production) Yes
? Pick a linter / formatter config: Basic
? Pick additional lint features: Lint on save
? Where do you prefer placing config for Babel, ESLint, etc.? (Use arrow keys)
> In dedicated config files
  In package.json
```

具体每个选项是什么意思和作用, 我这里就不多赘述了, 感兴趣的自己上 [VUE CLI](#) 上看文档, 里面有很详细的介绍.

```
PROBLEMS OUTPUT TERMINAL ... 1: node

Vue CLI v4.3.1
? Please pick a preset: Manually select features
? Check the features needed for your project: Babel, Router, Vuex, Linter
? Use history mode for router? (Requires proper server setup for index fallback in production) Yes
? Pick a linter / formatter config: Basic
? Pick additional lint features: Lint on save
? Where do you prefer placing config for Babel, ESLint, etc.? In dedicated config files
? Save this as a preset for future projects? (y/N)
```

最后一步让我们是否保存这一次新建工程的操作为自定义预设, 大家自己选吧, 图方便的可以选择 `Yes`, 按下 `enter` 后, 接下来就是漫长的安装环节;

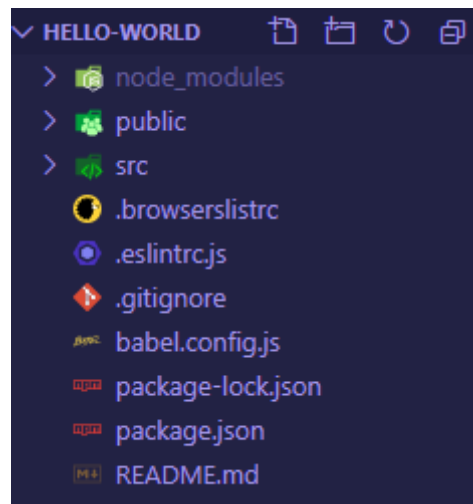
```
PROBLEMS OUTPUT TERMINAL ... 1: node
Vue CLI v4.3.1
🌟 Creating project in F:\vue学习\hellow-world.
📁 Initializing git repository...
⚙️ Installing CLI plugins. This might take a while...

[ ] ..... ] - fetchMetadata: sill pacote range manifest for webpack@^4.0.0 fetch
```

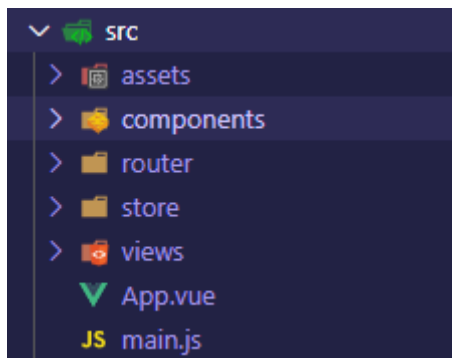
新建好工程后, 我们 `CTRL + K + CTRL + O` 打开新建好的工程文件夹, 现在我们来简单讲下各个文件的作用和如何在浏览器上预览新建的工程;

三、在本机上运行工程

1. 在运行工程之前, 我们先来讲讲新建好的文件结构;



- `node_modules` 是用来存放vue.js和其他一些插件文件的地方, 所有通过 `NPM` 安装的文件都会被存放到这里, 大家如果不是需要魔改插件的话基本上不需要去了解他;
- `public` 是用来存放 `ico` 图标和 `index.html` 的地方;
- `src` 是一个比较重要的文件夹, 是我们切图主要工作的地方, `assets` 文件夹用来存放css和图片文件, `components` 则是存放组件的地方, `router` 用来放置记录路由地址的js文件, `store` 用来放置记录状态管理的js文件, `views` 这个文件夹也是用来存放组件的地方, 实际项目中基本上用不到他, 大家可以酌情删掉, 把里面的单页面组件放到 `components` 文件夹里



`App.vue` 相当于我们平时静态切图的 `index.html`, 是工程启动的初始页面, `main.js` 是用来给我们需要的插件注册的一个地方, 点开他, 我们可以看到已经有两个插件 `router` 和 `vuex` 已经自动帮我们注册好了, 至于如何引入第三方插件, 我之后会详细说明;

```
JS main.js X
src > JS main.js > ...
1  import Vue from "vue";
2  import App from "./App.vue";
3  import router from "./router";
4  import store from "./store";
5
6  Vue.config.productionTip = false;
7
8  new Vue({
9    router,
10   store,
11   render: h => h(App)
12 }).$mount("#app");
13
```

2. 接下来是如何将整个工程跑起来, 也就是如何在浏览器上看到整个工程的页面, 我们直接在 `VSCODE` 的终端面板中输入:

```
npm run serve
```

稍等片刻便会出现如下进度

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: node
> hello-world@0.1.0 serve F:\vue学习\hello-world
> vue-cli-service serve

INFO Starting development server...
10% building 0/1 modules 1 active ...e学习\hello-world\node_modules\webpack\hot\dev-ser
10% building 2/5 modules 3 active ...s\eslint-loader\index.js??ref--13-0!F:\vue学习\hel
10% building 3/5 modules 2 active ...s\eslint-loader\index.js??ref--13-0!F:\vue学习\hel
10% building 4/5 modules 1 active ...s\eslint-loader\index.js??ref--13-0!F:\vue学习\hel
s
```

等进度100%后终端会如下图所示出现两个链接, 大家可以直接按住 `CTRL` 并点击链接就可以在浏览上打开项目页面;

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: node
App running at:
- Local: http://localhost:8081/
- Network: http://192.168.1.101:8081/

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

第一个链接用了本地连接, 第二个是局域网连接, 大家自己酌情使用, 打开后页面如果是下图的样子, 说明你做对了;👍

[Home](#) | [About](#)



Welcome to Your Vue.js App

For a guide and recipes on how to configure / customize this project,
check out the [vue-cli documentation](#).

Installed CLI Plugins

[babel](#) [router](#) [vuex](#) [eslint](#)

Essential Links

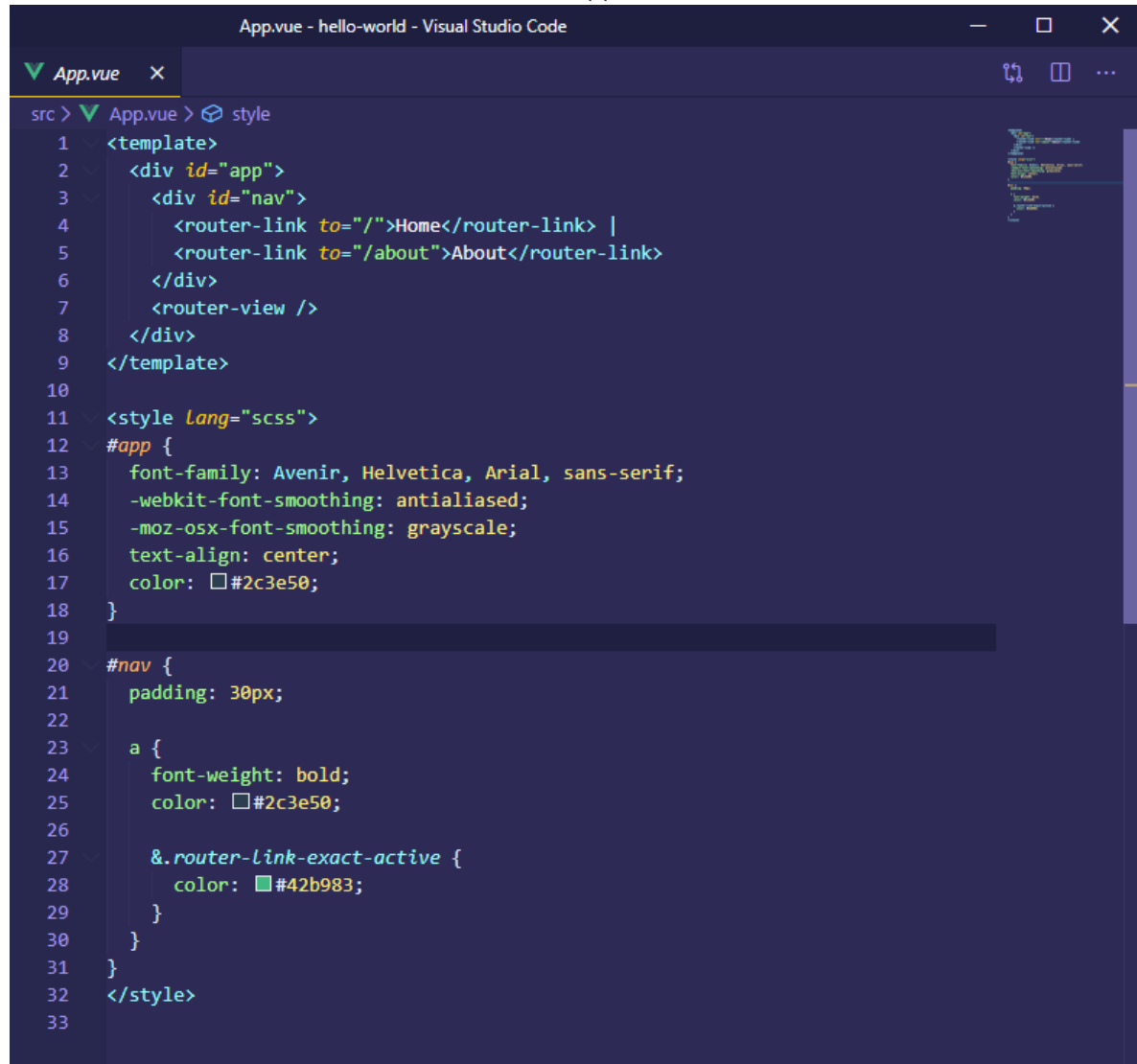
[Core Docs](#) [Forum](#) [Community Chat](#) [Twitter](#) [News](#)

Ecosystem

[vue-router](#) [vuex](#) [vue-devtools](#) [vue-loader](#) [awesome-vue](#)

四、开始切图

这个时候我们就可以开始正式切图了, 我们可以打开App.vue文件, 如下图

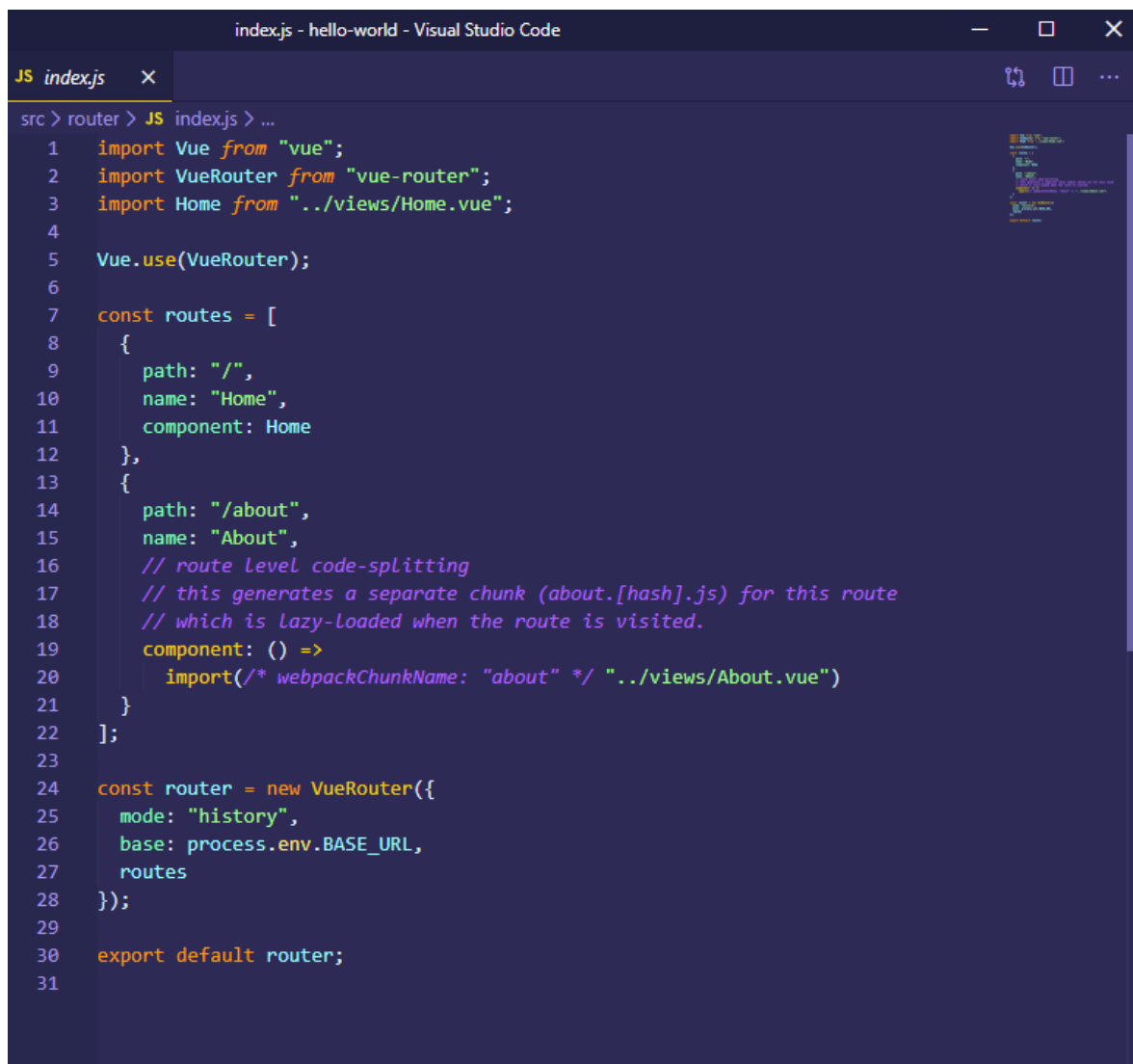


```
App.vue - hello-world - Visual Studio Code
src > App.vue > style
1 <template>
2   <div id="app">
3     <div id="nav">
4       <router-link to="/">Home</router-link> |
5       <router-link to="/about">About</router-link>
6     </div>
7     <router-view />
8   </div>
9 </template>
10
11 <style Lang="scss">
12 #app {
13   font-family: Avenir, Helvetica, Arial, sans-serif;
14   -webkit-font-smoothing: antialiased;
15   -moz-osx-font-smoothing: grayscale;
16   text-align: center;
17   color: #2c3e50;
18 }
19
20 #nav {
21   padding: 30px;
22
23   a {
24     font-weight: bold;
25     color: #2c3e50;
26
27     &.router-link-exact-active {
28       color: #42b983;
29     }
30   }
31 }
32 </style>
33
```

他的结构其实和我们平常接触的html结构很类似, 也都是由各种标签组成的, 也就是模板语法, 只有少量细节不一样, 要注意的是在 VUE 2.X 版本里, `template` 标签下面只能存在一个标签, 不能同时存在两个, 否则就会报错, 大家切图的时候注意下.

具体VUE的语法有哪些不一样, 我这里就不一一给大家讲解了, 官方网站 [Vue.js](https://vuejs.org) 讲的很详细, 大家自己点开去看;

接下来我简单讲下如何设置跳转链接, VUE本地页面的跳转都是通过 `Router` 这个插件来实现的, 我们首先打开 `src\router\index.js` 这个文件

A screenshot of the Visual Studio Code editor window titled 'index.js - hello-world - Visual Studio Code'. The editor shows a file named 'index.js' with the following code:

```
1 import Vue from "vue";
2 import VueRouter from "vue-router";
3 import Home from "../views/Home.vue";
4
5 Vue.use(VueRouter);
6
7 const routes = [
8   {
9     path: "/",
10    name: "Home",
11    component: Home
12  },
13  {
14    path: "/about",
15    name: "About",
16    // route level code-splitting
17    // this generates a separate chunk (about.[hash].js) for this route
18    // which is lazy-loaded when the route is visited.
19    component: () =>
20      import(/* webpackChunkName: "about" */ "../views/About.vue")
21  }
22 ];
23
24 const router = new VueRouter({
25   mode: "history",
26   base: process.env.BASE_URL,
27   routes
28 });
29
30 export default router;
```

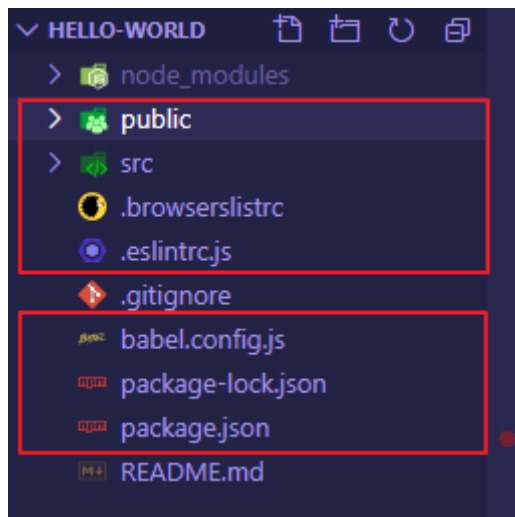
首先我们得 `import` 需要跳转的那个页面文件, 比如 `home`, 随后我们需要定义一下他的路由, 比如上图的路由路径定义为了 `"/"`, 然后是起一个 `name` (这个其实可要可不要), 最后是在 `component` 里声明一下引入的页面的名称, 这样路由的声明就完成了;

接下来我们看 `App.vue` 页面, 有两个和路由有关的标签, 分别是 `<router-link>` 和 `<router-view>` 两个标签, 顾名思义, 一个是用来点击的链接, 一个是用来展示点击链接后的内容, 这样我们就可以随意跳转本地页面了, 当然了, 你还是可以用 `<a>` 标签来跳转页面, 不过我建议用 `<router-link>` 这样跳转速度会更快些, 同时页面相同的元素会保留而不会刷新.

大家如果想要具体了解 `Router` 这个插件的强大功能, 建议访问 [Vue Router](#) 

四、和开发对接注意事项

- 和开发对接我们首先要清楚哪些文件时需要发给开发的, 我这里做了个图解说明, 具体时下面红框内的文件:



千万别把 `node_modules` 给打包进去了, 这玩意有几十兆这么大!!! 🤖

- 如果是开发把工程文件发给你, 你该怎么在把项目跑起来呢?
 1. 首先向开发索要上面红框的文件, 并存放到一个文件夹内;
 2. vscode 打开这个文件夹;
 3. 在终端面板里输入:

```
npm i
```

4. 等插件安装完成后, 输入:

```
npm run serve
```

这样就可以预览工程文件了;

- 我们切图的时候把页面分成模块切图, 如果很多不同页面有一个部分都长得很相似, 那么这个部分就可以单独起一个 `.vue` 单文件组件, 到时候开发会方便很多, 同时大家在切图的时候最好别用 `jquery` 库, 直接使用原生的 `Javascript` 最好, 因为vue框架已经包含了很多jq的功能;

五、如何引入第三方插件

我们切图经常需要用到插件, 在vue切图使用插件的时候注意几个点:

- 优先寻找支持 `NPM` 引入的插件, 那种什么 **Jquery插件库** 上的插件最好别用了;
- 大量插件都可以在 [Github](#) 上找到, 没有账号的自己注册一个;
- 大部分插件的git页面下方都会有如何使用的说明, 基本上看下说明就知道怎么用;

但由于我们这是入门手册, 有些童鞋不知道流程 🤖, 我这里还是讲一下(以[Vue-ECharts](#)为例):

1. 首先在工程的终端面板里输入:

```
npm install echarts vue-echarts
```

2. 然后打开 `main.js`, 按照下面的代码输入:

```
import Vue from 'vue'
import ECharts from 'vue-echarts' // refers to components/ECharts.vue in
webpack

// import ECharts modules manually to reduce bundle size
```

```
import 'echarts/lib/chart/bar'
import 'echarts/lib/component/tooltip'

// If you want to use Echarts extensions, just import the extension package
and it will work
// Taking ECharts-GL as an example:
// You only need to install the package with `npm install --save echarts-gl`
and import it as follows
import 'echarts-gl'

// register component to use
vue.component('v-chart', ECharts)
```

这里注意一下vue-echarts是分模块引入的, 比如你用到了柱状图, 那么你就需要单独引入bar组件

import 'echarts/lib/chart/bar', 引入了tooltip, 就得 import 'echarts/lib/component/tooltip', 这样可以减少项目启动需要注入的js;

3. 接下来是在vue文件中使用注册好的echarts插件, 加入一个 `v-chart` 标签:

```
<template>
  <v-chart :options="polar"/>
</template>
```

4. 值得注意的是这个组件有个需要自定义的对象 `polar`, 其实这个 `polar` 就是我们需要加入的配置项, 如下图, 在 `data` 里面写好 `polar` 配置项的内容:

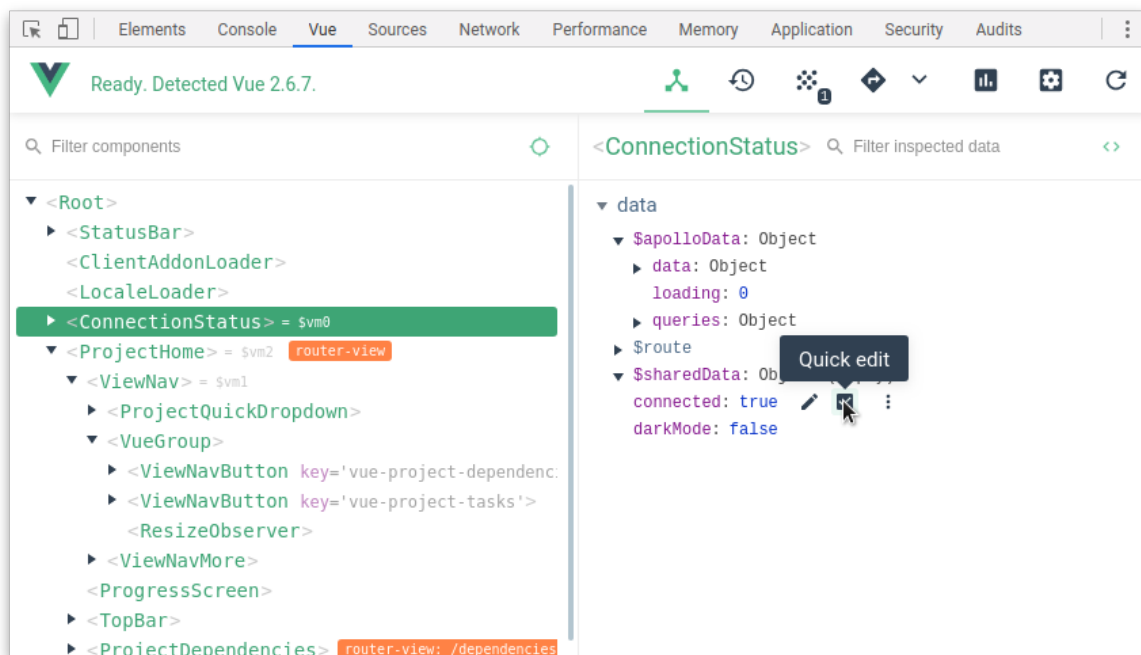
```
return {
  inputText: '',
  area: ['迪庆', '怒江', '丽江', '大理', '保山', '德宏', '临沧', '楚雄州', '昆明', '昭通'],
  bar1: {
    dataset: {
      source: bar1_data,
    },
    grid: {
      left: 20,
      top: 30,
      bottom: 10,
      right: 20,
    },
  },
  xAxis: [{
    show: false,
  }, {
    show: false,
  }],
  yAxis: {
    type: 'category',
    inverse: true,
    show: false
  },
  series: [
    {
      show: true,
      type: 'bar',
      barGap: '-100%',
      barWidth: 10,
      z: 2,
    }
  ]
}
```

具体配置项要怎么写, 我这里就不赘述了, 大家自己去看 [echarts配置项文档](#)

通过以上四步相信大家都知道怎么在工程内引入第三方插件了吧。☺

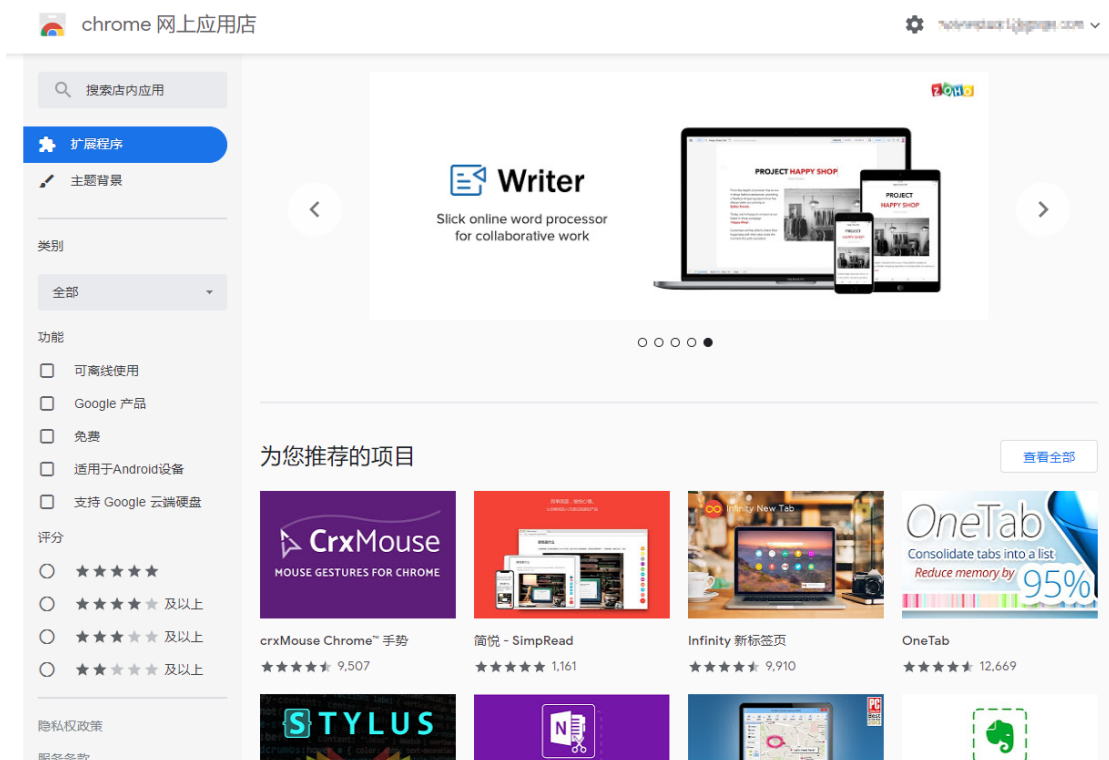
六、方便VUE切图的小工具

vue官方其实做了一个匹配VUE的开发者工具 [Vue DevTools](#)

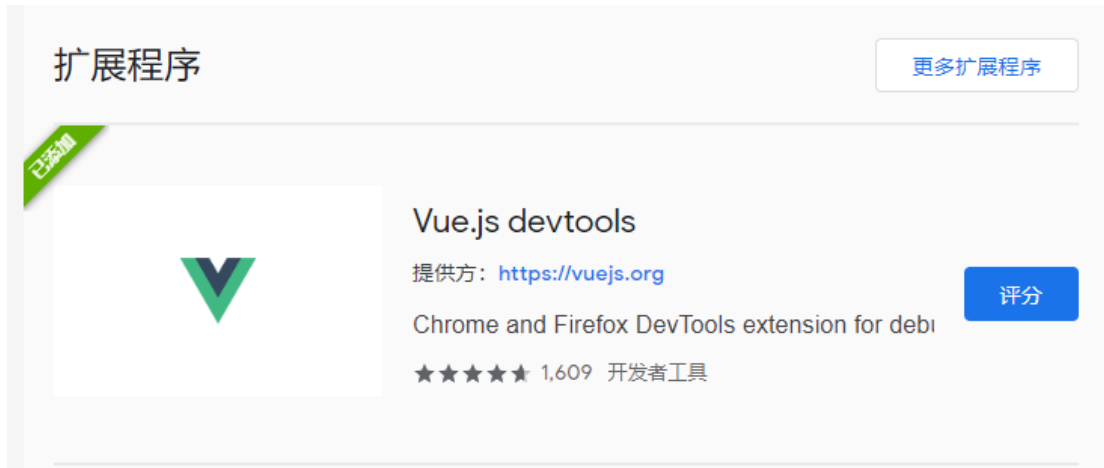


这小玩意可以在谷歌和火狐的拓展工具内直接安装,如果大家用的不是谷歌火狐,可以使用单独的 **Electron App** 版本,具体使用大家自己看说明,我这里就只讲下谷歌版本的使用方法:

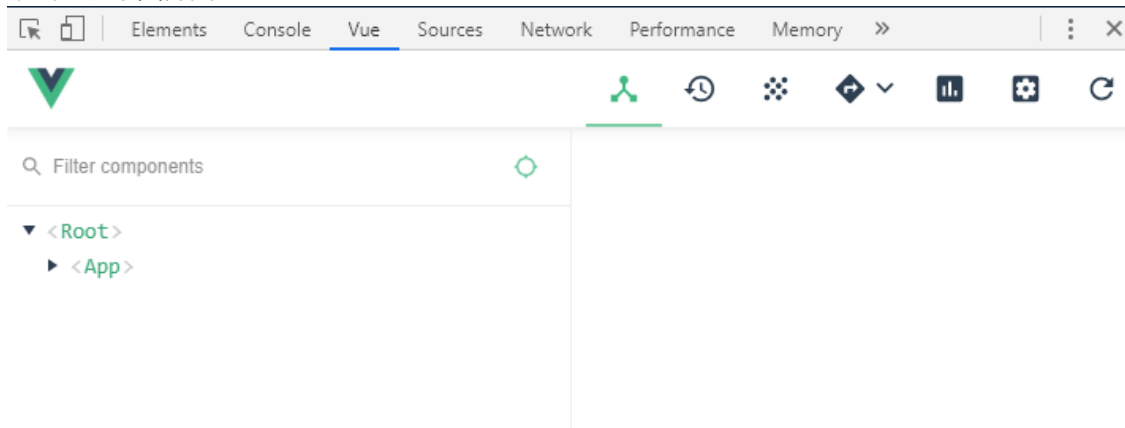
1. 打开谷歌网上商店,打不开的自己找下梯子翻墙



2. 搜索 `vue`, 找到 `vue.js devtools`, 然后添加至chrome即可;



3. 安装好后在你的vue工程页面按 `F12` 打开开发者工具, 在顶部的tab菜单中找到 `vue` 字样并点击, 就会出现下图界面:



这样你就可以开始用vue的开发者工具了, 你可以检查每个组件的属性和data值, 甚至可以修改预览效果, 非常方便;