前端学习

AVUE-CLI

后台模版框架讲解文档

一个很多骚操作的前端框架

目　录

小知识2

项目必读

介绍

avue-cli是一款基于avue和element-ui完全开源、免费的企业后端产品前端集成解决方案，采用最新的前端技术栈，已经准备好了大部分的项目准备工作，你可以快速进行二次开发

预览

预览

开发

```
# 克隆项目
git clone https://gitee.com/smallweigit/avue-cli.git

# 进入项目
cd avue-cli

# 安装依赖
npm install --registry=https://registry.npm.taobao.org

# 启动服务
npm run serve
```

功能

- 登录/注销
  - 用户名登录
  - 验证码登录
  - 第三方登陆(QQ,微信)
  - 人脸识别登录
- 错误的日志记录
- 灵活的10+多款主题自由配置
- 路由权限、菜单权限、登录权限

- 本地化持久存储api
- 页面缓冲
- 面向全屏幕尺寸的响应式适配能力
- 对国际化的支持
- 自动刷新token等机制
- 全新的前端错误日志监控机制
- 前端路由动态服务端加载
- 无限极动态路由加载
- 模块的可拆卸化,达到开箱即用
- 更多。。。

问答

有关问题和支持，请使用issues或加入QQ群.

issues

打开问题之前，请务必提供详细的问题过程和截图，不符合准则的问题将会被拒绝.

Changelog

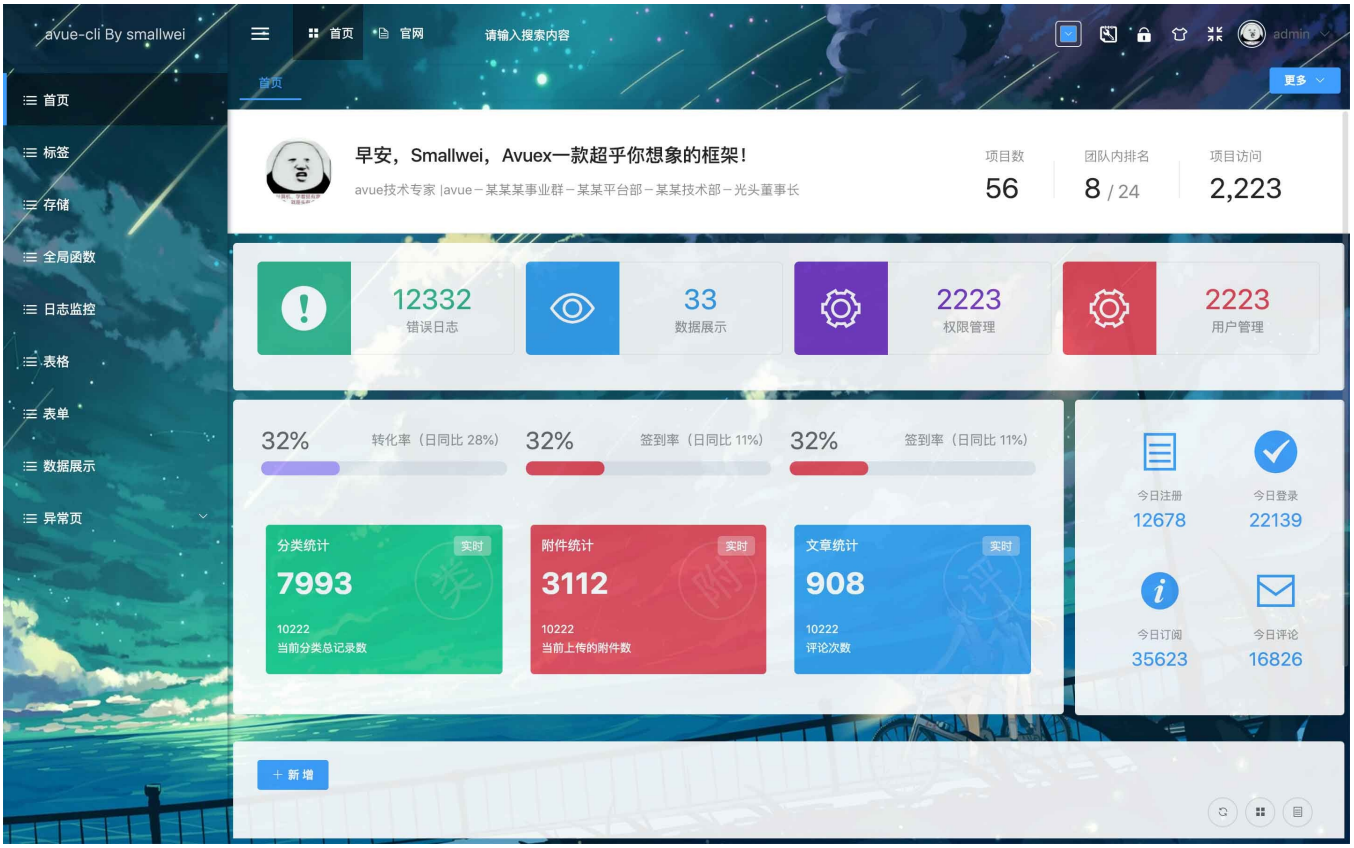
每个版本的详细更改记录在发行说明.

Page

登陆

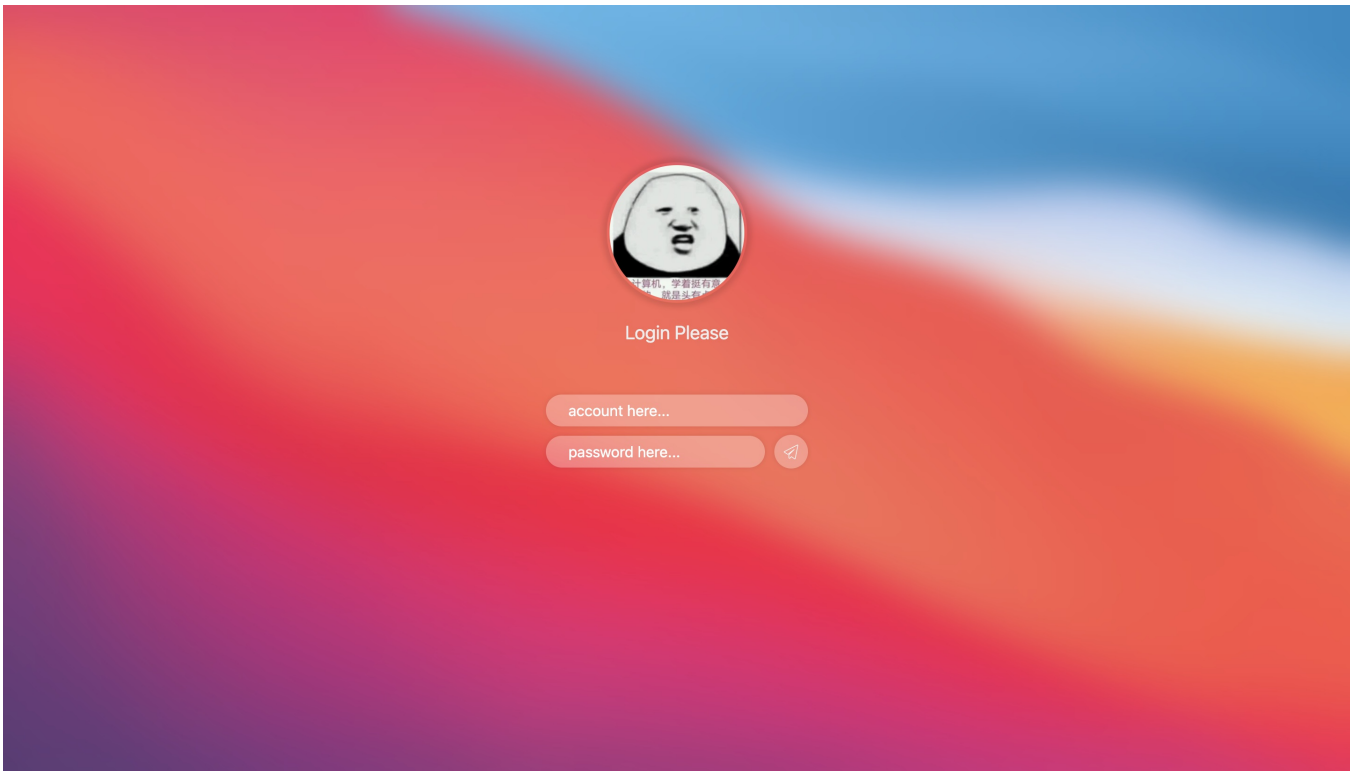
主页

炫酷主题

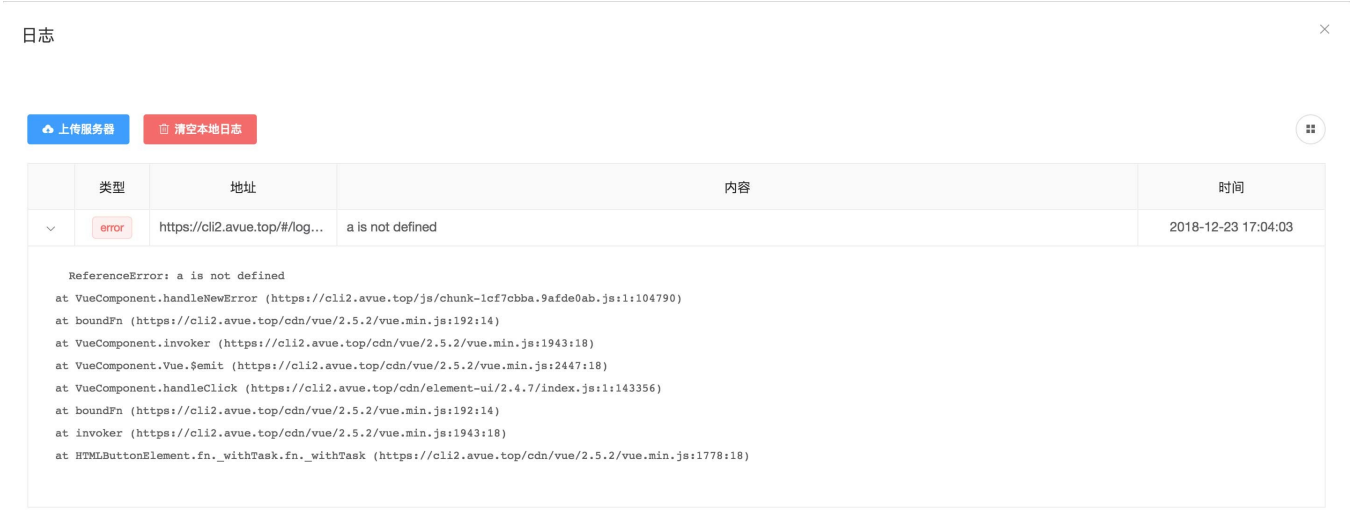


MAC主题

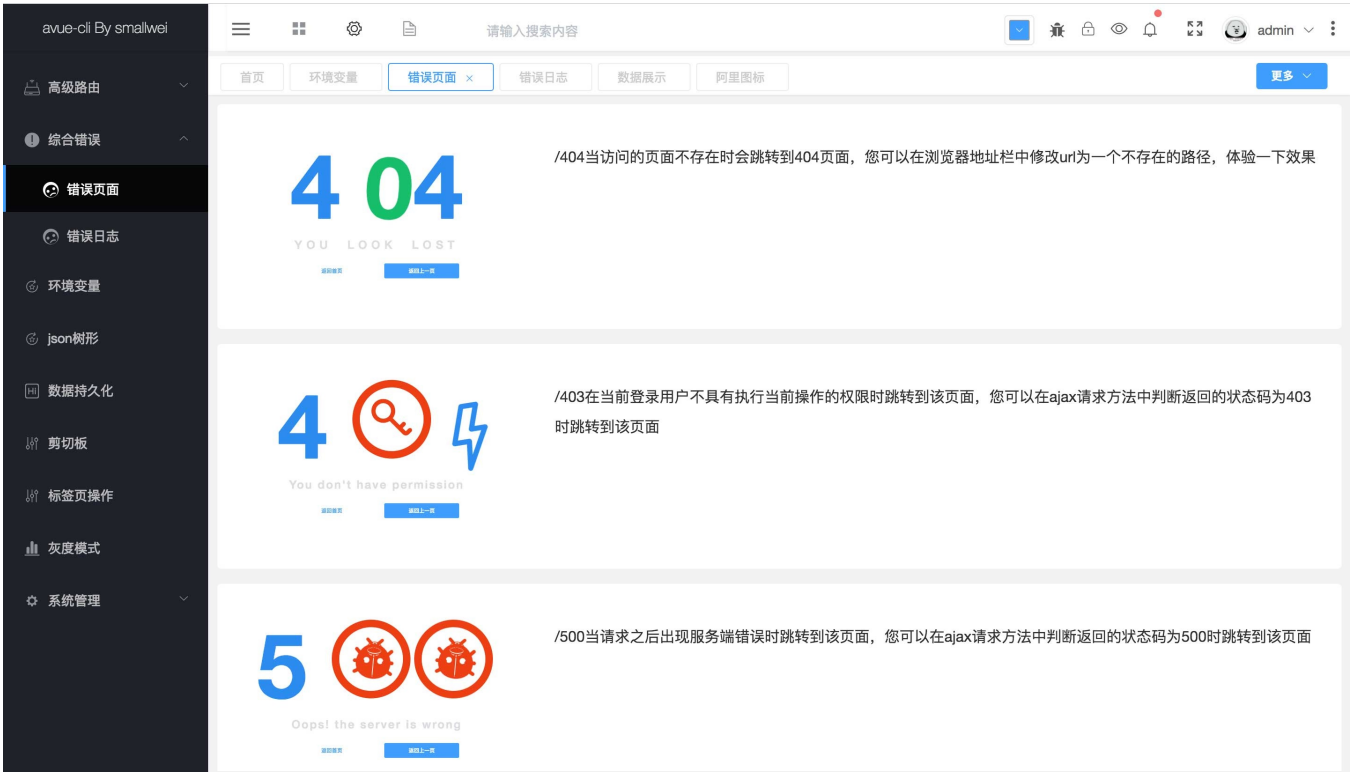

2021年08月25日 17:01
多级菜...
缓冲
官方网...
参数
全局因...
标签
error4...
存储
error4...
日志监控
error5...
返回顶部
关于
图钉


Login Please
account here...
password here...

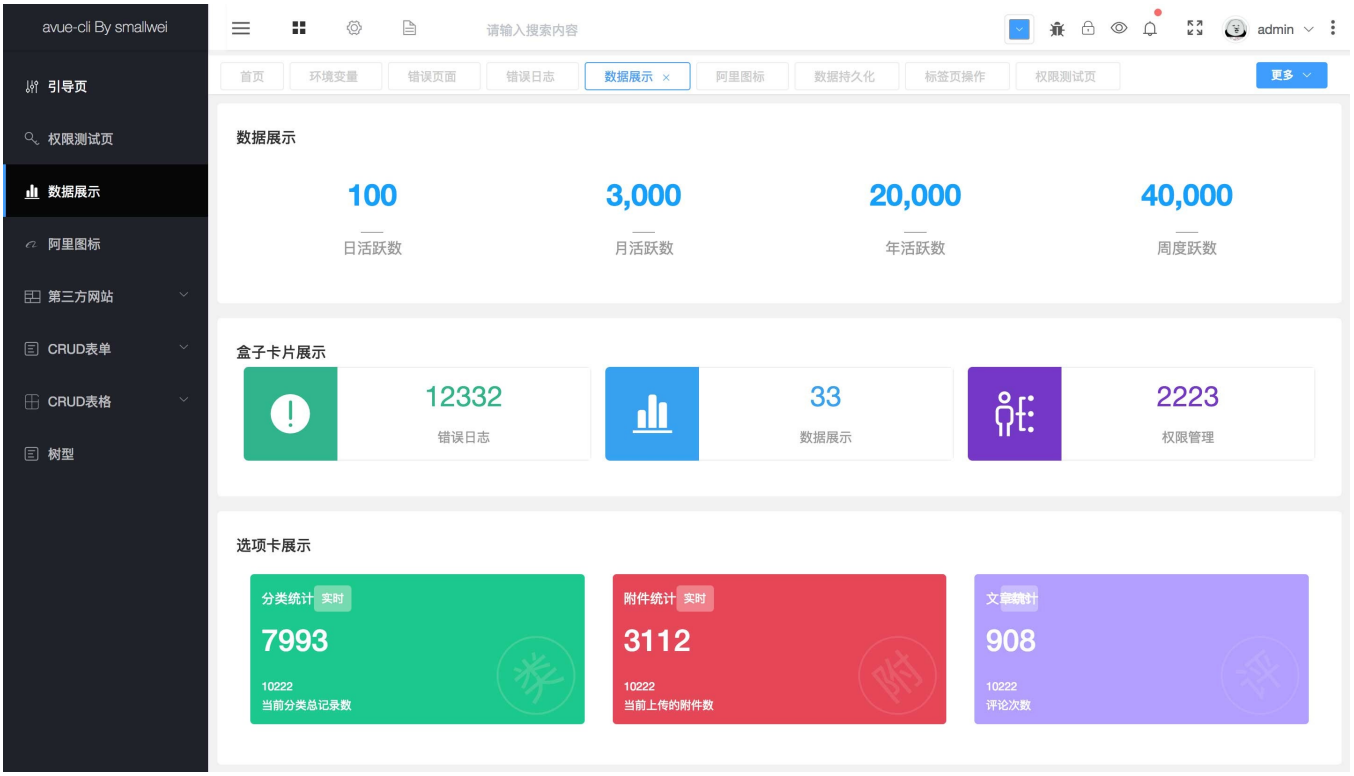
日志监控


日志
上传服务器
清空本地日志
类型
地址
内容
时间
error
https://cli2.avue.top/#/log...
a is not defined
2018-12-23 17:04:03
ReferenceError: a is not defined
at VueComponent.handleNewError (https://cli2.avue.top/js/chunk-1cf7cbba.9afde0ab.js:1:104790)
at boundFn (https://cli2.avue.top/cdn/vue/2.5.2/vue.min.js:192:14)
at VueComponent.invoker (https://cli2.avue.top/cdn/vue/2.5.2/vue.min.js:1943:18)
at VueComponent.Vue.$emit (https://cli2.avue.top/cdn/vue/2.5.2/vue.min.js:2447:18)
at VueComponent.handleClick (https://cli2.avue.top/cdn/element-ui/2.4.7/index.js:1:143356)
at boundFn (https://cli2.avue.top/cdn/vue/2.5.2/vue.min.js:192:14)
at invoker (https://cli2.avue.top/cdn/vue/2.5.2/vue.min.js:1943:18)
at HTMLButtonElement.fn._withTask.fn._withTask (https://cli2.avue.top/cdn/vue/2.5.2/vue.min.js:1778:18)

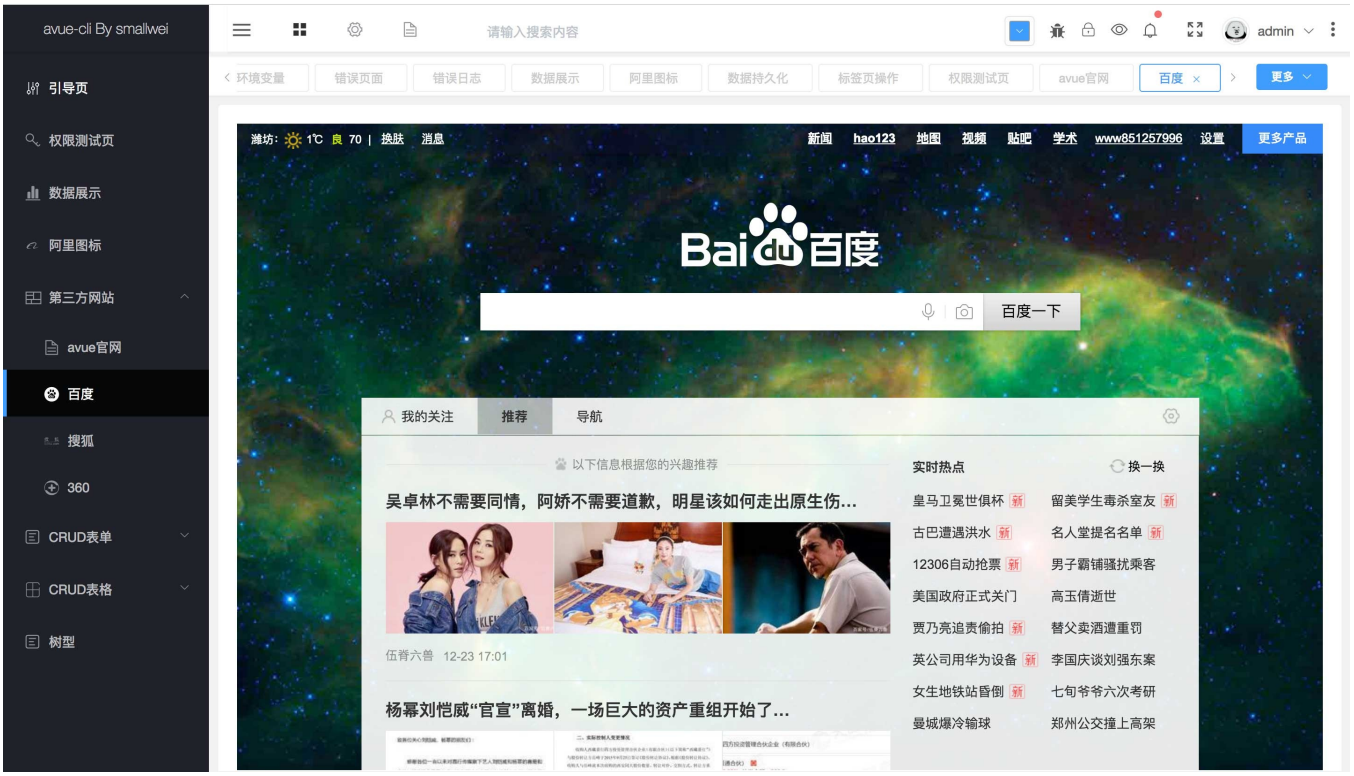
错误提示


avue-cli By smallwei
请输入搜索内容
admin
高级路由
综合错误
错误页面
错误日志
环境变量
json树形
数据持久化
剪切板
标签页操作
灰度模式
系统管理
首页
环境变量
错误页面
错误日志
数据展示
阿里图标
更多
404
YOU LOOK LOST
/404当访问的页面不存在时会跳转到404页面，您可以在浏览器地址栏中修改url为一个不存在的路径，体验一下效果
You don't have permission
/403在当前登录用户不具有执行当前操作的权限时跳转到该页面，您可以在ajax请求方法中判断返回的状态码为403时跳转到该页面
500
Oops! the server is wrong
/500当请求之后出现服务端错误时跳转到该页面，您可以在ajax请求方法中判断返回的状态码为500时跳转到该页面

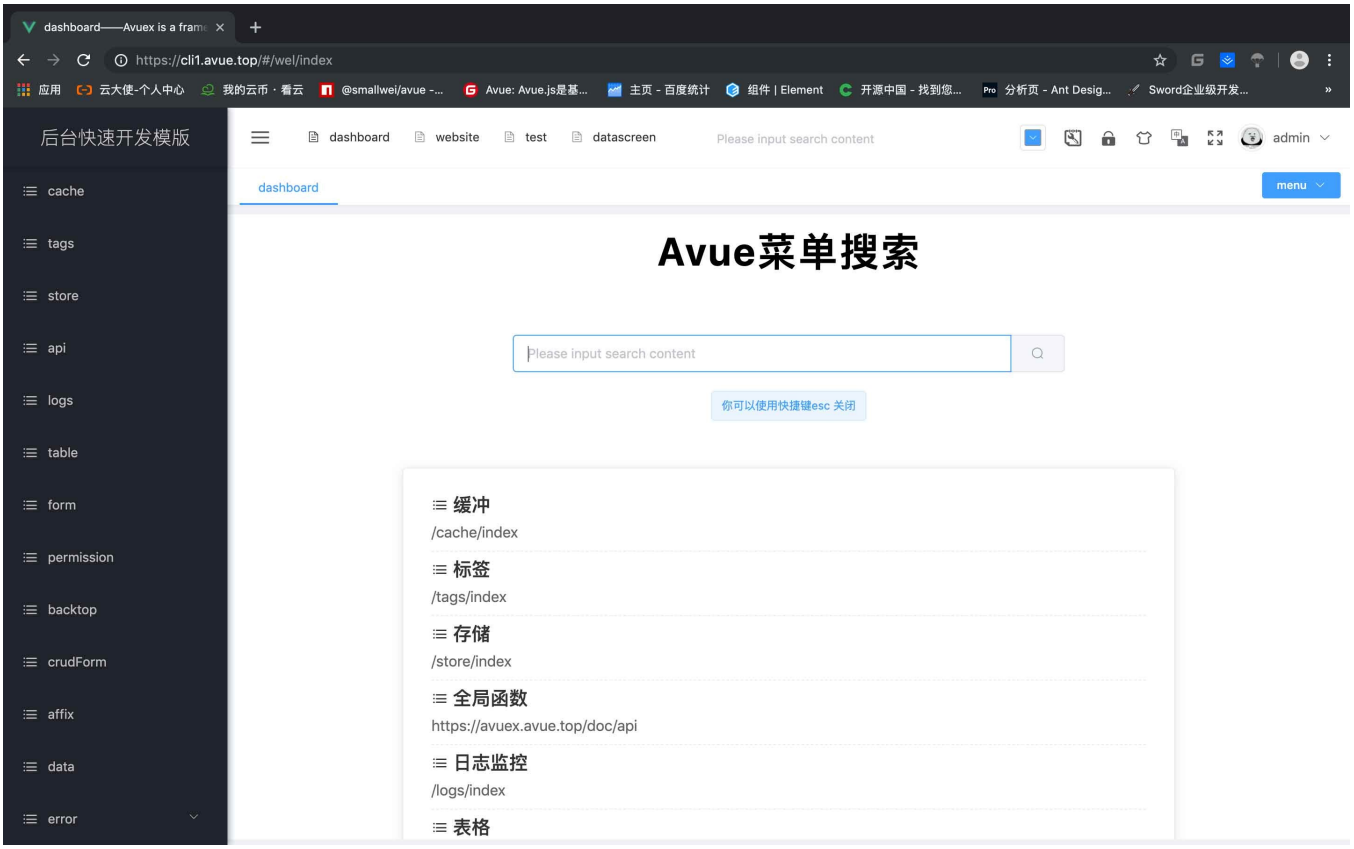
数据展示


avue-cli By smallwei
请输入搜索内容
admin
引导页
权限测试页
数据展示
阿里图标
第三方网站
CRUD表单
CRUD表格
树型
首页
环境变量
错误页面
错误日志
数据展示
阿里图标
数据持久化
标签页操作
权限测试页
更多
数据展示
100
日活跃数
3,000
月活跃数
20,000
年活跃数
40,000
周度跃数
盒子卡片展示
12332
错误日志
33
数据展示
2223
权限管理
选项卡展示
分类统计 实时
7993
10222
当前分类总记录数
附件统计 实时
3112
10222
当前上传的附件数
908
10222
评论次数

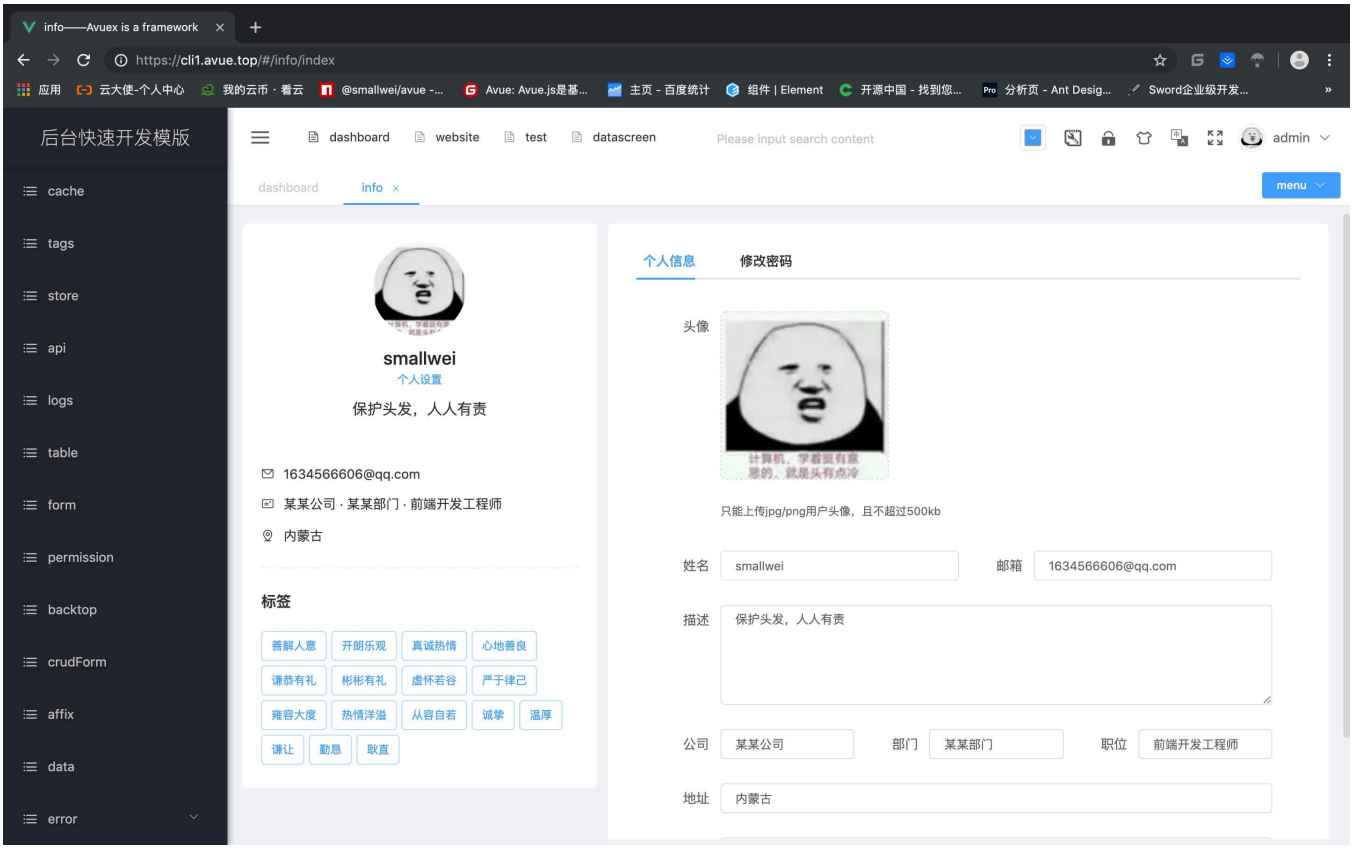
第三方网站


avue-cli By smallwei
请输入搜索内容
admin
引导页
权限测试页
数据展示
阿里图标
第三方网站
avue官网
百度
搜狐
360
CRUD表单
CRUD表格
树型
环境变量
错误页面
错误日志
数据展示
阿里图标
数据持久化
标签页操作
权限测试页
avue官网
百度
更多
潍坊: 1℃ 良 70 | 换肤 消息
新闻 hao123 地图 视频 贴吧 学术 www851257996 设置
更多产品
Bai du 百度
百度一下
我的关注
推荐
导航
以下信息根据您的兴趣推荐
吴卓林不需要同情，阿娇不需要道歉，明星该如何走出原生伤...
伍胥六兽 12-23 17:01
杨幂刘恺威“官宣”离婚，一场巨大的资产重组开始了...
实时热点
换一换
皇马卫冕世俱杯 新
古巴遭遇洪水 新
12306自动抢票 新
美国政府正式关门
贾乃亮追责偷拍 新
英公司用华为设备 新
女生地铁站昏倒 新
曼城爆冷输球
留美学生毒杀室友 新
名人堂提名名单 新
男子霸铺骚扰乘客
高玉倩逝世
替父卖酒遭重罚
李国庆谈刘强东案
七旬爷爷六次考研
郑州公交撞上高架

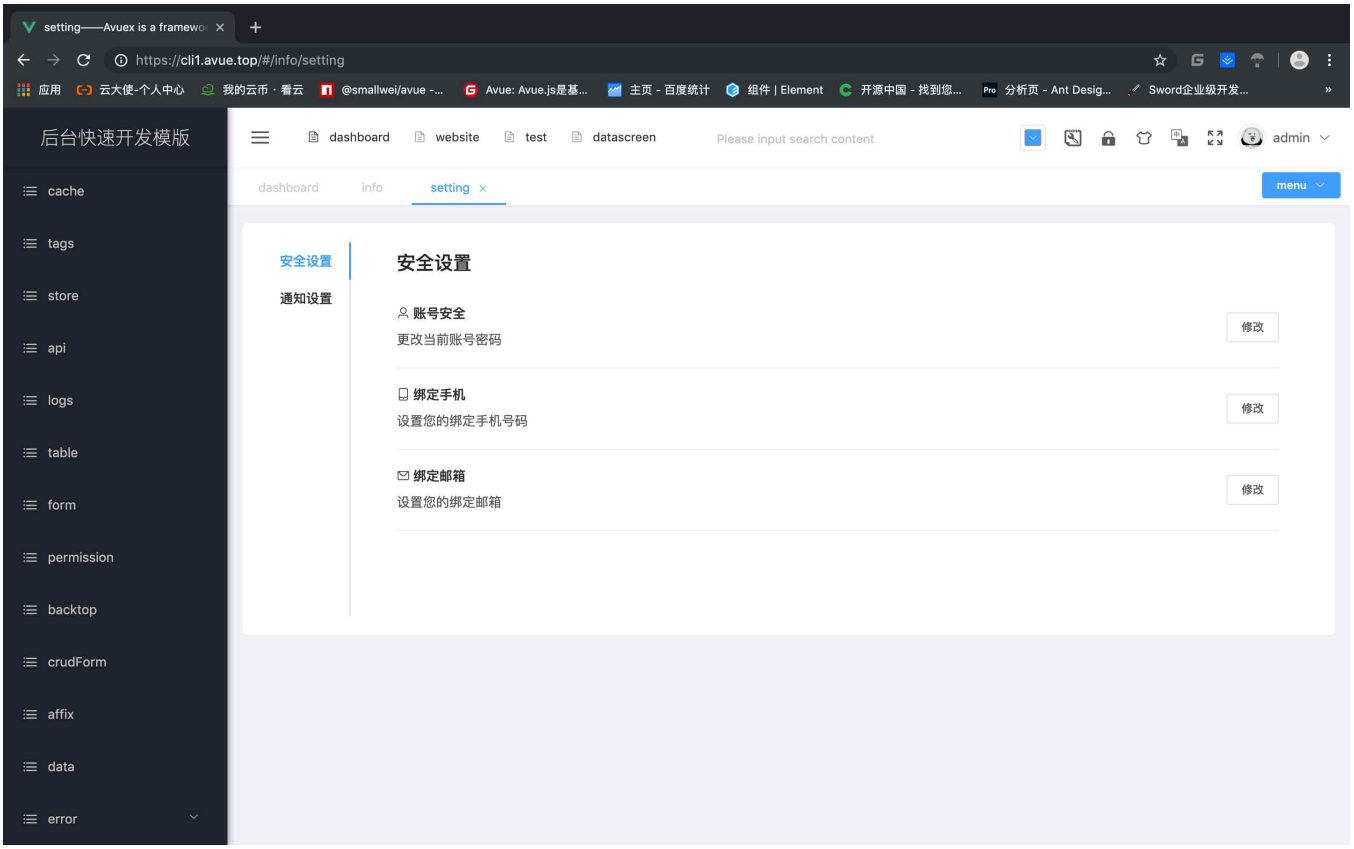
全局搜索


dashboard——Avuex is a fram
https://cli1.avue.top/#/wel/index
后台快速开发模版
dashboard
website
test
datascreen
Please input search content
admin
menu
cache
tags
store
api
logs
table
form
permission
backtop
crudForm
affix
data
error
Avue菜单搜索
Please input search content
你可以使用快捷键esc 关闭
缓冲
/cache/index
标签
/tags/index
存储
/store/index
全局函数
https://avuex.avue.top/doc/api
日志监控
/logs/index
表格

个人中心


info——Avuex is a framework
https://cli1.avue.top/#/info/index
后台快速开发模版
dashboard
info
menu
smallwei
个人设置
保护头发，人人有责
1634566606@qq.com
某某公司 · 某某部门 · 前端开发工程师
内蒙古
标签
善解人意
开朗乐观
真诚热情
心地善良
谦恭有礼
彬彬有礼
虚怀若谷
严于律己
雍容大度
热情洋溢
从容自若
诚挚
温厚
谦让
勤恳
耿直
个人信息
修改密码
头像
只能上传jpg/png用户头像，且不超过500kb
姓名
smallwei
邮箱
1634566606@qq.com
描述
保护头发，人人有责
公司
某某公司
部门
某某部门
职位
前端开发工程师
地址
内蒙古

个人设置



License

MIT

Copyright (c) 2017-present, Smallwei

环境搭建

node环境安装

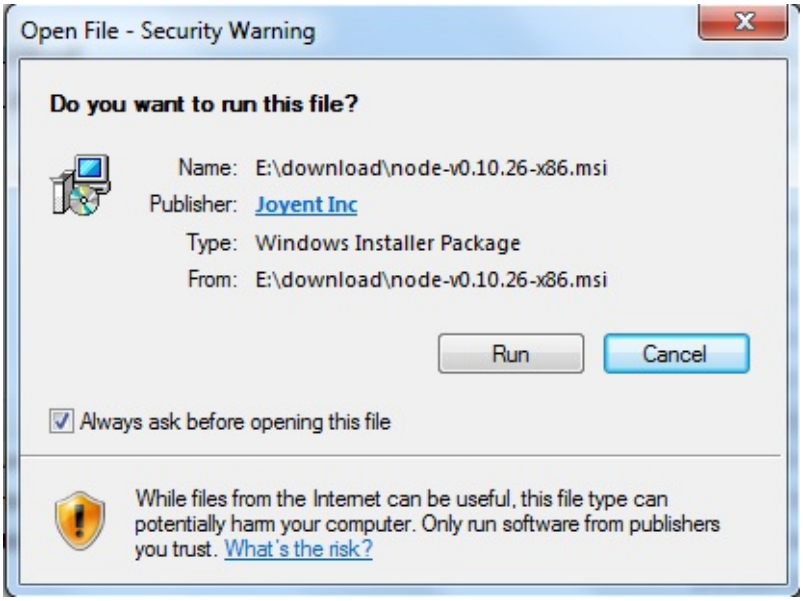
Window 上安装Node.js

32 位安装包下载地址 : https://nodejs.org/dist/v8.11.1/node-v8.11.1-x86.msi

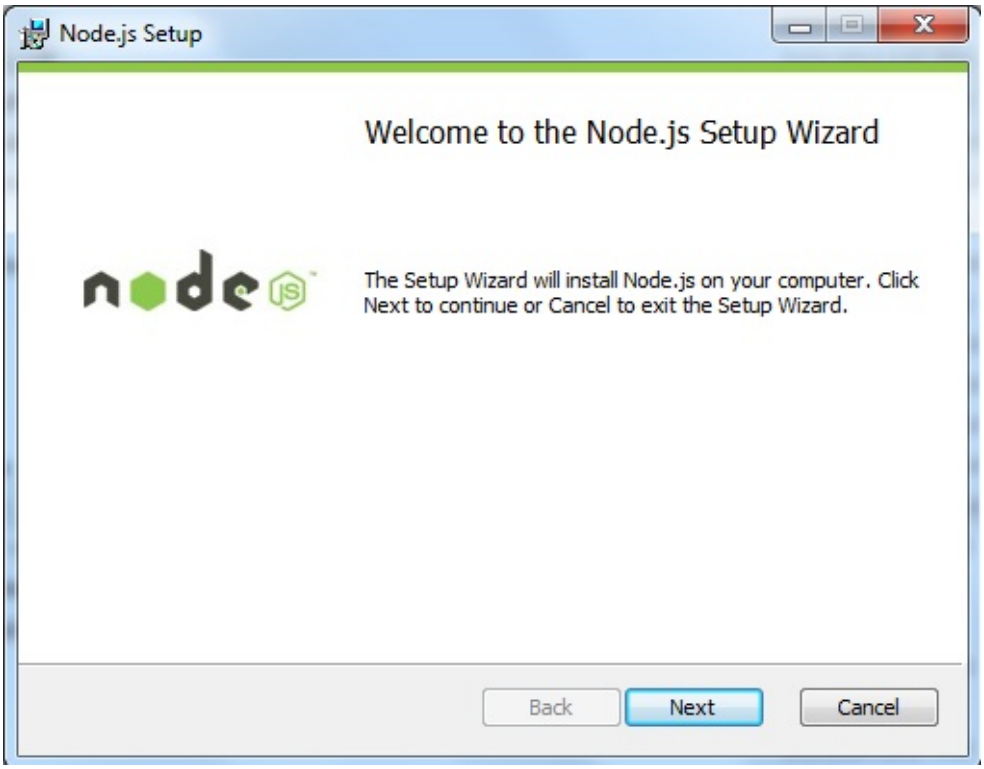
64 位安装包下载地址 : https://nodejs.org/dist/v8.11.1/node-v8.11.1-x64.msi

本文实例以node最新版本为例，其他版本类似， 安装步骤：

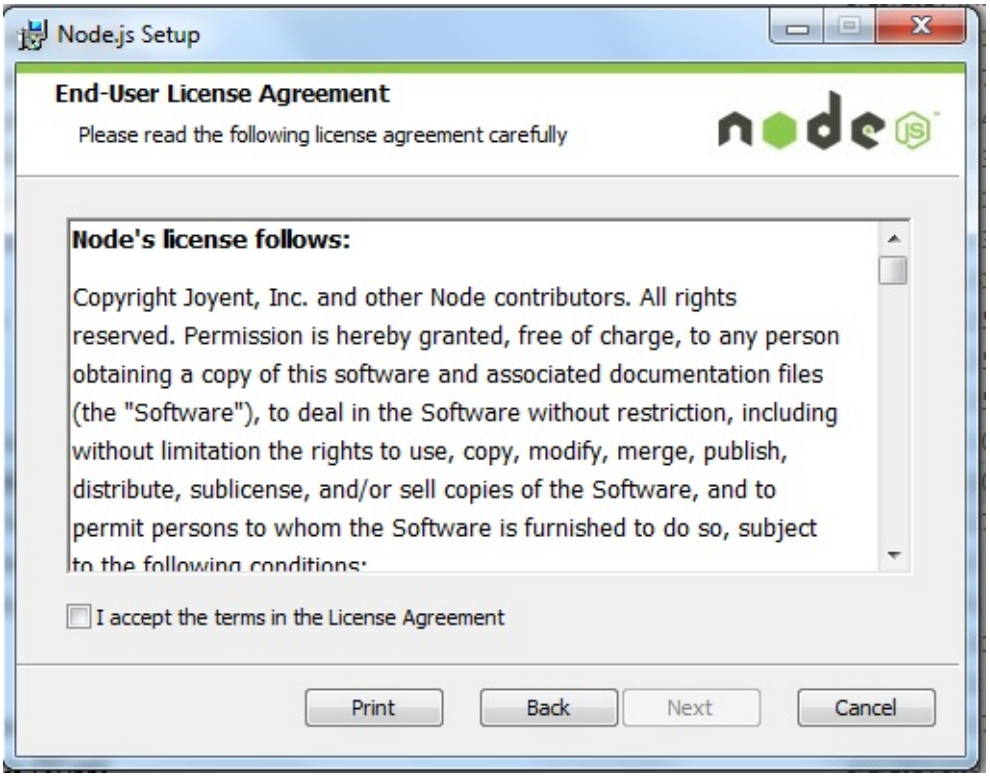
步骤 1 : 双击下载后的安装包，如下所示：



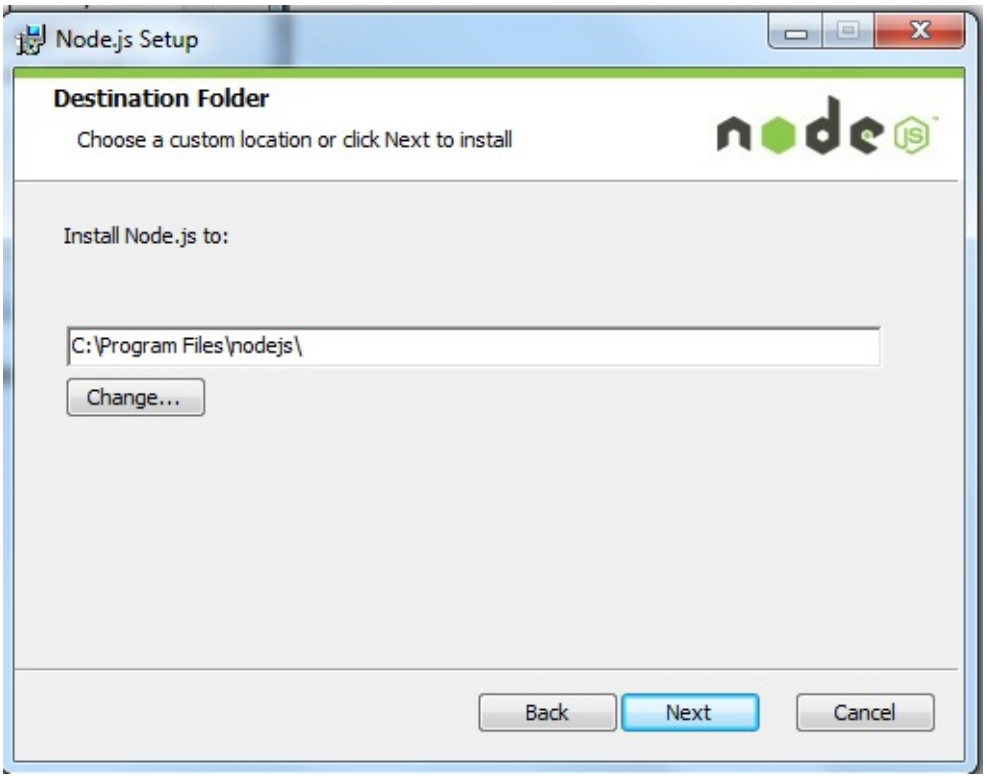
步骤 2 : 点击以上的Run(运行)，将出现如下界面：



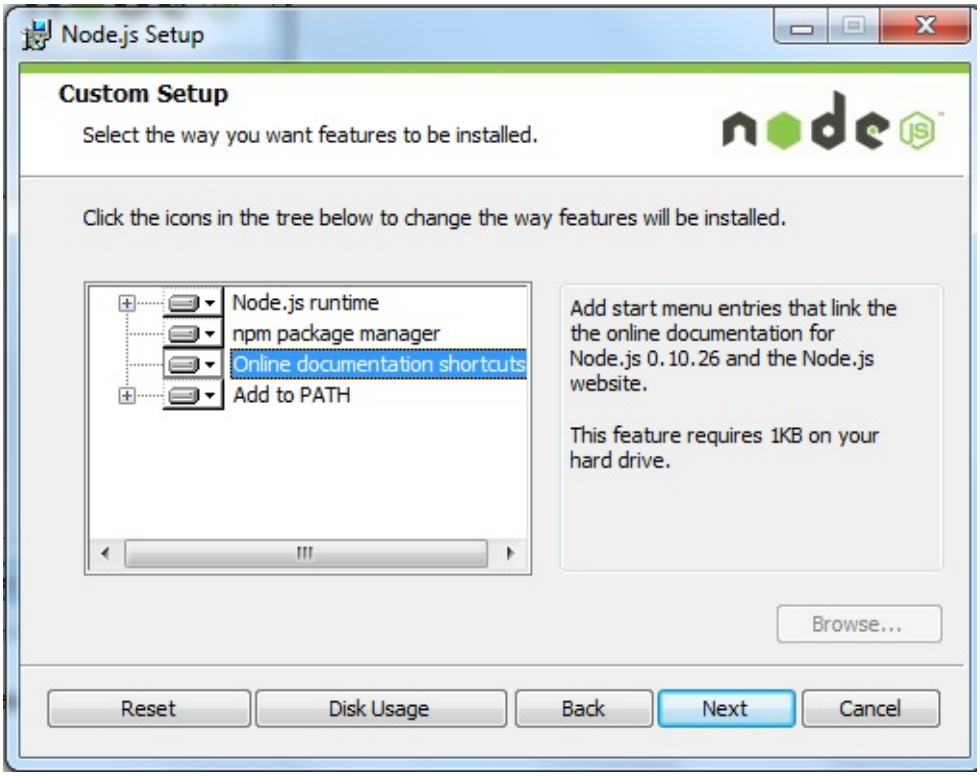
步骤 3 : 勾选接受协议选项，点击 next（ 下一步 ） 按钮 :



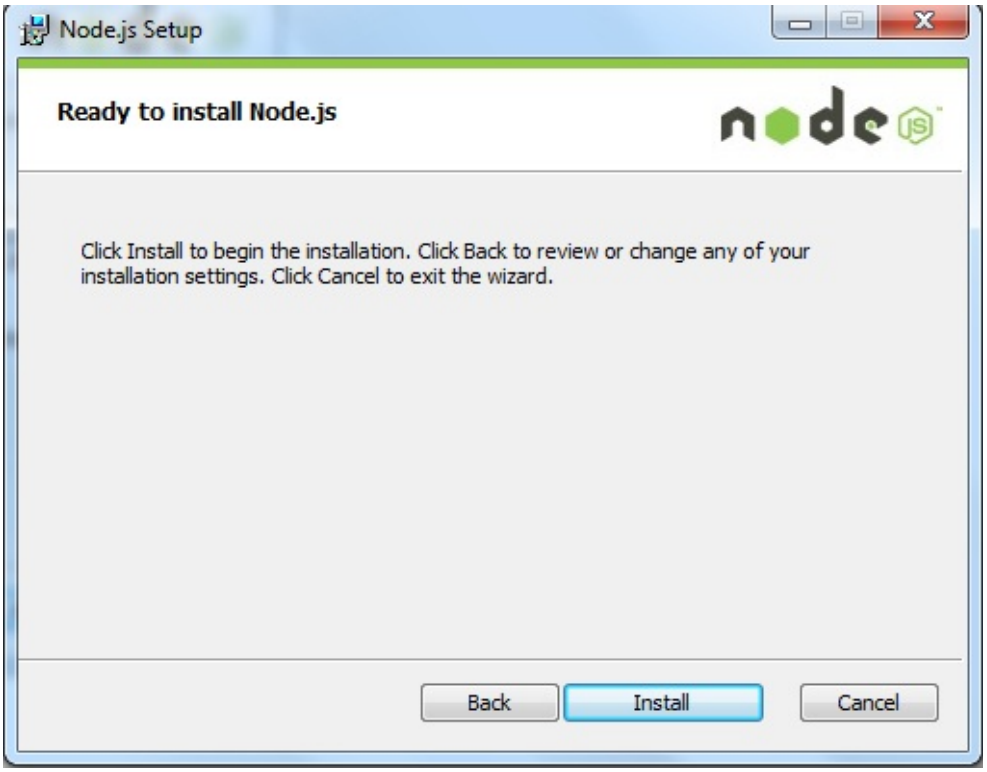
步骤 4 : Node.js默认安装目录为 "C:\Program Files\nodejs"，你可以修改目录，并点击 next（下一步）：



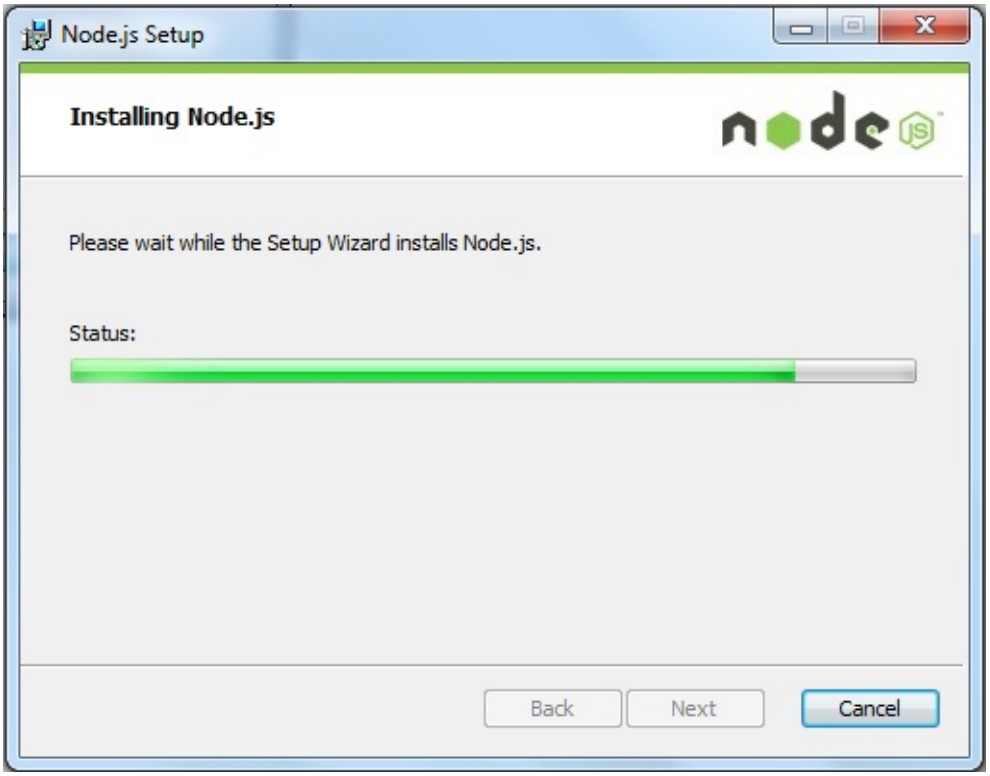
步骤 5 : 点击树形图标来选择你需要的安装模式，然后点击下一步 next（下一步）



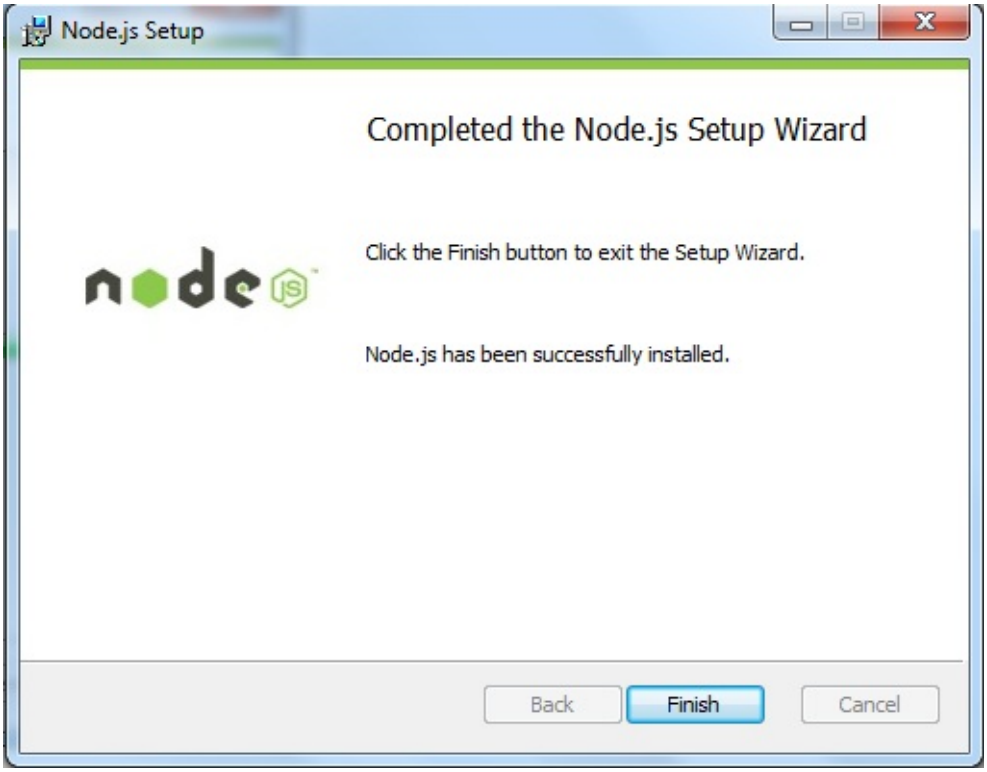
步骤 6 :点击 Install（安装） 开始安装Node.js。你也可以点击 Back（返回）来修改先前的配置。然后并点击 next（下一步）：



安装过程：

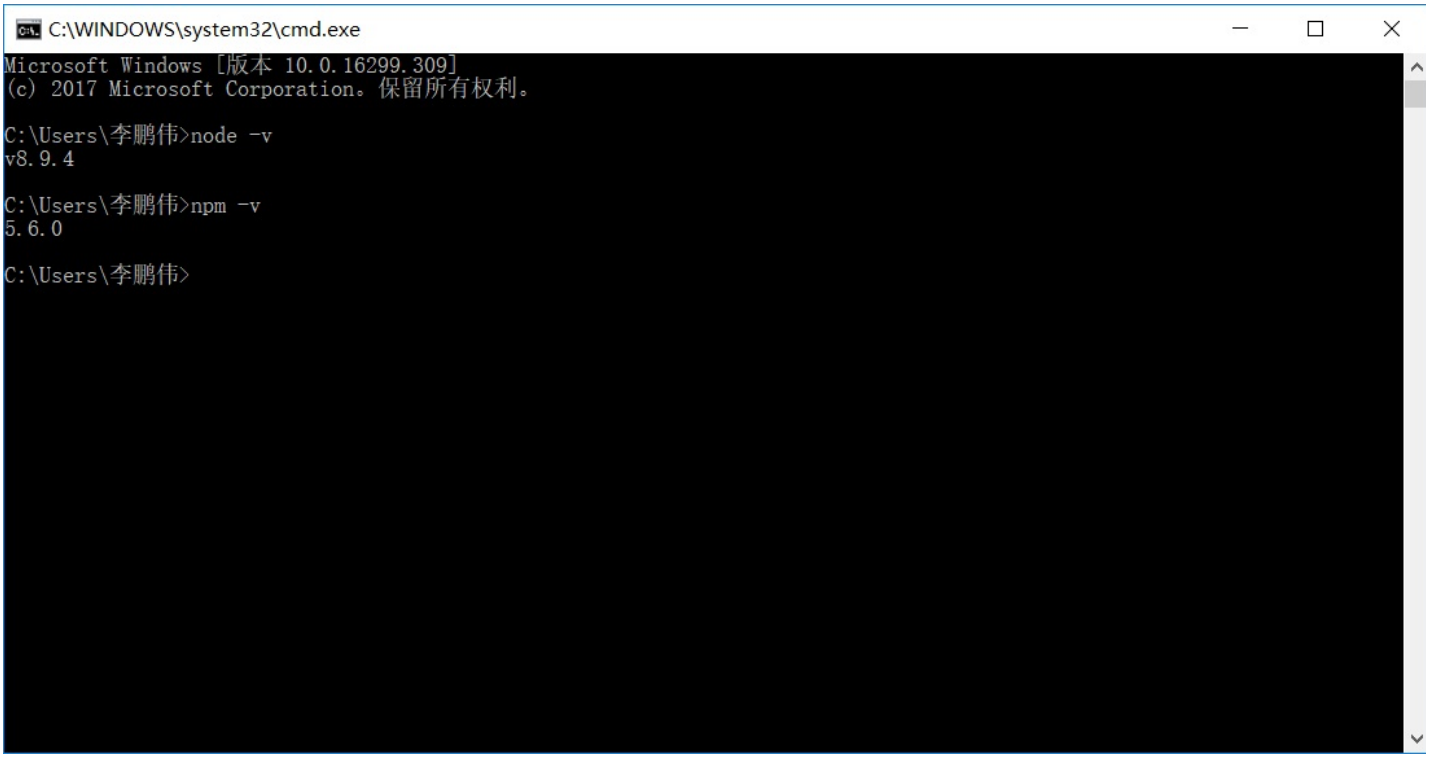


点击 Finish（完成）按钮退出安装向导。



检查Node.js版本

```
node-version
```


C:\WINDOWS\system32\cmd.exe
Microsoft Windows [版本 10.0.16299.309]
(c) 2017 Microsoft Corporation。保留所有权利。
C:\Users\李鹏伟>node -v
v8.9.4
C:\Users\李鹏伟>npm -v
5.6.0
C:\Users\李鹏伟>

webpack环境安装

使用npm 安装全局webpack

```
npm i webpack -g
```

项目中使用npm 安装webpack

进入项目目录

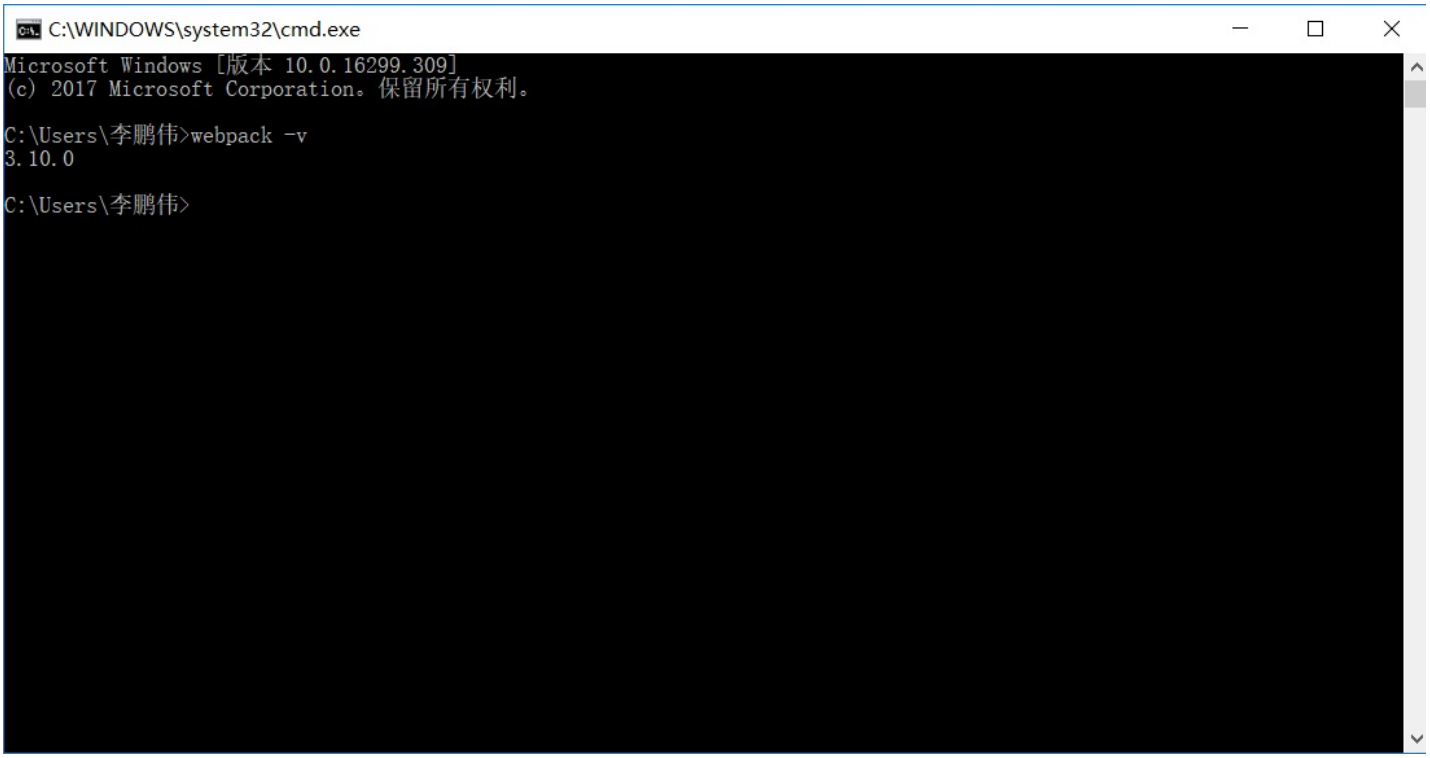
确定已经有 package.json，没有就通过 npm init 创建

安装 webpack 依赖

```
npm i webpack --save-dev
```

检查webpack版本

```
webpack-v
```



npm国内源切换

配置切换到淘宝源

```
npm config set registry https://registry.npm.taobao.org
```

使用淘宝cnpm

```
npm i -g cnpm
```

检测是否切换到了淘宝源

```
npm info underscore
.......
gitHead: 'e4743ab712b8ab42ad4ccb48b155034d02394e4d',
  dist:
   { shasum: '4f3fb53b106e6097fcf9cb4109f2a5e9bdfa5022',
     size: 34172,
     noattachment: false,
    //  有  registry.npm.taobao.org  等字样    说明切换成功
     tarball: 'http://registry.npm.taobao.org/underscore/download/underscore-1.
8.3.tgz' },
  directories: {},
  publish_time: 1427988774520 }
```

git知识学习

git安装

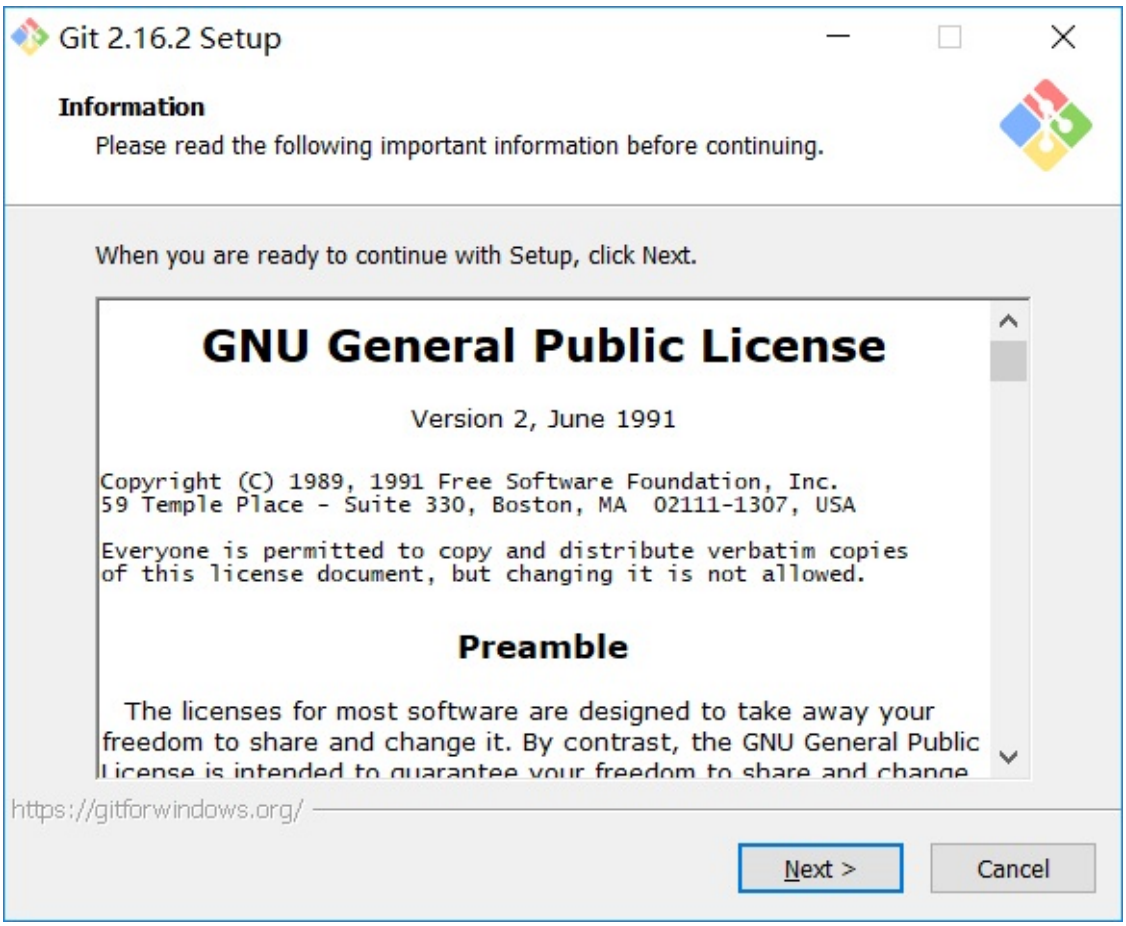
Window 上安装Git

32 位安装包下载地址 : https://github.com/git-for-windows/git/releases/download/v2.16.2.windows.1/Git-2.16.2-32-bit.exe

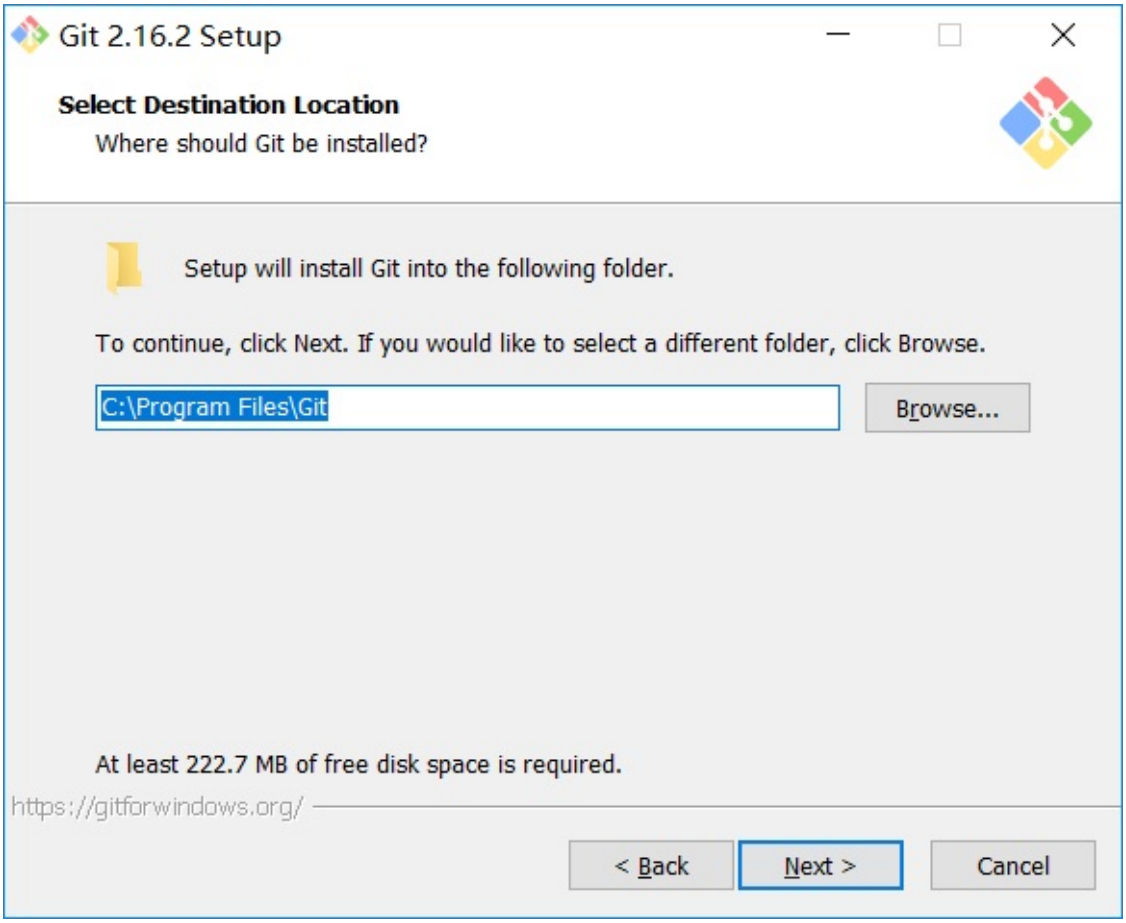
64 位安装包下载地址 : https://github.com/git-for-windows/git/releases/download/v2.16.2.windows.1/Git-2.16.2-64-bit.exe

本文实例以git最新版本为例，其他版本类似， 安装步骤：

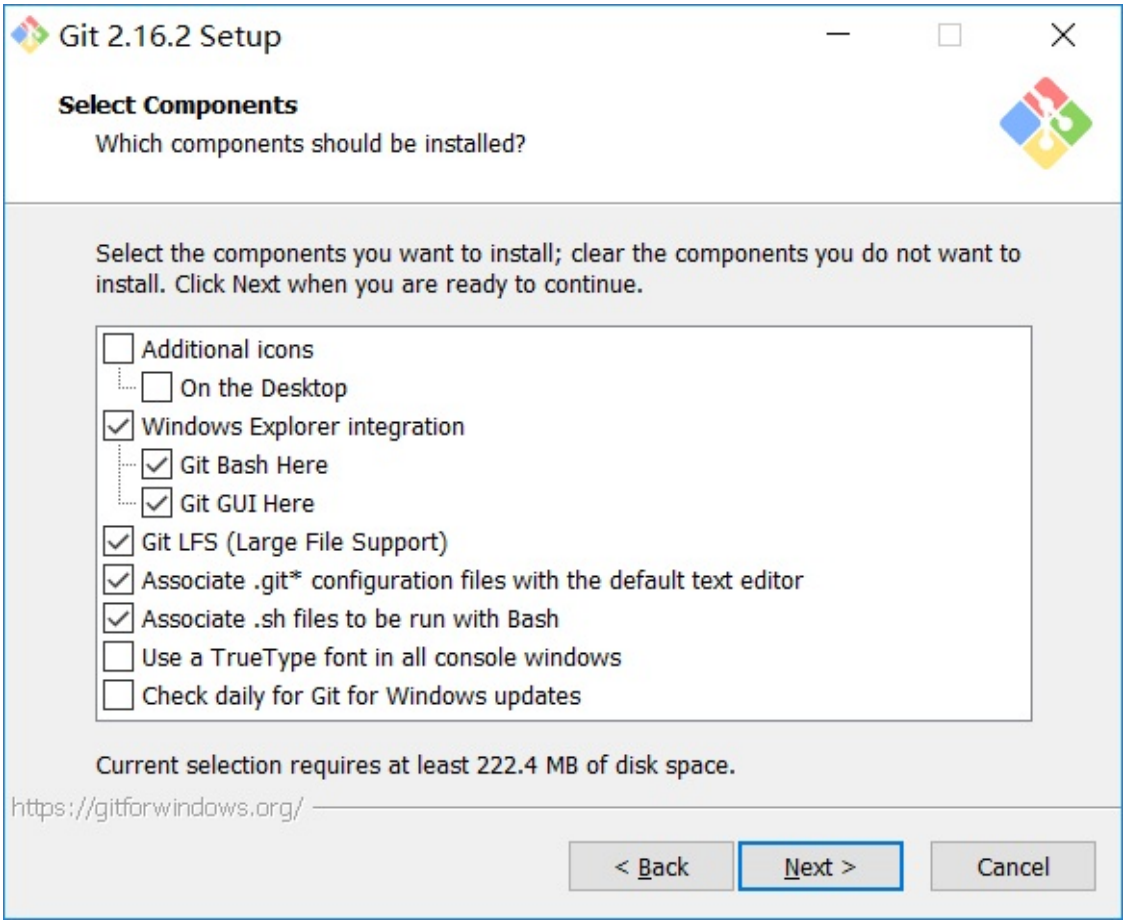
步骤 1 : 双击下载后的安装包，如下所示：



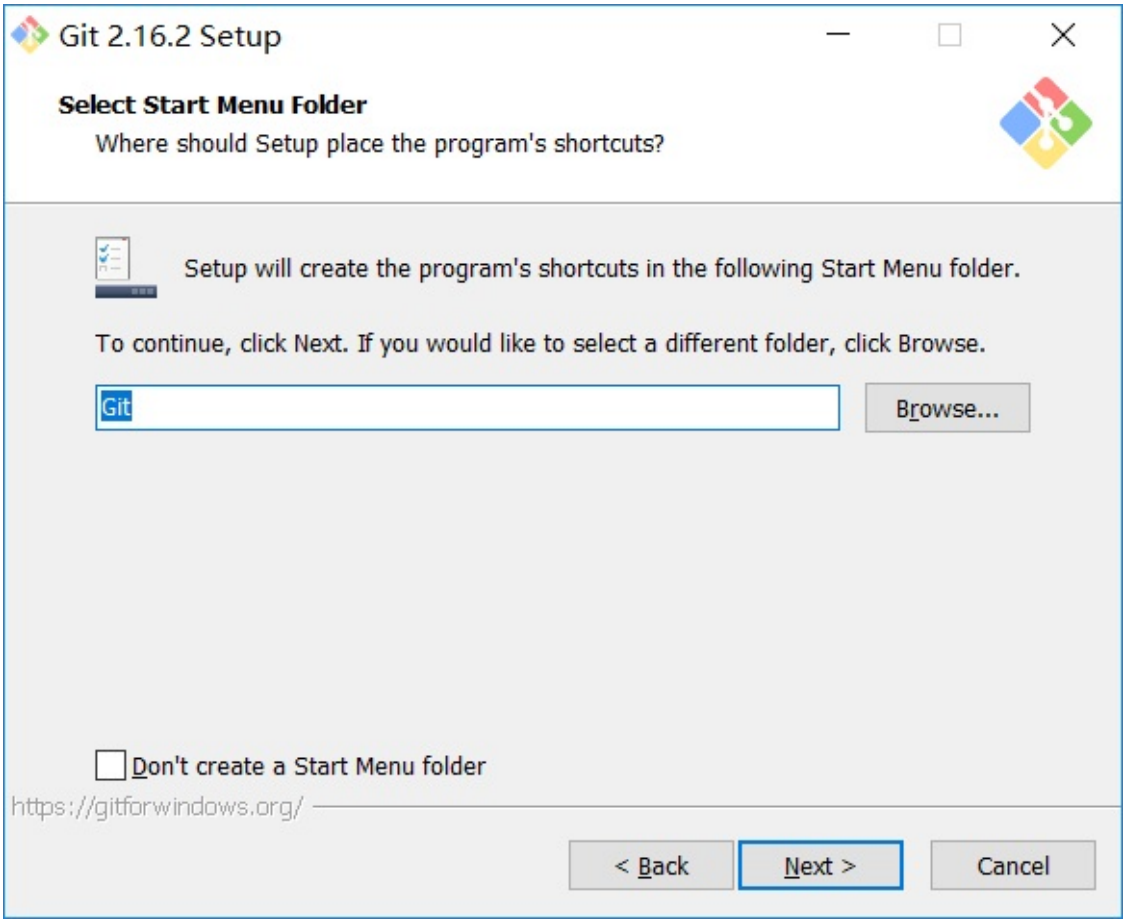
步骤2 : 选择安装路径点击next：



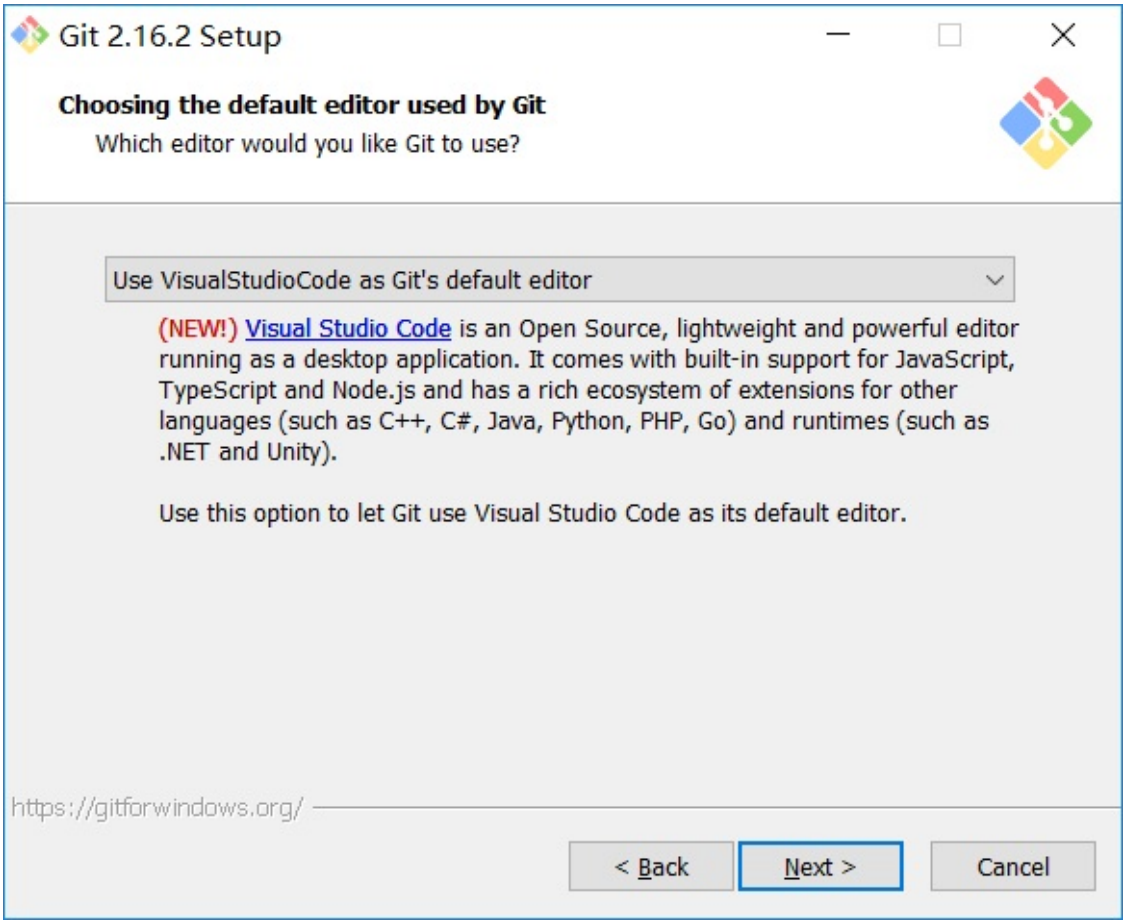
步骤3：按照默认配置点击next：



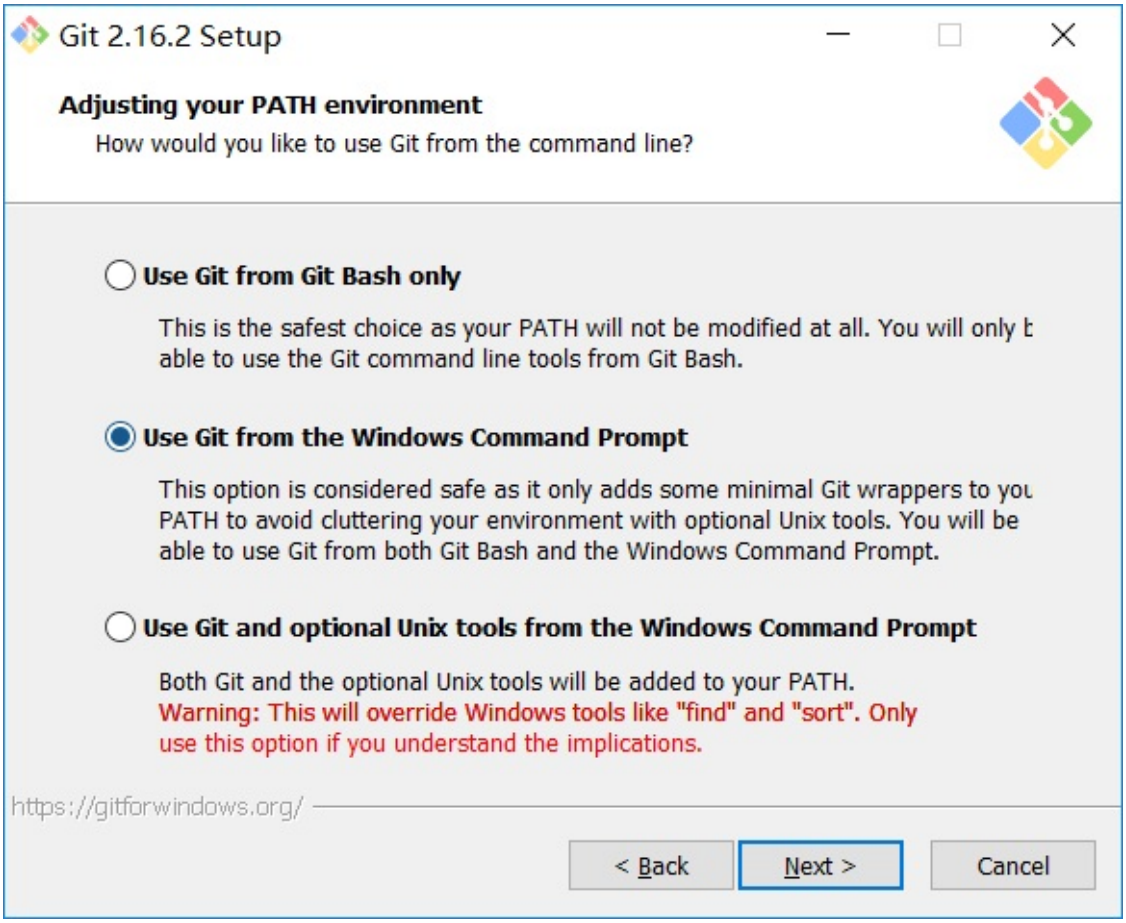
步骤4：没啥操作直接点击next：



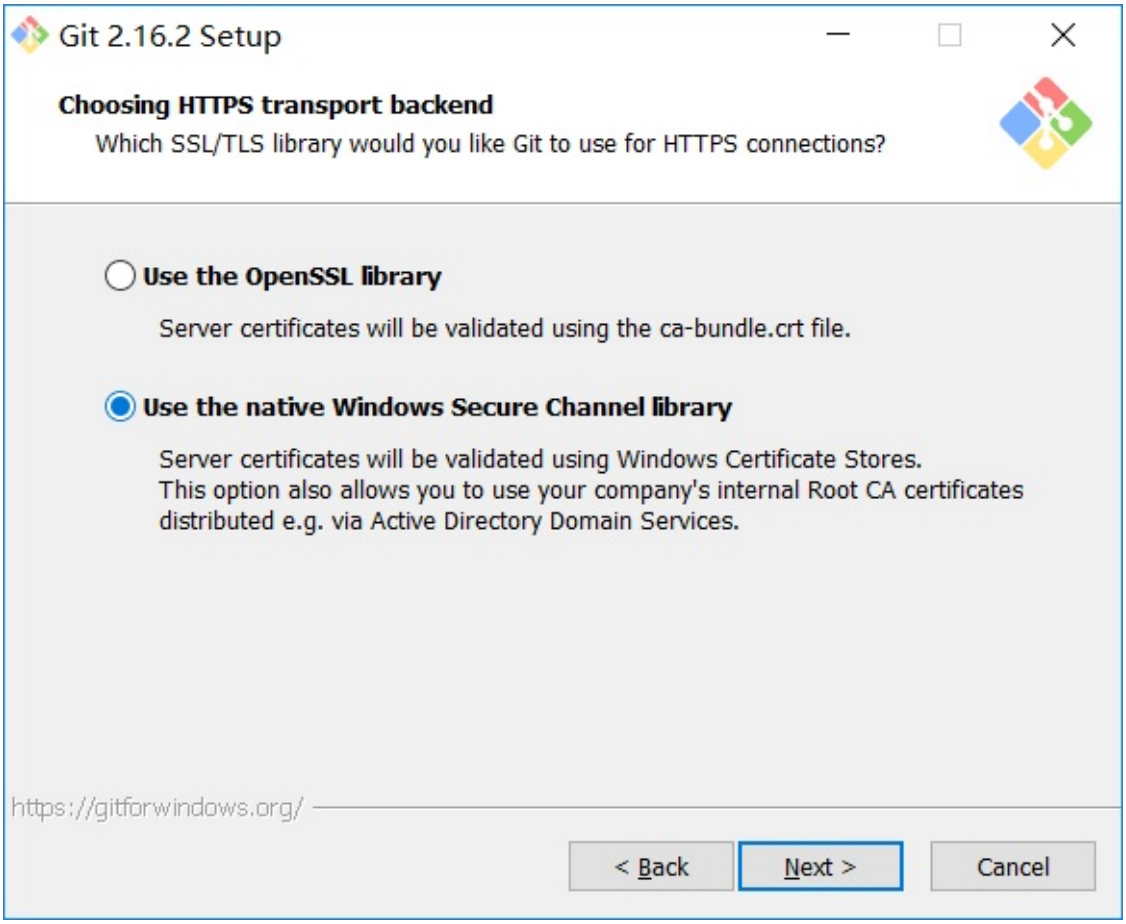
步骤5**：选择vscode作为默认，点击next：



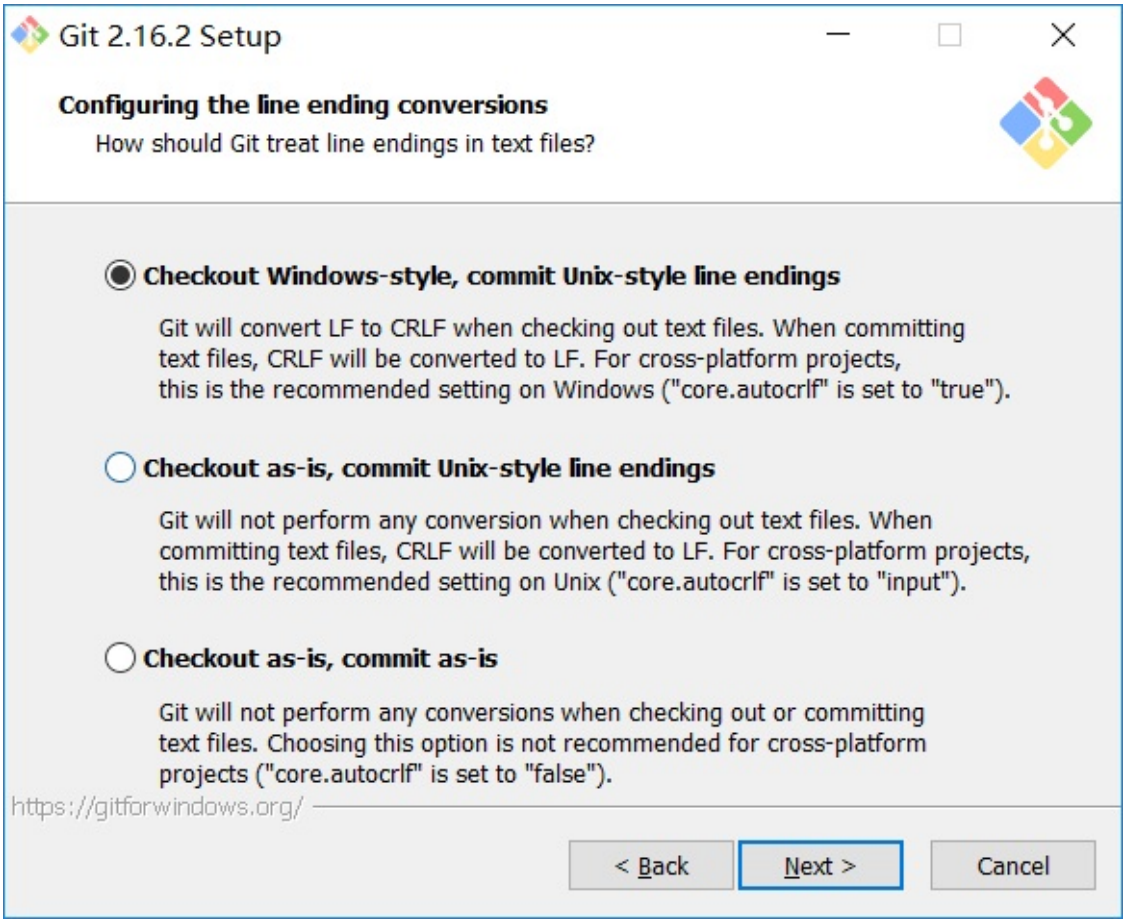
步骤6：选择第二项，点击next：



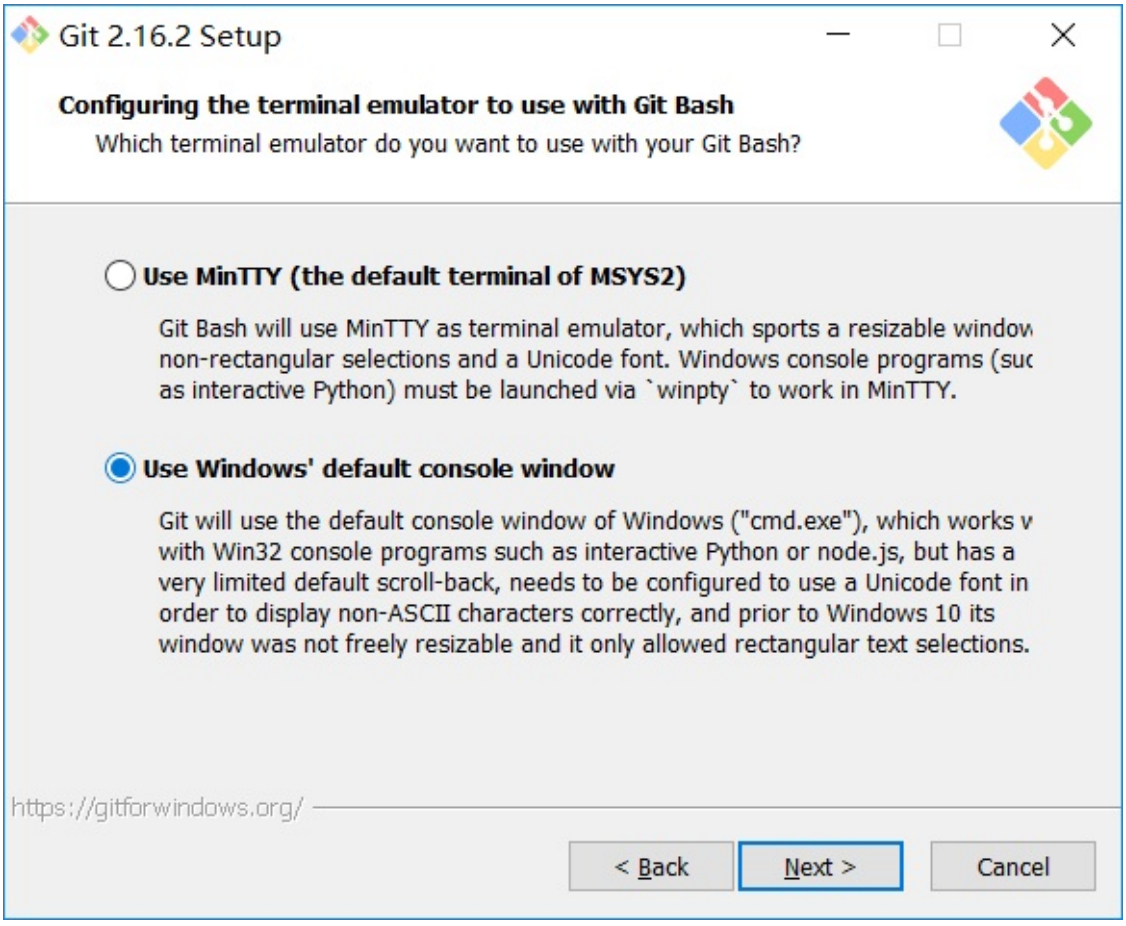
步骤7：也是选择第二项，点击next：



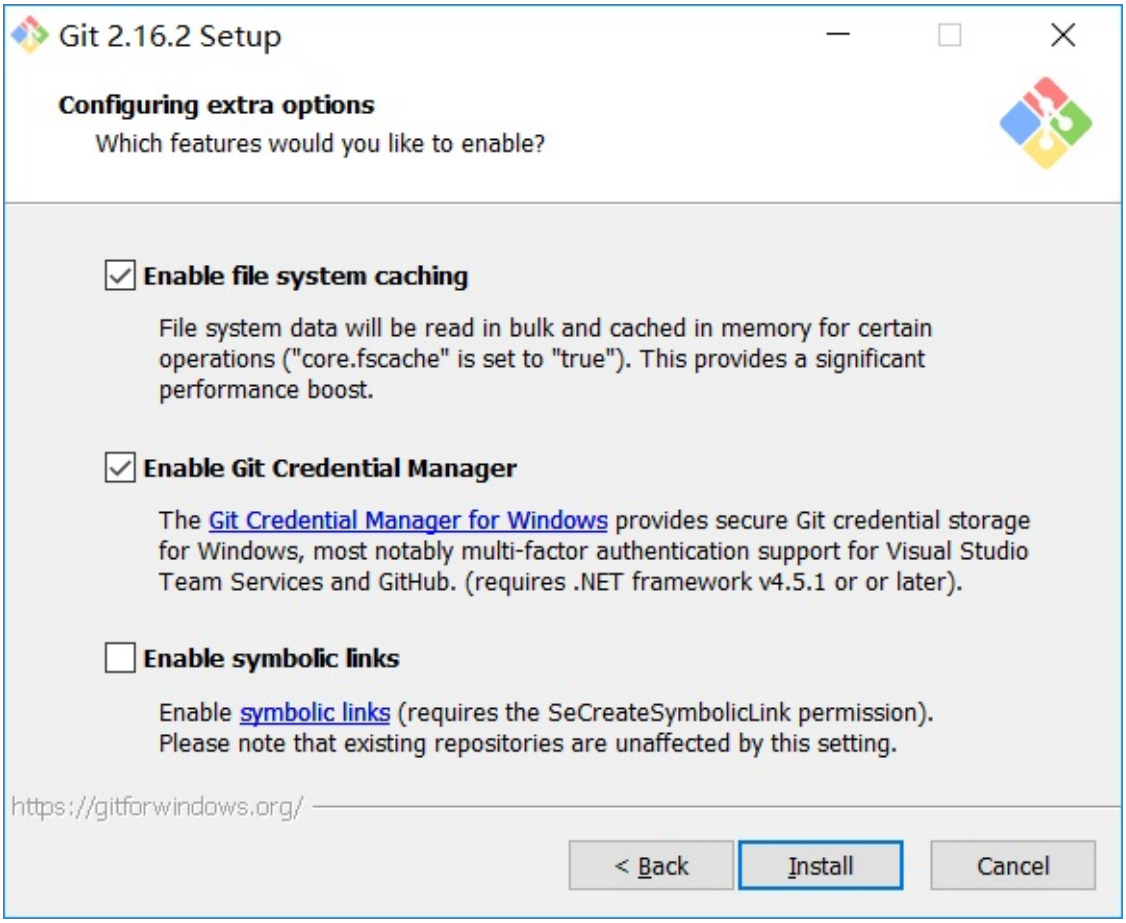
步骤8：选择第一项，点击next：



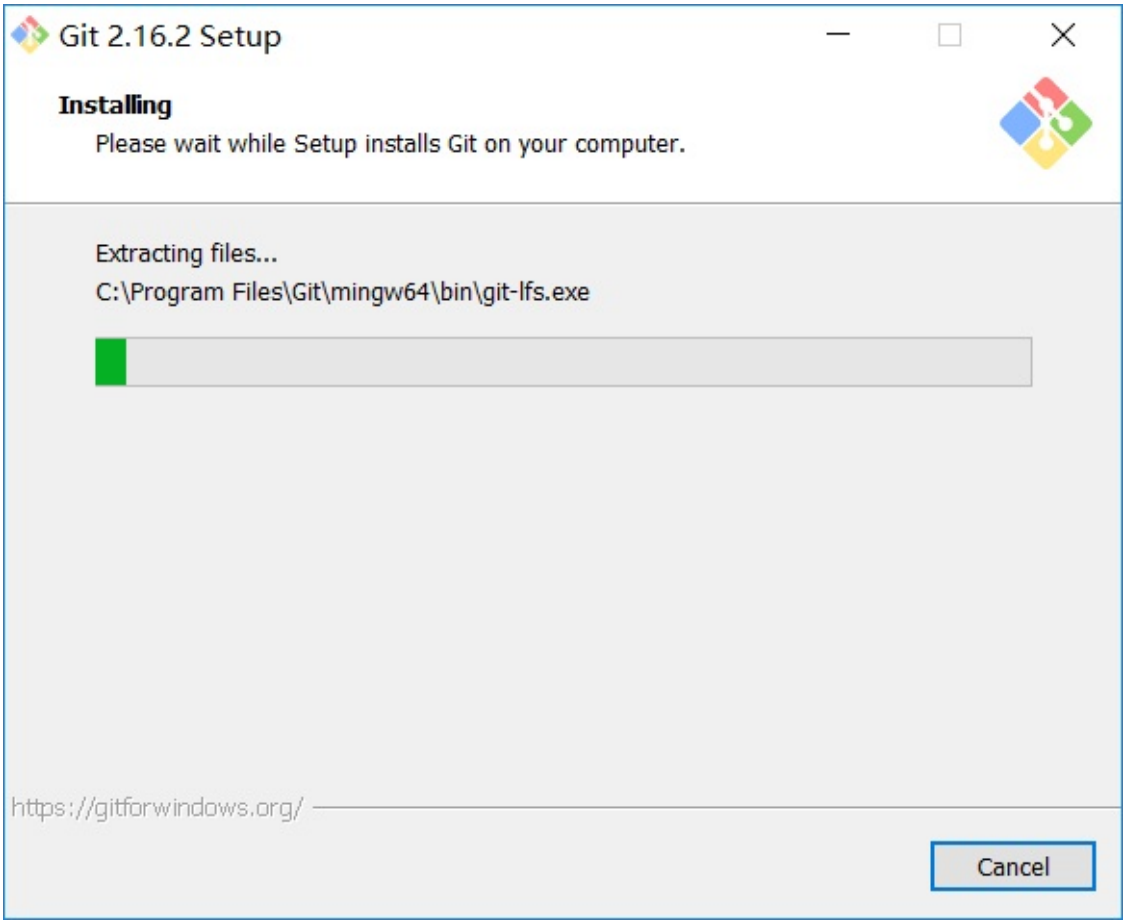
步骤9：选择第二项，点击next：



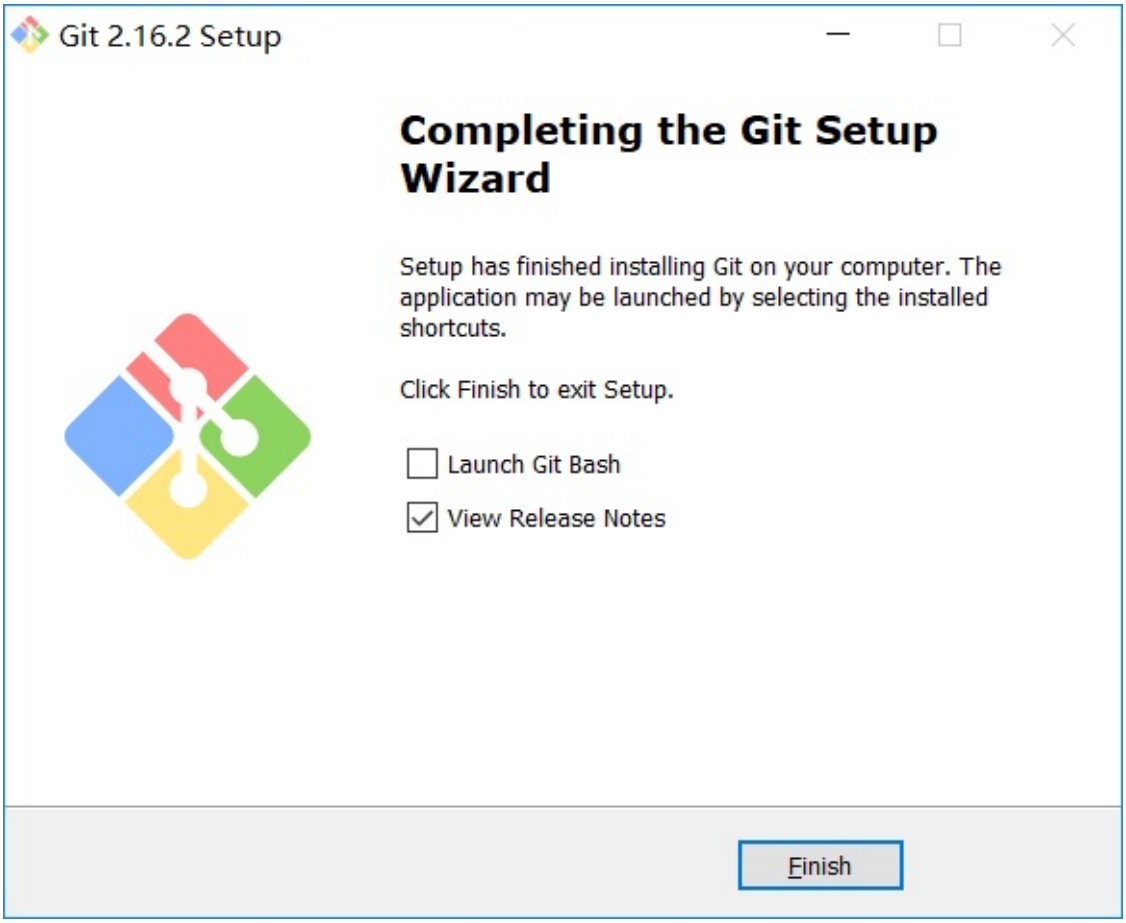
步骤10：默认配置直接点击next：



安装过程：



点击 Finish（完成）按钮退出安装向导。



检查git版本

```
git --version
```

git基本操作

设置姓名和邮箱

```
git config --global user.name "smallwei"
git config --global user.email "pengwei_li_flag@163.com"
```

克隆项目

```
git clone 项目地址

例：
git clone https://gitee.com/smallweigit/avue.git
```

拉取项目

```
git pull 本地分支名  远程分支名

例：
git pull origin master
```

提交文件

```
git add 你要提交的文件名或者路径
git commit -m '提交的注释'
git push 本地分支名 远程分支名

例：
git add test.sh
git commit -m 'test测试文件'
git push origin master
```

常用命令

```
    git init //初始化本地git环境
    git clone XXX//克隆一份代码到本地仓库
    git pull //把远程库的代码更新到工作台
    git pull --rebase origin master //强制把远程库的代码跟新到当前分支上面
    git fetch //把远程库的代码更新到本地库
    git add . //把本地的修改加到stage中
    git commit -m 'comments here' //把stage中的修改提交到本地库
    git push //把本地库的修改提交到远程库中
    git branch -r/-a //查看远程分支/全部分支
    git checkout master/branch //切换到某个分支
    git checkout -b test //新建test分支
```

```
git checkout -d test //删除test分支
git merge master //假设当前在test分支上面，把master分支上的修改同步到test分支上
git merge tool //调用merge工具
git stash //把未完成的修改缓存到栈容器中
git stash list //查看所有的缓存
git stash pop //恢复本地分支到缓存状态
git blame someFile //查看某个文件的每一行的修改记录（）谁在什么时候修改的）
git status //查看当前分支有哪些修改
git log //查看当前分支上面的日志信息
git diff //查看当前没有add的内容
git diff --cache //查看已经add但是没有commit的内容
git diff HEAD //上面两个内容的合并
git reset --hard HEAD //撤销本地修改
```

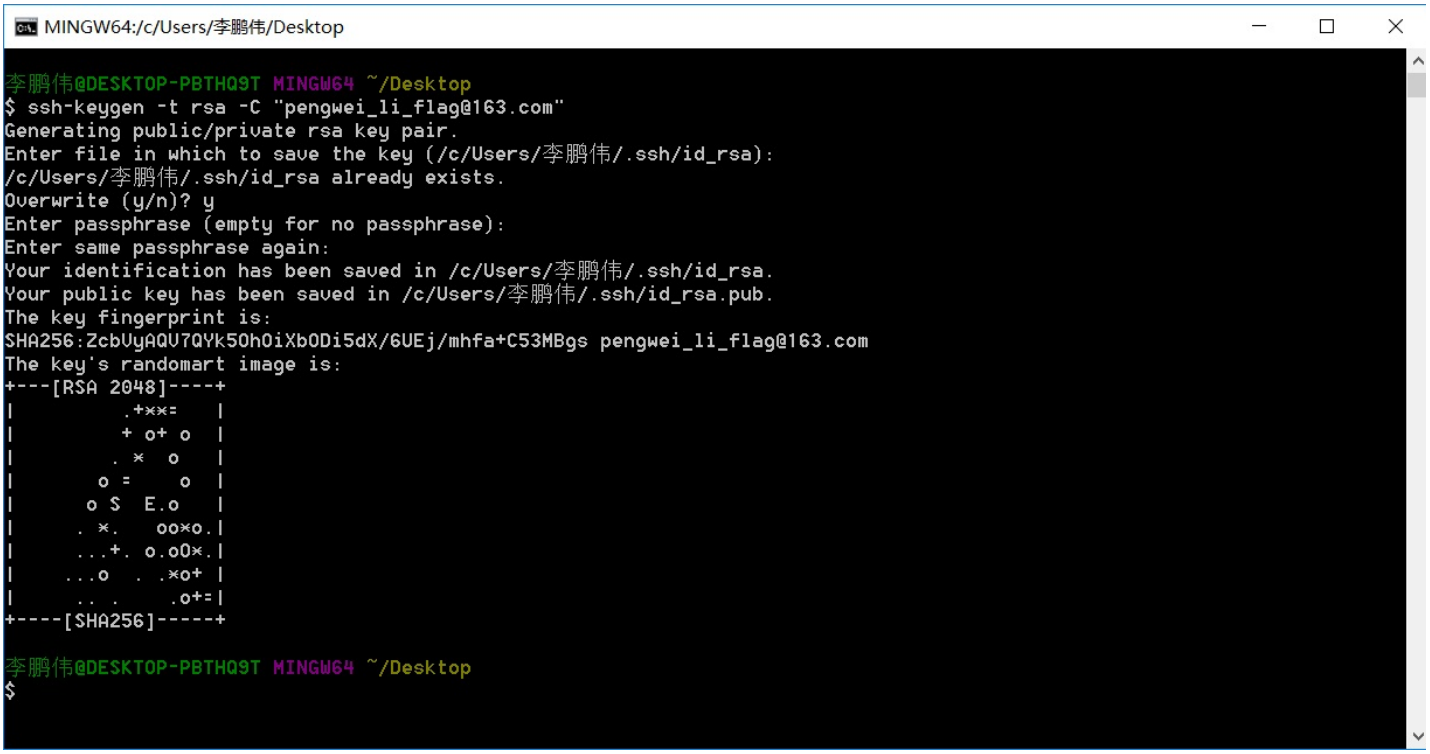
gitSSH配置

步骤 1:桌面右键菜单点击Git Bash Here

步骤 2:生成密钥

```
ssh-keygen -t rsa -C "你的电子邮箱"

例
ssh-keygen -t rsa -C "pengwei_li_flag@163.com"
```

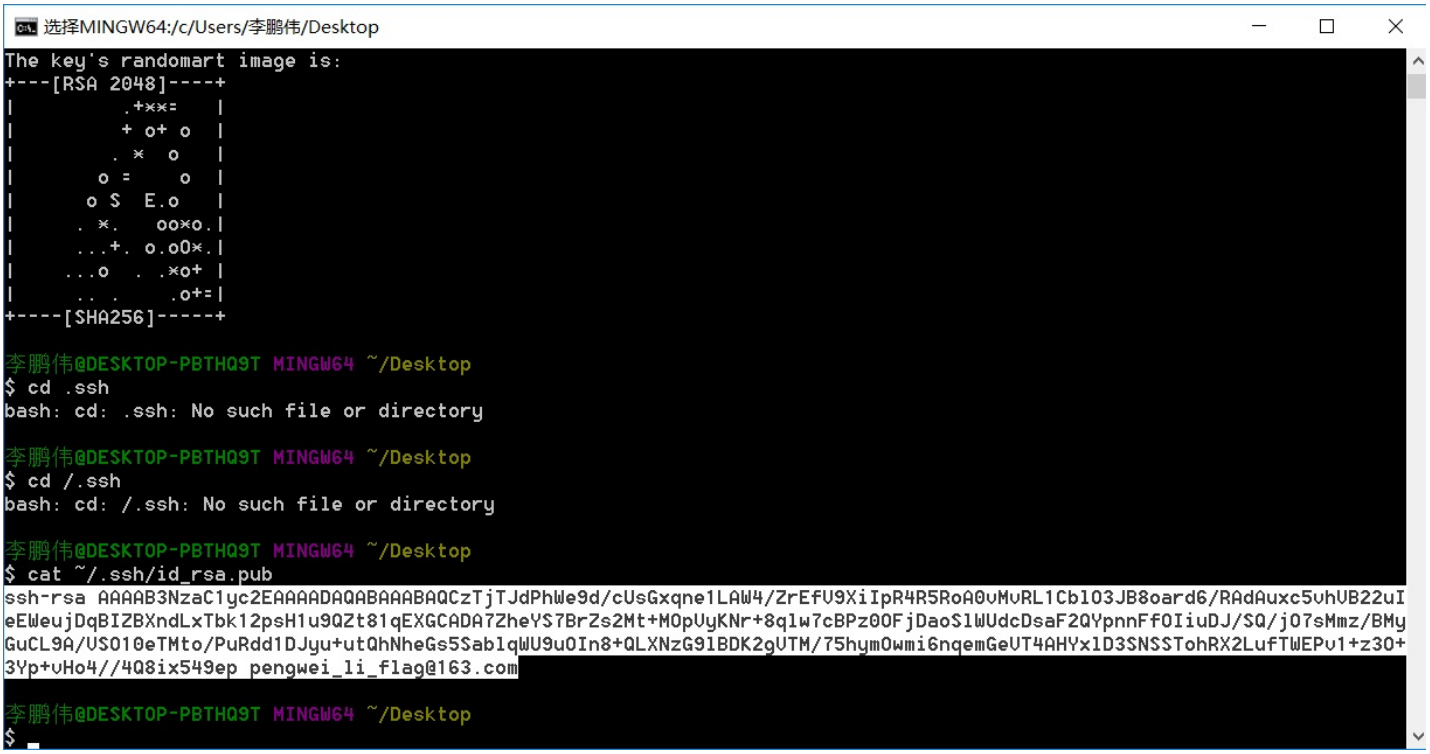
完后一直按Enter就行，出现了一大串字符，证明就成功了



步骤 3:查看密钥

```
cat ~/.ssh/id_rsa.pub
```

白色的部分就是你的密钥



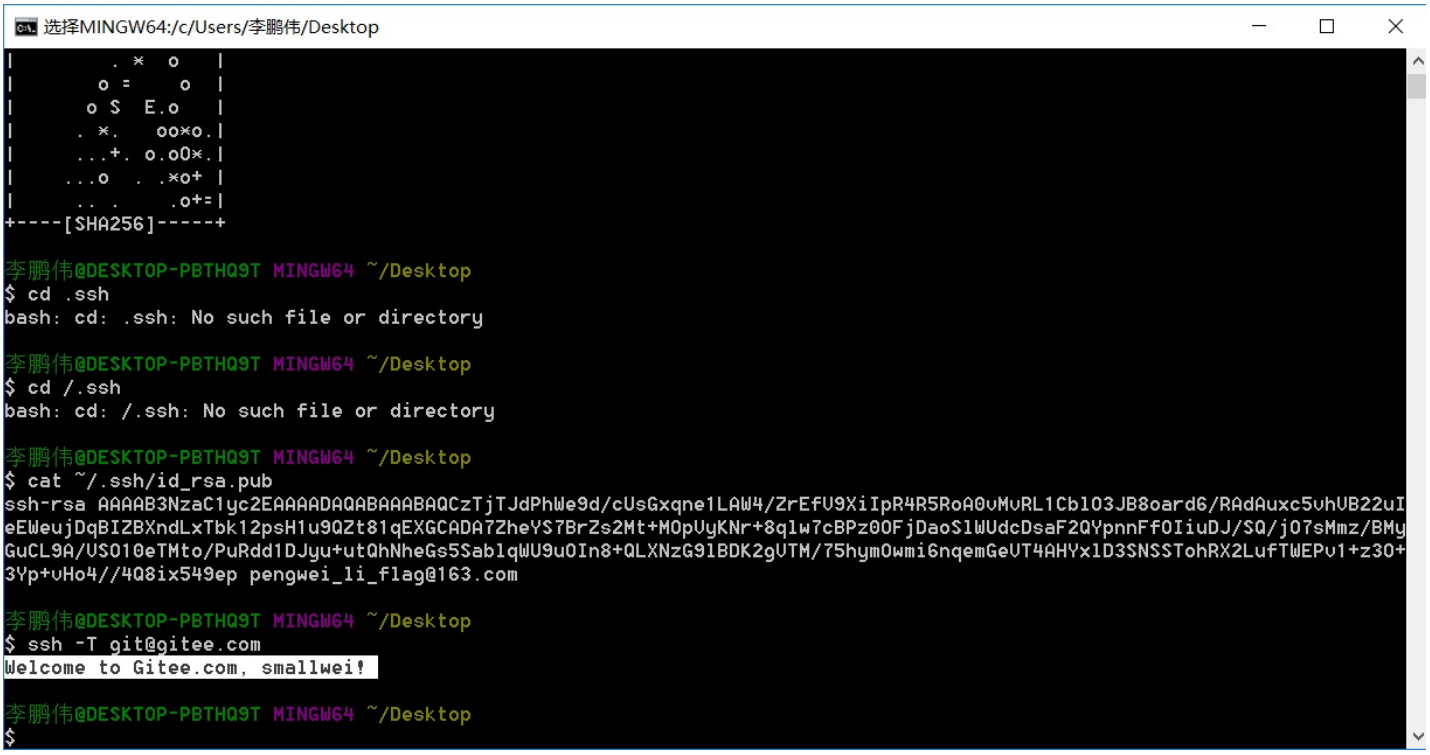
步骤4:已码云为例配置ssh

打开码云ssh 将你的密钥粘贴进去，点击保存

步骤 5:ssh链接测试

```
ssh -T git@gitee.com
```

看到welcome的字样ssh就配置成功了


选择MINGW64:/c/Users/李鹏伟/Desktop
+----[SHA256]-----+
李鹏伟@DESKTOP-PBTHQ9T MINGW64 ~/Desktop
$ cd .ssh
bash: cd: .ssh: No such file or directory
李鹏伟@DESKTOP-PBTHQ9T MINGW64 ~/Desktop
$ cd /.ssh
bash: cd: /.ssh: No such file or directory
李鹏伟@DESKTOP-PBTHQ9T MINGW64 ~/Desktop
$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCzTjTJdPhWe9d/cUsGxqne1LAW4/ZrEfV9XiIpR4R5RoA0vMvRL1Cb1O3JB8oard6/RAdAuxc5vhVB22uI
eEWeujDqBIZBXndLxTbk12psH1u9QZt81qEXGCADA7ZheYS7BrZs2Mt+MOpVyKNr+8q1w7cBPz0OFjDaoS1WUdcDsaF2QYpnnFfOIiuDJ/SQ/jO7sMmz/BMy
GuCL9A/VSO10eTMto/PuRdd1DJyu+utQhNheGs5SablqWU9uOIn8+QLXNzG91BDK2gVTM/75hymOwmi6nqemGeUT4AHYx1D3SNSSTohRX2LufTWEPv1+z3O+
3Yp+vHo4//4Q8ix549ep pengwei_li_flag@163.com
李鹏伟@DESKTOP-PBTHQ9T MINGW64 ~/Desktop
$ ssh -T git@gitee.com
Welcome to Gitee.com, smallwei!
李鹏伟@DESKTOP-PBTHQ9T MINGW64 ~/Desktop
$

vscode安装使用

安装

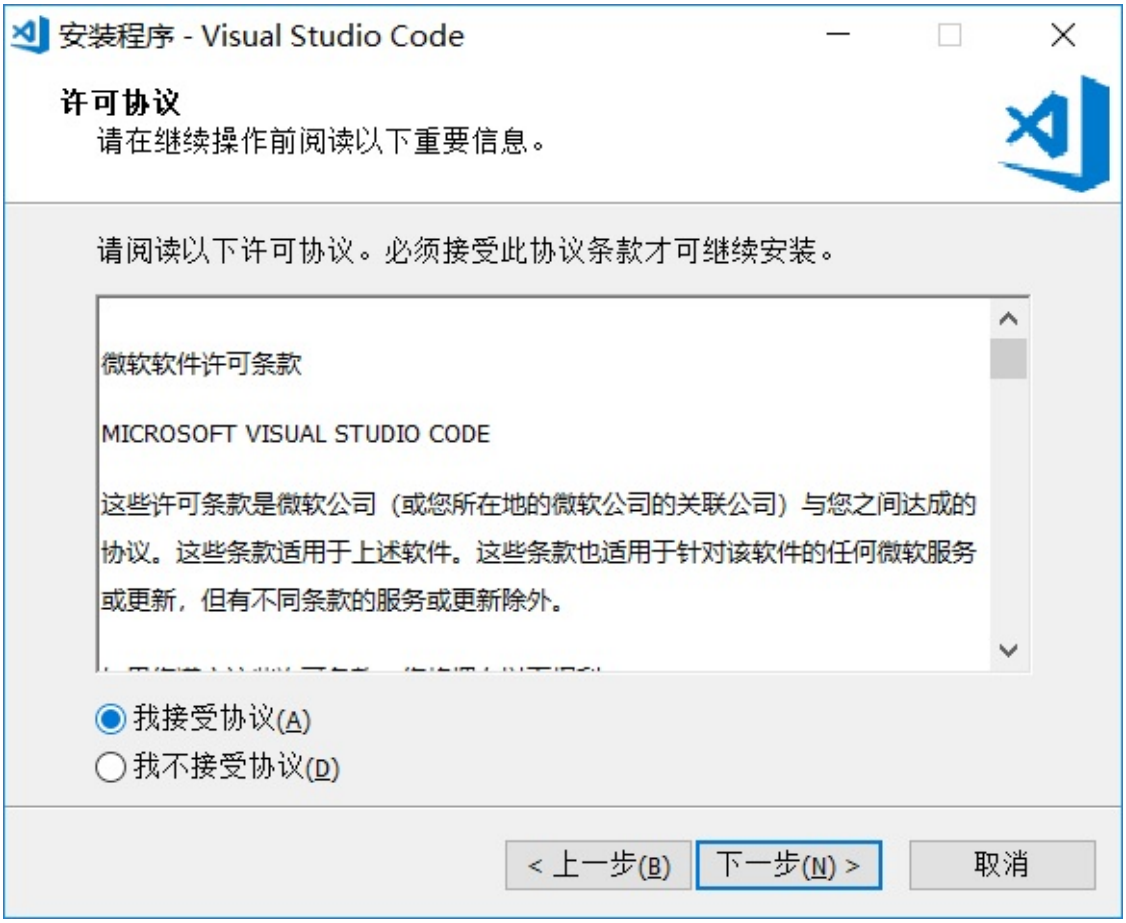
Window 上安装vscode

安装包下载地址 :https://code.visualstudio.com/Download

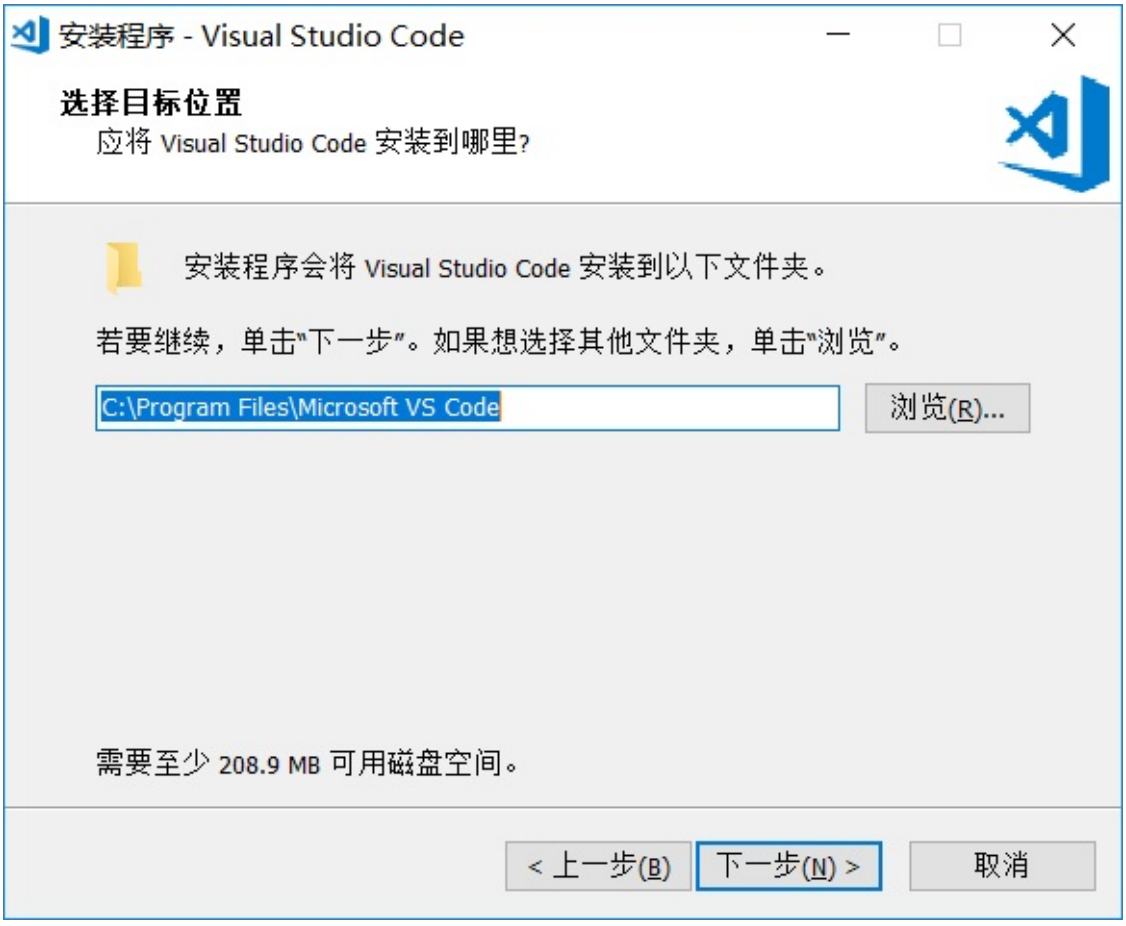
本文实例以node最新版本为例，其他版本类似， 安装步骤：

步骤 1：双击下载的安装包，进入安装向导界面；点击下一步

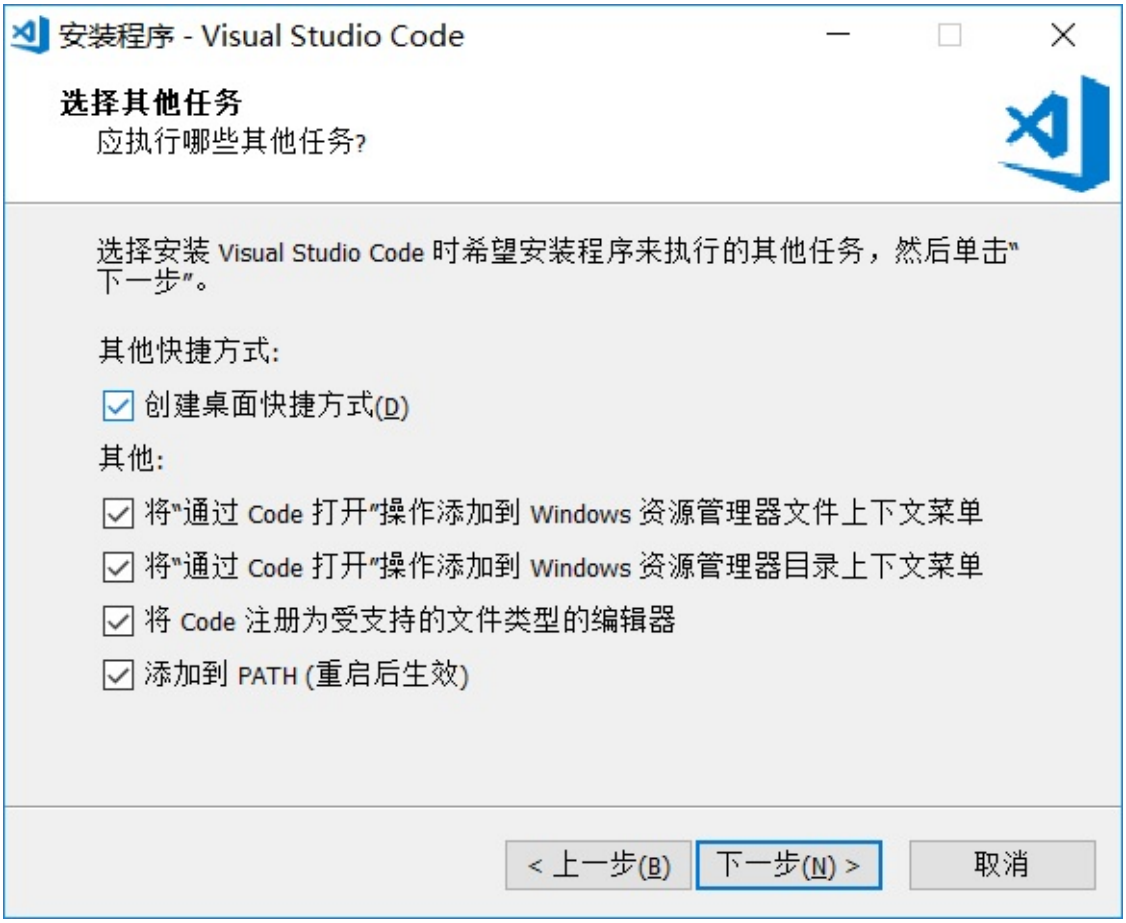
步骤 2：选择我接受协议；点击下一步



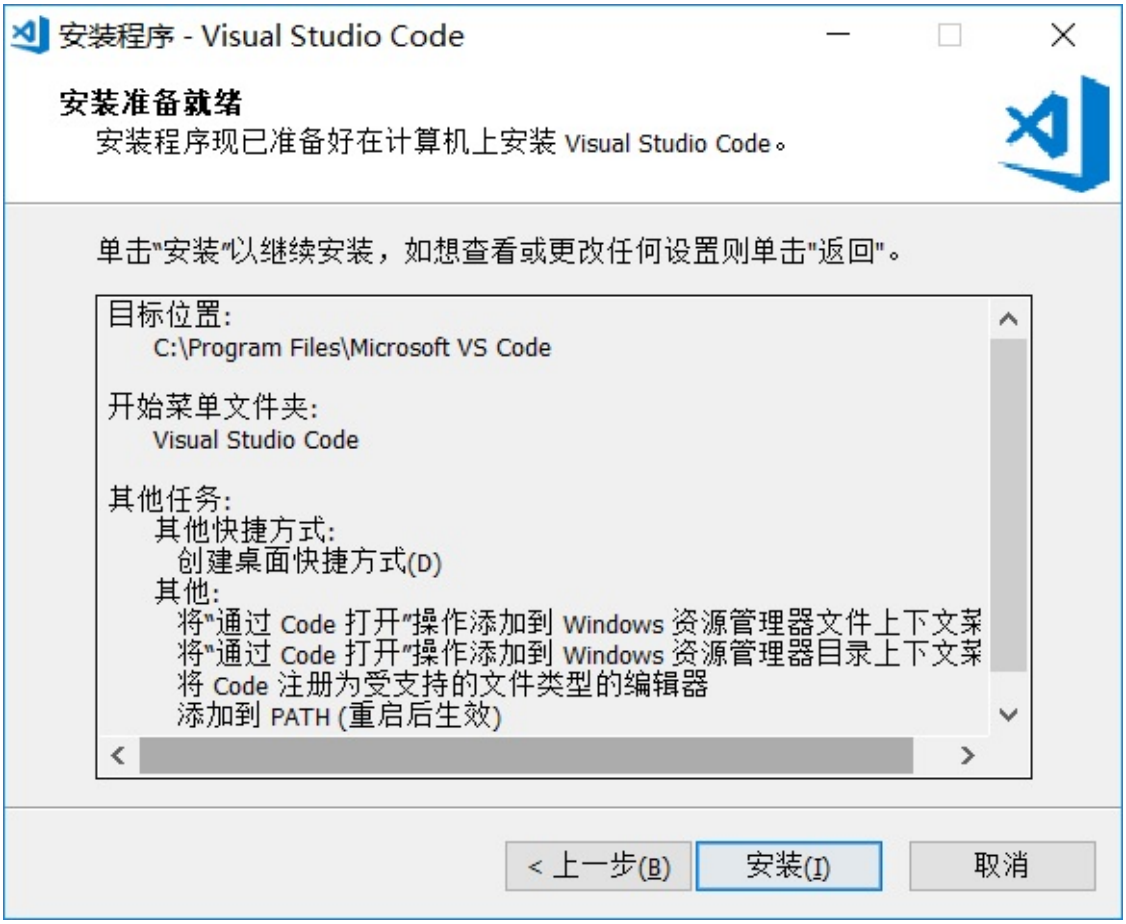
步骤 3：选择安装路径，也可以默认；点击下一步



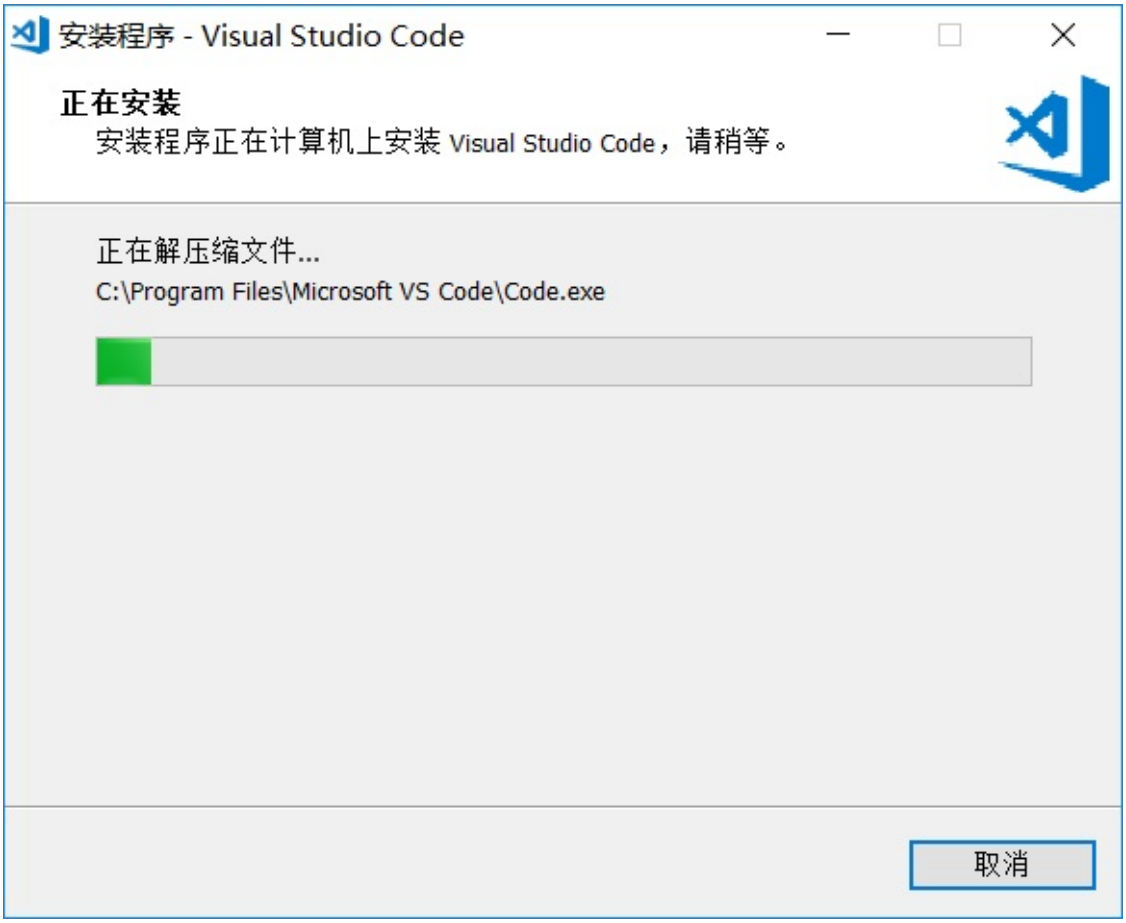
步骤 4：勾选自己需要的配置；点击下一步



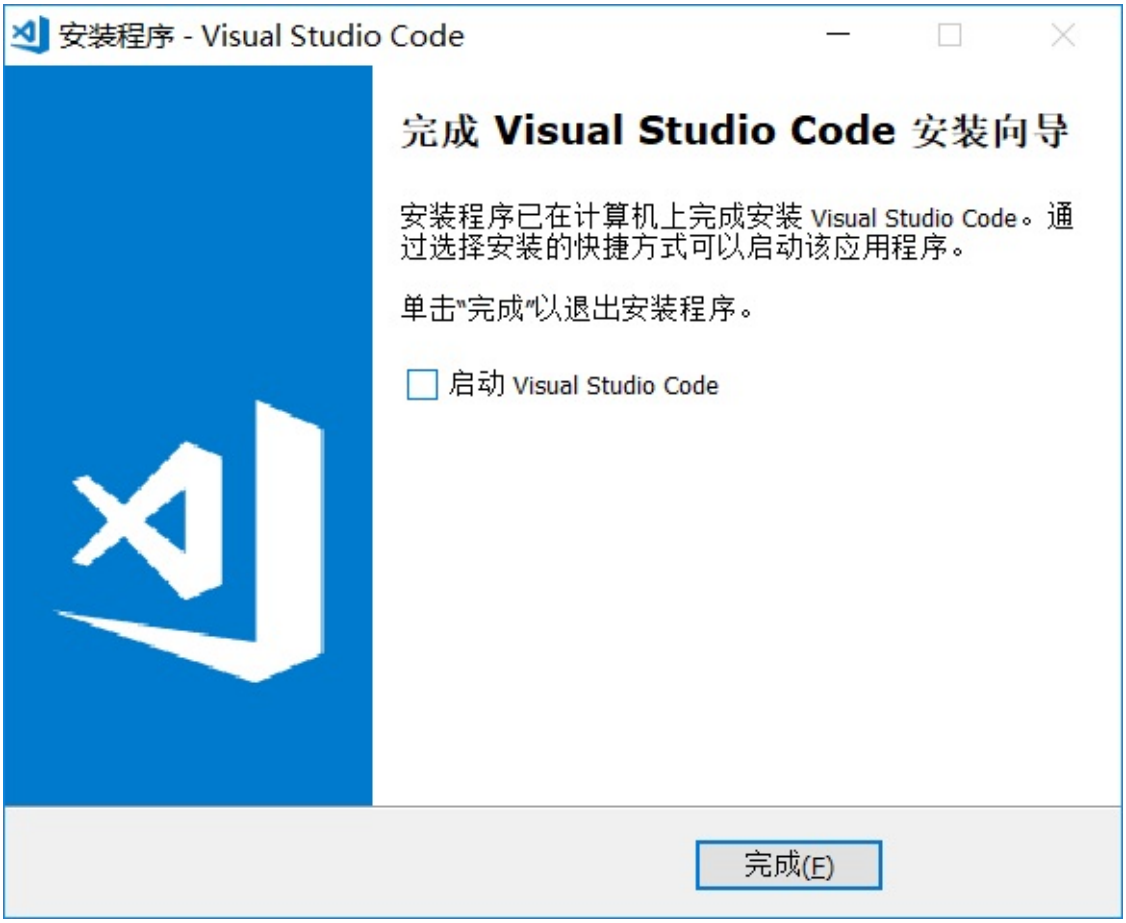
步骤 5：点击开始安装



安装过程：



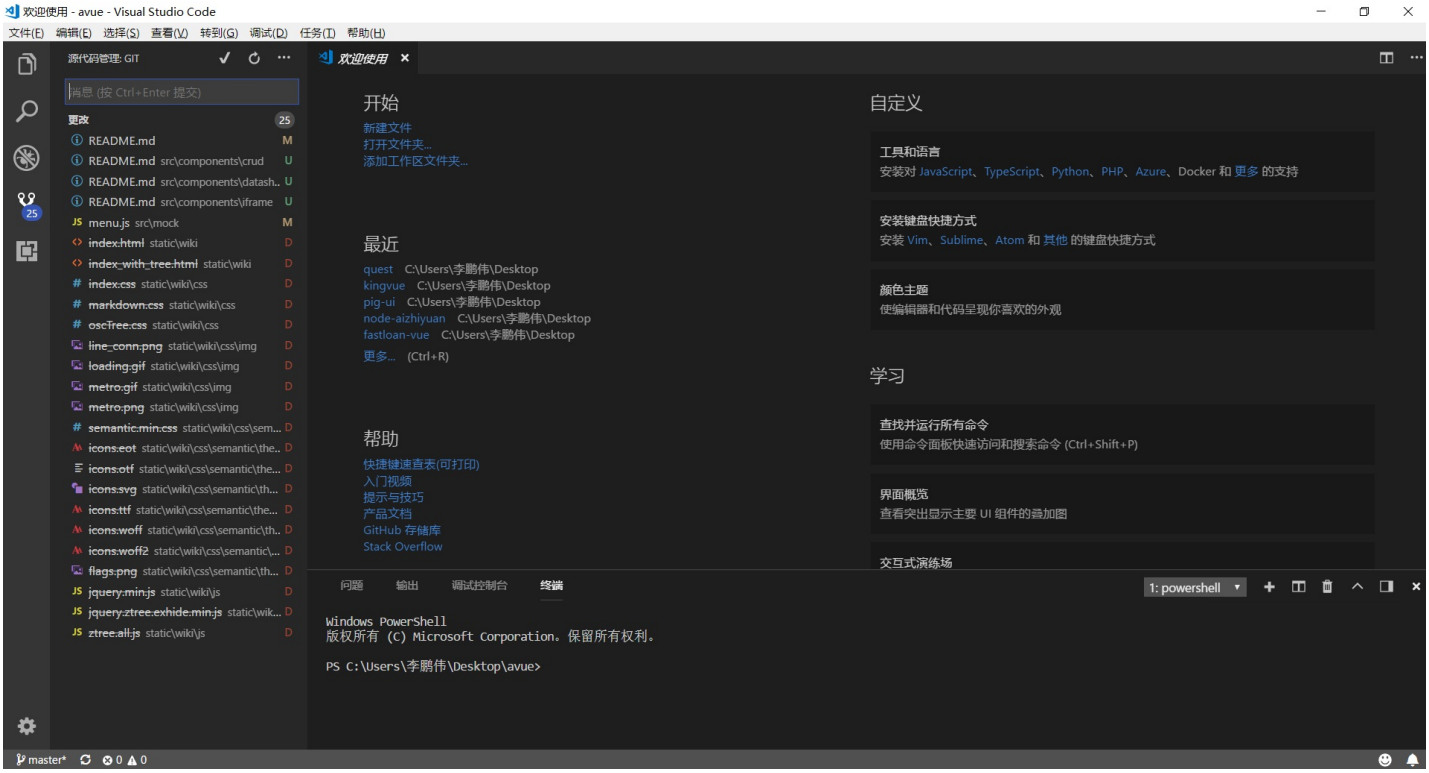
点击完成按钮退出安装向导。



git使用

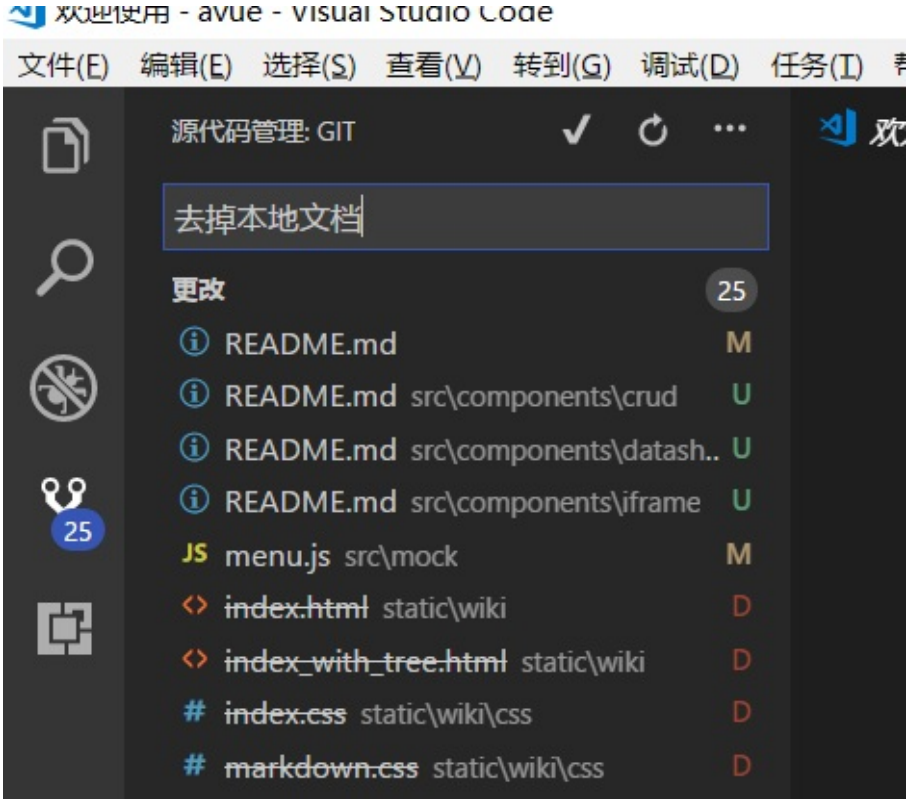
步骤 1 找到提交

显示你修改的文件



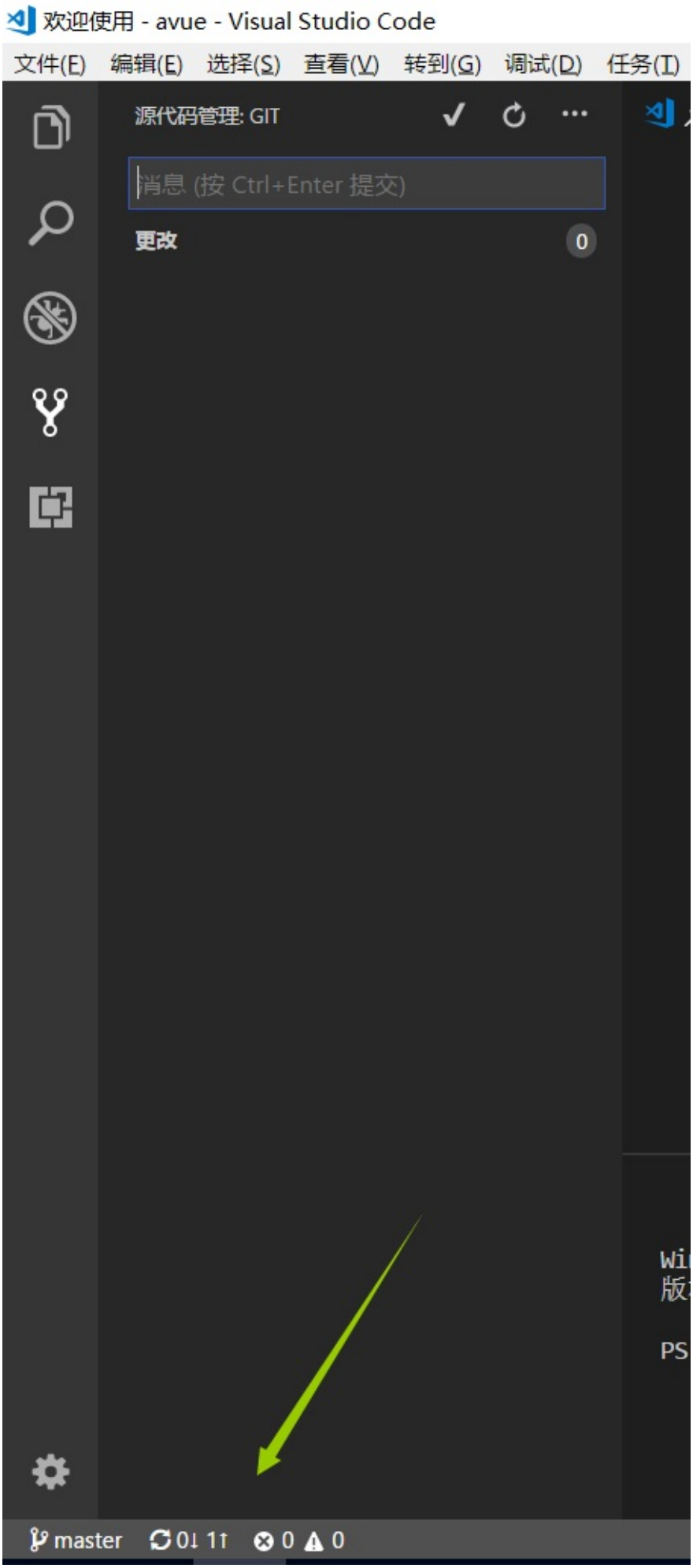
步骤 2 ：本地提交

输入框输入你要提交的内容，点击小对勾提交到本地



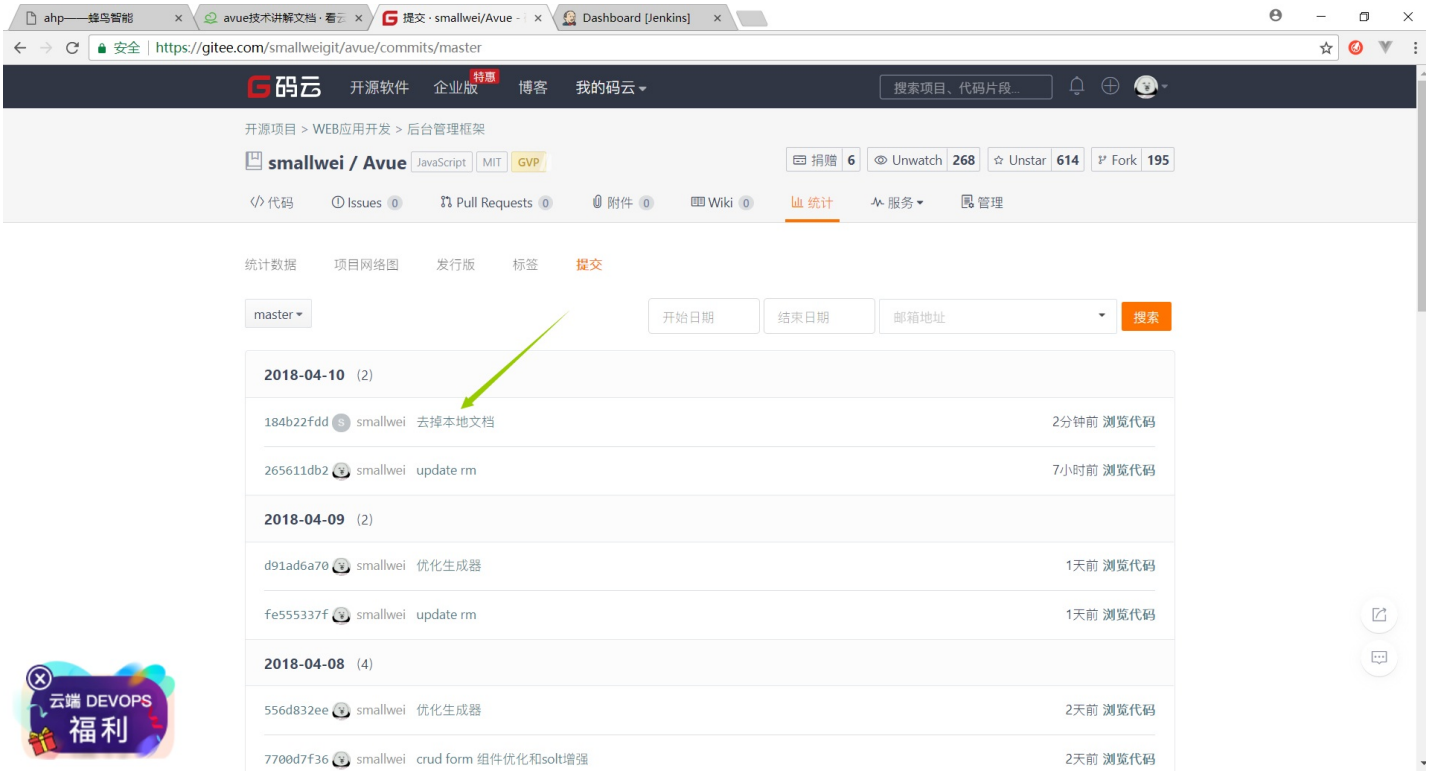
步骤 3 ：同步网络仓库

左边是需要pull的commit数量，右边是本地需要push的数量，点击同步图标


欢迎使用 - avue - Visual Studio Code
文件(F) 编辑(E) 选择(S) 查看(V) 转到(G) 调试(D) 任务(T)
源代码管理: GIT
消息 (按 Ctrl+Enter 提交)
更改
0
master
0↓ 1↑
0 0

步骤 4：同步成功

在码云上就可以看到刚提交的内容



快捷键

常用 General

按 Press	功能 Function
Ctrl + Shift + P , F1	显示命令面板 Show Command Palette
Ctrl + P	快速打开 Quick Open
Ctrl + Shift + N	新窗口/实例 New window/instance
Ctrl + Shift + W	关闭窗口/实例 Close window/instance

基础编辑 Basic editing

按 Press	功能 Function
Ctrl+X	剪切行（空选定） Cut line (empty selection)
Ctrl+C	复制行（空选定）Copy line (empty selection)
Alt+ ↑ / ↓	向上/向下移动行 Move line up/down
Shift+Alt + ↓ / ↑	向上/向下复制行 Copy line up/down
Ctrl+Shift+K	删除行 Delete line
Ctrl+Enter	在下面插入行 Insert line below
Ctrl+Shift+Enter	在上面插入行 Insert line above
Ctrl+Shift+\	跳到匹配的括号 Jump to matching bracket
Ctrl+] / [	缩进/缩进行 Indent/outdent line
Home	转到行首 Go to beginning of line
End	转到行尾 Go to end of line
Ctrl+Home	转到文件开头 Go to beginning of file
Ctrl+End	转到文件末尾 Go to end of file
Ctrl+↑ / ↓	向上/向下滚动行 Scroll line up/down
Alt+PgUp / PgDown	向上/向下滚动页面 Scroll page up/down
Ctrl+Shift+[	折叠（折叠）区域 Fold (collapse) region
Ctrl+Shift+]	展开（未折叠）区域 Unfold

	(uncollapse) region
Ctrl+K Ctrl+[	折叠（未折叠）所有子区域 Fold (collapse) all subregions
Ctrl+K Ctrl+]	展开（未折叠）所有子区域 Unfold (uncollapse) all subregions
Ctrl+K Ctrl+0	折叠（折叠）所有区域 Fold (collapse) all regions
Ctrl+K Ctrl+J	展开（未折叠）所有区域 Unfold (uncollapse) all regions
Ctrl+K Ctrl+C	添加行注释 Add line comment
Ctrl+K Ctrl+U	删除行注释 Remove line comment
Ctrl+/	切换行注释 Toggle line comment
Shift+Alt+A	切换块注释 Toggle block comment
Alt+Z	切换换行 Toggle word wrap

导航 Navigation

按 Press	功能 Function
Ctrl + T	显示所有符号 Show all Symbols
Ctrl + G	转到行... Go to Line...
Ctrl + P	转到文件... Go to File...
Ctrl + Shift + O	转到符号... Go to Symbol...
Ctrl + Shift + M	显示问题面板 Show Problems panel
F8	转到下一个错误或警告 Go to next error or warning
Shift + F8	转到上一个错误或警告 Go to previous error or warning
Ctrl + Shift + Tab	导航编辑器组历史记录 Navigate editor group history
Alt + ←/→	返回/前进 Go back / forward
Ctrl + M	切换选项卡移动焦点 Toggle Tab moves focus

搜索和替换 Search and replace

按 Press	功能 Function
Ctrl + F	查找 Find
Ctrl + H	替换 Replace

F3 / Shift + F3	查找下一个/上一个 Find next/previous
Alt + Enter	选择查找匹配的所有出现 Select all occurences of Find match
Ctrl + D	将选择添加到下一个查找匹配 Add selection to next Find match
Ctrl + K Ctrl + D	将最后一个选择移至下一个查找匹配项 Move last selection to next Find match
Alt + C / R / W	切换区分大小写/正则表达式/整个词 Toggle case-sensitive / regex / whole word

多光标和选择 Multi-cursor and selection**

按 Press	功能 Function
Alt +单击	插入光标 Insert cursor
Ctrl + Alt +↑/↓	在上/下插入光标 Insert cursor above / below
Ctrl + U	撤消上一个光标操作 Undo last cursor operation
Shift + Alt + I	在选定的每一行的末尾插入光标 Insert cursor at end of each line selected
Ctrl + I	选择当前行 Select current line
Ctrl + Shift + L	选择当前选择的所有出现 Select all occurrences of current selection
Ctrl + F2	选择当前字的所有出现 Select all occurrences of current word
Shift + Alt + →	展开选择 Expand selection
Shift + Alt + ←	缩小选择 Shrink selection
Shift + Alt + （拖动鼠标）	列（框）选择 Column (box) selection
Ctrl + Shift + Alt +（箭头键）	列（框）选择 Column (box) selection
Ctrl + Shift + Alt + PgUp / PgDown	列（框）选择页上/下 Column (box) selection page up/down

丰富的语言编辑 Rich languages editing

按 Press	功能 Function

Ctrl + 空格	触发建议 Trigger suggestion
Ctrl + Shift + Space	触发器参数提示 Trigger parameter hints
Tab	Emmet 展开缩写 Emmet expand abbreviation
Shift + Alt + F	格式化文档 Format document
Ctrl + K Ctrl + F	格式选定区域 Format selection
F12	转到定义 Go to Definition
Alt + F12	Peek定义 Peek Definition
Ctrl + K F12	打开定义到边 Open Definition to the side
Ctrl + .	快速解决 Quick Fix
Shift + F12	显示引用 Show References
F2	重命名符号 Rename Symbol
Ctrl + Shift + . / ,	替换为下一个/上一个值 Replace with next/previous value
Ctrl + K Ctrl + X	修剪尾随空格 Trim trailing whitespace
Ctrl + K M	更改文件语言 Change file language

编辑器管理 Editor management

按 Press	功能 Function
Ctrl+F4, Ctrl+W	关闭编辑器 Close editor
Ctrl+K F	关闭文件夹 Close folder
Ctrl+\	拆分编辑器 Split editor
Ctrl+ 1 / 2 / 3	聚焦到第1，第2或第3编辑器组 Focus into 1st, 2nd or 3rd editor group
Ctrl+K Ctrl+ ←/→	聚焦到上一个/下一个编辑器组 Focus into previous/next editor group
Ctrl+Shift+PgUp / PgDown	向左/向右移动编辑器 Move editor left/right
Ctrl+K ← / →	移动活动编辑器组 Move active editor group

文件管理 File management

按 Press	功能 Function
Ctrl+N	新文件 New File

Ctrl+O	打开文件... Open File...
Ctrl+S	保存 Save
Ctrl+Shift+S	另存为... Save As...
Ctrl+K S	全部保存 Save All
Ctrl+F4	关闭 Close
Ctrl+K Ctrl+W	关闭所有 Close All
Ctrl+Shift+T	重新打开关闭的编辑器 Reopen closed editor
Ctrl+K	输入保持打开 Enter Keep Open
Ctrl+Tab	打开下一个 Open next
Ctrl+Shift+Tab	打开上一个 Open previous
Ctrl+K P	复制活动文件的路径 Copy path of active file
Ctrl+K R	显示资源管理器中的活动文件 Reveal active file in Explorer
Ctrl+K O	显示新窗口/实例中的活动文件 Show active file in new window/instance

显示 Display

按 Press	功能 Function
F11	切换全屏 Toggle full screen
Shift+Alt+1	切换编辑器布局 Toggle editor layout
Ctrl+ = / -	放大/缩小 Zoom in/out
Ctrl+B	切换侧栏可见性 Toggle Sidebar visibility
Ctrl+Shift+E	显示浏览器/切换焦点 Show Explorer / Toggle focus
Ctrl+Shift+F	显示搜索 Show Search
Ctrl+Shift+G	显示Git Show Git
Ctrl+Shift+D	显示调试 Show Debug
Ctrl+Shift+X	显示扩展 Show Extensions
Ctrl+Shift+H	替换文件 Replace in files
Ctrl+Shift+J	切换搜索详细信息 Toggle Search details
Ctrl+Shift+C	打开新命令提示符/终端 Open new command prompt/terminal

Ctrl+Shift+U	显示输出面板 Show Output panel
Ctrl+Shift+V	切换Markdown预览 Toggle Markdown preview
Ctrl+K V	从旁边打开Markdown预览 Open Markdown preview to the side

调试 Debug

按 Press	功能 Function
F9	切换断点 Toggle breakpoint
F5	开始/继续 Start/Continue
Shift+F5	停止 Stop
F11 / Shift+F11	下一步/上一步 Step into/out
F10	跳过 Step over
Ctrl+K Ctrl+I	显示悬停 Show hover

集成终端 Integrated terminal

按 Press	功能 Function
Ctrl+`	显示集成终端 Show integrated terminal
Ctrl+Shift+`	创建新终端 Create new terminal
Ctrl+Shift+C	复制选定 Copy selection
Ctrl+Shift+V	粘贴到活动端子 Paste into active terminal
Ctrl+↑ / ↓	向上/向下滚动 Scroll up/down
Shift+PgUp / PgDown	向上/向下滚动页面 Scroll page up/down
Ctrl+Home / End	滚动到顶部/底部 Scroll to top/bottom

项目启动

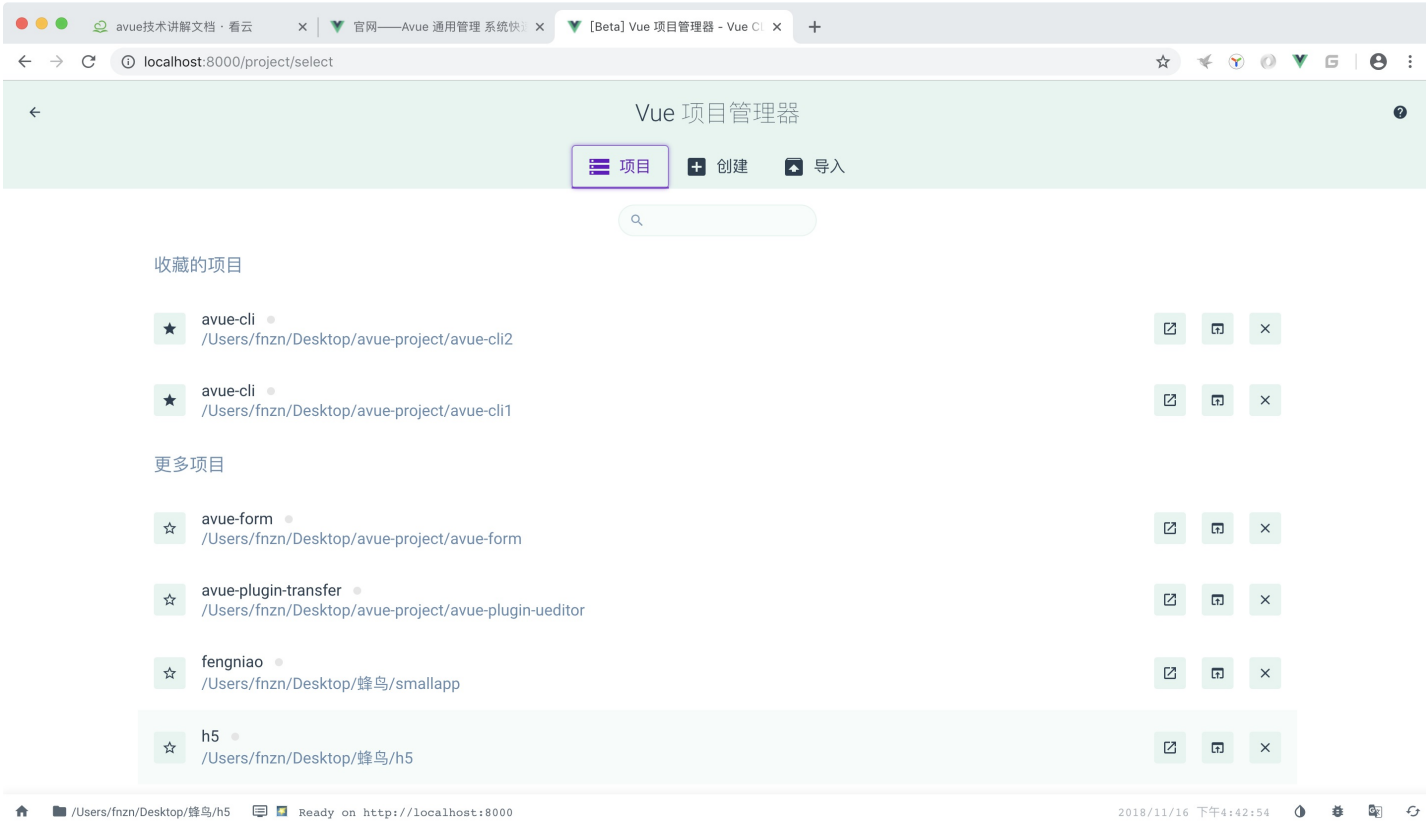
启动项目

npm/yarn启动

```
# 克隆项目
git clone https://gitee.com/smallweigit/avue-cli.git

# 进入项目
cd avue-cli

# 安装依赖
npm install --registry=https://registry.npm.taobao.org

# 启动服务
npm run serve
```

可视化管理界面启动

```
npm install -g @vue/cli 全局安装vue脚手架最新版
vue --version 查看版本是否为3.x版本
vue-ui 运行管理工具，导入avue-cli项目
```



源码目录结构

源码目录结构

```
├──api  //后台api接口
├──components //基础公共组件
   ├──basic-block.vue //基础模版1
   ├──basic-container //基础模板2
   ├──error-page.vue //错误模版
   ├──iframe //第三方网站模版
├──config //项目全局变量配置文件
   ├──env.js //阿里巴巴图标库地址,项目api接口地址等
   ├──website.js //项目的一些通用配置
├──const    //静态数据
├──mac  //mac主题文件
├──lang //国际化配置文件
├──mock //mock.js模拟数据(项目没有与服务器交互api文件调用这里模拟数据)
   ├──index.js //mock控制器
   ├──menu.js //菜单模拟数据
   ├──user.js //用户模拟数据
├──page //登录页和主页的一些功能核心组件
├──router   //项目的路由文件和ajax拦截器
   ├──views //业务路由
   ├──axios.js//ajax拦截器配置
   ├──page  //底层框架路由
   ├──avue-router   //动态路由核心源码
   ├──router.js //你的业务路由配置文件
├──store    //vuex全局组件共享变量和方法
  ├──moudles
    ├──tags.js  //导航栏持久化
    ├──common.js    //公用数据持久化
    ├──user.js  //用户相关持久化
    ├──logs.js //错误日志持久化
├──styles   //全局样式文件存放
    ├──animate //全局动画
            ├──vue-transition.scss 过度动画
    ├──theme //主题配置
            ├──black.scss 黑色主题
            ├──gradual.scss 渐变主题
            ├──index.scss 主题的配置
            ├──star.scss 炫彩主题
    ├──common.scss  //基础样式
    ├──media.scss //多终端适配
    ├──element-ui.scss  //覆盖ele的样式
    ├──mixin.scss  //工具类包(滚动条等)
    ├──sidebar.scss //侧边菜单
    ├──tags.scss  //选项标签
```

```
    ├──top.scss //顶部
    ├──variables.scss  //全局scss变量
├──util //全局工具包存放
   ├──auth.js       //(cookie)授权相关的工具包
   ├──store.js    //本地存储缓存的工具包
   ├──date.js  //日期工具类
   ├──util.js   //基础工具包
   ├──validate.js  //验证工具包
├──views    //业务逻辑页面存放
├──App.vue  //vue主组件入口文件
├──error.js //全局错误日志文件配置
├──main.js  //主文件入口
├──permission.js    //全局权限配置
```

登录授权

登录授权

1.用户点击登录时调用 `vuex` 中的 `LoginByUsername`

```
//根据用户名登录
LoginByUsername ({ commit }, userInfo = {}) {
  const user = encryption({
    data: userInfo,
    type: 'Aes',
    key: 'avue',
    param: ['useranme', 'password']
  });
  return new Promise((resolve) => {
    loginByUsername(user.username, user.password, userInfo.code, userInfo.redom
Str).then(res => {
      const data = res.data.data;
      commit('SET_TOKEN', data);
      commit('DEL_ALL_TAG');
      commit('CLEAR_LOCK');
      resolve();
    })
  })
}
```

2.用户密码加密授权部分，如果你没有这部分需求，可以把这里注释

```
 const user = encryption({
    data: userInfo,
    type: 'Aes',
    key: 'avue',
    param: ['useranme', 'password']
  });
```

3.将加密的数据发送请求服务端

```
 loginByUsername(user.username, user.password, userInfo.code, userInfo.redomStr)
.then(res => {})
```

4.服务端返回数据后掉用 `vuex` 中的 `SET_TOKEN` 方法，进行一系列的 `token` 操作,

```
commit('SET_TOKEN', data);

SET_TOKEN: (state, token) => {
  setToken(token) //放入cookie中
  state.token = token;
  setStore({ name: 'token', content: state.token }) //放入session中
}
```

同时会把token放入headers中来请求接口

```
//router/axios.js
if (getToken() && !isToken) {
    config.headers[website.Authorization] = 'Bearer ' + getToken() // 让每个请求携带token--['Authorization']为自定义key 请根据实际情况自行修改
  }
```

5.登录成功后在 `permission.js` 全局权限判用户的信息是否存在，如果不存在掉用 `vuex` 中的 `GetUserInfo` 方法请求用户信息

```
//permission.js
if (store.getters.roles.length === 0) {
    store.dispatch('GetUserInfo').then(() => {
      next()
    }).catch(() => {
      store.dispatch('FedLogOut').then(() => {
        next({ path: '/login' })
      })
    })
  }

//store/modules/user.js
 GetUserInfo ({ commit }) {
  return new Promise((resolve, reject) => {
    getUserInfo().then((res) => {
      const data = res.data.data;
      commit('SET_USERIFNO', data.userInfo); //放入用户信息
      commit('SET_ROLES', data.roles); //放入角色信息
      commit('SET_PERMISSION', data.permission) //放入按钮权限信息
      resolve(data);
    }).catch(err => {
      reject(err);
    })
  })
```

权限详解

菜单权限

菜单权限

登录成功后进入系统，会在首页中调用获取菜单的方法，根据前台传过去的 `token` ，后台返回相应用户权限的菜单数据

```
//store/modules/user.js
GetMenu ({ commit }, parentId) {
  return new Promise(resolve => {
    getMenu(parentId).then((res) => {
      const data = res.data.data
      let menu = deepClone(data);
      menu.forEach(ele => formatPath(ele, true));
      commit('SET_MENUALL', menu)
      commit('SET_MENU', menu)
      resolve(menu)
    })
  })
},
```

调用api接口获取数据

```
 getMenu(parentId).then((res) => {})
```

对菜单数据处理，处理成符合 `vue-router` 的动态路由,详细处理方法可以看 `src/router/avue-router.js` 中的 `formatPath`

```
menu.forEach(ele => formatPath(ele, true));
```

把处理后的数据放入 `vuex` 中

```
commit('SET_MENUALL', menu)
commit('SET_MENU', menu)
```

按钮权限

按钮权限

按钮权限，是根据用户登录成功后，再返回的个人信息中带着一个permission字段，里面包含了按钮的权限，再加载到vuex全局变量permission中，根据v-if标签去判断

```
GetUserInfo ({ commit }) {
     return new Promise((resolve, reject) => {
       getUserInfo().then((res) => {
         const data = res.data.data;
         commit('SET_USERIFNO', data.userInfo);
         commit('SET_ROLES', data.roles);
         commit('SET_PERMISSION', data.permission)
         resolve(data);
       }).catch(err => {
         reject(err);
       })
     })
   },
```

将后台返回的数据中按钮的权限信息挂在到 `vuex` 中

```
  commit('SET_PERMISSION', data.permission)
```

格式如下

```
permission：[
        'sys_crud_btn_add',
        'sys_crud_btn_export',
        'sys_menu_btn_add',
        'sys_menu_btn_edit',
        'sys_menu_btn_del',
        'sys_role_btn1',
        'sys_role_btn2',
        'sys_role_btn3',
        'sys_role_btn4',
        'sys_role_btn5',
        'sys_role_btn6',
    ]
```

格式命名如下 系统模块_功能模块名_按钮名

比如首页下面的用户的新增按钮:index_user_add

调用

```
<el-button v-if="permission.sys_crud_btn_add">新 增</el-button>

computed: {
   ...mapGetters(["permission"])
 }
```

路由权限

路由权限

路由权限指的是在系统不同的页面需要不同的权限来划分，比如

1. 没有登录的用户访问授权页面跳转到登录页
2. 登录的用户访问登录页跳转到主页
3. 其他等等

项目中的路由权限控制主要在 `/src/permission.js` 中做了处理,利用 `vue-router` 的 `beforeEach` 钩子

判断是否添加到标签栏中

```
const value = to.query.src || to.fullPath;
const label = to.query.name || to.name;
if (meta.isTab !== false && !validatenull(value) && !validatenull(label)) {
    store.commit('ADD_TAG', {
        label: label,
        value: value,
        params: to.params,
        query: to.query,
        group: router.$avueRouter.group || []
    });
}
```

用户登录后和没登录的相关逻辑

- 如果用户有无 `token` 时候访问页面的逻辑

```
if (getToken()) {
     ...
}else {
    //页面判断是否需要认证，不需要认证直接访问
    if (meta.isAuth === false) {
      next()
    } else {
      next('/login')
    }
}
```

- 如果系统锁屏,访问如何页面都会跳转到锁屏页面

```
 if (store.getters.isLock && to.path != lockPage) {
    next({ path: lockPage })
 } else if (to.path === '/login') {
    next({ path: '/' })
 } else {
    ...
}
```

- 如果用户信息为空则获取用户信息，获取用户信息失败跳转到登录页

```
if (store.getters.roles.length === 0) {
    store.dispatch('GetUserInfo').then(() => {
        next({...to, replace: true })
    }).catch(() => {
        store.dispatch('FedLogOut').then(() => {
            next({ path: '/login' })
        })
    })
}
```

动态设置浏览器标题

```
router.afterEach(() => {
    NProgress.done();
    const title = store.getters.tag.label;
    //根据当前的标签也获取label的值动态设置浏览器标题
    router.$avueRouter.setTitle(title);
});
```

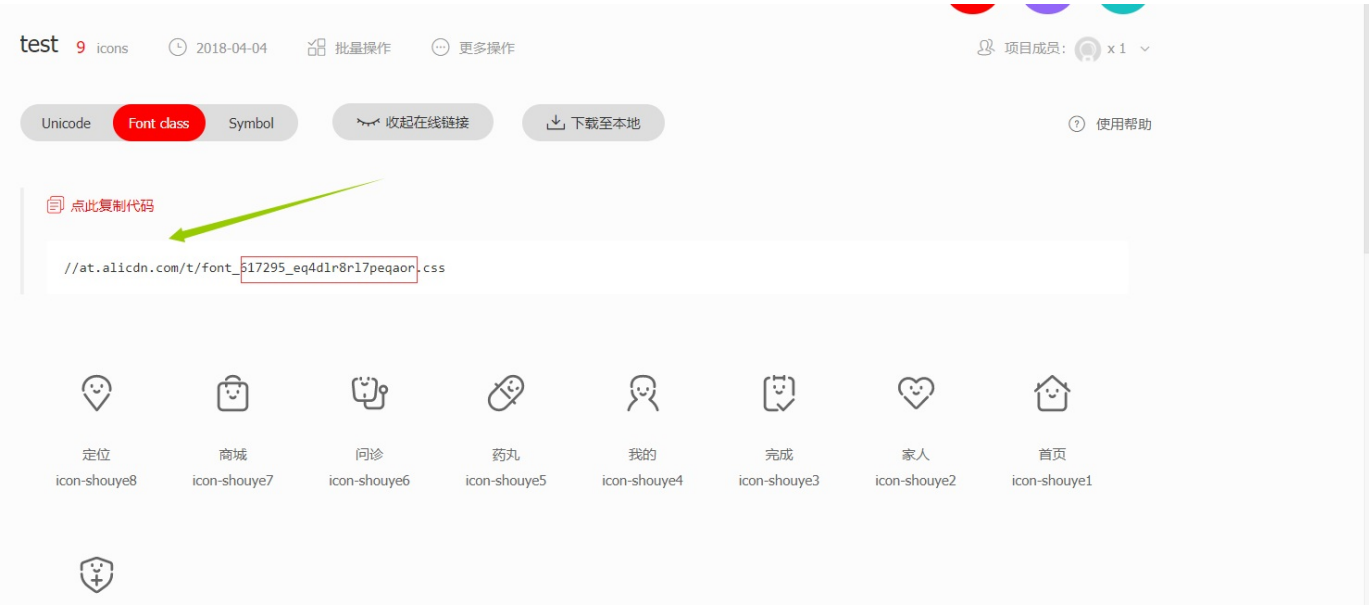
图标配置

图标配置

项目采用了2中加载图标的方式（本地 / 网络）

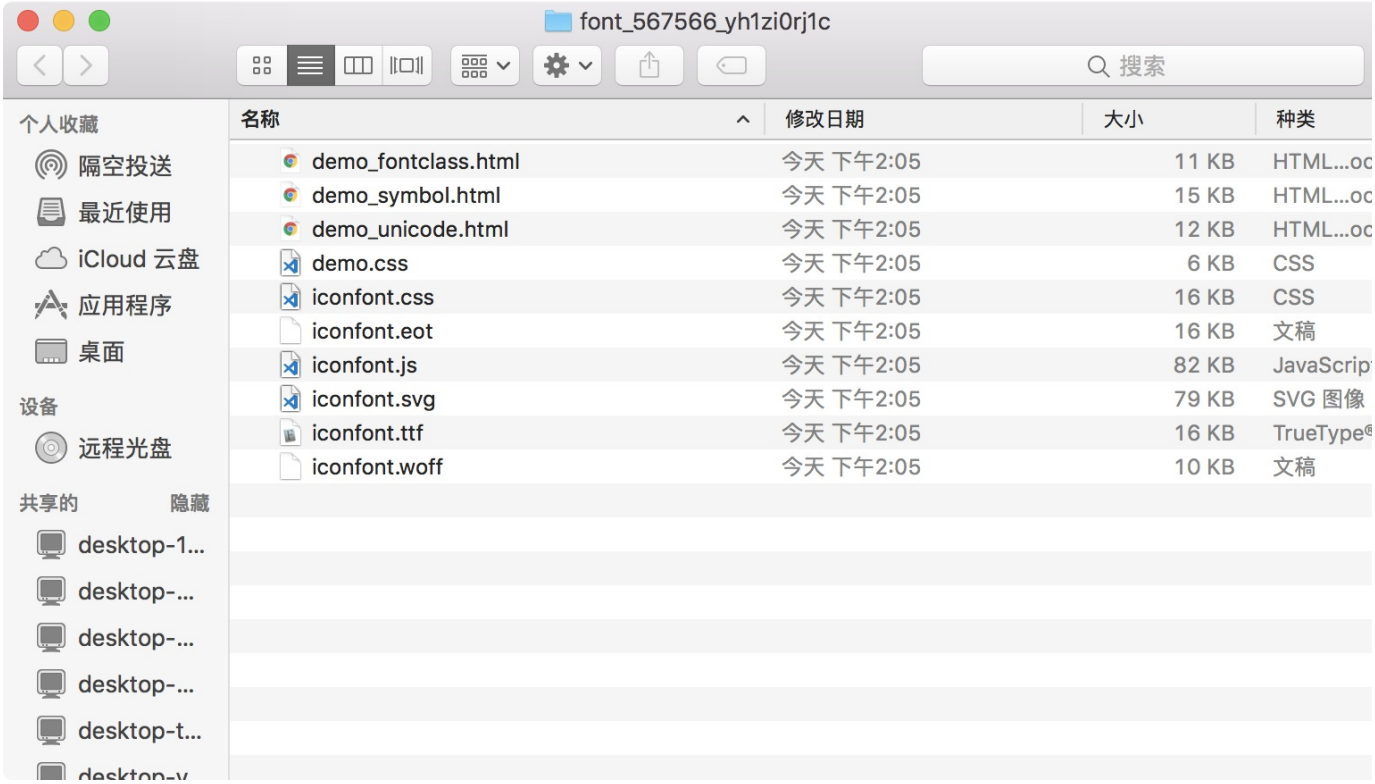
准备工作

- 先去阿里巴巴图标库注册一个账号
- 完后新建一个项目
- 选择图标导入项目
- 生成在线链接 / 或者下载图标文件



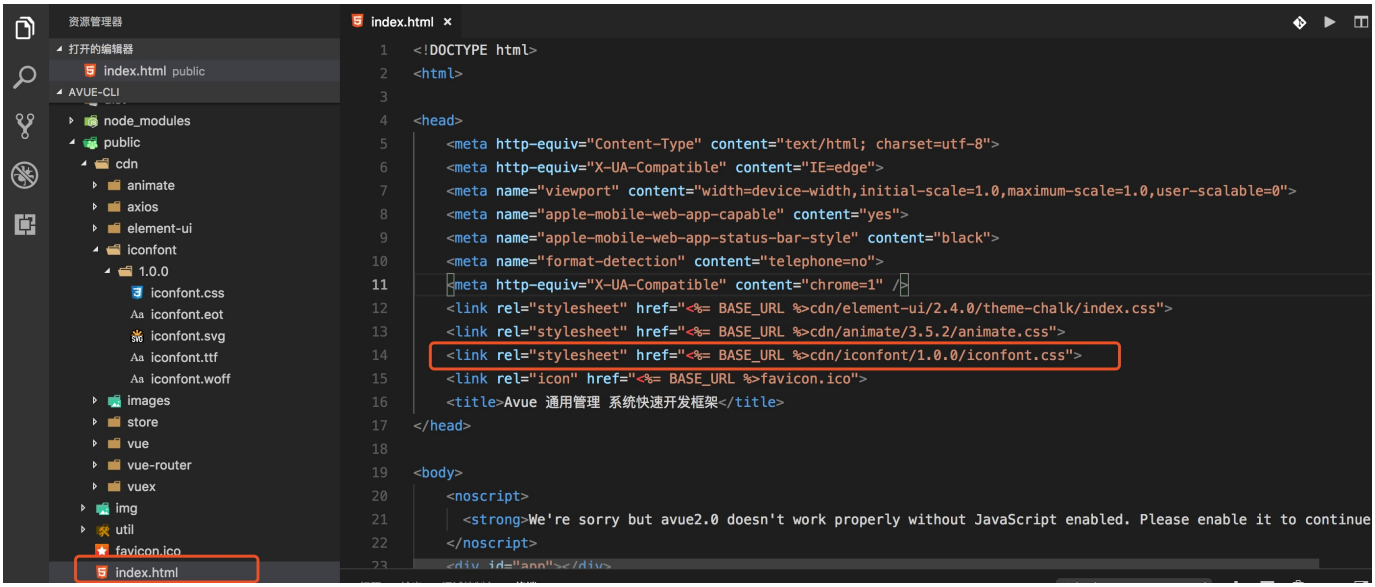
本地加载

1. 下载图标源文件



2. 复制相关文件到/public/cdn/iconfont/1.0.0/目录下

3. 在html中引入样式地址



4. 加入iconfont.css中的全局样式

```
[class^="icon-"]{
    font-family: "iconfont" !important;
    /* 以下内容参照第三方图标库本身的规则 */
    font-size: 18px !important;
    font-style: normal;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}
```

网络加载

直接把生成的连接放入index.html中的header头部引入即可

图标使用

在系统中采用class方式即可使用图标，左侧菜单配置图标直接写icon-图标名称即可

图表库地址

https://www.iconfont.cn/collections/detail?cid=14298

```
<i class="icon-图标名称"></i>
```

样式配置

样式配置

项目使用了sass框架（面向对象css开发）,按照功能和模块来的分出来不同的样式模块。

```
├──animate //全局动画
├──theme //主题配置
├──common.scss  //基础样式
├──media.scss //多终端适配
├──element-ui.scss  //覆盖ele的样式
├──mixin.scss  //工具类包(滚动条等)
├──sidebar.scss //侧边菜单
├──tags.scss  //选项标签
├──top.scss   //顶部
├──variables.scss  //全局scss变量
```

划分依据

在公用底层组件中有 `top` ， `menu` ， `sidebar` ， `tags` 等几大组件，他们的样式独立维护，便可以单独分出模块，其他划分根据自己系统的业务需求和实际用途合理划分即可，底层组件如下

- sidebar 侧边栏
- tags 标签栏
- top 顶部栏
- index 主框架
- 其他

路由配置

路由配置

项目采用了 `vue-router` 路由，由于页面比较多的化写在一个js文件里难以维护管理，所以按照不用的模块和用户去划分路由，最后在导入 `/src/router/router.js` 文件里，由 `addRoutes` 方法添加进去

静态路由

- 基础路由（/src/router/page/index.js）
  - 登陆页
  - 主页
  - 锁屏页
  - 404，403，401错误页面
  - ...
- 业务路由（/src/router/views/)
  - 增删改查页面
  - 用户管理
  - 角色管理
  - 菜单管理
  - ...
- 其他
  - ...

```
import PageRouter from './page/'
import ViewsRouter from './views/'
Router.addRoutes([...PageRouter, ...ViewsRouter]);
```

如果你的模块比较多，可以按照下面的例子划分

```
├──router
  ├──user/index.js  //用户路由
  ├──base/index.js   //基础路由
  ├──user/index.js   //用户路由
  ├──logs/index.js   //日志路由
  ├──dic/index.js   //字典模块路由
```

这里需要的是菜单的路由是不用配置，他会请求菜单数据利用 `avue-router` 方法中动态自动添加的。

配置说明

在对应路由的meta中配置对应的参数即可

- keepAlive：是否开启缓冲
- isTab：是否添加到标签栏里
- isAuth：是否需要认证才能打开

```
{
        path: '/login',
        name: '登录页',
        component: () =>
            import ( /* webpackChunkName: "page" */ '@/page/login/index'),
        meta: {
            keepAlive: true,
            isTab: false,
            isAuth: false
        }
    }
```

ajax配置

ajax配置

项目采用了前后端分离，采用 `ajax` 发送 `json` 的数据格式和服务器交互，因此在 `ajax` 也要配置一些全局配置（项目采用了axios框架）,全局配置如下

1. 发送 `ajax` 的等待进度条
2. 全局 `status` 错误日志的处理和提示
3. 请求中区分是否携带 `token`
4. 发送数据是否需要表单序列化

axios有2大部分拦截器

- request发送数据之前拦截器

```
...
    NProgress.start() // start progress bar
    const isToken = (config.data || {}).isToken === false
    if (getToken() && !isToken) {
        config.headers['Authorization'] = 'Bearer ' + getToken() // 让每个请
求携带token--['Authorization']为自定义key 请根据实际情况自行修改
    }
    //headers中配置serialize为true开启序列化
    if (config.methods === 'post' && config.headers.serialize) {
        config.data = serialize(config.data);
        delete config.data.serialize;
    }
    return config
}, error => {
    return Promise.reject(error)
});
...
```

- response收到数据之后的拦截器

```
...
axios.interceptors.response.use(res => {
        NProgress.done();
        const status = Number(res.status) || 200;
        const statusWhiteList = website.statusWhiteList || [];
        const message = res.data.message || '未知错误';
        //如果请求为200则放过，否者默认统一处理,或者在website中配置statusWhiteList
白名单自行处理
        if (status !== 200 && !statusWhiteList.includes(status)) {
            Message({
```

```
                message: message,
                type: 'error'
            })
            return Promise.reject(new Error(message))
        }
        return res;
    }, error => {
        NProgress.done();
        return Promise.reject(new Error(error));
    })
    ...
```

错误日志捕获

错误捕获配置

在 `src/error.js` 文件里主要代码,主要利用的了 `vue` 的 `errorHandler` 方法

```
Vue.config.errorHandler = function(err, vm, info) {

    Vue.nextTick(() => {
        //添加错误日志
        store.commit('ADD_LOGS', {
            type: 'error',
            message: err.message,
            stack: err.stack,
            info
        })
        if (process.env.NODE_ENV === 'development') {
            console.group('>>>>>> 错误信息 >>>>>>')
            console.log(info)
            console.groupEnd();
            console.group('>>>>>> Vue 实例 >>>>>>')
            console.log(vm)
            console.groupEnd();
            console.group('>>>>>> Error >>>>>>')
            console.log(err)
            console.groupEnd();
        }
    })
}
```

二次开发

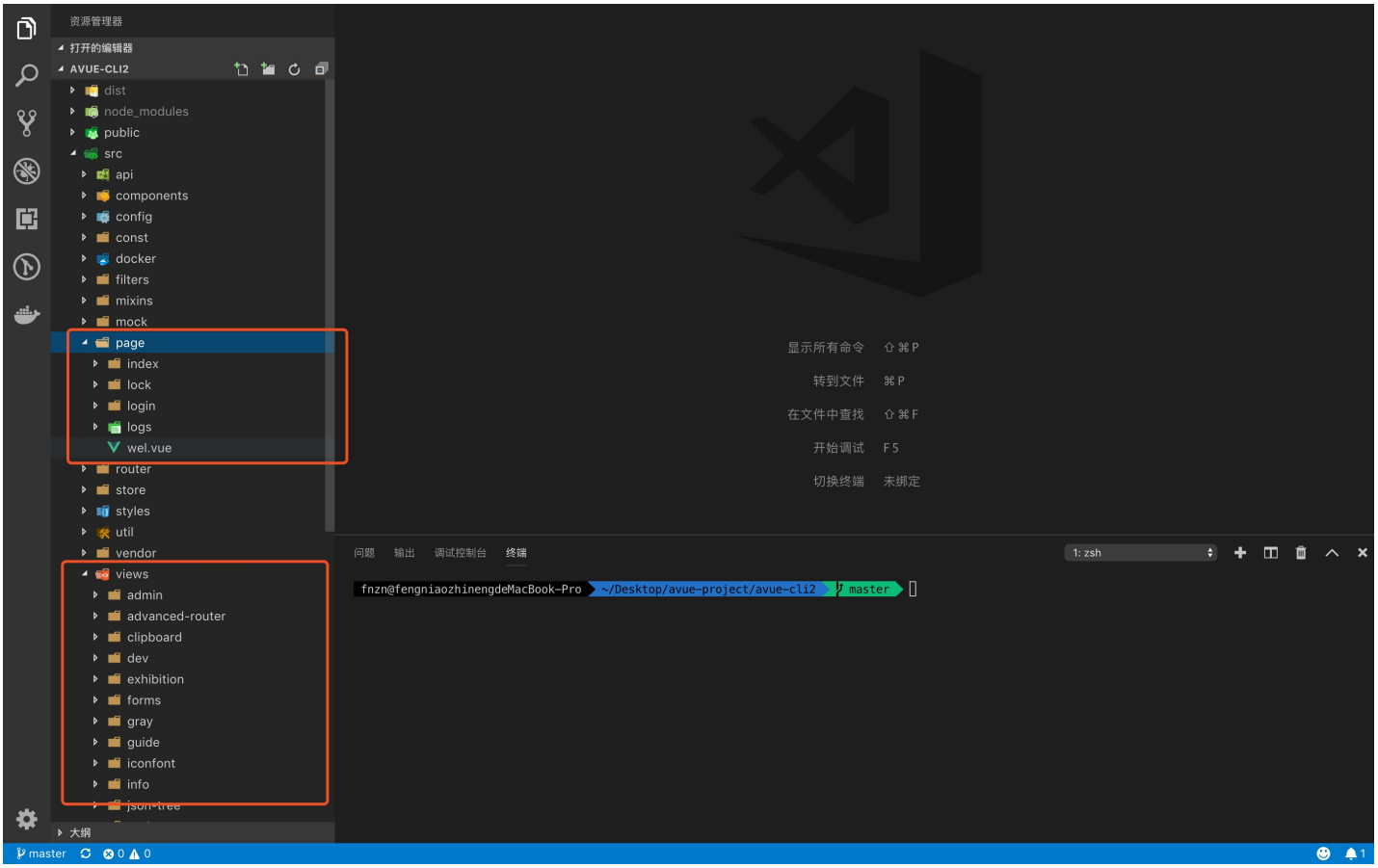
如何添加页面

如何添加页面

项目中主要分了2部分页面，一部分是底层页面page文件夹，另一个是views开发文件夹

page——主要存放底层的框架组件，如标签栏，左侧菜单栏，登录等（page最好不要修改，可以保证每次升级平滑过度）。

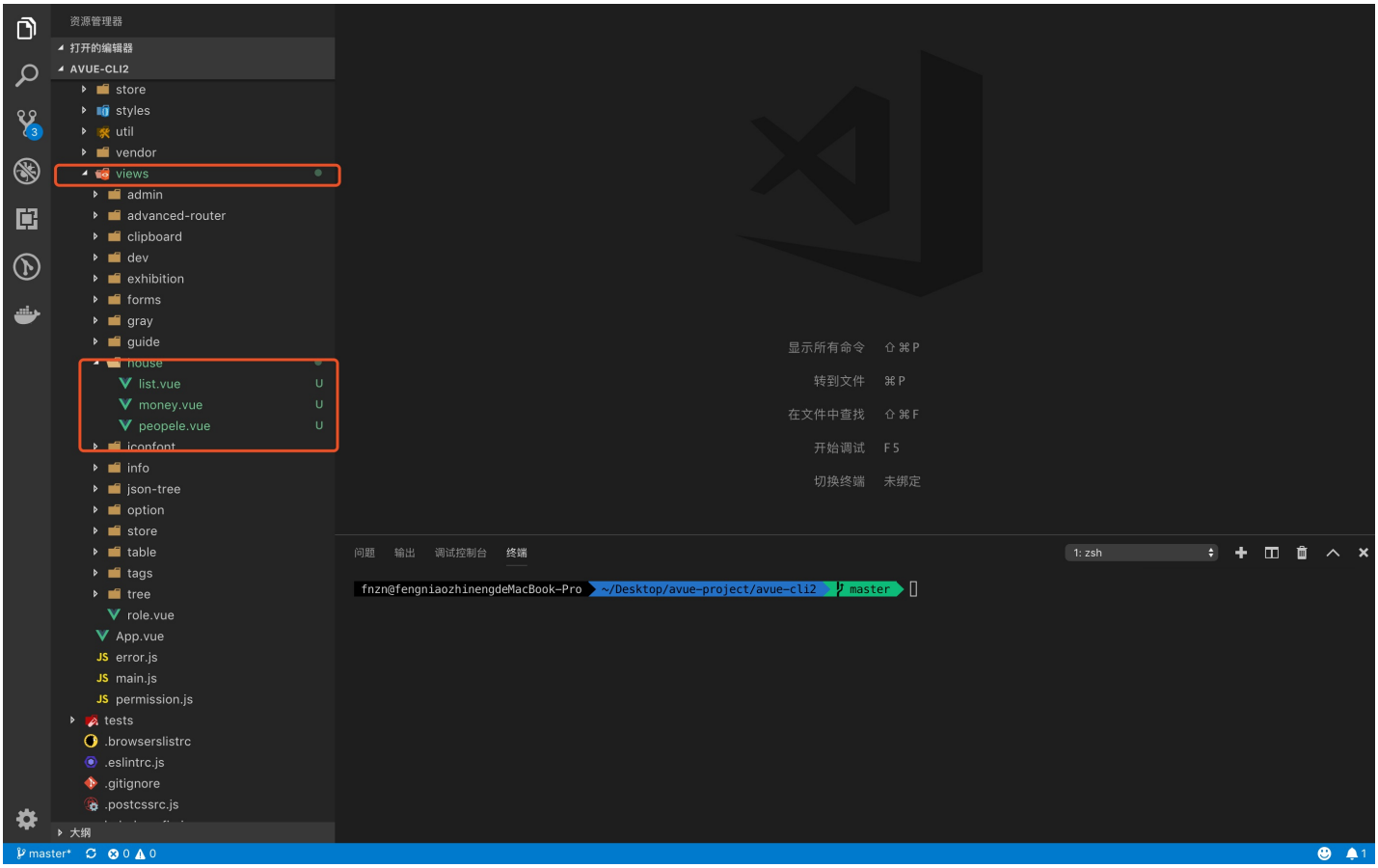
views——主要存放业务页面和项目页面，你可以在views里面按照模块再划分文件夹。



例子

我们比如开发一个【售房模块】，比如有【房子列表】，【客户经理列表】，【销售额度列表】等。建立如下目录

```
├──house //售房模块
        ├──list.vue //房子的列表
        ├──money.vue //销售额度的列表
        ├──people.vue //客户经理列表
        ├──default.vue //其他模块
```

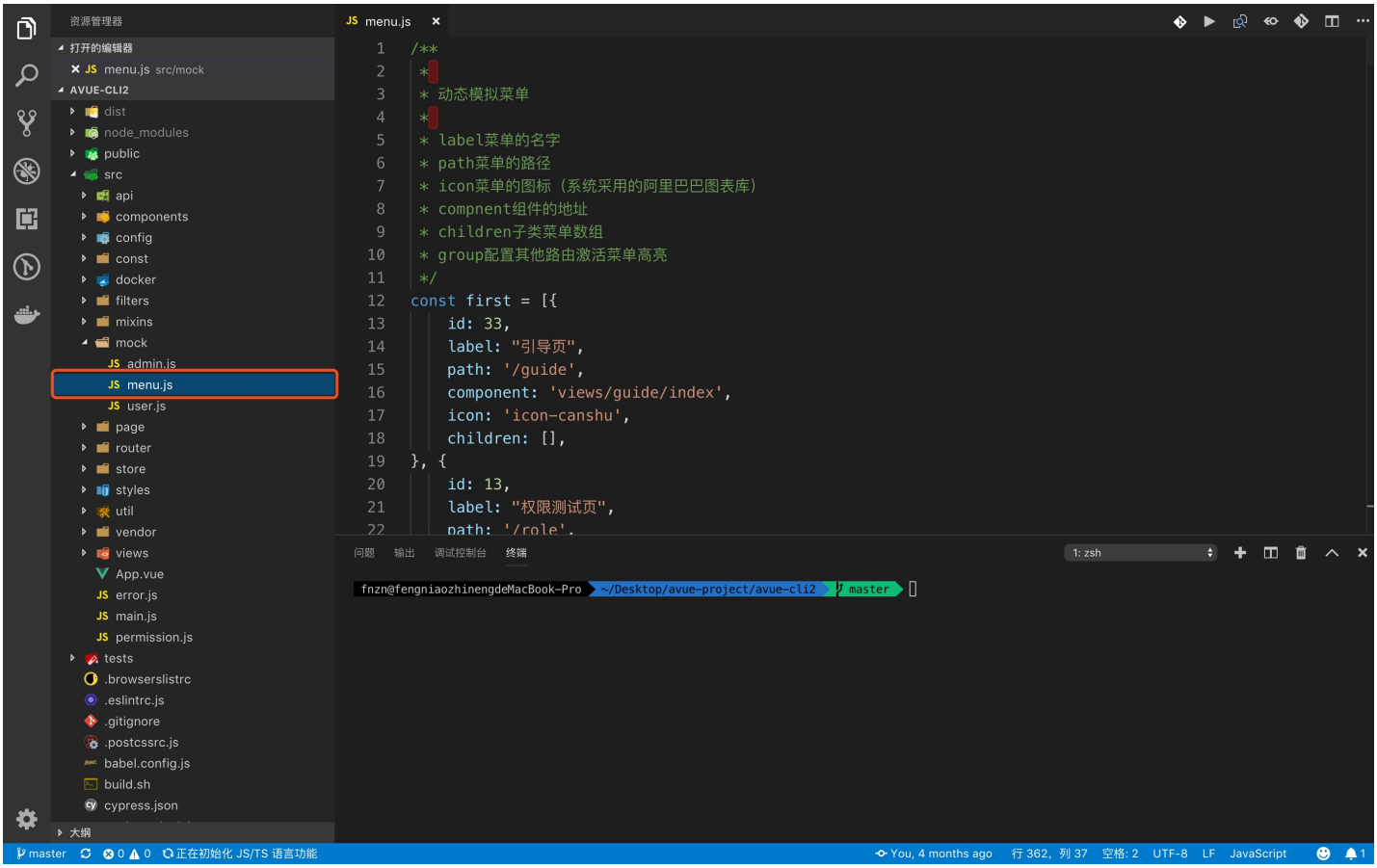


这样我就建立好了页面，接下来去配置【路由和菜单】

如何添加路由和菜单

如何添加左侧菜单

项目采用的是模拟数据，所以在项目中找到mock模拟数据修改



参数说明

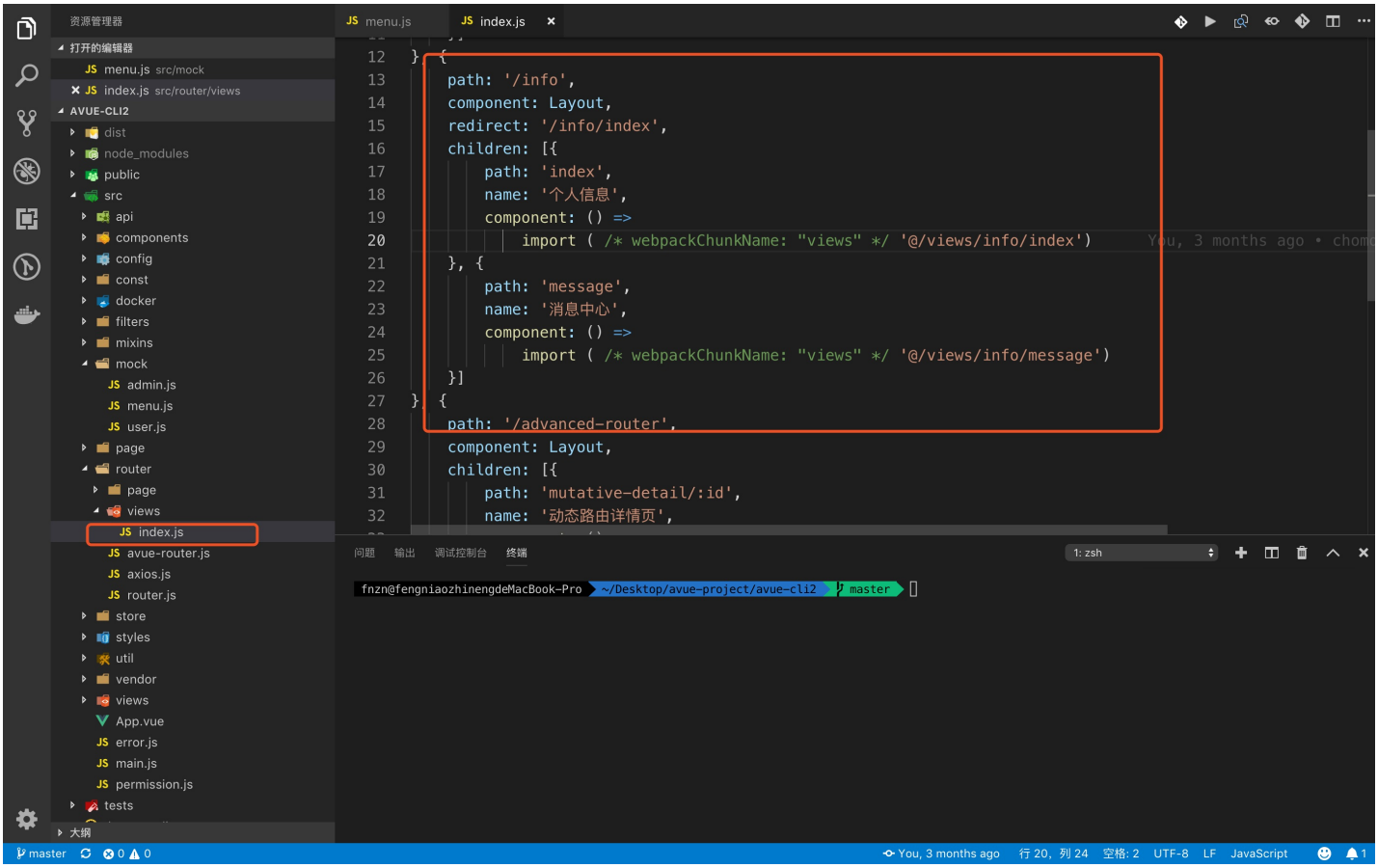
- label——菜单的名字
- path——菜单的路径
- icon——菜单的图标
- compnent——前端组件的地址
- children——子类菜单
- group——配置其他路由激活菜单高亮

1.系统用的是动态路由加载到vue-router中，所哟无需在router里面再去配置。

2.如果非左侧菜单的路由，没有动态加载，则需要单独去router里面配置，具体配置规则参考【路由配置详解】

3.compnent前端组件的地址第三方网站的话直接写完整的url即可。

4.icon是图标地址，项目采用阿里巴巴图表库，具体配置参考【图标配置】



例子

我们上次建立【售房模块】模块的页面，分析可以得知。

【房子列表】/【客户经理列表】需要展示在左侧菜单里，

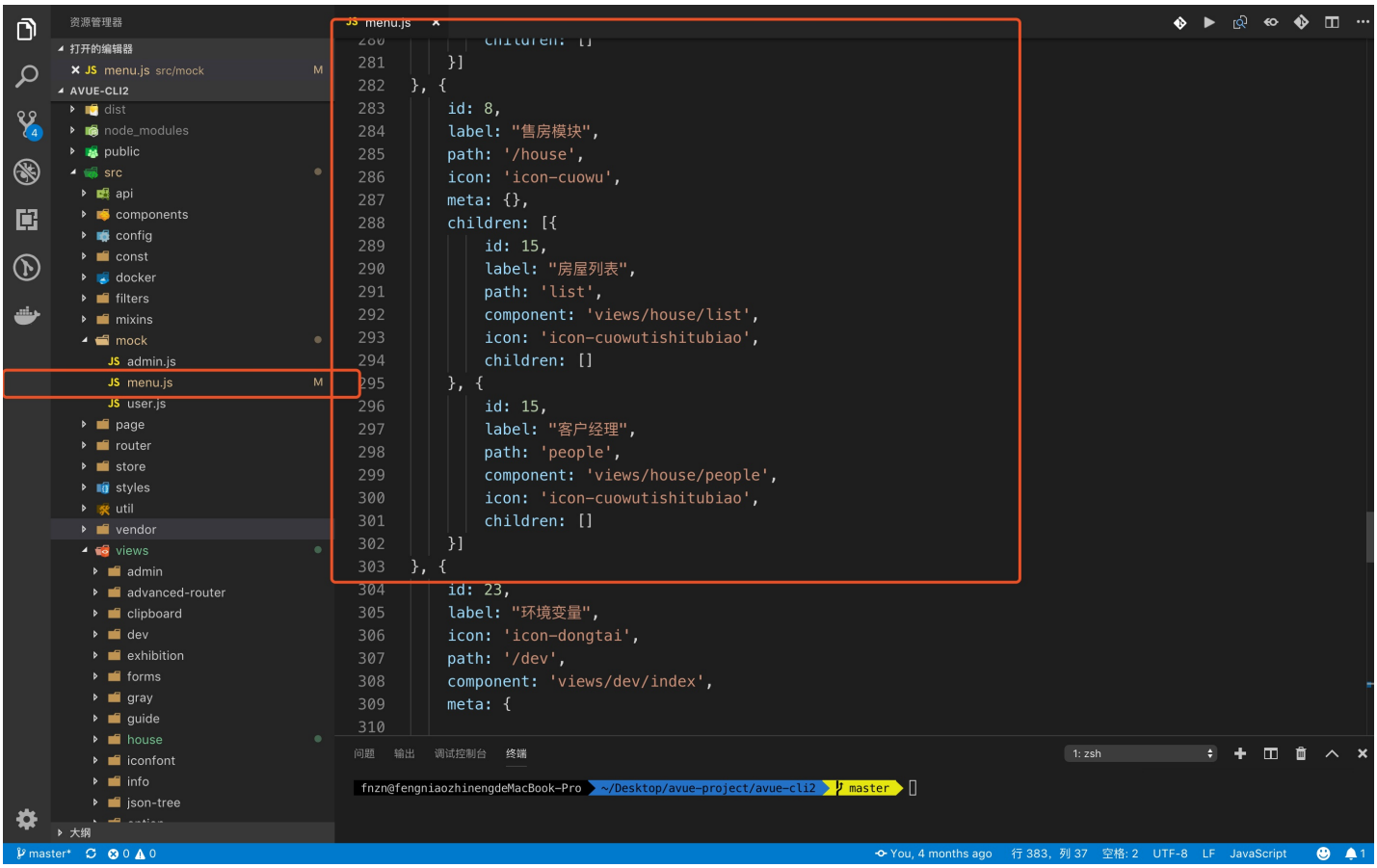
【销售额度列表】需要进入【客户经理】列表页面完后点击详情查看，无需配置到左侧菜单里

配置左侧菜单路由

...

```
    id: 8,
    label: "售房模块",
    path: '/house',
    icon: 'icon-cuowu',
    meta: {},
    children: [{
        id: 15,
        label: "房屋列表",
        path: 'list',
        component: 'views/house/list',
        icon: 'icon-cuowutishitubiao',
        children: []
    }, {
        id: 15,
        label: "客户经理",
        path: 'people',
```

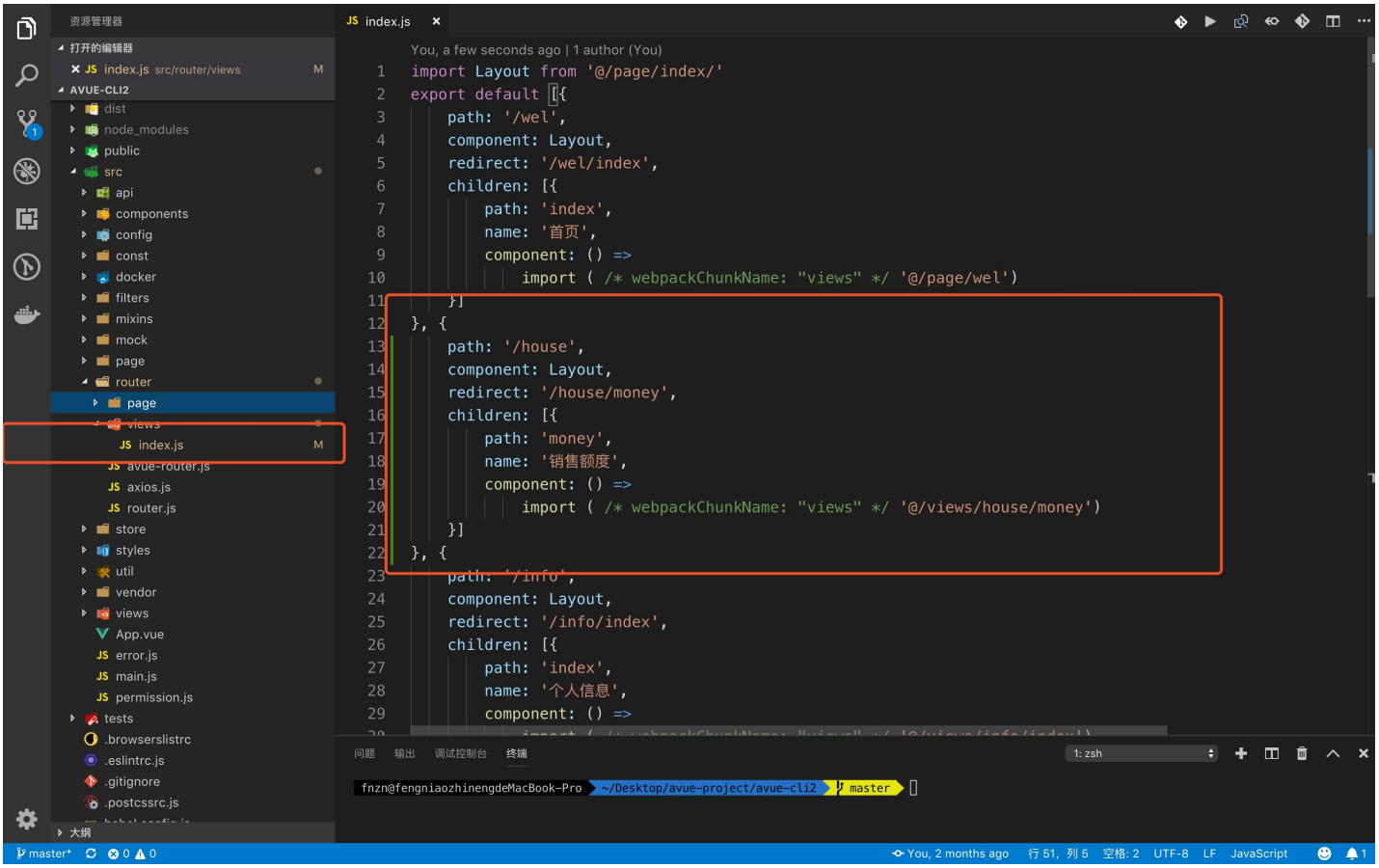
```
        component: 'views/house/people',
        icon: 'icon-cuowutishitubiao',
        children: []
    }]
}
...
```



这样子就会在左侧显示出来，接下来配置不是左侧菜单的路由

配置路由

```
...
 {
    path: '/house',
    component: Layout,
    redirect: '/house/money',
    children: [{
        path: 'money',
        name: '销售额度',
        component: () =>
            import ( /* webpackChunkName: "views" */ '@/views/house/money')
    }]
}
...
```



这样我们就完成了左侧菜单和非左侧菜单的路由

如何添加api(ajax请求)

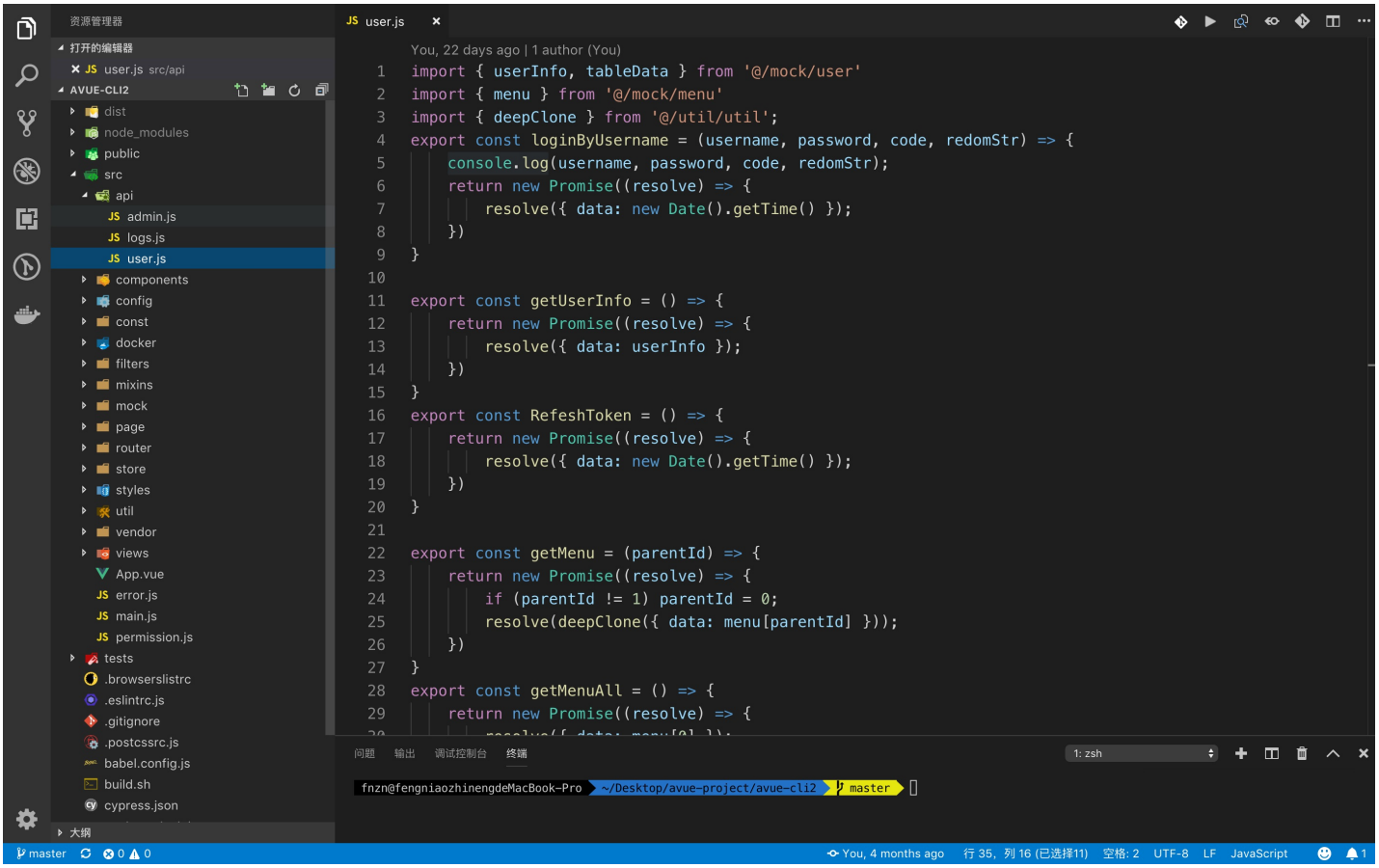
如何添加api(ajax请求)

由于项目是前后端分离，所以采用ajax请求的方式，单手我们前端也要模块化，所哟ajax请求不能混写在页面里，一来不好维护，二来没有打包模块化的需求。
项目中单独了一个api文件夹用来存放模块的请求，同时和views页面视图一样也可以按照功能模块划分。

例子

下面采用的是模拟数据，并没有ajax请求，采用mock拦截ajax的方式模拟出数据。替换真实的ajax即可，ajax框架很多，你可以选择一个你常用的，项目采用了axios包。

```
import request from '@/router/axios'
...
export function fetchList(query) {
    return request({
        url: '/admin/user/page',
        method: 'get',
        params: query
    })
}
...
```



在对应的业务页面中导入这个方法调用，完后加载数据，编写相应的逻辑即可。

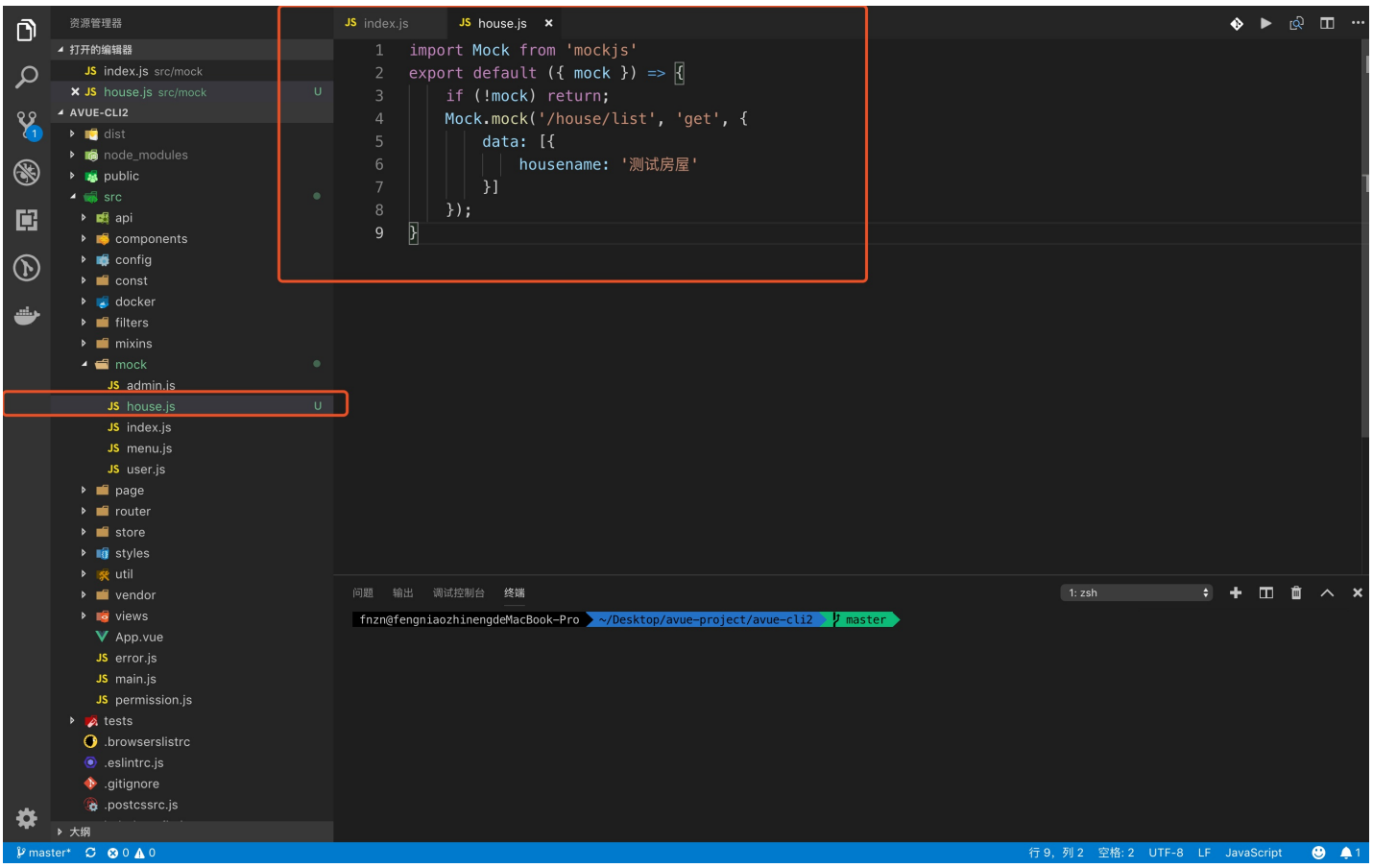
如何添加mock数据

如何添加mock数据

如果在开发过程中，后端并没有提供我们真实的api接口，我们可以采用mock的方式来拦截ajax模拟数据，项目中 `src/mock` 目录就是模拟数据

例子

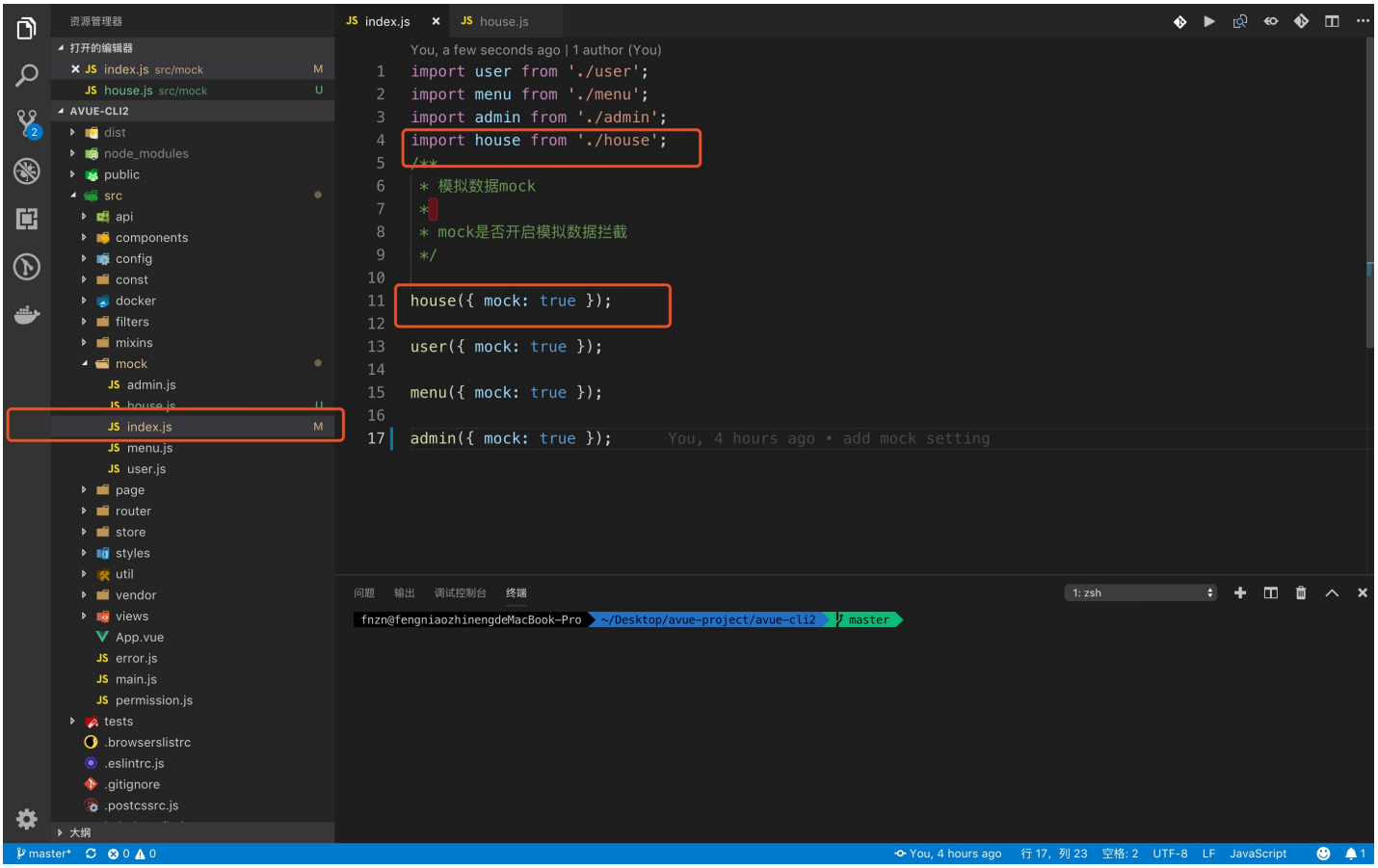
我们以售房模块的房子列表为例子，建立一个mock数据，目录结构如下




```
import Mock from 'mockjs'
export default ({ mock }) => {
    if (!mock) return;
    Mock.mock('/house/list', 'get', {
        data: [{
            housename: '测试房屋1'
        },{
            housename: '测试房屋2'
        }]
    });
}
```

我们用 `mock` 这个个参数来控制是否加载模拟数据，当然你也可以配置更多参数自由去控制，之后我们在

index.js引入我们新建立的 `house` 模块，设置 `mock` 为 `true` 开启模拟数据，上一篇【如何添加api(ajax请求)】中就可以调用了，当你需要对接真实的接口时，设置 `mock` 为 `false` 即可。



本地cdn

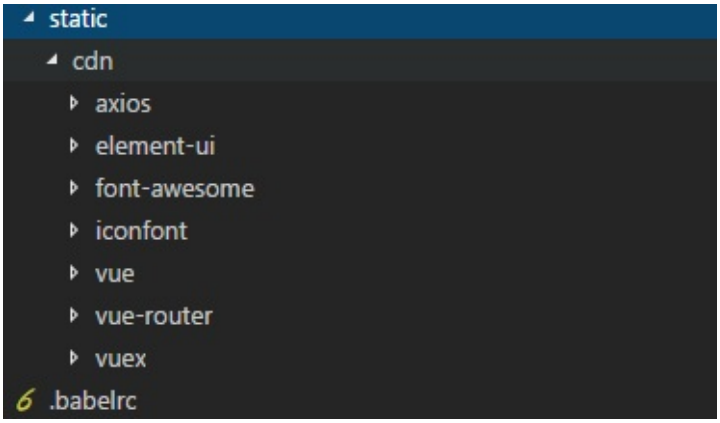
背景

第三方库文件可以使用第三方的cdn，或者把资源文件下载到static文件里，从而减少了vendor的体积大小，webpack的打包忽略第三方库的配置（webpack.base.conf.js）

```
externals: {
  'vue': 'Vue',
  'vue-router': 'VueRouter',
  'vuex':'Vuex',
  'axios': 'axios',
  'element-ui':'ELEMENT',
}
```

要是内网环境下，你又懒得搭建cdn，就把相应的库文件下载到本地来使用，像往常一样引入html里就可以

本地库文件



引入ihtml文件

```
<body>
        <div id="app">
            <!--vue-ssr-outlet-->
        </div>
        <script src="/static/cdn/vue/2.5.2/vue.min.js" charset="utf-8"></script>

        <script src="/static/cdn/vuex/2.4.1/vuex.min.js" charset="utf-8"></scri
pt>
        <script src="/static/cdn/vue-router/3.0.1/vue-router.min.js" charset="u
tf-8">  </script>
        <script src="/static/cdn/axios/1.0.0/axios.min.js" charset="utf-8"></sc
ript>
        <script src="/static/cdn/element-ui/2.0.5/index.js" charset="utf-8"></s
```

```
 cript>
    </body>
```

打包优化

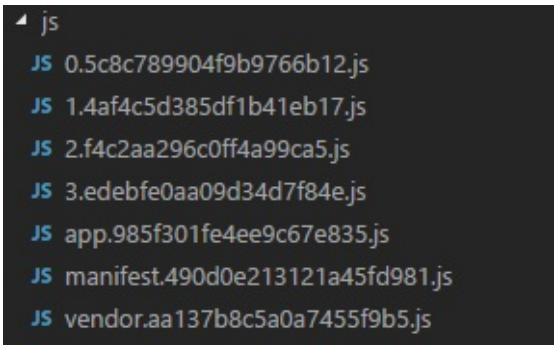
背景

vue是一个单例页面，所有页面都是由js动态渲染，因此js的体积会越来越大，从而影像页面的渲染性能

异步组件

官方推出了异步组件路由，是一个解决js打包后体积过大问题的方案，可以很好的吧vue组件按组区分，从而打出不同的js文件，解决了单一js打包大的问题。

以下是一个异步组件打包的最终文件



- 后面的数字为hash值，为了js不会缓冲
- app为vue以及vue-router等第三方库的额外配置代码
- vendor为第三方库插件的本身代码
- mainfest为第三方库runtime代码
- 数组部分就是不同分组的异步打包文件（几个组就有几个js文件）

注意：千万不要把js的组件分的颗粒度很小，js文件多了也会影响效率，浏览器加载js的线程一般为4-5个

官方异步组件传送门

webpack打包配置

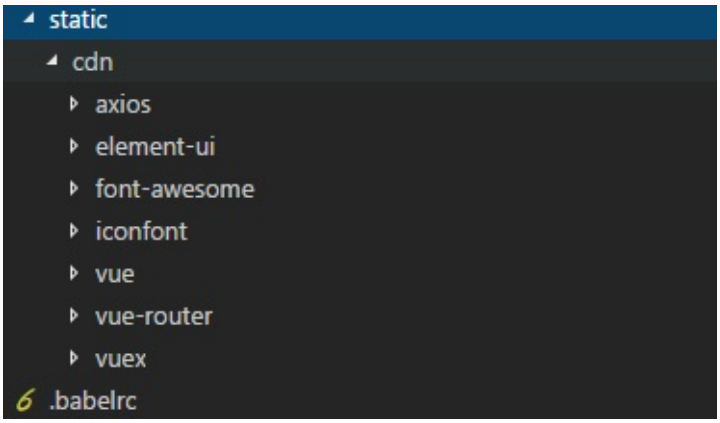
第三方库文件可以使用第三方的cdn，从而减少了vendor的体积大小。

webpack的打包忽略第三方库的配置 （webpack.base.conf.js）

```
externals: {
  'vue': 'Vue',
  'vue-router': 'VueRouter',
  'vuex':'Vuex',
  'axios': 'axios',
  'element-ui':'ELEMENT',
}
```

要是内网环境下，你又懒得搭建cdn，就把相应的库文件下载到本地来使用，像往常一样引入html里就可以

本地库文件



引入html文件

```
    <body>
        <div id="app">
            <!--vue-ssr-outlet-->
        </div>
        <script src="/static/cdn/vue/2.5.2/vue.min.js" charset="utf-8"></script>

        <script src="/static/cdn/vuex/2.4.1/vuex.min.js" charset="utf-8"></scri
pt>
        <script src="/static/cdn/vue-router/3.0.1/vue-router.min.js" charset="u
tf-8"></script>
        <script src="/static/cdn/axios/1.0.0/axios.min.js" charset="utf-8"></sc
ript>
        <script src="/static/cdn/element-ui/2.0.5/index.js" charset="utf-8"></s
cript>
    </body>
```

bootcdn官方传送门

实验证明方法可以减少60%的体积文件，渲染速度大大提升

SSR

ssr就是把前端的渲染改成又后台直接渲染，将渲染好的页面直接呈现给客户，此方法可用于网站的百度推广和seo优化有重大的意义,常用的后台渲染框架有nuxtjs

4.项目拆分

一个项目中的不同模块拆分出来，都可以独立成vue项目，采用iframe标签进行跳转，验证方面你可以制作一个公用的js验证库与服务端进行交互。（有个类似于后台微服务的体系）

此方法还在实验阶段，只是个人的想法

生产部署

nginx部署

nginx部署

将打包好的dist文件全部复制出来，放入任意一个目录中

本次例子放到

../nginx-1.13.10/html/

配置nginx配置文件监听80端口指向上面的目录

html是相对nginx的安装路径

```
...
server {
        listen       80;
        server_name  localhost;
        location / {
            root   html;
            index  index.html index.htm;
        }
        #这里对应后端的服务地址
        location ~* ^/(api|auth|app|entrance-guard) {
           proxy_pass 后台服务ip;
           #proxy_set_header Host $http_host;
           proxy_connect_timeout 15s;
           proxy_send_timeout 15s;
           proxy_read_timeout 15s;
           proxy_set_header X-Real-IP $remote_addr;
           proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        }
}
...
```

启动nginx，访问http://localhost即可,如果部署到你的外网服务器，访问外网ip即可

tomcat部署

tomcat部署

将打包好的dist文件夹里文件复制出来，放入tomcat的webapp下面

新建ROOT文件夹

进入bin文件夹，startup.bat启动tomcat即可tomcat是8080端口

启动tomcat，访问http://localhost:8080即可,如果部署到你的外网服务器，访问外网ip即可

docker部署

docker部署

将dist文件夹和dockerfile文件放到一起

docker部署

dockerfile文件

```
FROM nginx
VOLUME /tmp
ENV LANG en_US.UTF-8
ADD ./dist/ /usr/share/nginx/html/
EXPOSE 80
EXPOSE 443
```

执行打包镜像

```
docker build -t avue:1.0 .
```

执行运行容器

```
docker run -itd -p 7777:80 -v /mnt/:/mnt/ --name avue --restart unless-stopped
 avue:1.0
```

引入query

引入Jquery

在public目录下放入jquery相关包，在index.html引入该包,这时候即注册成window下面的全局变量

在项目中直接使用即可

```
//调用
window.$('#test')
```

当然你也可以挂载到Vue的原型链上

```
//引入
Vue.prototype.$ = window.$
//调用
this.$('#test')
```

干货分享

干货分享

学习相关

- JavaScript链接：http://pan.baidu.com/s/1eQ3yxcm 密码：onfp
- es6深入学习: https://www.kancloud.cn/kancloud/es6-in-depth/45513
- VUE链接： https://pan.baidu.com/s/1XgnmxrKBg5nl6wtuM8juQQ 密码：pa64

关网网站

- Vue中文帮助网站：http://cn.vuejs.org/
- Vue github开源地址：https://github.com/vuejs/vue
- Vue.js专业中文社区：http://www.vue-js.com/
- 前端开发Vue相关：http://www.opendigg.com/tags/front-vue

小知识1

小知识1

vue和jquery的区别

- 赋值

> jquery

```
<input id="test" type="text"/>
```

```
//获取dom
var dom = document .getElementById('test);
//取值
console.log(dom.val());
//赋值
dom.val('这里赋值');

~~输入框输入值后，取值赋值都是手动操作，手动赋值后需要手动更新视图(输入框才会更新)
```

> vue

```
<input v-model="test" type="text"/>
```

```
//取值
console.log(this.text);
//赋值
this.text = '这里赋值';
```

~~输入框输入值后，自动赋值变量，手动赋值的时候也会改变视图

- 循环

> jquery

```
<div id="test"></div>
```

```
//获取dom
var dom = document .getElementById('test);
var list=[{
    label:'测试',
    value:1
}]
var str = '';
for(var i =0;i<list.length;i++){
    str = '<span id="+list[i].value+">+list[i].label+'</span>'
}
dom.innerHtml(str);
~~list值改变每次更新视图都需要操作字符串，同时复杂的dom用字符串的形式难以维护
```

> vue

```
<div>
   <span :id="item.value" v-for="item in list">{{item.label}}</span>
</div>
```

```
var list=[{
    label:'测试',
    value:1
}]
```

~~直接在dom上自动绑定,复杂dom也好维护，同时list数据改变，自动更新视图

- vue和react以及angular的区别

> 流行的三个框架思想都是一至，在写法，调用，以及性能上可能有着不同的差别

1. angular -> react - > vue
2. vue,react -> angular
3. react -> vue

~~ angular - > angularjs

---

jquery注意

- 循环拼接字符串

```
var str = '';
for(var i =0;i<list.length;i++){
```

```
    str = '<span id="'+list[i].value+'">'+list[i].label+'</span>'
}
dom.innerHtml(str);
```

这样子层次复杂很难维护，可以 采用如下写法

```
<div id="test"></div>
<script id="temp">
    <span>姓名：${label}</span>
    <span>名称：${value}</span>
</script>
```

```
var temp =  document .getElementById('temp');
var dom =  document .getElementById('test');
var str = temp.innerHtml;
for(var i =0;i<list.length;i++){
  var ele = list[i];
  str = str.replace('${label}',ele.label);
  str = str.replace('${value}',ele.value);
}
dom.innerHtml(str);
```

~~ 这样子好处是不用拼接字符串 ，以类模版的形式

- 单引号和双引号

> 页面中单引号和双引号都是生效的，es6最新规定字符串全部用单引号，标签属性用双引号

```
//单引号
 '<span id="'+list[i].value+'">'+list[i].label+'</span>'

//双引号,这里的属性双引号需要转义
 "<span id=\""+list[i].value+"\">"+list[i].label+'</span>'
```

- 回调地狱——ajax

  深层次的代码嵌套,代码难以维护，以函数的方式分割

```
//回调嵌套
//处理逻辑1
$.ajax(...,function(res){
       var list =res;
           ...
        //处理逻辑2
       $.ajax(...,function(res){
```

```
            var list =res;
                ...
            //处理逻辑3
            $.ajax(...,function(res){
                var list =res;
                    ...
                })
        })
})
```

```
//链式调用
//处理逻辑1
function test1(params){
    return new Promise(function(resolve,reject){
          $.ajax(...,function(res){
                    var list =res;
                    resolve(rest)
        })
    })

}
//处理逻辑2
function test2(params){
   return new Promise(function(resolve,reject){
          $.ajax(...,function(res){
                    var list =res;
                    resolve(rest)
        })
    })
}
//处理逻辑3
function test3(){
     return new Promise(function(resolve,reject){
          $.ajax(...,function(res){
                    var list =res;
                    resolve(rest)
        })
    })
}


test1().then(function(res){
    ...
    return test2(res);
}).then(function(res){
    ...
     return test3(res);
})).then(function(res){
    ...
```

```
})
```

- 字符串凭借处理逻辑

  在拼接的字符串中串逻辑

```
//正常写法
label = list[i].label;
label = label +'/test';
value = list[i].value;
value = value +'/test';
 '<span id="'+value+'">'+label+'</span>'


//巧用数组写法

function value(list){
   value = list[i].value;
   value = value +'/test';
}
var list = [
    '<span id="',
    (function(){
        label = list[i].label;
        label = label +'/test';
        return label;
    })(),
    '">',
    value(list),
    '</span>'
]

console.log(list.jion(''));

~~ layui框架中大量拼接字符串都是使用该方法
~~ split 和join（字符串拼接常用的方法）
```

- js和jquery尽量不要混合何用

jquery是js的增强包有着满足大量需求api包，不要忽略jquey的包手动去写，

这样子让代码不易维护，如果找不到合适api，自己造api;

```
//普通写法
for(var i =0;i<list.length;i++){
  var ele = list[i];
  console.log(ele,i);
```

```
}

//jquery写法
$.each(list,function(index,ele){
    console.log(ele,index);
});
```

自己造api也就是自己封装工具包，例如加水印，图片压缩，日期格式化等，也可以直接使用别人的工具包，不能满足自己需求的，二次开发，封装成自己的包。
开发几年后，就会发现自己攒了很多自己的库（提供开发效率），比较好的库可以开源让更多人参与维护和使用。

- github(全球)
- gitee(国内)
- 例子-水印生成器

  view-source:https://canvas.avue.top/

  https://canvas.avue.top/

vue注意

- {{}}和v-text区别

  如果网速慢的情况下花括号会显示出来，而v-text知道数据出来才会显示
- v-model绑定对象的时候

  双向绑定数据的时候一定要在data里面申明，否则无法绑定数据

```
<input v-model="test" type="text"/>
```

```
data(){
    return {
        text:'',
    }
}
```

~~ 赋值时候视图无更新的情况，可以用this.$set方法强制赋值
~~ 如果绑定对象是深层次的结构对象，没在data里面申明深层次的，而且要在v-model下水用,一定要用this.$set赋值

- vue也内置了dom操作方法

项目中使用vue时，最好不好用jquery包，因为他也内置了dom操作包，如果混合使用由于生命周期的问题会产生很多未知问题

~~给标签加一个ref属性，直接this.$ref.name即可

```
<div ref="test"></div>
```

```
mounted(){
   this.$ref.test.className= '样式名称'
}
```

小知识2

小知识1

一、常见的菜单列表

我们以一个菜单的案例展开，先来看一个平常程序员怎么常规的vue文件如何写？

以element-ui为基础组件的部分代码

```
<el-row :span="24">
    <el-col span="8" @click="event1">
        <i class="icon-caidan1" :style="{backgroundColor:color1}"></i>
        <p>菜单1</p>
    </el-col>
    <el-col span="8" @click="event2">
        <i class="icon-caidan2" :style="{backgroundColor:color2}"></i>
        <p>菜单2</p>
    </el-col>
    <el-col span="8" @click="event3">
        <i class="icon-caidan3" :style="{backgroundColor:color3}"></i>
        <p>菜单3</p>
    </el-col>
    <el-col span="8" @click="event4">
        <i class="icon-caidan4" :style="{backgroundColor:color4}"></i>
        <p>菜单4</p>
    </el-col>
</el-row>
```

这就是大多数人们写的代码，有着大量的重复代码，而且难以高效的维护，如果让再加10个菜单，他就会ctrl+c/ctrl+v复制10个出来，代码量特别的大

二、数据驱动视图

> （数据+模版=视图）
>
> 我们分析代码发现有很多重复的部分，无疑是改变了几个关键的信息

- 图标
- 标题
- 背景色

- 点击事件
- 其他参数

  我们把上述的参数具体为了一个对象数组，也就是我们说的(数据层)

```
[
    {
        title:'菜单1',
        icon:'icon-caidan1',
        event:'event1,
        style:{
            backgroundColor:'color1'
            color:'color1'
        }
    }
    ...
]
```

我们想想如果用循环的方式遍历这个数组对象，把相关的关键内容填充到相应的位置，换言之就是插入到我们的卡槽中,我们接下来构建我们的卡槽(模版层)

```
<el-row :span="24" v-for="(item,index) in list" :key="index">
    <el-col span="8" @click="goLink(item)">
        <i :class="item.icon" :style=loadStyle(item)></i>
        <p v-text="item.title"></p>
    </el-col>
</el-row>
```

接下来我们写goLink和loadStyle，逻辑写起来比较简单

```
...
event1(item){
    //每一个菜单对应的事件逻辑
},
...
goLink(item){
    this[item.event](item);
},
loadStyle(item){
    const style = item.style;
    return style
}
```

- list就是我们前面构建的数组对象

- goLink我们点击事件的处理方法，并将item当前数据对象传入方法
- loadStyle我们样式加载处理方法

接下来我们将数据和模版组合到一起就呈现我们看到的视图了，从而达到了数据驱动视图的质量高、维护性高的优点

再接着我们上面的老板的需求，让我们加10个20个都不用去写重复代码，只要在数据数组中配置好要加的东西即可

番外

在js里面，经常需要使用js往页面中插入html内容。

比如这样：

```
var number = 123;
$('#d').append('<div class="t">'+number+'</div>');
```

- 如果html很短还好说，但是遇到描述里面的这么大段，直接用字符串存储会很困难，因为不光要处理单引号，还需要很多「+」号把字符串一个个连接起来，十分的不方便。
- 给script设置type="text/template"，标签里面的内容不会被执行，也不会显示在页面上，但是可以在另一个script里面通过获取插入到页面中。这样就把大段的HTML操作从js里面分离开了。

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
    </head>
    <body>
        <div id="user_info"></div>
    </body>
    <script type="text/tmplate" id="tmplate">
        <div>
            <ul>
                <li>姓名：{{name}}</li>
                <li>年龄：{{age}}</li>
                <li>电话：{{phone}}</li>
            </ul>
        </div>
    </script>

    <script type="application/javascript">
        //实例参数
        var user = { name: "陈立明", age: 23, phone: "15932582632"};
        //模板
```

```
        var template = document.getElementById("tmplate").innerHTML;
        //使用mustache.js进行模板解析填充数据
        var view = template.replace('{{name}}',user.name)
        view = view.replace('{{age}}',user.age)
        view = view.replace('{{phone}}',user.phone)
        document.getElementById("user_info").innerHTML = view;
    </script>
</html>
```