环境：node 16.13.1

yarn 1.19.2

vue 3

electron 18.2.4

typescript 4.1.5

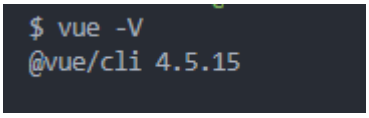
地址 https://github.com/Lateautumn00/zhzy-tveep.git

搭建vue3

一、全局安装vue/cli

```
npm install -g @vue/cli
// 或
yarn global add @vue/cli
```

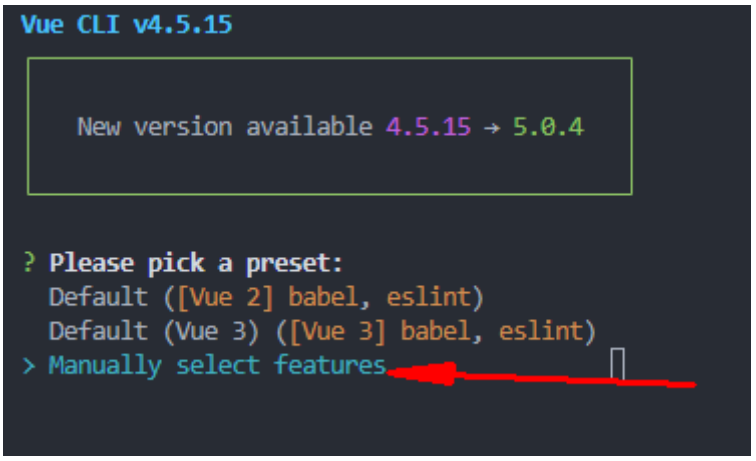
验证全局是否安装成功



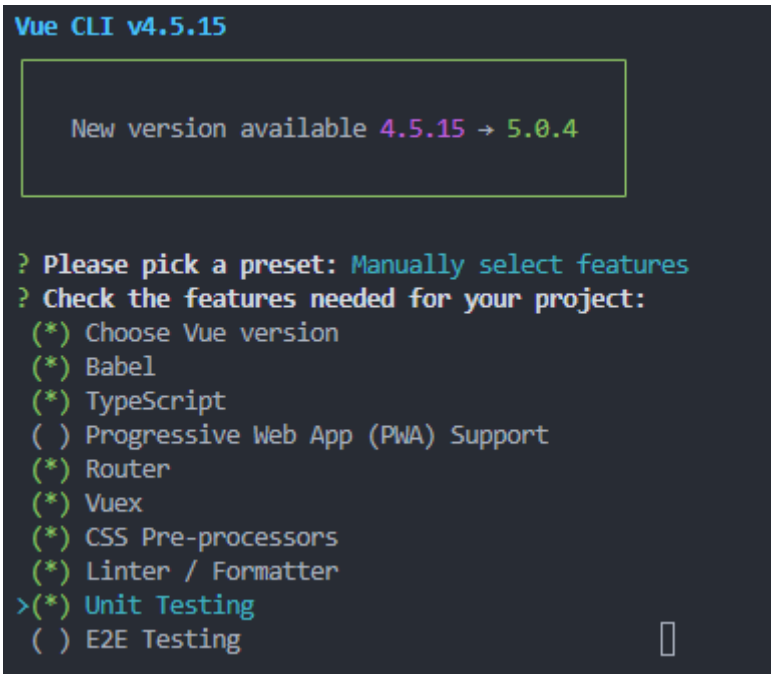
二、初始化项目

1、选择自定义

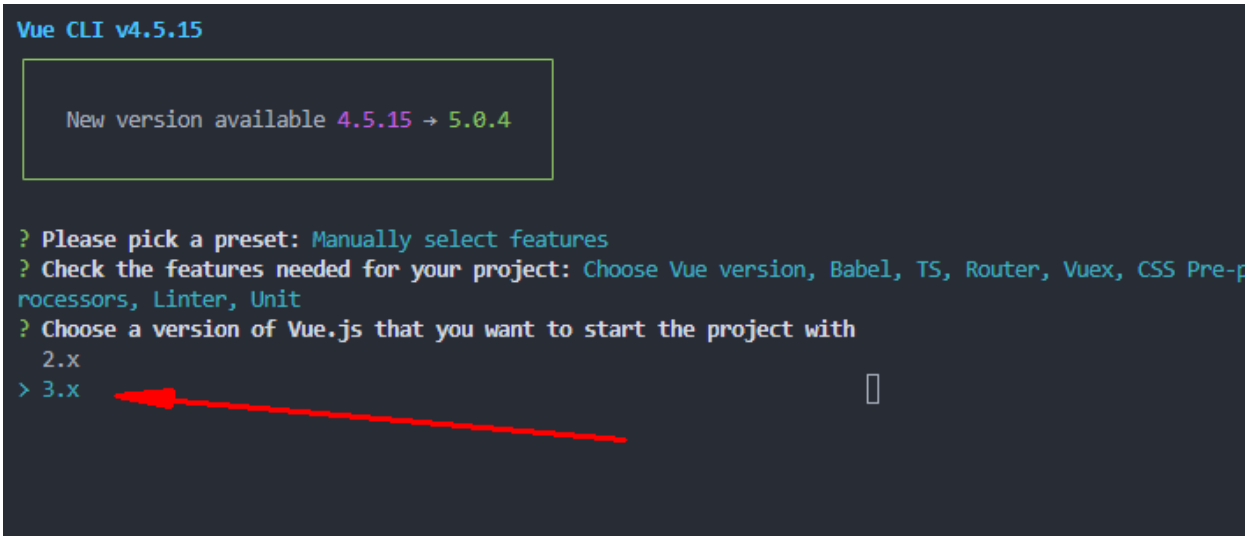
```
vue create zhzy-tveep
```



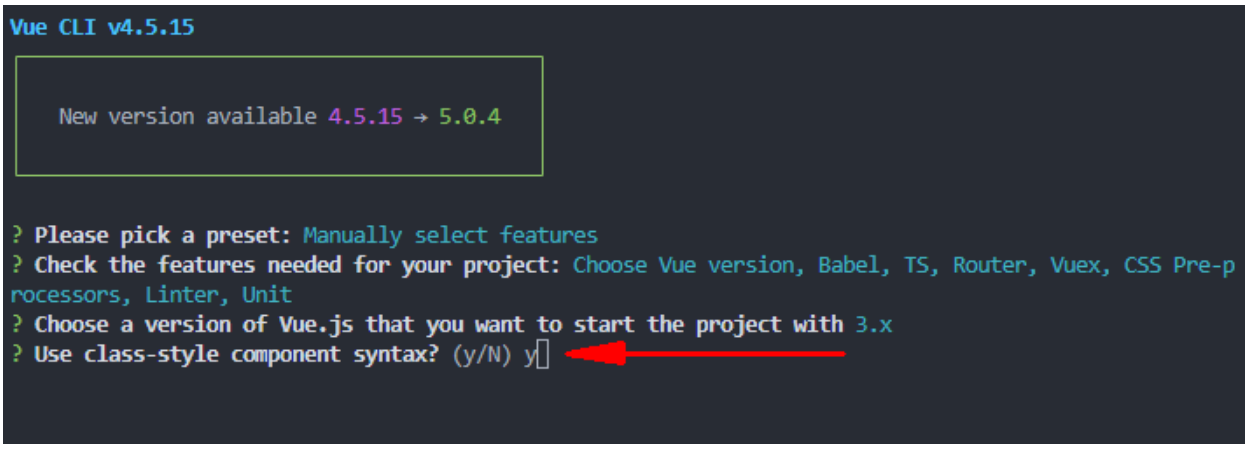
2、使用上下箭头选择，按空格确认选择



3、这里选择3.x



4、是否使用class风格



5、是否使用babel做转义

```
Vue CLI v4.5.15

  New version available 4.5.15 → 5.0.4

? Please pick a preset: Manually select features
? Check the features needed for your project: Choose Vue version, Babel, TS, Router, Vuex, CSS Pre-p
rocessors, Linter, Unit
? Choose a version of Vue.js that you want to start the project with 3.x
? Use class-style component syntax? Yes
? Use Babel alongside TypeScript (required for modern mode, auto-detected polyfills, transpiling JSX
)? (Y/n) y
```

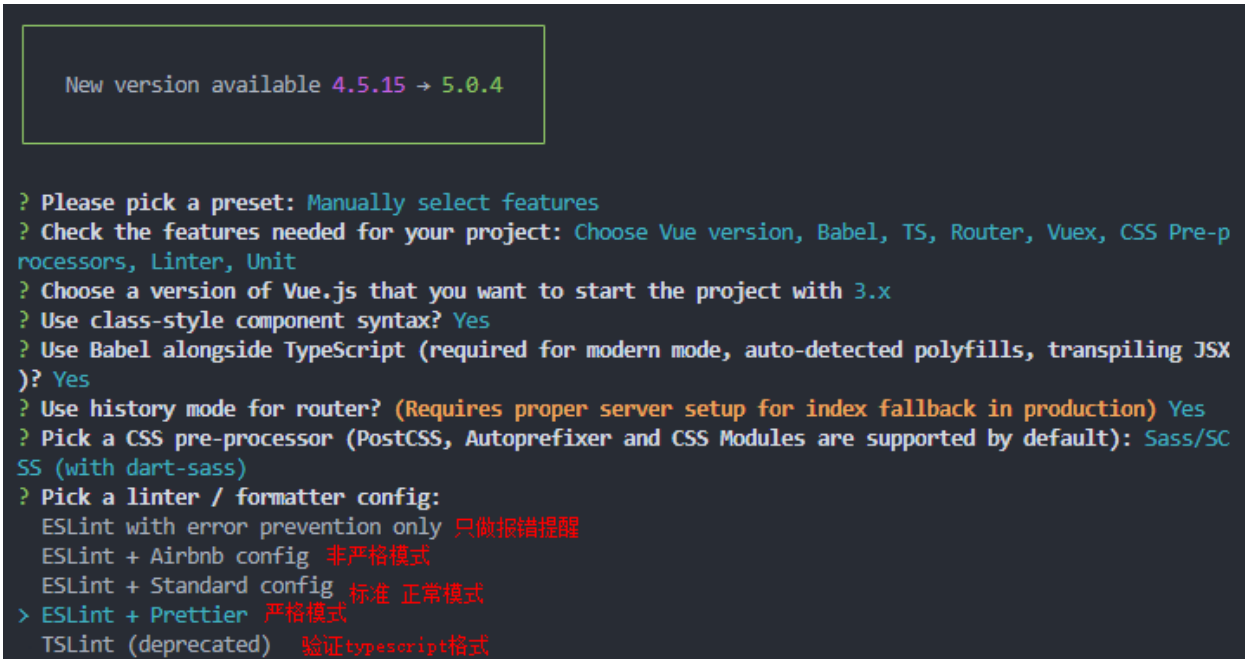
6、是否使用history模式

```
Vue CLI v4.5.15

  New version available 4.5.15 → 5.0.4

? Please pick a preset: Manually select features
? Check the features needed for your project: Choose Vue version, Babel, TS, Router, Vuex, CSS Pre-p
rocessors, Linter, Unit
? Choose a version of Vue.js that you want to start the project with 3.x
? Use class-style component syntax? Yes
? Use Babel alongside TypeScript (required for modern mode, auto-detected polyfills, transpiling JSX
)? Yes
? Use history mode for router? (Requires proper server setup for index fallback in production) (Y/n)
 y
```

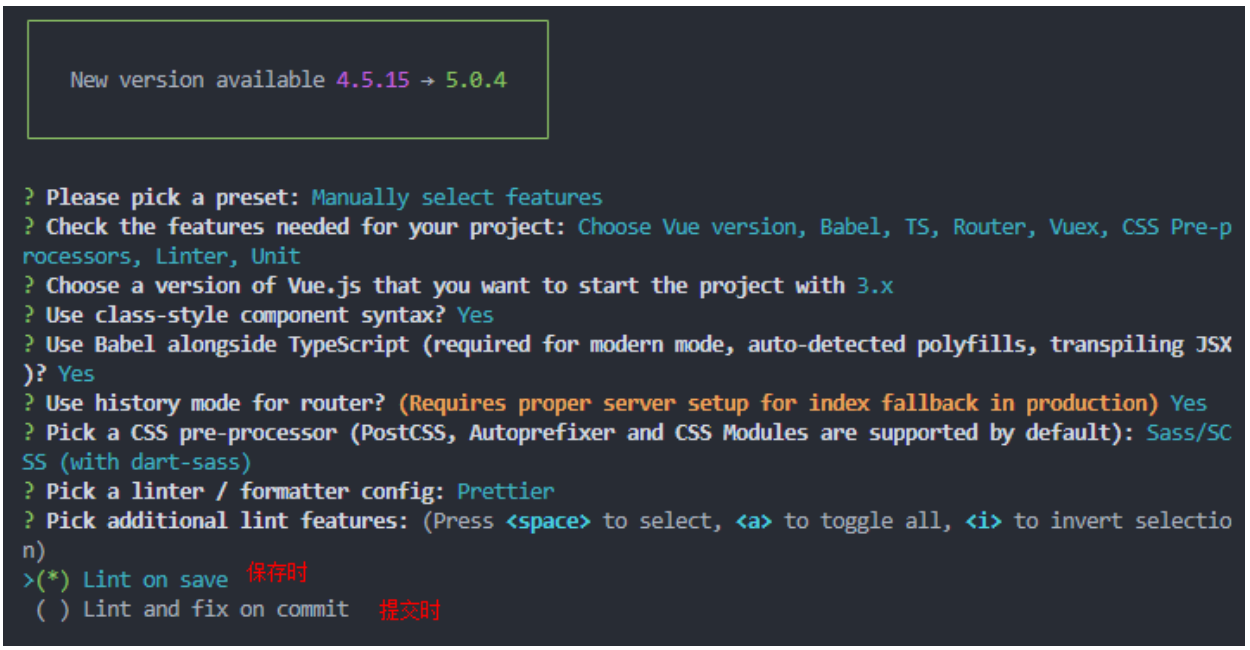
7、选择css预处理器

```
Vue CLI v4.5.15

  New version available 4.5.15 → 5.0.4

? Please pick a preset: Manually select features
? Check the features needed for your project: Choose Vue version, Babel, TS, Router, Vuex, CSS Pre-p
rocessors, Linter, Unit
? Choose a version of Vue.js that you want to start the project with 3.x
? Use class-style component syntax? Yes
? Use Babel alongside TypeScript (required for modern mode, auto-detected polyfills, transpiling JSX
)? Yes
? Use history mode for router? (Requires proper server setup for index fallback in production) Yes
? Pick a CSS pre-processor (PostCSS, Autoprefixer and CSS Modules are supported by default): (Use ar
row keys)
> Sass/SCSS (with dart-sass)  自动编译
  Sass/SCSS (with node-sass)  保存后才生效
  Less
  Stylus
```

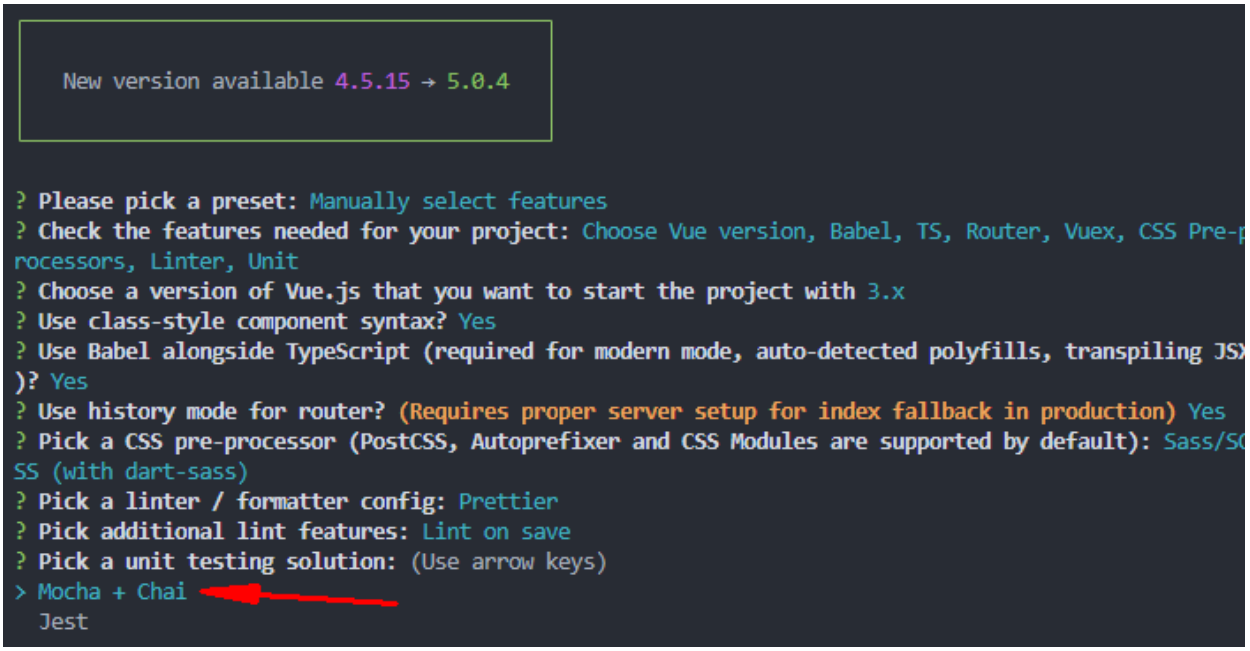
8、选择验证模式



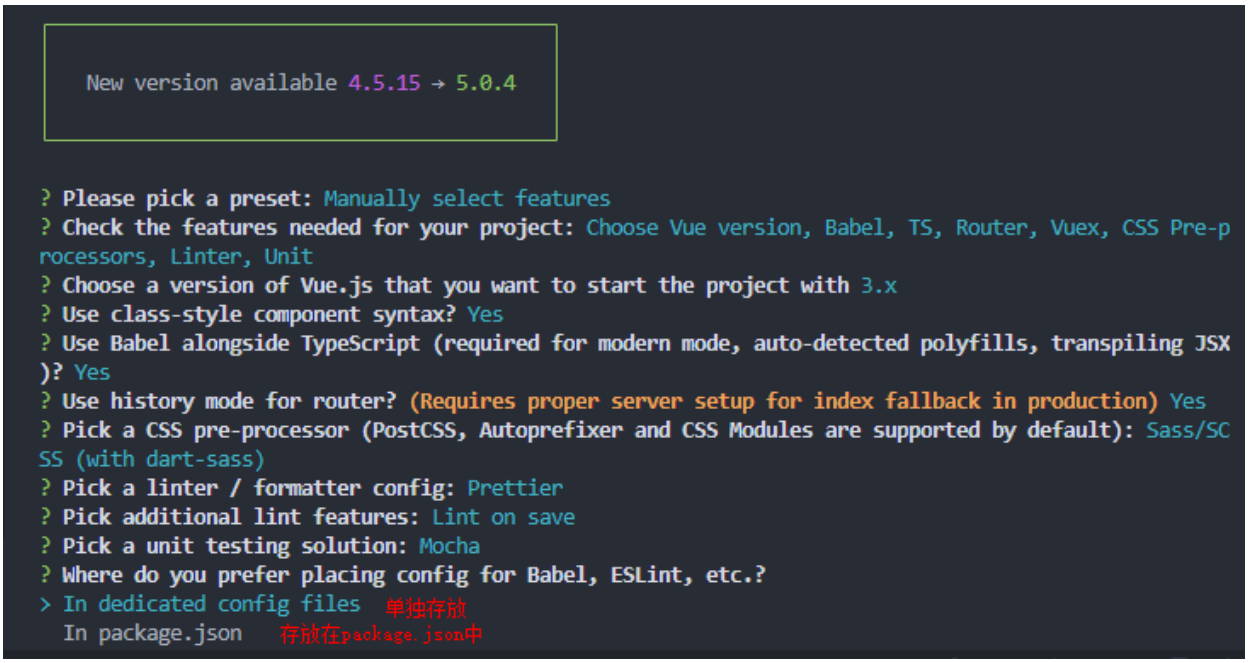
9、选择什么时候进行eslint检查



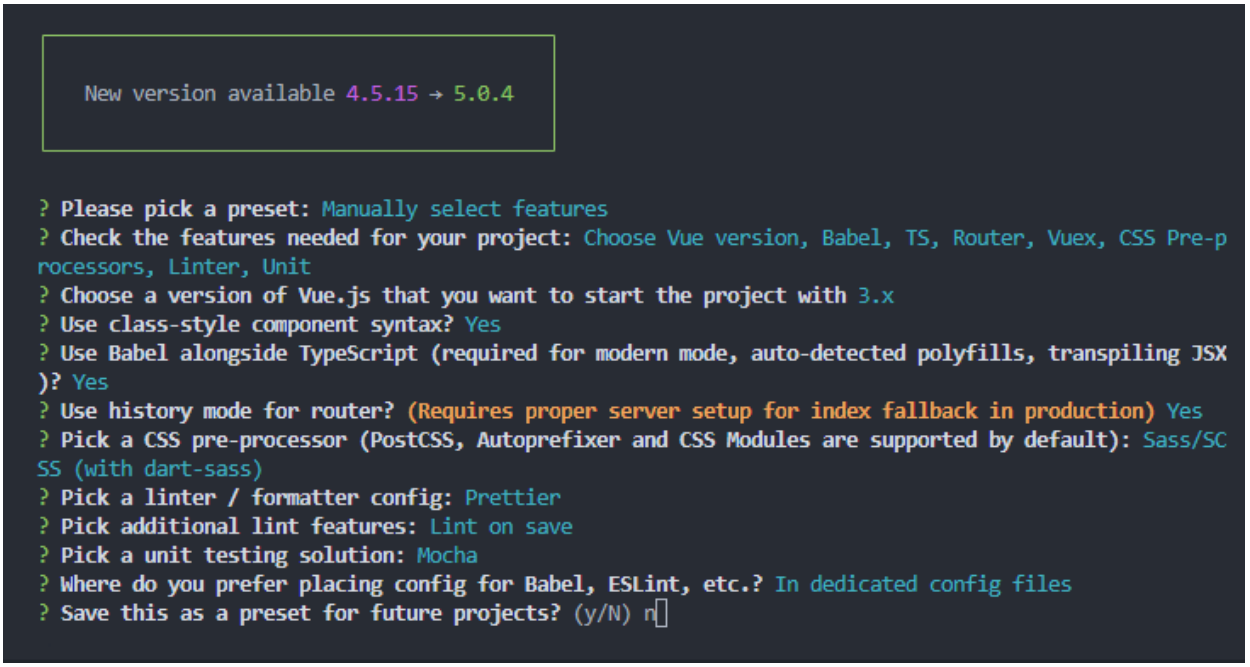
10、选择断言库



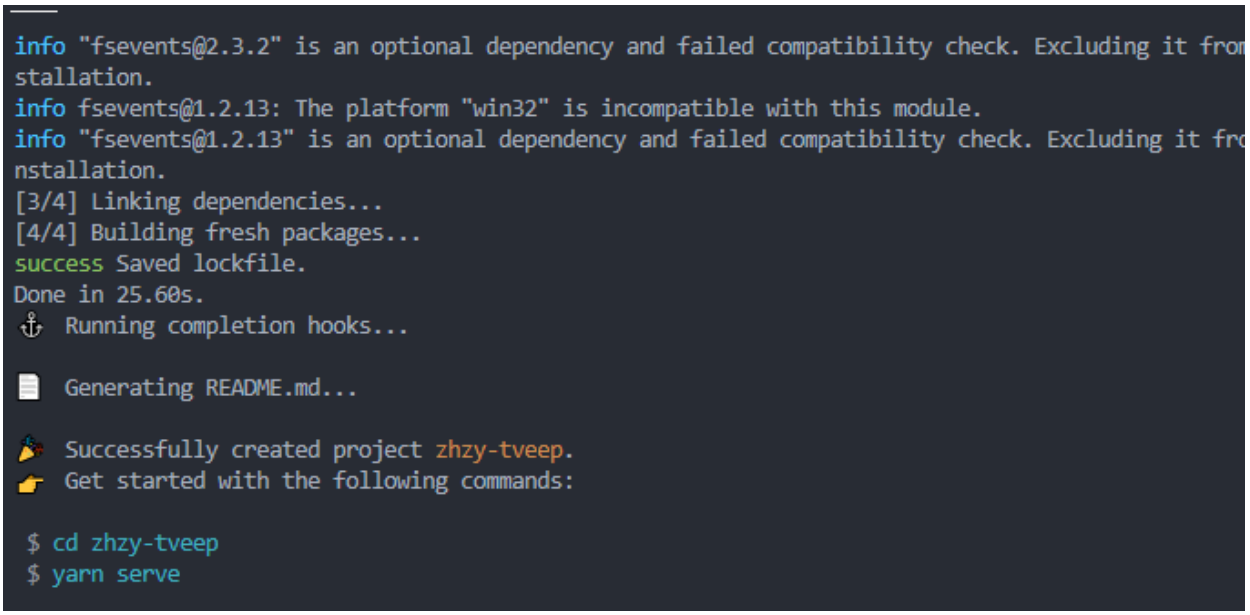
11、eslint、babel配置存放位置



12、是否保存这些选项组合

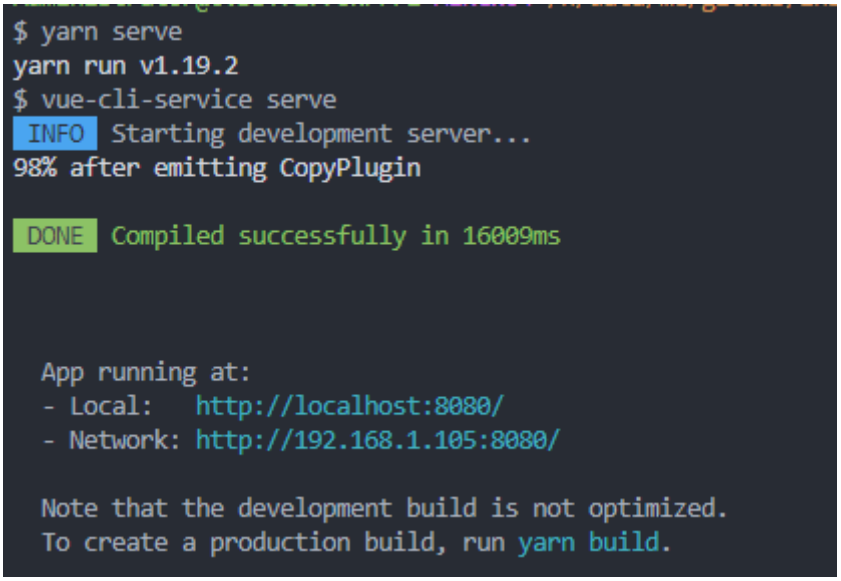


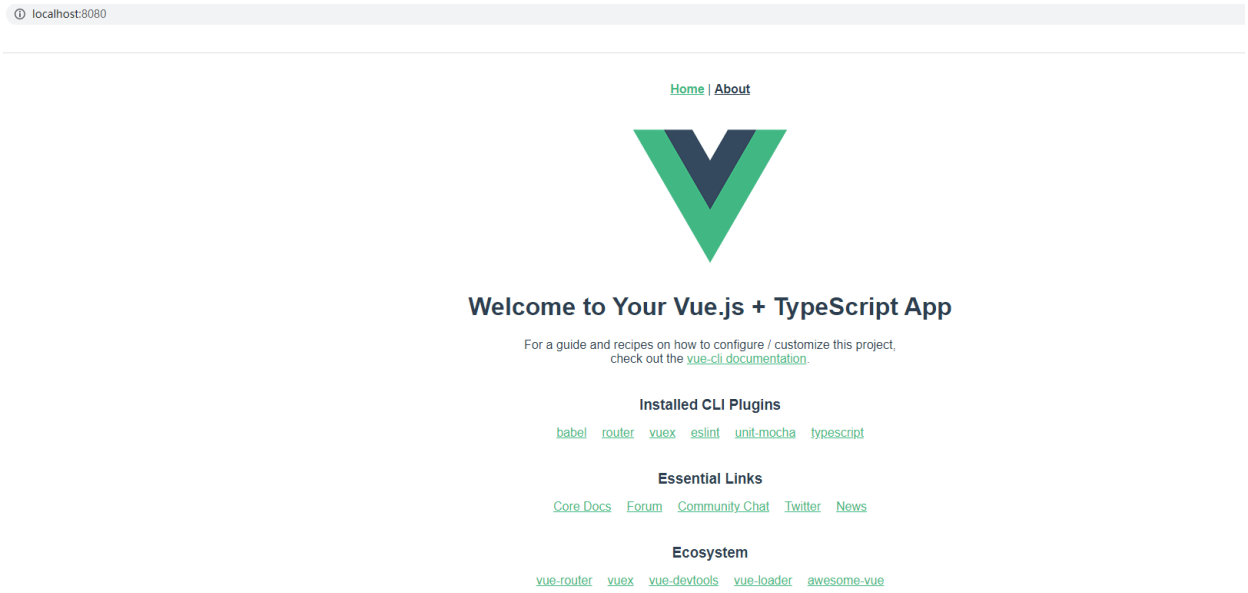
13、安装完成



14、验证是否成功

```
cd zhzy-tveep
yarn serve
```





搭建electron

1、安装 （过程有点慢，耐心等待；建议替换yarn源或翻墙）

```
安装electron
yarn add electron --dev
uarn add electron-devtools-installer --dev
yarn add vue-cli-plugin-electron-builder
```

2、在src目录下新建 background.ts,作为electron入口

```
import { app, protocol, BrowserWindow, ipcMain } from 'electron'
import { createProtocol } from 'vue-cli-plugin-electron-builder/lib'
import path from 'path'
```

```
4 //import installExtension, { VUEJS3_DEVTOOLS } from 'electron-devtools-in
staller'
5 const isDevelopment = process.env.NODE_ENV !== 'production'
6 // Scheme must be registered before the app is ready
7 protocol.registerSchemesAsPrivileged([
8   { scheme: 'app', privileges: { secure: true, standard: true } }
9 ])
10 let win: any = null
11 async function createWindow() {
12   // Create the browser window.
13   win = new BrowserWindow({
14     useContentSize: true,
15     width: 1220,
16     height: 640,
17     minWidth: 1220,
18     minHeight: 640,
19     //transparent: true, //窗口透明  设置后还原窗口win.restore()无效
20     //backgroundColor: '#000', //背景颜色
21     title: '知乎者也', //标题
22     movable: true,
23     webPreferences: {
24       // Required for Spectron testing
25       // enableRemoteModule: !!process.env.IS_TEST,
26       // Use pluginOptions.nodeIntegration, leave this alone
27       // See nklayman.github.io/vue-cli-plugin-electron-builder/guide/se
curity.html#node-integration for more info
28       nodeIntegration: process.env
29         .ELECTRON_NODE_INTEGRATION as unknown as boolean,
30       contextIsolation: !process.env.ELECTRON_NODE_INTEGRATION,
31     }
32   })
33   if (process.env.NODE_ENV === 'production') {
34     //正式
35     createProtocol('app')
36     win.loadURL('app://./index.html') // Load the index.html when not in
development
37     //win.webContents.openDevTools()
38   } else {
39     //开发
40     // Load the url of the dev server if in development mode
41     await win.loadURL(process.env.WEBPACK_DEV_SERVER_URL as string)
```

```
    if (!process.env.IS_TEST) win.webContents.openDevTools()
  }
  // 当应用所有窗口关闭要做的事情
  win.on('closed', () => {
    win = null
  })
}
// Quit when all windows are closed.
app.on('window-all-closed', () => {
  // On macOS it is common for applications and their menu bar
  // to stay active until the user quits explicitly with Cmd + Q
  if (process.platform !== 'darwin') {
    app.quit()
  }
})
app.on('activate', () => {
  // On macOS it's common to re-create a window in the app when the
  // dock icon is clicked and there are no other windows open.
  if (BrowserWindow.getAllWindows().length === 0) createWindow()
})
// This method will be called when Electron has finished
// initialization and is ready to create browser windows.
// Some APIs can only be used after this event occurs.
app.on('ready', async () => {
  //外网 加载慢 去掉
  //   if (isDevelopment && !process.env.IS_TEST) {
  //     // Install Vue Devtools
  //     try {
  //       await installExtension(VUEJS3_DEVTOOLS)
  //     } catch (e) {
  //       console.error('Vue Devtools failed to install:', e.toString
())
  //     }
  //   }
  if (win === null) createWindow()
})
// Exit cleanly on request from parent process in development mode.
if (isDevelopment) {
  if (process.platform === 'win32') {
    process.on('message', (data) => {
```

```
      if (data === 'graceful-exit') {
        app.quit()
      }
    })
  } else {
    process.on('SIGTERM', () => {
      app.quit()
    })
  }
}

```

3、修改package.json文件，新增入口属性以及运行命令

```
"scripts": {
        "test:unit": "vue-cli-service test:unit",
        "lint": "vue-cli-service lint",
        "web:serve": "vue-cli-service serve --open",
        "web:build": "vue-cli-service build",
        "electron:build": "vue-cli-service electron:build",
        "electron:serve": "vue-cli-service electron:serve",
        "postinstall": "electron-builder install-app-deps",
        "postuninstall": "electron-builder install-app-deps"
    },
```

4、根目录下新建vue.config.js 用与保存打包配置

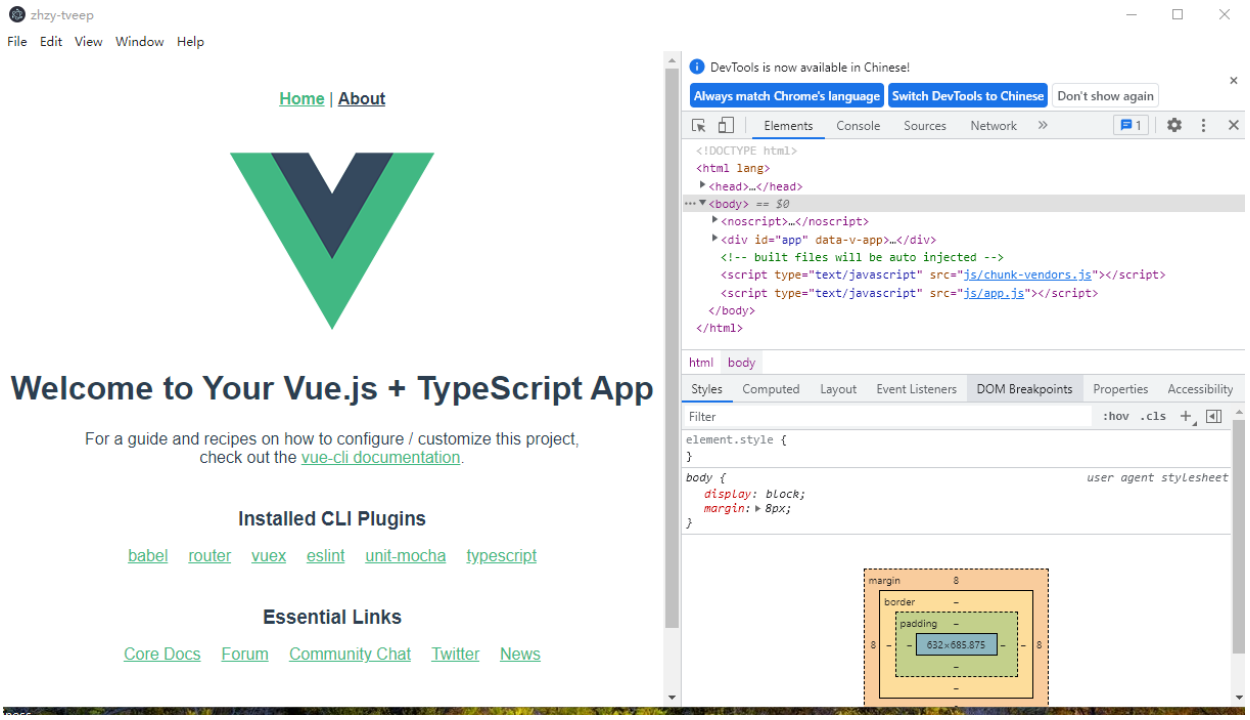
```
module.exports = {
    runtimeCompiler: true,
    publicPath: './',
    //electron 13 把"build":{}从package.json移除，在vue.config.js里写
    pluginOptions: {
        electronBuilder: {
            nodeIntegration: true,
            builderOptions: {
                productName: 'zhzy', //打包名称
                 appId: 'cn.zhzy.app',
                 copyright: 'Copyright © 2022',
                 publish: [
                     {
                         provider: 'generic',
                         url: ''
```

```
        }
      ],
      nsis: {
        allowToChangeInstallationDirectory: true,
        createDesktopShortcut: true,
        createStartMenuShortcut: true,
        shortcutName: 'zhzy',
        perMachine: true,
        oneClick: false
      },
      dmg: {
        contents: [
          {
            x: 410,
            y: 150,
            type: 'link',
            path: '/Applications'
          },
          {
            x: 130,
            y: 150,
            type: 'file'
          }
        ]
      },
      //productName 包名称 version 包版本(package.json)  ext后缀
      mac: {
        icon: 'public/icons/icon.icns',
        artifactName: '${productName}_setup_${version}.${ext}'
      },
      win: {
        icon: 'public/icons/icon.ico',
        artifactName: '${productName}_setup_${version}.${ext}'
      },
      linux: {
        icon: 'public/icons',
        artifactName: '${productName}_setup_${version}.${ext}'
      }
```

```
            }
        }
    }
}

```

5、运行看效果

```
yarn run electron:serve
```



引入element-plus

1、安装

```
yarn add element-plus
yarn add @element-plus/icons-vue
```

2、新建src/plugins/element.ts文件

```
import ElementPlus from 'element-plus'
import 'element-plus/theme-chalk/index.css'
export default (app: any) => {
  app.use(ElementPlus)
}
```

3、修改main.ts

```
import { createApp } from 'vue'
import App from './App.vue'
import router from './router'
import store from './store'
```

```
import installElementPlus from './plugins/element'
import * as ElementPlusIconsVue from '@element-plus/icons-vue'
const app = createApp(App)
for (const [key, component] of Object.entries(ElementPlusIconsVue)) {
  app.component(key, component)
}
installElementPlus(app)
app.use(store).use(router).mount('#app')
```

引入axios

1、安装

```
yarn add axios
background.ts webPreferences中增加 webSecurity: false, //解决跨域
```

2、新建src/plugins/request.ts

```
import axios from 'axios'
const service = axios.create({
  baseURL: '', // 此处写死时无法兼容不同api地址， 地址固定一个 这里可写死
  timeout: 5000
  // withCredentials: true // send cookies when cross-domain requests
})
// Request interceptors
service.interceptors.request.use(
  (config) => {
    // Add X-Access-Token header to every request, you can add other cus
tom headers here
    // if (UserModule.token) {
    //   config.headers.Authorization = UserModule.token
    // }
    return config
  },
  (error) => {
    Promise.reject(error)
  }
)
// Response interceptors
service.interceptors.response.use(
  (response) => {
    const res = response.data
    if (res.statusCode !== 1000) {
      console.error('错误')
      return Promise.reject(new Error(res.message || 'Error'))
```

```
    } else {
      return res
    }
  },
  (error) => {
    console.error('错误')
    return Promise.reject(error)
  }
)

export default service
```

3、新建配置文件app/consig/index.ts

```
interface Config {
  url: string
}
const config: Config = {
  url: ''
}
if (process.env.NODE_ENV === 'production') {
  config.url = 'http://127.0.0.1'
} else {
  config.url = 'http://127.0.0.1'
}
export default config
```

4、新建app/api/demo.ts

```
import request from '@/plugins/request'
import config from '@/config/index'
export const getList = (params: any) =>
  request({
    url: `${config.url}/sys/getTime`,
    method: 'get',
    params
  })
export const create = (data: any) =>
  request({
    url: `${config.url}/sys/getTime`,
    method: 'post',
    data
  })

```

5、使用 见 FrameHome.vue

6、新建app/types文件夹，将shims-vue.d.ts移入并新建axios.d.ts

隐藏菜单栏

在background.ts增加

```
autoHideMenuBar: true, //是否隐藏菜单栏
```

无边框模式

1、在background.ts增加

```
 frame: false, //无边框   不使用无边框模式 需要屏蔽掉  并将router/index.ts 路由中的替换  原始边框路由
```

2、background.ts增加

```
//登录窗口最小化
ipcMain.on('app-min', function () {
  win.minimize()
})
//登录窗口最大化、还原窗口
ipcMain.on('app-max', function () {
  if (win.isMaximized()) {
    win.restore()
  } else {
     win.maximize()
  }
})
ipcMain.on('app-close', function () {
  //win.close() //在设置无边框后失效
  win = null //主窗口设置为null 防止内存溢出
  app.exit() //直接退出应用程序
})

```

3、在src目录下新建proload.ts

```
// import { contextBridge, ipcRenderer } from 'electron'
// contextBridge.exposeInMainWorld('electron', { ipcRenderer }) //使用这种会报错，因为contextIsolation: false是非安全的，设置为true 此文件就获取不到
import { ipcRenderer } from 'electron'
;(window as any).ipcRenderer = ipcRenderer
```

4、在background.ts webPreferences新增

```
preload: path.join(__dirname, 'preload.js'), //预加载
```

无边框模式下自定义关闭、最大化、还原、最小化 完整代码 请拉取代码查看components Header.vue

版本升级

1、安装

```
yarn add electron-updater
```

2、src目录下新建upgrade.ts

```
import { ipcMain } from 'electron'
import { autoUpdater } from 'electron-updater'
let mainWindow: any = null
export function upgradeHandle(window: any, feedUrl: any) {
  const msg = {
    error: '检查更新出错 ...',
    checking: '正在检查更 ...',
    updateAva: '检测到新版本 ...',
    updateNotAva: '已经是最新版本 ...',
     downloadProgress: '正在下载新版本 ...',
     downloaded: '下载完成，开始更新 ...'
  }
  mainWindow = window
  autoUpdater.autoDownload =
    process.env.VUE_APP_UPGRADE === 'automatic' ? true : false //true 自动升级 false 手动升级
  //设置更新包的地址
  autoUpdater.setFeedURL(feedUrl)
  //监听升级失败事件
  autoUpdater.on('error', function (message: any) {
    sendUpdateMessage({
      cmd: 'error',
      title: msg.error,
      message: message
    })
  })
  //监听开始检测更新事件
  autoUpdater.on('checking-for-update', function (message: any) {
    sendUpdateMessage({
      cmd: 'checking-for-update',
      title: msg.checking,
      message: message
    })
```

```
  })
  //监听发现可用更新事件
  autoUpdater.on('update-available', function (message: any) {
    sendUpdateMessage({
      cmd: 'update-available',
      title: msg.updateAva,
      message: message
    })
  })
  //监听没有可用更新事件
  autoUpdater.on('update-not-available', function (message: any) {
    sendUpdateMessage({
      cmd: 'update-not-available',
      title: msg.updateNotAva,
      message: message
    })
  })
  // 更新下载进度事件
  autoUpdater.on('download-progress', function (message: any) {
    sendUpdateMessage({
      cmd: 'download-progress',
      title: msg.downloadProgress,
      message: message
    })
  })
  //监听下载完成事件
  autoUpdater.on(
    'update-downloaded',
    function (
      event: any,
      releaseNotes: any,
      releaseName: any,
      releaseDate: any,
      updateUrl: any
    ) {
      sendUpdateMessage({
        cmd: 'update-downloaded',
        title: msg.downloaded,
        message: {
          releaseNotes,
```

```
            releaseName,
            releaseDate,
            updateUrl
          }
        })
        //退出并安装更新包
        autoUpdater.quitAndInstall()
      }
    )
    //接收渲染进程消息，开始检查更新
    ipcMain.on('checkForUpdate', () => {
      //执行自动更新检查
      autoUpdater.checkForUpdates()
    })
    ipcMain.on('downloadUpdate', () => {
      // 下载
      autoUpdater.downloadUpdate()
    })
}
//给渲染进程发送消息
function sendUpdateMessage(text: any) {
  mainWindow.webContents.send('message', text)
}

```

3、background.ts增加

```
import { upgradeHandle } from '@/upgrade'
if (process.env.NODE_ENV === 'production') {
    //正式
    createProtocol('app')
    win.loadURL('app://./index.html') // Load the index.html when not in development
    if (process.env.VUE_APP_UPLOAD !== '')
      upgradeHandle(win, process.env.VUE_APP_UPLOAD) //检测版本更新
    //win.webContents.openDevTools()
  }
```

4、通过.env.development/.env.production设置 开发与正式环境

```
# 配置生产环境 系统变量 建议配置一些 常用 或者 数据较单一的配置
NODE_ENV=production
# 版本号  应与package.json version 保持统一
VUE_APP_VERSION=0.1.0
```

```
# 升级地址
VUE_APP_UPLOAD= http://192.168.1.51:8011/upload/
# 升级方式 automatic 自动升级 manual 手动升级
VUE_APP_UPGRADE=manual

```

5、将打包后的dist_electron 目录下的 exe文件 和 latest.yml上传到 服务器
VUE_APP_UPLOAD
解决打包后白屏

```
const router = createRouter({
  // history: createWebHistory(process.env.BASE_URL),
  history: process.env.IS_ELECTRON
    ? createWebHashHistory(process.env.BASE_URL)
    : createWebHistory(process.env.BASE_URL), //解决打包后白屏
  routes
})
```

更换图标
window
如下方式可解决打包后窗口左上角图标不显示问题
1、将生成的icon粘贴到当前目录
2、修改vue.config.js
"win": {
    "icon": "build/icons/icon.ico" （为256*256图片）
  }
3、运行yarn run electron:serve or yarn run electron:build

mac
图标格式icns
推荐在线转换格式 https://cloudconvert.com/png-to-icns