# 基本概念

Node.js是基于Chrome的V8 JavaScript引擎构建的，让开发者可以使用JavaScript来编写服务器端应用程序 。它本质上是一个可以在服务器上运行JavaScript代码的环境。

# 主要特点

## 1、单线程模型

Node.js采用单线程模型，这意味着它在一个进程中只使用一个线程来处理所有请求 。这种设计减少了内存消耗，使其能在低硬件条件下支持高并发。

## 2、非阻塞IO

非阻塞I/O是Node.js的核心特性，它允许Node.js在执行I/O操作（如读取文件、网络请求）时不会阻塞主线程 。

一个形象的比喻：

- 阻塞I/O就像餐厅有多个服务员（多线程），每个服务员只负责一桌客人，从点菜到结束都被占用。

- 非阻塞I/O则像餐厅只有一个服务员（单线程），但这个服务员告诉客人「你先看菜单，好了叫我」，然后去服务其他客人，提高了效率。

## 3、事件驱动架构

Node.js使用事件驱动模型，通过事件循环机制处理并发操作 。这种架构使得Node.js能够高效地处理大量并发连接

# nodejs下载地址

https://nodejs.cn/download/

https://registry.npmmirror.com/binary.html?path=node/v12.22.9/

https://nodejs.p2hp.com/download/releases/

# 查看node.js的版本信息

https://nodejs.org/en/about/previous-releases

# nodejs和npm的关系

1、Node.js 是一个基于 Chrome V8 引擎的 JavaScript 运行环境，用于在服务器端运行 JavaScript 代码。它提供了许多内置模块和工具，使开发者能够轻松构建高性能的网络应用程序。

2、NPM（Node Package Manager）是 Node.js 的包管理器，用于管理和发布 JavaScript 包。它是 Node.js 安装时默认包含的一个命令行工具，可以通过命令行界面来安装、升级、卸载和管理 JavaScript 包。

3、总结一下，Node.js 是一个 JavaScript 运行环境，NPM 是 Node.js 的包管理器。Node.js 提供对 NPM 的支持，使开发者能够使用 NPM 来安装和管理第三方 JavaScript 包。同时，NPM 依赖于 Node.js 环境才能正常运行。通过 package.json 文件，可以方便地管理项目的元数据和依赖关系。这种紧密的关系使得 Node.js 和 NPM 成为开发 JavaScript 应用程序的重要工具。

# npm查询包的地址

https://www.npmjs.com/package/package

# npx

## 基本概念

npx全称为"Node Package eXecuter"（Node包执行器），它是npm的一个强大的补充工具，主要用于执行Node.js包中的命令行工具和其他可执行文件。

## npx与npm的区别

### npm

1、包管理器：用于安装、管理和发布 Node.js 包

2、本地安装：将包安装到项目的 node\_modules 目录

3、全局安装：将包安装到系统全局位置

### npx

1、包执行器：直接执行 npm 包中的命令

2、临时下载：如果包不存在，会临时下载并执行

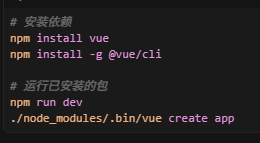
3、避免全局安装：无需全局安装即可使用工具。

### 主要区别

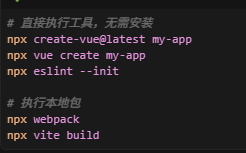


### 使用场景对比

#### npm 使用场景



#### npx使用场景



### 优势

#### npm 优势

1、离线使用：安装后可离线使用

2、版本固定：版本稳定可控

3、性能更好：无需重复下载

#### npx 优势

1、无需全局安装：减少全局包污染

2、版本灵活：可指定特定版本执行

3、节省空间：不占用永久存储空间

4、最新版本：总是使用最新版本

### 实际应用



### 总结

npm 用于包管理，npx 用于包执行。npx 让工具使用更便捷，避免了全局安装的问题。

## 使用场景

### 1、一次性使用工具

当你只需要使用一个工具一次，不想全局安装它

### 2、测试不同版本的包

快速测试一个包的不同版本

### 3、运行本地安装的命令行工具

简化访问项目内部安装的CLI工具

### 4、启动临时服务

如 npx http-server 快速启动一个静态文件服务器

### 5、执行GitHub上的代码

直接运行开源代码而无需克隆仓库

# PNPM

## 基本概念

PNPM（Performant NPM）是一个高效的 JavaScript 包管理器，旨在优化 Node.js 环境中的包安装过程。它是对 NPM（Node Package Manager）和 Yarn 的一种改进，提供了更快速、更高效的依赖管理解决方案。PNPM 的核心优势在于其磁盘空间节省和安装速度，它通过硬链接技术优化包的存储和共享，极大地提高了依赖安装的效率。

## 核心特性

### 1、磁盘空间优化

（1）硬链接机制：pnpm 使用硬链接将包从全局存储链接到项目的 node\_modules

（2）去重存储：相同版本的包只在磁盘上存储一份

（3）节省空间：比 npm/yarn 节省高达 95% 的磁盘空间

### 2、安装速度

1、并行安装：同时下载和安装多个包

2、缓存机制：已下载的包直接从缓存使用

3、增量安装：只安装变更的依赖

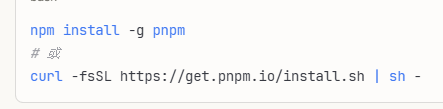
### 3、严格的依赖管理

1、真正的扁平化node\_modules

2、防止幽灵依赖（phantom dependencies）

3、只能访问显式声明的依赖

## 安装PNPM



## 常用命令

### 安装依赖

pnpm install # 等同于 npm install

pnpm i # 简写

### 添加依赖

pnpm add react # 等同于 npm install react

pnpm add -D webpack # 开发依赖

## 对比总结



## 总结

PNPM是现代前端项目的优秀选择，特别适合需要管理多个项目或注重性能的开发者。

# windows环境安装Nodejs

https://zhuanlan.zhihu.com/p/442215189

https://developer.aliyun.com/article/832078（参考这个）

# windows怎么卸载Nodejs

1、从卸载程序卸载程序和功能，也可以直接右键node.js的安装包并选择卸载。

2、重新启动（或者重新启动任务管理器杀死所有与节点相关的进程）

3、从下列的目录中找到相关的内容并删除掉

C:\Program Files (x86)\nodejs

C:\Program Files\nodejs

C:\Users\{User}\AppData\Roaming\npm（或%appdata%\npm）

C:\Users\{User}\AppData\Roaming\npm-cache（或%appdata%\npm-cache）

4、检查%PATH%环境变量以确保没有引用Nodejs或npm存在

5、重新启动电脑

# linux环境搭建

## 1、安装必要的工具

sudo dnf install -y curl

## 2、添加 NodeSource 仓库

curl -fsSL https://rpm.nodesource.com/setup\_18.x | sudo bash -

## 3、安装 Node.js 18

sudo dnf install -y nodejs

## 4、验证安装

node --version

npm --version

# 官方镜像地址

https://registry.npmjs.org

# 淘宝镜像地址

https://registry.npmmirror.com

# npm怎么配置代理

命令行界面配置

## 设置HTTP代理

npm config set proxy http://proxy-server:port

## 设置 HTTPS 代理

npm config set https-proxy https://proxy-server:port

## 带用户名密码的代理

npm config set proxy http://username:password@proxy-server:port

npm config set https-proxy http://username:password@proxy-server:port

## 注意事项

1、代理配置优先级：命令行参数 > 环境变量 > 配置文件

2、公司网络建议咨询网络管理员获取正确的代理地址。

3、配置后记得测试是否生效。

# 如何升级npm的版本

## 1、检查当前版本



## 2、升级到最新稳定版

npm install -g npm@latest

## 3、验证升级结果

npm --version

# nvm-windows

## 基本概念

nvm 全名 **node.js version management**，顾名思义是一个nodejs的[版本管理工具](https://so.csdn.net/so/search?q=%E7%89%88%E6%9C%AC%E7%AE%A1%E7%90%86%E5%B7%A5%E5%85%B7&spm=1001.2101.3001.7020)。通过它可以安装和切换不同版本的nodejs。

## 优点

1、解决不同项目对 Node.js 版本的不同需求。

2、切换版本方便快捷

## 下载地址

https://github.com/coreybutler/nvm-windows/releases

## 安装教程

https://blog.csdn.net/HuangsTing/article/details/113857145

### 1、卸载node

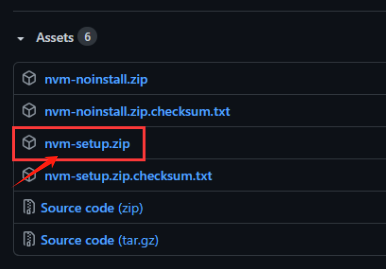
如果你已经安装了node，那么你需要先卸载node（不然安装nvm可能会失败），如果你没有安装那直接跳过这一步到下一步。

打开控制面板 -> 打开程序和功能 -> 右上角搜索输入node -> 右键卸载

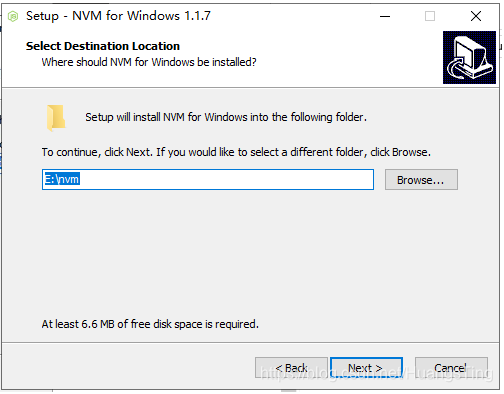
为了确保彻底删除node在看看你的node安装目录中还有没有node文件夹，有的话一起删除。再看看C:\Users\用户名 文件夹下有没有.npmrc以及.yarnrc等等统统删除。再去看看你的环境变量有没有node相关的，有的话也一起删除了。

### 2、安装nvm

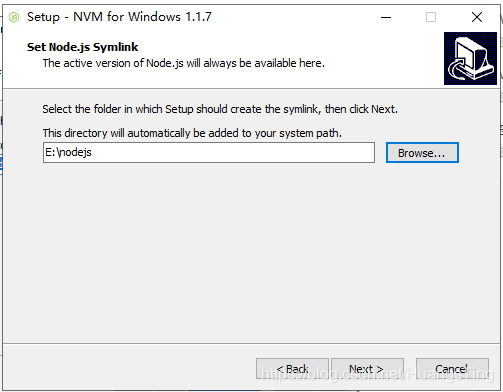
去github [下载最新的 nvm](https://github.com/coreybutler/nvm-windows/releases) 找到 nvm-setup.zip 点击下载。



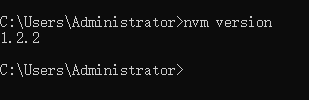
选择[nvm安装](https://so.csdn.net/so/search?q=nvm%E5%AE%89%E8%A3%85&spm=1001.2101.3001.7020)路径，注意路径不要有空格和中文。



选择nodejs的安装路径。

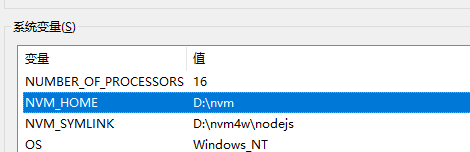


点击next，安装完成后在终端输入nvm version，能查到版本号说明安装成功了。

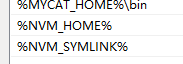


### 3、配置环境变量

安装的时候就自动配置环境变量了，不用手动去配置



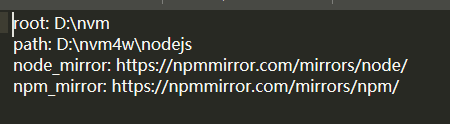
path中的环境变量如下：



### 4、配置路径和下载源（非必须）

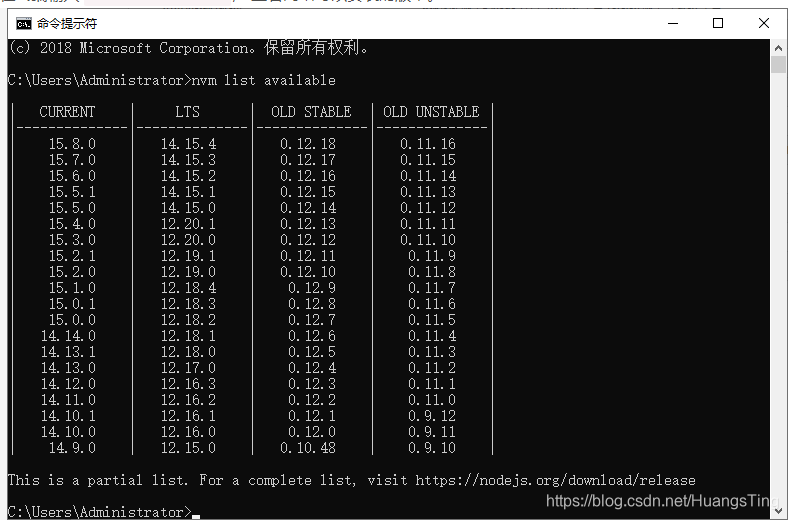
下载node.js时候可能国内限速，此时可以通过修改nvm安装路径下的”settings.txt“文件来提速。在命令行输入nvm root可以找到nvm安装路径 -> 找到 settings.txt 文件 -> 修改下载源

文件内容如下：

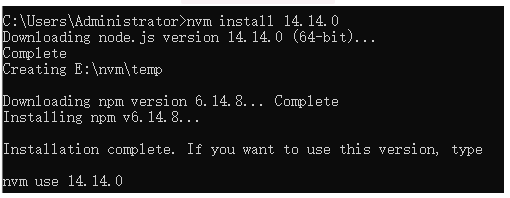


## 使用nvm安装node

在终端输入nvm list available， 查看网络可以安装的版本。

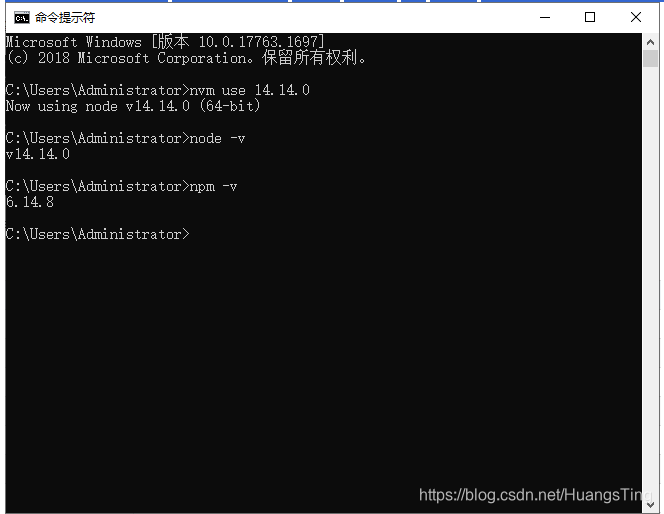


选择一个版本安装，比如 nvm install 14.14.0。

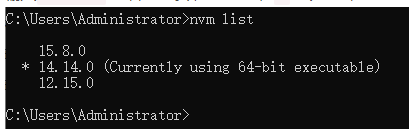


使用这个版本的node，nvm use 14.14.0。

输入node -v 和 npm -v可以查询到node和npm的版本号。



输入nvm list 查看已经安装的node，\*号表示当前使用的node版本。



## nvm常用的命令

### nvm version

查看版本号

### nvm root

找到nvm安装路径

### nvm list available

查看网络可以安装的版本

### nvm install 版本号

安装某个版本的node

### nvm use 版本号

使用某个版本的node

### nvm list

查看已经安装的版本

### nvm on

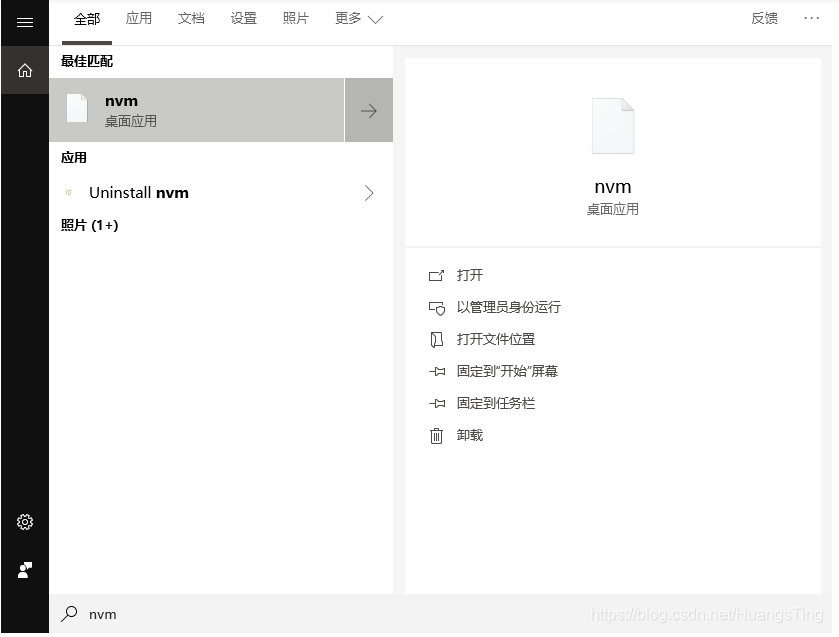
打开nodejs版本控制

### nvm off

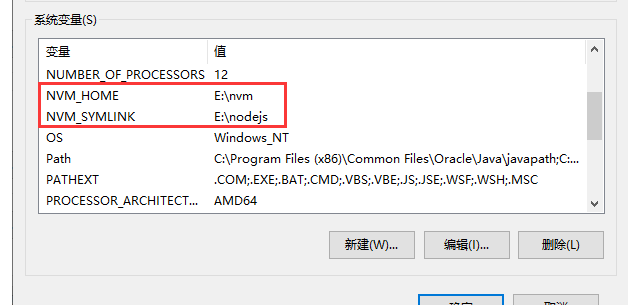
关闭nodejs版本控制

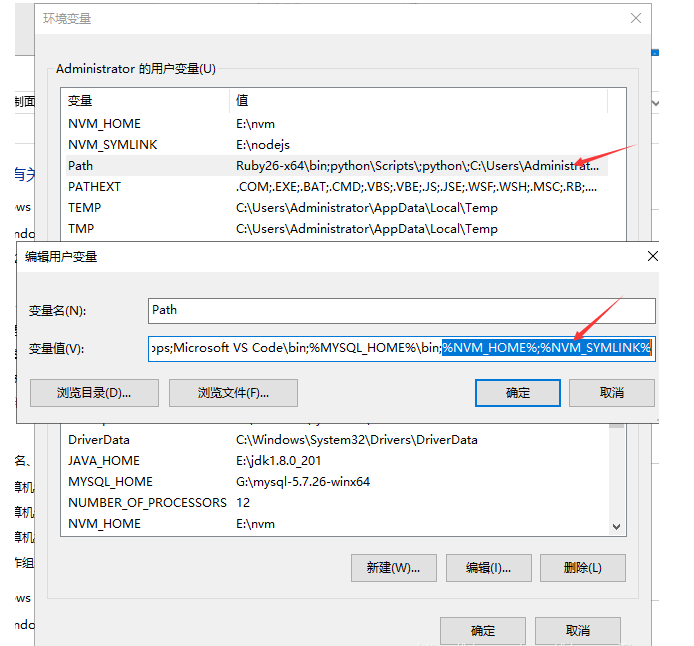
## 如何卸载nvm

1、可以通过下面方式卸载或找到nvm安装路径，直接将文件删掉。



2、卸载完后记得删除nvm环境变量配置（环境变量是在安装时候自动生成的）





## 遇到的问题

1、在安装nvm的时候没有卸载node，导致使用nvm安装完之后，node和npm都不可用。

2.在第一次使用nvm安装node后，要记得使用 nvm use 切换下node版本，以及用 nvm on 打开nodejs版本控制，不然这时候node和npm也都不可用。

3.全局安装完npm的时候，需要关掉终端窗口重新打开，才能装得上cnpm。

4.选择nvm的安装文件夹时，注意文件夹名不要出现中文和空格。

# cnpm

## 基本概念

cnpm是个中国版的npm，是淘宝定制的 cnpm (gzip 压缩支持) 命令行工具，代替默认的 npm : 如果因为网络原因无法使用 npm 下载，那 cnpm 这个就派上用场了。

## 安装

要使用 cnpm，你首先需要将其全局安装到你的计算机上。你可以使用以下命令进行安装。

npm install -g cnpm --registry=https://registry.npm.taobao.org

安装完成后，你就可以使用 cnpm 命令代替平常使用的 npm 命令来进行包管理操作了。

需要注意的是，虽然 cnpm 可以提供更快的下载速度，但它并不是官方推荐的 npm 工具。因此在某些情况下，可能会导致一些兼容性问题或其他不可预见的问题。如果你在使用 cnpm 时遇到了问题，你可以尝试切换回 npm，或者查看官方 npm 文档以解决你遇到的问题。

# yarn

## 基本概念

**Yarn** 是一个快速、可靠且安全的 JavaScript 包管理工具，用于管理项目中的依赖包（即第三方库或工具）。它是 npm（Node.js 的默认包管理器）的替代品，旨在解决 npm 的一些性能和安全问题。

## 安装方法

### 通过npm安装 Yarn

在命令提示符或 PowerShell 中输入以下命令

npm install -g yarn

安装完成后，输入以下命令检查 Yarn 是否安装成功

yarn -v

如果显示版本号，说明安装成功

### 安装指定版本的yarn

使用 npm install -g yarn@<version>（简单且通用）

例如，安装 Yarn 1.22.10

npm install -g yarn@1.22.10

## 常用的命令

### yarn -v

查看版本号

### yarn cache clean

清除缓存

### yarn install

安装依赖

### yarn config set strict-ssl false

禁用严格 SSL 验证

### yarn config get registry

查看镜像源地址

### yarn config set registry https://registry.npmmirror.com

设置镜像源地址

## 问题

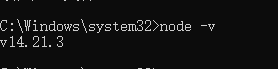
1、

# 常用的命令

## node常用的命令

### node -v

查看Node的版本



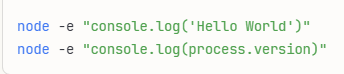
### node [文件名]

执行 JavaScript 文件



### node -e "代码"

直接执行 JavaScript 代码



## 内置快捷命令

### npm start

等价于 npm run start

### npm test

等价于 npm run test

### npm stop

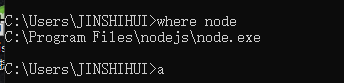
等价于 npm run stop

### npm restart

等价于 npm run restart

## where node

查看node安装文件的位置。



## npm -v

查看npm的版本

## npm init -y

### 基本概念

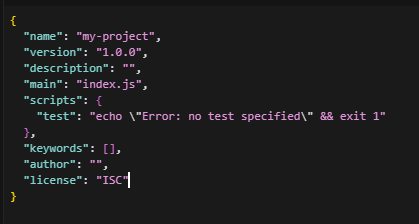
npm init：初始化一个新的 npm 项目，创建 package.json 文件

-y 参数：自动回答所有问题为 "yes"，使用默认值

### 作用

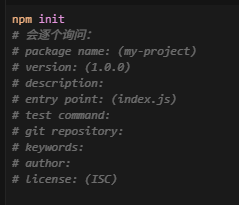
快速创建 package.json 文件，跳过交互式问答过程。

### 生成的 package.json 示例



### 对比

#### 不使用 -y 参数



#### 使用 -y 参数



### 默认值来源

name：当前文件夹名称

version：1.0.0

description：空字符串

main：index.js

scripts：默认 test 脚本

author：从 git 配置或 npm 配置获取

license：ISC

### 自定义默认值



### 使用场景

1、快速创建新项目

2、脚本自动化

3、不需要自定义项目信息时

4、学习和测试环境

### 总结

npm init -y 是快速初始化 npm 项目的便捷命令，适用于不需要自定义配置的场景。

## npm ci

### 基本概念

npm ci 是 npm 5.7.0 版本引入的命令，用于在项目中进行干净、确定性的依赖项安装。

### 干净安装

npm ci 命令会根据项目的 package-lock.json 文件或 npm-shrinkwrap.json 文件来进行依赖项的安装。它会忽略 package.json 文件中的 node\_modules 目录和 package.json 中的版本范围等信息，从而实现干净的依赖项安装。

### 确定性安装

npm ci 命令保证在不同的环境中安装相同的依赖项和版本，确保了项目的可重复性。它会严格按照 package-lock.json 或 npm-shrinkwrap.json 中记录的依赖版本来进行安装，而不会考虑 package.json 中的版本范围。

### 快速安装

由于 npm ci 命令会直接使用 package-lock.json 或 npm-shrinkwrap.json 中记录的精确版本号和依赖关系，因此它可以更快速地安装依赖项。相比于 npm install 命令，npm ci 不需要进行依赖解析和冗余计算，从而加快了安装过程。

### 适用场景

npm ci 命令特别适用于 CI/CD（持续集成/持续部署）环境和生产环境的依赖项安装。它可以确保在不同的环境中始终安装相同的依赖版本，减少由于不确定性而引起的问题，同时也提高了构建和部署的速度。

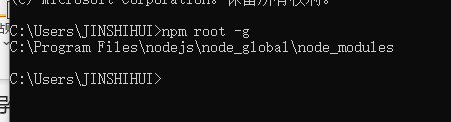
需要注意的是，npm ci 命令要求项目必须存在 package-lock.json 或 npm-shrinkwrap.json 文件，而且这两个文件必须与 package.json 保持一致。如果这些文件不存在或者与 package.json 不一致，npm ci 命令将无法执行。

### 总结

npm ci 命令是一种干净、确定性、快速的依赖项安装方式，可用于确保项目在不同环境中安装相同的依赖版本，并且适用于 CI/CD 环境和生产环境。它通过使用 package-lock.json 或 npm-shrinkwrap.json 文件来实现这些功能。

## npm root -g

查看全局目标路径



## npm list

查看当前目录下已安装的node包

## set NODE OPTIONS=--openssl-legacy-provider

在 Node.js 中，NODE\_OPTIONS 是一个环境变量，它可以用来指定一组选项，以影响 Node.js 运行时的行为。该环境变量允许您在启动 Node.js 进程时设置一些运行时选项，以满足特定需求或调整 Node.js 的行为。

通过设置 NODE\_OPTIONS 环境变量为 --openssl-legacy-provider，您告诉 Node.js 在启动时使用旧版的 OpenSSL 提供程序，以确保与您的应用程序或依赖库的兼容性。

请注意，set NODE\_OPTIONS=--openssl-legacy-provider 命令是针对 Windows 操作系统的设置环境变量的语法。在其他操作系统（如 Linux 或 macOS）上，可能需要使用不同的命令或语法来设置环境变量。

## npm config get registry

查看当前 npm 使用的 registry（镜像源）

## npm config set registry 镜像源地址

设置当前npm的镜像源

npm config set registry https://registry.npmjs.org/

## npm config get prefix

### 基本概念

查看npm的**全局包安装路径前缀**，即全局安装的包存放在哪个目录。

### 作用

1、显示全局包的根目录位置

2、用于排查全局包找不到的问题

3、确认环境变量PATH配置

## npm i

npm i 是 npm install 的简写命令，用于安装项目所需的依赖包。

常见用法如下：

直接输入 npm i，会根据 package.json 文件安装所有依赖。

npm i <包名> 可以安装指定的包。

## npm install

### 基本概念

用于在当前目录下安装和管理 Node.js 模块。具体来说，它可以从 npm 的默认软件源或其他指定的源中下载、安装、更新和卸载各种 Node.js 模块。当你执行 npm install 命令时，npm 会首先检查当前目录下的 package.json 文件，根据其中的依赖描述来安装相应的 Node.js 模块。如果没有找到 package.json 文件，npm 会将所有安装的模块直接放在 node\_modules 目录下。通常情况下，package.json 文件会包含应用程序所需的所有依赖信息，以及开发者运行应用程序所需的命令等信息。如果你想安装一个新的 Node.js 模块，可以在终端中执行 npm install <module-name> 命令。

例如，如果你想安装 Express 框架，可以执行 npm install express 命令。如果你不指定版本号，则默认安装最新版本。

### 常用选项

#### -g

全局安装模块，这样可以在任何目录下使用。

#### --save 或 -S

将模块添加到 package.json 文件的依赖列表中。

#### --save-dev 或 -D

将模块添加到 package.json 文件的开发依赖列表中。

#### --registry=<registry-url>

指定 npm 的下载源。

除此之外，还可以通过 npm install 命令来更新和卸载 Node.js 模块。例如，如果你想升级 Express 框架到最新版本，可以执行 npm update express 命令，如果你想卸载某个模块，可以执行 npm uninstall <module-name> 命令。

### 总结

npm install 是 Node.js 开发中必备的命令之一，可以简化 Node.js 模块的安装和管理，提高开发效率。

## npm start

启动应用

npm start 是npm的**内置快捷命令**，而不是自定义脚本，所以不需要加run。

## npm run dev

### 基本概念

npm run dev 是一个常见的命令，通常在开发环境中使用。它用于运行项目中的开发脚本或者启动开发服务器。

1、npm：是 Node.js 的包管理器，用于执行 npm 命令。

2、run：是 npm 的一个子命令，用于运行项目中的自定义脚本。

3、dev：是自定义的脚本名称，可以根据项目的需求进行命名。

一般来说，npm run dev 会执行项目中定义的 "dev" 脚本。这个脚本可以在项目的 package.json 文件中的 "scripts" 部分进行配置。例如：



在这个例子中，当你运行 npm run dev 命令时，它实际上会执行 node server.js 命令。这通常用于启动一个开发服务器，用于开发过程中的实时预览、热更新和调试等。

在实际项目中，npm run dev 的功能可能会有所不同，具体取决于项目的配置和需求。你可以根据你的项目需要，在 package.json 文件中的 "scripts" 部分进行相应的配置和定义。

## npm run build

### 基本概念

npm run build 是一个在 Node.js 项目中常用的命令。它用于构建（build）项目，通常是将源代码转换、打包、优化等操作，生成可部署的生产环境代码。

## npm run build –report

## npm cache clean -f

npm cache clean -f 是 npm（Node Package Manager）提供的一个命令，用于清除 npm 的缓存数据。这个命令的含义和作用如下：

npm: npm 是 Node.js 包管理器，用于下载、安装和管理 Node.js 模块。

cache: 指的是 npm 缓存，npm 会在本地缓存下载的模块和其他资源，以便在后续安装时可以快速访问。

clean: 表示清理缓存。

-f: 是 --force 的简写，表示强制执行操作，不进行任何确认或交互。

当你运行 npm cache clean -f 命令时，npm 会强制清空本地的 npm 缓存，包括已下载的包、缓存的数据等。这样做可以帮助解决一些问题，比如缓存中的某些数据损坏或过期导致安装或下载出现问题。

需要注意的是，清除 npm 缓存可能会导致一些依赖重新下载，因此在执行该命令之前，请确保你了解清除缓存可能带来的影响，并确保有必要的备份或准备工作。

## npm install -g yarn

npm install -g yarn 这个命令是用来在全局（global）安装 Yarn 包管理工具的。下面我会详细解释这条命令的各个部分：

**1、npm**：npm 是 Node.js 包管理器（Node Package Manager）的缩写，它是 Node.js 平台的默认包管理工具，用于安装、升级和管理 JavaScript 包和依赖项。

**2、install**：install 是 npm 的一个命令，用于安装指定的包或依赖项。

**3、-g**：-g 是 npm 命令的一个选项，表示在全局范围内安装包。当使用 -g 选项时，安装的包将在系统的全局路径中可用，可以在任何位置直接调用。

**4、yarn**：yarn 是另一个流行的 JavaScript 包管理工具，与 npm 类似，但有一些不同之处。Yarn 提供了更快的下载速度、更一致的依赖项解析以及其他一些改进。

因此，npm install -g yarn 这个命令的含义是在全局范围内使用 npm 安装 Yarn 包管理工具。这样一来，您就可以在命令行中直接使用 yarn 命令来管理 JavaScript 项目的依赖项和构建过程。

## npm i node-sass -S

### 作用

安装 **node-sass** 包到项目中，用于将Sass/SCSS文件编译为CSS。

### 说明

1、node-sass 已被弃用

2、-S 参数在npm 5+版本中是默认行为，无需添加

### 替代方案

npm install sass

## npm info express

### 基本概念

查看npm包 **express** 的详细信息，包括版本、依赖、维护者等元数据

### 作用

1、查看包的最新版本

2、了解包的基本信息

3、检查包的维护状态

4、查看历史版本列表

## npm fund

### 基本概念

npm fund 是一个显示项目依赖包资助信息的命令。

### 用法

npm fund

### 主要功能

1、显示当前项目依赖包中哪些接受资助/捐赠

2、展示资助链接和联系方式

3、帮助开发者了解如何支持开源项目

### 输出内容

1、包名称

2、资助平台（GitHub Sponsors、Open Collective等）

3、资助链接

4、依赖层级关系

### 常用选项

npm fund --json - JSON格式输出

npm fund <package-name> - 查看特定包的资助信息

### 总结

这个命令主要用于了解项目依赖的开源维护者的资助渠道，方便开发者支持他们使用的开源项目。

# ESLint

## 基本概念

ESLint 是一个用于 JavaScript 代码静态分析的开源工具。它能够帮助开发者在编写 JavaScript 代码时发现和修复常见的问题，以确保代码质量和一致性。ESLint 可以通过定义和应用规则来检测代码中的潜在问题，并提供了丰富的插件和配置选项，使得开发团队可以根据自己的需求来定制规则。

## 核心功能

### 1、代码质量检查

（1）检测未使用的变量

（2）发现语法错误

（3）识别潜在的bug

（4）检查代码逻辑问题

### 2、代码风格统一

（1）强制缩进规则

（2）统一引号使用

（3）规范命名约定

（4）控制代码格式

### 3、最佳实践建议

（1）避免全局变量

（2）推荐现代语法

（3）提醒性能优化

（4）安全编码建议

## 安装ESLint

### npm install eslint --save-dev

项目本地安装（推荐）

### npm install -g eslint

全局安装

### yarn add eslint --dev

使用yarn安装

## 初始化配置

### npx eslint --init

交互式配置

### npm init @eslint/config

这两种方法都可以

交互式配置会询问：

1、如何使用ESLint？（检查语法/发现问题/强制代码风格）

2、使用什么模块类型？（ES6 modules/CommonJS/None）

3、使用什么框架？（React/Vue.js/None）

4、是否使用TypeScript？

5、代码运行在哪里？（Browser/Node）

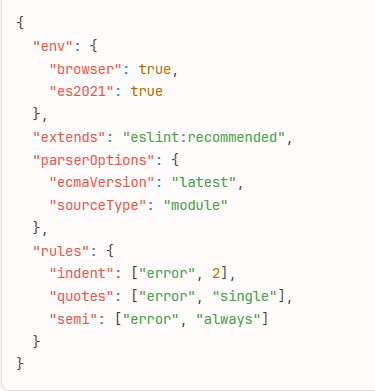
6、配置文件格式？（JavaScript/YAML/JSON）

## 配置文件示例

### .eslintrc.js 文件

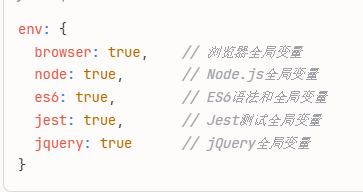


### .eslintrc.json 文件

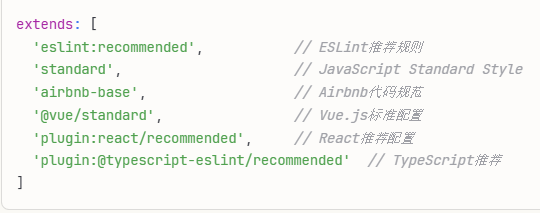


## 配置详解

### 1、env（环境）



### 2、extends（扩展配置）



### 3、parser（解析器）

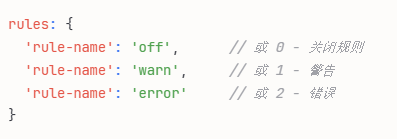


### 4、plugins（插件）



## 规则配置

### 规则严重级别



### 常用规则示例



## 使用方法

### 1、命令行使用



### 2、package.json脚本



### 3、编辑器集成

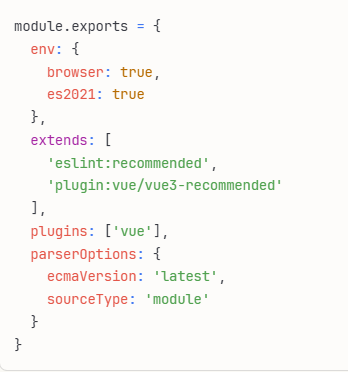
VS Code

安装ESLint扩展，在settings.json中配置：



## 常用配置预设

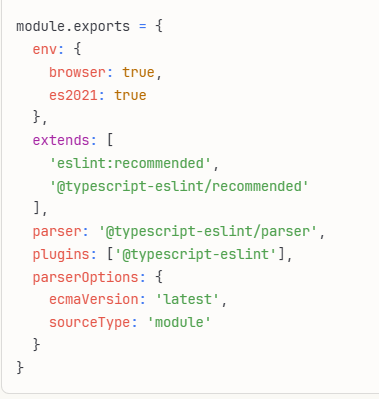
### Vue.js项目配置



### React项目配置



### TypeScript项目配置



## 忽略文件

### .eslintignore文件



### 文件内忽略



## 自定义规则

### 创建自定义规则



## 最佳实践

### 1、团队协作

1、统一配置文件

2、设置pre-commit钩子

3、在CI/CD中集成检查

4、定期更新规则

### 2、性能优化



### 3、渐进式迁移



## 总结

ESLint是现代JavaScript开发的必备工具，正确配置和使用能显著提高代码质量和团队协作效率。建议从基础配置开始，逐步根据项目需求调整规则。

# 模块

## axios

## babel-polyfill

## node-sass

## yauzl

yauzl 是一个 Node.js 模块，用于在 Node.js 环境中操作 ZIP 文件。它提供了解压缩和压缩 ZIP 文件的功能，使开发者能够轻松地在他们的 Node.js 应用程序中处理 ZIP 文件。

以下是 yauzl 模块的一些主要功能和用途：

**解压缩ZIP文件**：yauzl 可以帮助你从 ZIP 文件中提取出文件和文件夹，并将其解压到指定的目录中。这对于需要动态解压缩 ZIP 文件的应用程序非常有用。

**压缩文件为ZIP格式**：除了解压缩，yauzl 还可以帮助你将文件和文件夹压缩成一个 ZIP 文件。这在需要将多个文件打包成一个 ZIP 文件进行传输或存储时非常方便。

**处理ZIP文件中的内容**：通过 yauzl，你可以访问 ZIP 文件中的每个文件的内容，包括读取文件的数据、元数据等信息。这使得你可以在应用程序中按需处理 ZIP 文件中的内容。

**异步操作**：yauzl 支持异步操作，这意味着你可以在 Node.js 应用程序中方便地处理大型的 ZIP 文件而不会阻塞主线程。

总的来说，yauzl 模块为 Node.js 提供了处理 ZIP 文件的能力，使开发人员能够更轻松地在他们的应用程序中实现 ZIP 文件的解压缩和压缩功能。

# 问题

## 1、'vue-cli-service' 不是内部或外部命令，也不是可运行的程序或批处理文件。

### 解决方法

这个错误通常是因为Vue CLI Service没有正确安装或环境变量配置问题。

1、删除现有依赖



2、重新安装依赖



3、验证安装



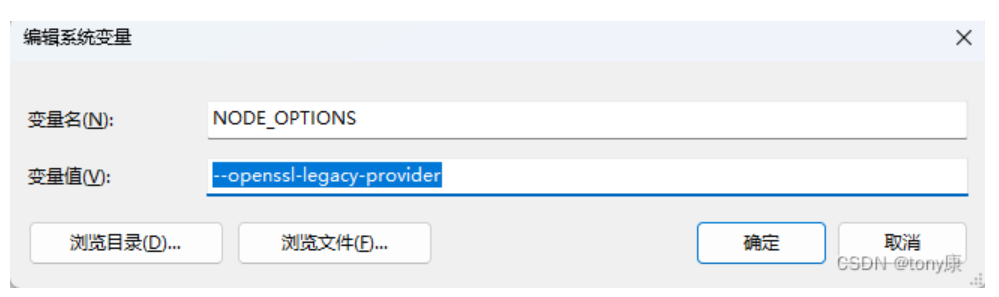
## 2、解决报错10% building 2/5 modules 3 active ...loader\lib\index.js!

### 错误原因

[node版本](https://so.csdn.net/so/search?q=node%E7%89%88%E6%9C%AC&spm=1001.2101.3001.7020)高于16，项目的版本不高于16，node17之后更新了[OpenSSL](https://so.csdn.net/so/search?q=OpenSSL&spm=1001.2101.3001.7020)。

### 解决方法

1、**需要设置环境变量来解决**



2、**降低node版本**

使用[nvm](https://so.csdn.net/so/search?q=nvm&spm=1001.2101.3001.7020)将node修改为不高于16的版本。

## 3、'webpack-dev-server' 不是内部或外部命令，也不是可运行的程序 或批处理文件

### 解决方法

#### 1、检查当前项目状态



#### 2、清理现有依赖



#### 3、安装webpack-dev-server



#### 4、重新安装所有依赖



#### 5、验证安装



#### 6、配置package.json脚本

在package.json中添加或修改scripts



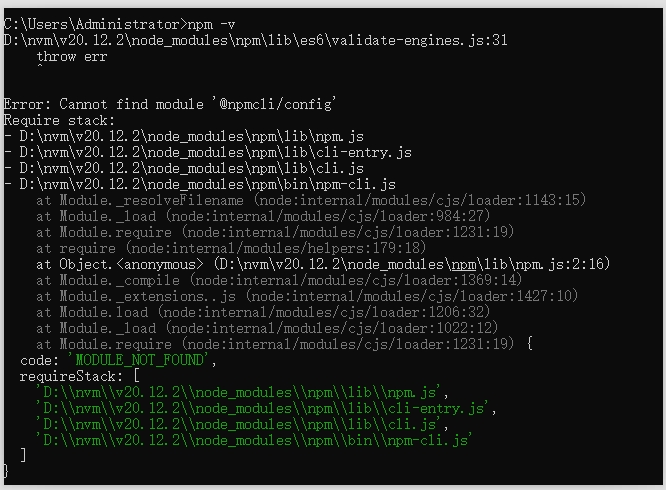
#### 7、测试运行



## 4、windows报错node: --openssl-legacy-provider is not allowed in NODE\_OPTIONS解决

https://blog.csdn.net/zhu\_liu\_kun/article/details/130840151

## 5、Error：Cannot find module @npmcli/config



这个错误是在输入npm -v命令的时候报的

### 错误分析

1、npm 无法找到 @npmcli/config 模块，这是 npm 正常运行所必需的

2、错误堆栈显示问题出在 npm 的内部模块加载过程中

3、这通常表明 npm 安装不完整或已损坏

### 解决方法

1、重新安装 Node.js 和 npm