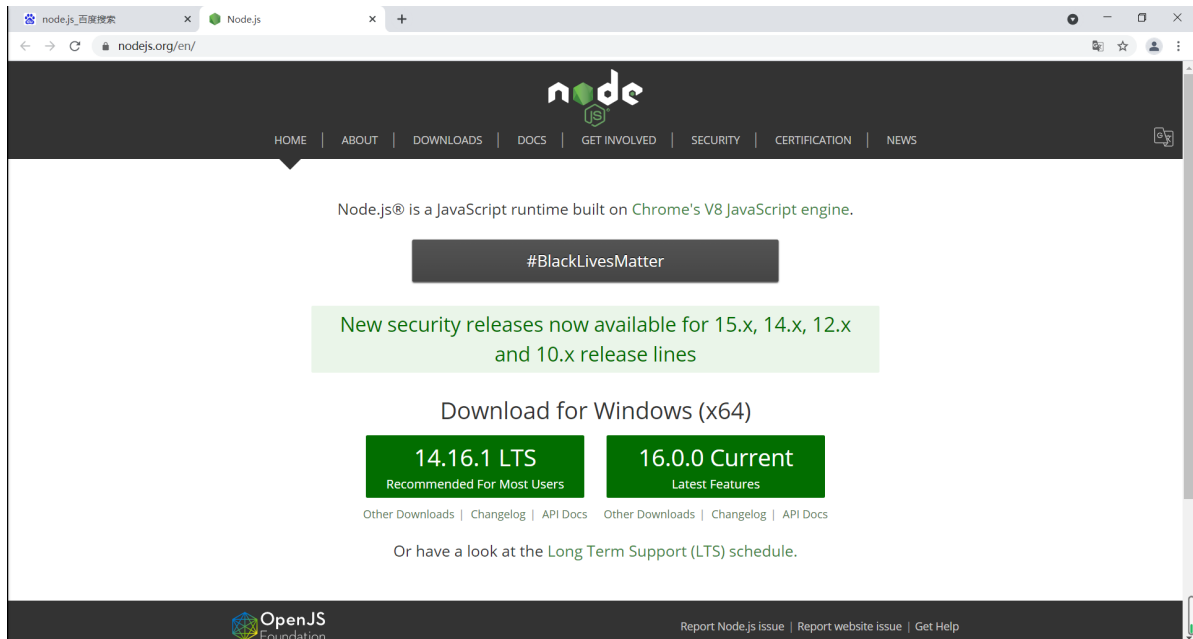


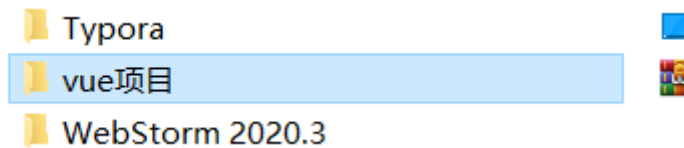
## 1.\*\*前端项目的准备\*\*

在百度搜索node.js官网，并下载，安装

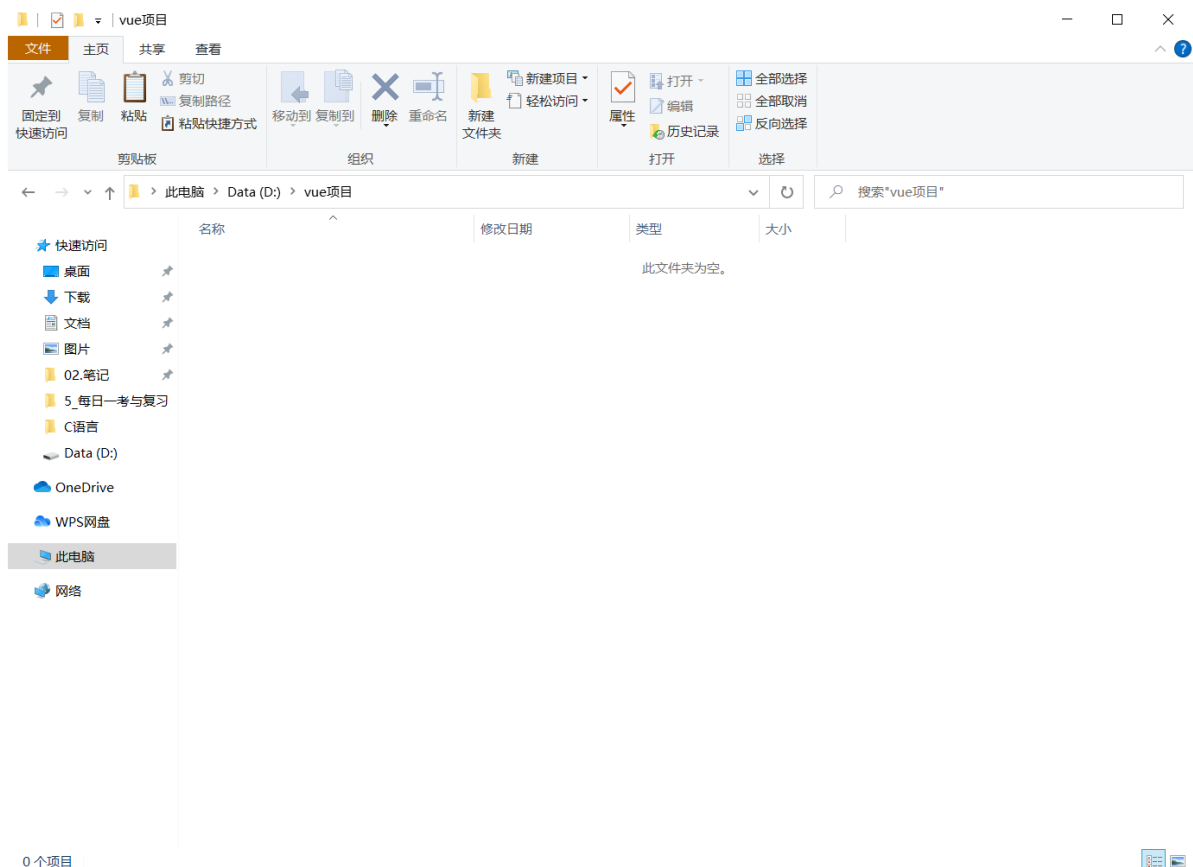


之后就进行vue的创建

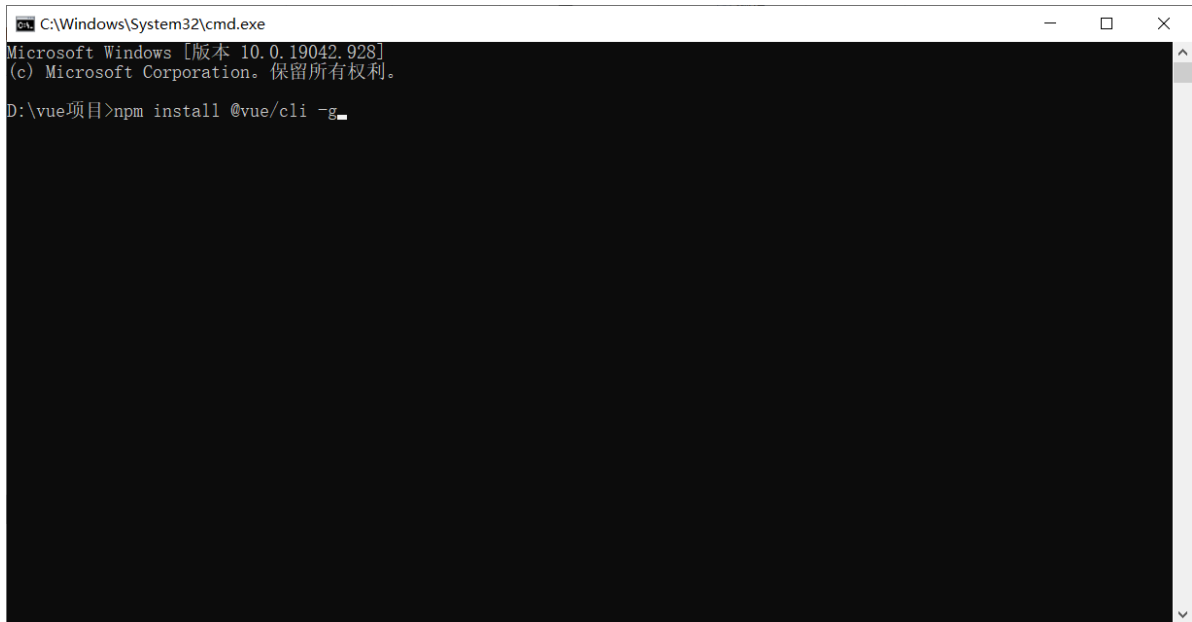
(1) 可以在D盘创建一个文件夹 //这里我命名为vue项目



(2) 打开文件夹，在命令行输入cmd



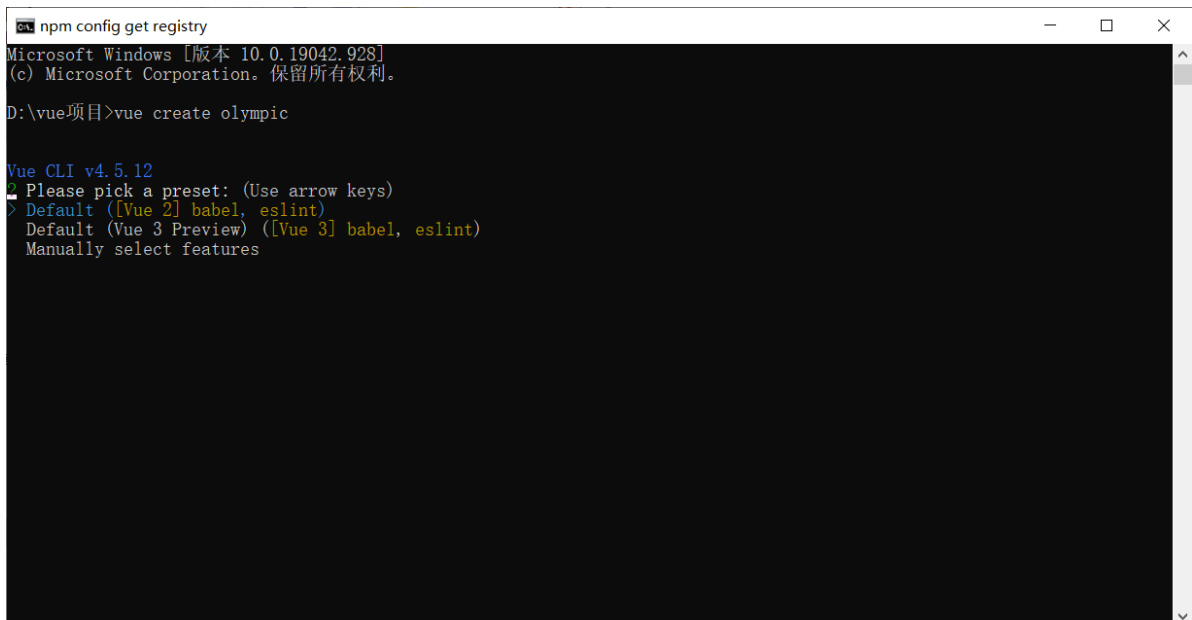
(3)通过命令npm install @vue/cli -g //注意一台电脑只需执行一次即可，下一次可不执行此语句



```
C:\Windows\System32\cmd.exe
Microsoft Windows [版本 10.0.19042.928]
(c) Microsoft Corporation。保留所有权利。

D:\vue项目>npm install @vue/cli -g
```

(4) 通过vue create + 项目名称 的方式创建vue //这里我vue项目名称为 olympic，命名不可大写。



```
npm config get registry
Microsoft Windows [版本 10.0.19042.928]
(c) Microsoft Corporation。保留所有权利。

D:\vue项目>vue create olympic

Vue CLI v4.5.12
? Please pick a preset: (Use arrow keys)
> Default ([Vue 2] babel, eslint)
  Default (Vue 3 Preview) ([Vue 3] babel, eslint)
  Manually select features
```

之后会出现选项 //如有不同可百度具体查询

### 1.1.2. 工程的创建

使用命令行执行

具体的配置项如下:

手动选择特性 (按住`↓`选择Manually select features 选中后按回车)

```
npm config get registry
Microsoft Windows [版本 10.0.19042.928]
(c) Microsoft Corporation。保留所有权利。

D:\vue项目>vue create olympic

Vue CLI v4.5.12
? Please pick a preset:
  Default ([Vue 2] babel, eslint)
  Default (Vue 3 Preview) ([Vue 3] babel, eslint)
> Manually select features
```

集成（按住`[↓]`分别选择Router, Vuex, CSS Pre-processors 按空格选中。全部选中后按回车）

```
? Check the features needed for your project:
(*) Babel
(*) TypeScript
(*) Progressive Web App (PWA) Support
(*) Router
(*) Vuex
(*) CSS Pre-processors
(*) Linter / Formatter
(*) Unit Testing
(*) E2E Testing
```

vue create vision是否选用历史模式的路由

```
? Please pick a preset: Manually select features
? Check the features needed for your project: Babel, Router, Vuex, CSS Pre-processors, Linter
? Use history mode for router? (Requires proper server setup for index fallback in production) (Y/n) n
```

选择 Less 作为 CSS 的预处理器

```
? Please pick a preset: Manually select features
? Check the features needed for your project: Babel, Router, Vuex, CSS Pre-processors, Linter
? Use history mode for router? (Requires proper server setup for index fallback in production) No
? 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
```

选择 ESLint 的配置

```
? Please pick a preset: Manually select features
? Check the features needed for your project: Babel, Router, Vuex, CSS Pre-processors, Linter
? Use history mode for router? (Requires proper server setup for index fallback in production) No
? Pick a CSS pre-processor (PostCSS, Autoprefixer and CSS Modules are supported by default): Less
? Pick a linter / formatter config:
  ESLint with error prevention only
  ESLint + Airbnb config
> ESLint + Standard config
  ESLint + Prettier
```

什么时候进行 Lint 提示

```
? Please pick a preset: Manually select features
? Check the features needed for your project: Babel, Router, Vuex, CSS Pre-processors, Linter
? Use history mode for router? (Requires proper server setup for index fallback in production) No
? Pick a CSS pre-processor (PostCSS, Autoprefixer and CSS Modules are supported by default): Less
? Pick a linter / formatter config: Standard
? Pick additional lint features: (Press <space> to select, <a> to toggle all, <i> to invert selection)
> (*) Lint on save
  (*) Lint and fix on commit
```

如何存放 Babel, ESLint 等配置文件

```
? Please pick a preset: Manually select features
? Check the features needed for your project: Babel, Router, Vuex, CSS Pre-processors, Linter
? Use history mode for router? (Requires proper server setup for index fallback in production) No
? Pick a CSS pre-processor (PostCSS, Autoprefixer and CSS Modules are supported by default): Less
? Pick a linter / formatter config: Standard
? Pick additional lint features: (Press <space> to select, <a> to toggle all, <i> to invert selection)Lint on save
? Where do you prefer placing config for Babel, ESLint, etc.? (Use arrow keys)
> In dedicated config files
  In package.json
```

是否保存以上配置以便下次创建项目时使用

```
? Please pick a preset: Manually select features
? Check the features needed for your project: Babel, Router, Vuex, CSS Pre-processors, Linter
? Use history mode for router? (Requires proper server setup for index fallback in production) No
? Pick a CSS pre-processor (PostCSS, Autoprefixer and CSS Modules are supported by default): Less
? Pick a linter / formatter config: Standard
? Pick additional lint features: (Press <space> to select, <a> to toggle all, <i> to invert selection)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) n
```

配置选择完之后, 就开始创建项目了, 这个过程需要一些时间:

当项目就创建完成了, 会看到这个提示

```
$ cd vision
$ npm run serve

WARN Skipped git commit due to missing username and email in git config.
You will need to perform the initial commit yourself.
```

将目录使用 vscode 或HB打开

### 1.1.3.\*\*删除无关代码\*\*

删除 src/components/HelloWorld.vue 这个文件

删除 src/views/About.vue 和 views/Home.vue 这两个文件

修改 App.vue 中的代码,将布局和样式删除, 变成如下代码:

```
1 <template>
2   <div id="app">
3     <router-view></router-view>
4   </div>
5 </template>
6 <style lang="less">
7 </style>
8
```

修改 router/index.js 中的代码,去除路由配置和 Home 组件导入的代码

```
1 import Vue from 'vue'
2 import VueRouter from 'vue-router'
3
4 Vue.use(VueRouter)
5
6 const routes = [
7 ]
8
9 const router = new VueRouter({
10   routes
11 })
12
13 export default router
```

修改 .eslintrc.js 中的代码（将 rules 替换，避免一些代码规范的报错）：

```

1  rules: {
2    'no-tabs': 'off',
3    'indent': ['off', 2],
4    'space-before-function-paren': 0,
5    'no-console': process.env.NODE_ENV === 'production' ? 'warn' : 'off',
6    'no-debugger': process.env.NODE_ENV === 'production' ? 'warn' : 'off'
7  }

```

## 1.2. 项目的基本配置

在项目根目录下创建 vue.config.js 文件

在文件中增加代码：

```

1  // 使用vue-cli创建出来的vue工程，webpack的配置是被隐藏起来了的
2
3  // 如果想覆盖webpack中的默认配置，需要在项目的根路径下增加vue.config.js文件
4
5  module.exports = {
6    devServer: {
7      port: 8080,
8      open: true
9    }
10 }
11

```

## 1.3. \*\*全局echarts 对象\*\*

### 1.3.1. \*\*引入\*\* echarts 包

将资料文件夹中的 static 目录复制到 public 目录之下

在 public/index.html 文件中引入 echarts.min.js 文件，//在public/index.html 的 body中写入以下代码：

```

1
2  <script src="static/lib/echarts.min.js"></script>

```

### 1.3.2. \*\*挂载到\*\* Vue 原型上

在 src/main.js 文件中写入如下代码

```

1  .....
2
3  // 将全局echarts对象挂载到Vue的原型对象上
4
5  vue.prototype.$echarts = window.echarts
6
7  .....

```

由于在 index.html 中已经通过script标签引入了 echarts.js 文件夹，故在 window 全局对象中是存在 echarts 全局对象，写入刚才的代码就可将其挂载到 Vue 的原型对象上

### 1.3.3.\*\*使用全局\*\* echarts 对象

在其他组件中使用

在具体的echarts实例中调用echarts.js只需用如下代码：

```
1 | this.$echarts
```

## 1.4. axios 的处理

### 1.4.1.\*\*安装\*\* axios 包

```
1 | npm install axios
```

### 1.4.2.\*\*封装\*\* axios 对象

在 src/main.js 文件中配置 axios 并且挂载到Vue的原型对象上，写入如下代码：

```
1 | .....  
2 |  
3 | import axios from 'axios'  
4 |  
5 | vue.prototype.$http = axios  
6 |  
7 | .....
```

### 1.4.3.\*\*使用\*\* axios 对象

#### 在其他组件中使用

在具体需要异步获取数据的地方调用axios只需用如下代码：

```
1 | this.$http
```