第 1 章 Node.js 和 TypeScript 基础

1

本章将带领读者了解 Node.js 及 TypeScript 的基础知识。

首先将全面介绍 Node.js 开发概述，包括为什么要学习 Node.js、Node.js 的定义及其特点，并深入探讨 JavaScript 中 var、let 和 const 的区别，还会介绍必备的开发工具，为读者提供一个良好的开端。然后，将介绍如何搭建 Node.js 运行环境，包括 Node.js 运行环境的安装方法、可能出现的安装失败及其解决办法，以及代码中有无分号的问题。接下来，将带领读者快速入门 Node.js，这部分内容包括 Node.js 的组成、基础语法以及全局对象 global。最后，将介绍 nvm 的简介、安装和使用，以及 Visual Studio Code 的使用方法，包括忽略 node_module 目录、安装 Visual Studio Code 插件、打开并运行项目，以及配置 Visual Studio Code 和进行搜索。

通过对本章内容的学习，读者将对 Node.js 和 TypeScript 有更深入的了解，并能够熟练应用它们于实际项目开发当中。

本章学习目标

- 能够知道 Node.js 是什么
- 能够安装 Node.js 运行环境
- 能够知道系统环境变量 PATH 的作用
- 能够使用 Node.js 环境执行代码
- 能够切换 Node.js 指定版本
- 熟悉 Node.js 基础语法
- 熟悉 Visual Studio Code 的使用

1.1 Node.js 开发概述

Node.js（简称 Node）的推出，不仅能帮助开发者自动化处理更多琐碎费时的工作，更打破了前端和后端的语言边界，让 JavaScript（简称 JS）流畅地运行在服务器端。

Node.js 作为 JavaScript 的运行环境，大大地提高了前端开发的效率，增加了 Web 应用的丰富性。对 JavaScript 开发者来讲，有了 Node.js 几乎可以无障碍地深入实践服务端开发，并且无须学习新的

编程语言。Node.js 的生态圈也是目前最为活跃的技术领域之一，大量的开源工具和模块可以让我们开发出高性能的服务端应用。

1.1.1　为什么要学习 Node.js

Node.js 从 2009 年出现至今，一直风靡全球，微软也已经将它集成进 Visual Studio 了。我们知道微软总是喜欢将一些它觉得比较好的东西集成到自己的产品中，这也侧面说明了 Node.js 的优秀。

（1）学习 Node.js 的好处：

- 能够和后端程序员配合得更加紧密。
- 可以将网站业务逻辑前置，学习前端技术需要后端技术（ajax）的支撑。
- 扩宽知识视野，能够站在更高的角度审视整个项目。
- 可以独立完成一些 Web 应用的开发。

（2）服务器端（后端）开发要做的事情：

- 实现网站的业务逻辑。
- 数据的增删改查（CRUD）。

（3）为什么选择 Node.js?

- 可以使用 JavaScript 语法开发后端应用。
- 一些公司要求前端工程师掌握 Node.js 开发。
- 生态系统活跃，有大量开源库可以使用。
- 前端开发工具大多基于 Node.js 开发。

Node.js 可能不是那些从未接触过服务端开发的初学者的最佳选择，因为要想真正精通服务端编程，传统的技术栈如 Java、PHP 和.NET 更具优势。Node.js 不适宜初学者的原因主要是它比较偏底层并且极其灵活；而 Java、PHP 和.NET 更适合初学者的原因在于它们抽象化了更多的底层细节。虽然 Node.js 并不是最适合服务端开发的入门技术，但这并不意味着它不强大。实际上，Node.js 在有经验的开发者手中可以发挥出极高的性能。对于前端开发者而言，Node.js 是成为高级前端开发工程师的一个重要技能。

1.1.2　什么是 Node.js

Node.js 既不是一种编程语言，也不是一个框架，而是一个运行时环境。它允许在服务器端运行 JavaScript 代码。在 Node.js 中，我们不会找到浏览器特定的对象，如 BOM（浏览器对象模型）或 DOM（文档对象模型）。它主要支持 ECMAScript（简称 ES）标准中规定的核心 JavaScript 语言功能。不过，Node.js 扩展了 JavaScript 的能力，提供了一系列服务器级别的 API，使得 JavaScript 能够执行文件系统操作、创建 HTTP 服务等服务器端特定任务。相对于浏览器中的 JavaScript，这些是新增的能力，因为在浏览器中，出于安全考虑，JavaScript 通常没有访问文件系统的权限。

1）Node.js 简介

我们可以通过以下描述来建立对 Node.js 的大致印象：

- Node.js 让 JavaScript 具备了与 Java 和.Net 一样开发 Web 应用的能力。
- Node.js 是一个由 C++编写的基于 Chrome V8 引擎的 JavaScript 运行环境。
- Node.js 的运行速度非常快，性能非常好，它对一些特殊用例进行了优化，提供了替代的 API，使得 Chrome V8 在非浏览器环境下运行得更好。
- Node.js 使用了一个事件驱动、非阻塞式 I/O 的模型，从而轻量又高效。
- Node.js 的包管理器 npm，是全球最大的开源库生态系统。

2）ECMAScript 包含的内容

Node.js 主要支持 ECMAScript 标准中规定的核心 JavaScript 语言功能，而 ECMAScript 包含的内容如下：

- 变量
- 方法
- 数据类型
- 内置对象
- Array
- Object
- Date
- Math

3）运行环境

- 浏览器（软件）能够运行 JavaScript 代码，浏览器就是 JavaScript 代码的运行环境，浏览器是不认识 Node.js 代码的。
- Node.js（软件）能够运行 JavaScript 代码，Node.js 就是 JavaScript 代码的运行环境。

在写作本书时，Node.js 的最新版本为 Node.js 18.17.1。

官方网站为：https://nodejs.org。
中文网站为：http://nodejs.cn。

1.1.3　Node.js 的特点

Node.js 具有以下特点：

（1）基于事件驱动和非阻塞 I/O 模型：Node.js 采用事件循环机制，通过异步非阻塞 I/O 操作来处理多个并发请求，使得服务器能够高效地处理大量的并发连接。

（2）运行在 V8 JavaScript 引擎上：Node.js 使用 Google 开发的 V8 引擎，将 JavaScript 代码直接编译为机器码，提高了执行速度和性能。

（3）轻量级和高效：Node.js 的核心设计理念是轻量级和高效。它具有精简的内核，只包含了必要的功能，减少了资源占用和启动时间，提升了应用程序的响应速度。

（4）单线程，但支持多线程：尽管 Node.js 是单线程的，但它通过事件循环和异步机制实现了高并发。此外，Node.js 还提供了 Cluster 模块和 Worker 线程池等工具，可以利用多核 CPU 实现多

线程处理，进一步提高并发能力。

（5）适合构建实时应用：由于 Node.js 的高性能和事件驱动的特性，使得它非常适合构建实时应用，如聊天应用、游戏服务器、推送服务等。

（6）模块化和生态系统丰富：Node.js 支持模块化开发，通过 NPM（Node Package Manager）提供了丰富的第三方模块，能够快速构建和集成各种功能。

（7）跨平台：Node.js 可以在多个操作系统上运行，包括 Windows、macOS、Linux 等，提供了良好的跨平台支持。

总体而言，Node.js 以其高性能、可扩展性、灵活性和丰富的生态系统，成为流行的服务器端开发平台，广泛应用于 Web 开发、后端服务、命令行工具等领域。

Node.js 采用 JavaScript 与非阻塞 Socket 结合的方式，其独特之处在于对 I/O 操作的处理。与其他语言不同，Node.js 始终不会阻止程序执行，而是要求持续处理新任务。这使得它非常适合网络编程，特别是在服务器端需要与多个客户端通信以及处理网络连接的情况。Node.js 鼓励使用非阻塞模式，这一特性使得其在进行服务器端开发时，相较传统编程语言更为便捷。

1.1.4　var、let 和 const 的区别

在 JavaScript 和 Node.js 中，var、let 和 const 是用于声明变量的关键字，它们之间有一些区别。

（1）var 是在 ES 5 中引入的关键字，用于声明变量。它有以下特点：

- 函数作用域：var 声明的变量的作用域是整个函数内部。
- 变量提升：使用 var 声明的变量会被提升到函数的顶部，即使声明在后面也可以在之前使用。
- 可重复声明：可以重复使用 var 声明同一个变量，而不会产生错误。
- 没有块级作用域：var 声明的变量不存在块级作用域，在{}内部声明的变量会泄露到外部。

（2）let 是在 ES 6 中引入的关键字，用于声明块级作用域的变量。它具有以下特点：

- 块级作用域：let 声明的变量的作用域是包含它的最近的块（{}）内部。
- 不会提升变量：使用 let 声明的变量不会被提升，必须在声明后才能使用。
- 不可重复声明：在同一个作用域内，不能通过 let 对同一个变量进行重复声明。
- 可以修改值：let 声明的变量可以修改其值。

（3）const 也是在 ES 6 中引入的关键字，用于声明常量。它具有以下特点：

- 块级作用域：const 声明的变量的作用域是包含它的最近的块（{}）内部。
- 不会提升变量：使用 const 声明的变量不会被提升，必须在声明后才能使用。
- 不可重复声明：在同一个作用域内，不能通过 const 对同一个变量进行重复声明。
- 必须赋初始值：const 声明的变量必须在声明时赋初始值，并且之后不能再修改其值。

总的来说，var 是函数作用域，没有块级作用域和常量概念；let 是块级作用域的变量声明方式，可以修改值；const 同样是块级作用域的变量声明方式，但其值一旦被赋值后就不能再修改，因此被称为常量。

在实际开发中，推荐使用 let 和 const 来代替 var，以获得更好的作用域管理和变量约束。

1.1.5　开发工具

俗话说“工欲善其事必先利其器”，在进行编码开发之前，我们有必要先选择并安装好相应的开发工具。当前使用得最多的前端 IDE 当属 Visual Studio Code（简称 VS Code）和 WebStorm，而 WebStorm 是收费的，所以更推荐读者使用 Visual Studio Code，本书所有代码也均由 Visual Studio Code 编写。Visual Studio Code 相较于 WebStorm 来说更加轻量级，而丰富的插件库使得 Visual Studio Code 具备强大的扩展和自定义功能。Visual Studio Code 官网地址：https://code.visualstudio.com/。

Visual Studio Code 具有以下特点：

- 开源，免费。
- 自定义配置。
- 集成 git。
- 智能提示强大。
- 支持各种文件格式，如.html、.jade、.css、.less、.sass、.xml、.vue 等。
- 调试功能强大。
- 各种方便的快捷键。
- 强大的插件扩展。

1.2　Node.js 运行环境搭建

本节介绍如何搭建 Node.js 运行环境。

1.2.1　Node.js 运行环境安装

进入 Node.js 官网 https://nodejs.org，下载版本为 18.17.1 LTS 的 SDK，由于笔者的计算机是 Windows 11 64 位的系统，因此会看到如图 1-1 所示界面。

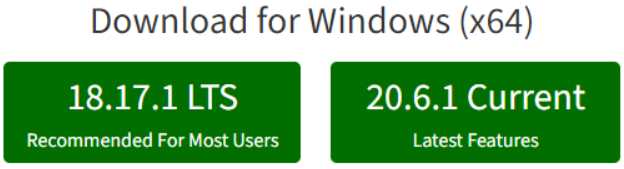

图 1-1

界面中有两个版本，分别是：

- LTS：Long Term Support，长期支持版、稳定版。
- Current：拥有最新特性的实验版。

在这里，选择左侧的稳定版（18.17.1 LTS）进行下载。如果要下载其他 SDK，可以进入下载页

面 https://nodejs.org/en/download/，选择需要的安装包进行下载。

下载完成之后，直接双击安装包进行安装，安装过程中不断单击“下一步”按钮便可顺利完成安装。

默认的安装路径是 D:\Program Files\nodejs。

按 Ctrl+R 快捷键，输入 CMD 命令，打开控制台窗口，进入 Node.js 所在的安装目录，然后输入 node -v 查看当前的 Node.js 版本信息，操作如下所示。

```
C:\Users\zouqj>d:
D:\>cd D:\Program Files\nodejs
D:\Program Files\nodejs>node -v
v18.17.1
D:\Program Files\nodejs>
```

如果能看到 Node.js 的版本号，则说明安装成功。

如果我们直接在控制台窗口中输入 node -v 时报错，错误信息如图 1-2 所示，这是因为 Node 安装目录写入环境变量失败。解决办法是将 Node 安装目录添加到环境变量中。

图 1-2

那么应该如何配置系统环境变量呢？下面以 Windows 10 为例介绍配置步骤。

步骤 01 在计算机桌面上选中“此电脑”并右击，在弹出的快捷菜单中选择“属性”命令。

步骤 02 在弹出的页面中单击“高级系统设置”，将弹出“系统属性”对话框。

步骤 03 在“系统属性”对话框中单击“环境变量”按钮，将弹出“系统变量”对话框，如图 1-3 所示。

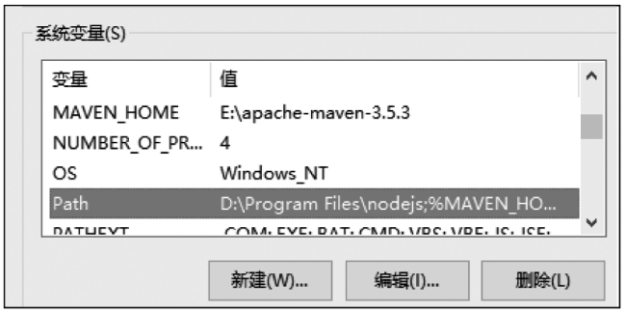

图 1-3

步骤 04 在“系统变量”对话框中单击“编辑”按钮，将弹出“编辑系统变量”对话框，在“变量值”文本框的最后添加分号，再加上 Node.js 的安装路径，例如;D:\Program Files\nodejs\，如图 1-4 所示。

步骤 05 最后单击“确定”按钮，即可完成环境变量的配置。

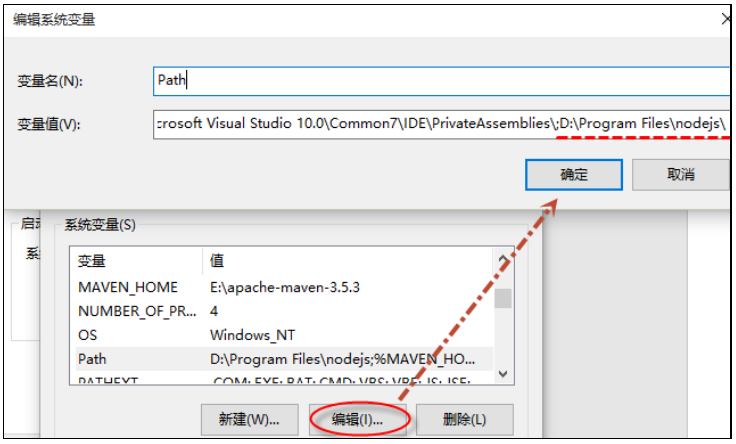

图 1-4

1.2.2　Node.js 环境安装失败的解决办法

在安装 Node.js 的时候如果报错，我们就要根据错误提示码来分析产生错误的原因。

常见错误代码有 2502、2503，错误提示如图 1-5 所示。

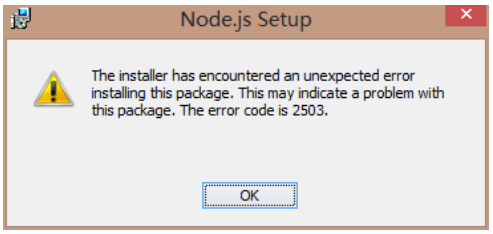

图 1-5

失败原因是系统账户权限不足。解决办法如下：

- 首先以管理员身份运行 powershell 命令行工具。
- 然后输入运行安装包的命令：msiexec/package node 安装包位置。

安装过程中遇到错误时，不要慌，我们可以去网上搜索解决方案，我们要学会自己组织搜索关键字来查询相关解决方法，例如在百度搜索框中输入“Node.js 安装 2502”。

1.2.3　代码有无分号的问题

代码结尾是否写分号这是编码习惯的问题，有些人的编码风格就是代码后面不写分号，有些人又喜欢写分号，在同一个项目中统一规则即可。

当采用无分号的代码风格的时候，只需要注意以下情况就不会有什么问题：

当一行代码是以“(”“[”“`”开头的时候，在前面补上一个分号用以避免一些语法解析错误。因此，我们会发现在一些第三方的代码中，一些代码行以一个“;”开头。

建议：无论代码是否有分号，如果一行代码是以“(”“[”“`”开头的，则最好都在其前面补上一个分号。其实现在很多 IDE 都可以配置自动给代码结尾加分号，但为了减少出错的可能，笔者

个人建议还是自己加上分号为妙。

1.3　Node.js 快速入门

本节将介绍如何快速入门 Node.js。

1.3.1　Node.js 的组成

JavaScript 由 3 部分组成：ECMAScript、DOM、BOM。

Node.js 由 ECMAScript 及 Node 环境提供的一些附加 API 组成，包括文件、网络、路径等一些更加强大的 API，如图 1-6 所示。

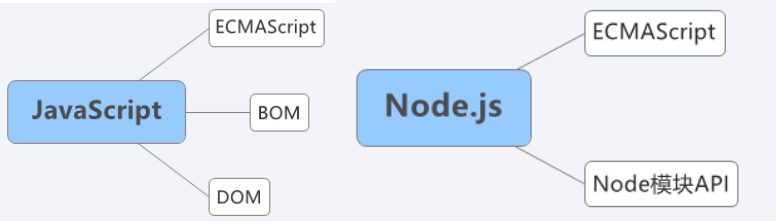

图 1-6

1.3.2　Node.js 基础语法

所有 ECMAScript 语法在 Node 环境中都可以使用。在 Node 环境下，使用 node 命令执行后缀为.js 的文件即可运行 ECMAScript 代码。

示例：新建一个 hello-china.js 文件。首先，输入如下代码：

```
var msg = '我和我的祖国';
console.log(msg);
```

然后在当前文件所在目录输入 node hello-china.js，运行结果如下：

```
PS D:\WorkSpace\node_mongodb_vue3_book\codes\chapter1\js> node hello-china.js
我和我的祖国
```

1.3.3　Node.js 全局对象 global

在浏览器中全局对象是 window，在 Node 中全局对象是 global。

Node 中全局对象下有以下方法：

- console.log()：在控制台中输出。
- setTimeout()：设置超时定时器。
- clearTimeout()：清除超时定时器。
- setInterval()：设置间歇定时器。

- clearInterval(): 清除间歇定时器。

可以在任何地方使用这些方法，使用时可以省略 global。

示例：global 的使用。

global.js 代码如下：

```
global.console.log('你让我独自斟满这碗红尘的酒');
global.setTimeout(function () {
  console.log('借来晚风下口 敢与寂寞交手');
}, 1000);
```

在 CMD 控制台或者 Visual Studio Code 终端都可以执行上述代码。以 Visual Studio Code 终端为例，如图 1-7 所示新建一个终端，然后运行代码。

图 1-7

运行结果如下：

```
PS D:\WorkSpace\node mongodb vue3 book\codes\chapter1\js> node global.js
你让我独自斟满这碗红尘的酒
借来晚风下口 敢与寂寞交手
```

1.4　nvm 的安装与使用

nvm（Node Version Manager）是一个用于管理多个 Node.js 版本的工具，可以让开发人员在同一台机器上同时安装不同版本的 Node.js。通过 nvm，我们可以轻松地在项目之间切换 Node.js 版本，以确保每个项目都使用适合其需求的特定版本。当前 nvm 的最新版本为 v1.1.11。

1. 安装 nvm

nvm 的安装步骤如下：

步骤 01　在 https://github.com/coreybutler/nvm-windows/releases 网页下载 nvm-setup.exe 文件，如图 1-8 所示。

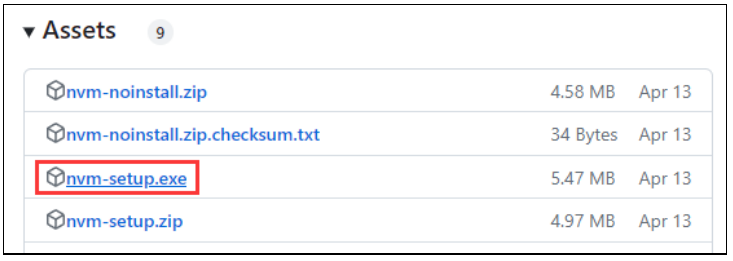

图 1-8

步骤 02　下载后，双击 nvm-setup.exe 进行安装。为了方便，可以给 nvm 配置环境变量。如图 1-9 所示。

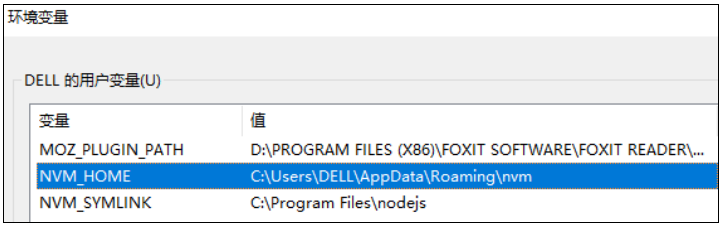

图 1-9

步骤 03　在 CMD 控制台执行 nvm version 命令，查看当前 nvm 安装的版本：

```
C:\Users\DELL>nvm version
1.1.11
```

2. 使用 nvm

安装了 nvm 后，就可以使用以下命令来管理 Node.js 版本：

- nvm install <version>：安装指定版本的 Node.js。例如，要安装 Node.js 14.17.6，可以运行 nvm install 14.17.6。
- nvm use <version>: 切换到指定版本的 Node.js。例如，要使用 Node.js 14.17.6，可以运行 nvm use 14.17.6。
- nvm ls 或者 nvm list：列出已安装的所有 Node.js 版本。
- nvm current：显示当前正在使用的 Node.js 版本。
- nvm alias <name> <version>：为指定版本创建别名。例如，nvm alias default 14.17.6 将 v14.17.6 设置为默认版本。

通过以上命令，我们可以方便地安装、切换和管理不同的 Node.js 版本，以适应各个项目的需求。

1.5　Visual Studio Code 的使用

当我们使用 Visual Studio Code 作为前端代码开发工具时，需要做一些配置，例如安装插件、忽略 node_module 目录等，这样可以极大地提升开发效率。

Visual Studio Code 官网下载地址为 https://vscode.en.softonic.com/。

1.5.1　忽略 node_module 目录

在 Visual Studio Code 中，为了提升开发工具的性能，我们通常会忽略 node_module 目录。忽略后，node_module 目录将不会显示在 Visual Studio Code 当中。

操作步骤如下：

步骤 01　在 Visual Studio Code 中，在菜单栏上依次单击 File（文件）→Preferences（首选项）→Settings（设置）命令，如图 1-10 所示。

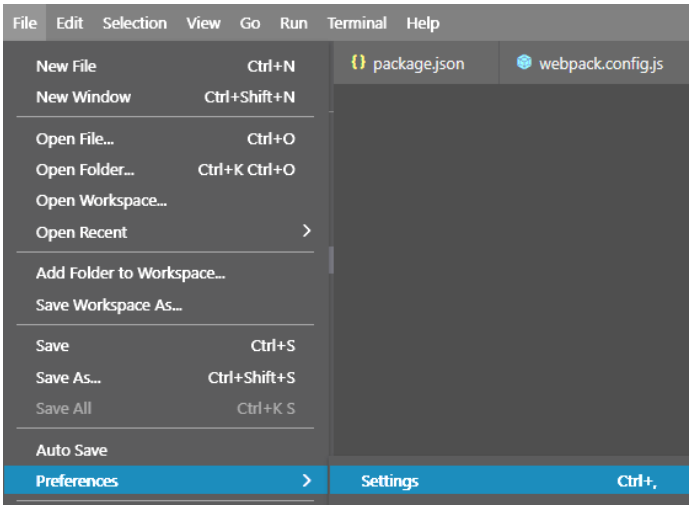

图 1-10

步骤 02　在控制面板中输入 setting.json，打开 setting.json，如图 1-11 所示。

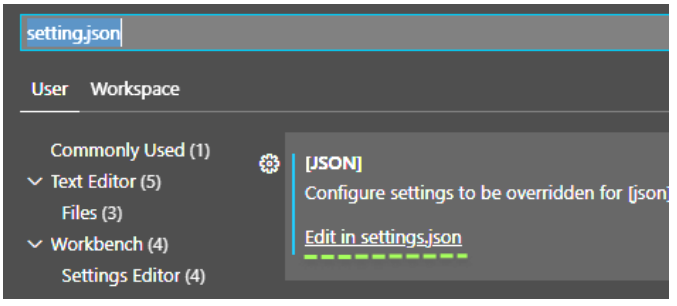

图 1-11

步骤 03　修改 setting.json，添加以下代码：

```
"files.exclude": {
    "**/node_modules": true
},
```

这样就可以忽略 node_modules 目录。

1.5.2　安装 Visual Studio Code 插件

当使用 Visual Studio Code 来开发基于 Vue 3+TypeScript 的 Web 应用时，安装合适的 Visual Studio Code 插件可以极大地提升我们的开发体验和开发效率。

Visual Studio Code 中的许多插件都需要我们自己去安装，此处以安装一个 Vue 的插件 Vue Peek 为例进行介绍。打开 Visual Studio Code，单击左侧最下面的图标，然后在搜索框中输入“Vue Peek”，再在搜索出来的列表中找到这个插件，单击 Install 按钮进行安装，如图 1-12 所示。

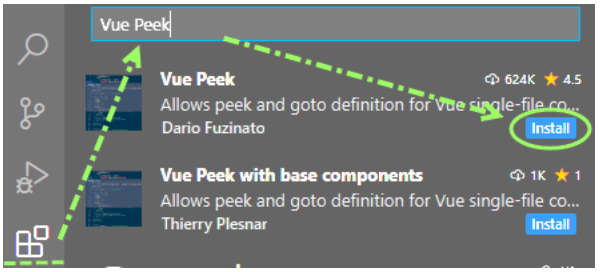

图 1-12

其他插件的安装方法与此类似。

Visual Studio Code 支持中文显示，不过笔者还是习惯英文显示，读者可以根据自己的喜好选择。建议安装以下插件：

- Mongo Snippets for Node-js：基于 Node.js 的 Mongo 代码模板。
- Node.js Modules Intellisense：Node.js 智能提示。
- Bootstrap 3 Snippets：Bootstrap3 代码模板。
- Volar：支持 Vue 3 语法高亮、语法检测，还支持 TypeScript 和基于 vue-tsc 的类型检查功能。
- Vue VSCode Snippets：Vue 的代码片段插件，可以通过输入快捷指令自动插入代码片段。例如输入“vbase-3”。
- IntelliSense for CSS class names in HTML：在 HTML 页面中可以对 CSS 的 class 类名进行智能提示。
- Auto Close Tag：自动闭合标签。它支持 HTML、Handlebars、XML、PHP、Vue、JavaScript、Typescript、JSX 等。
- Auto Rename Tag：自动完成另一侧标签的同步修改。
- Vue Peek: 该插件用来拓展 Vue 代码编辑的体验，可以让我们快速跳转到组件、模块定义的文件。
- Prettier - Code formatter：代码格式化。
- HTML CSS Support：支持 HTML、CSS、SCSS、LESS 等语法的智能提示和代码补全。
- CSS Tree：根据选中的 HTML 或者 JSX 自动生成 CSS 代码树。
- Live Server：可以快速启动本地 HTML 页面。

可选择安装的插件如下：

- Chinese (Simplified) Language Pack for Visual Studio Code：中文（简体）语言包。
- EsLint：代码风格检查工具，可以通过配置文件自定义规则，主要用于语法纠错。

使用 Volar 时需要注意以下事项：

- 首先要禁用 Vetur 插件，避免冲突。
- 推荐使用 CSS/LESS/SCSS 作为<style>的语言，因为它们基于 vscode-css-language 服务提供了可靠的语言支持。
- 如果使用 postcss/stylus/sass，就需要安装额外的语法高亮扩展。postcss 使用 language-postcss，stylus 使用 language-stylus 拓展，sass 使用 Sass 拓展。
- Volar 不包含 ESLint 和 Prettier，而官方的 ESLint 和 Prettier 扩展支持 Vue，所以需要自行安装。

过去 Vue 2 用得比较多的语法插件是 Vetur。而在 Vue 3 中则使用 Volar，用于 Vue 3 的智能代码提示、语法高亮、智能感知、Emmet 等。Volar 替代了 Vue 2 中的 Vetur 插件。Vetur 和 Volar 之间存在一些冲突，需要使用哪个版本的 Vue 就安装对应的插件。

1.5.3　打开并运行项目

这里，笔者准备从 GitHub 上下载一个示例项目，下载地址为 https://github.com/un-pany/v3-admin-vite/tree/main。下载操作如图 1-13 所示。

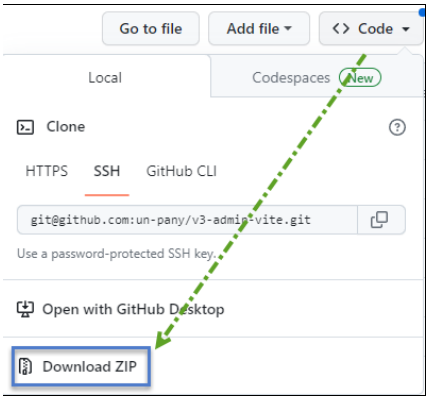

图 1-13

将项目下载到本地并进行解压，解压后的目录名称为 v3-admin-vite-main。下面开始打开并运行这个示例项目。

步骤 01　打开 Visual Studio Code，在 IDE 的菜单栏中依次单击“File”→“Open Folder…”，然后选中我们的项目代码目录，例如“D:\WorkSpace\node_mongodb_vue3_book\codes\chapter1\v3-admin-vite-main”。

我们查看 package.json 文件，看到使用的 Vue 版本是 3.3.4，如图 1-14 所示。

图 1-14

因此，要运行这个项目，需要使用高版本的 Node.js，这里使用当前最新版本 Node.js 18.17.1。

步骤 02　在 IDE 的菜单栏中依次单击“Terminal”→“New Terminal”，然后在控制台输入“yarn”，

安装项目需要的依赖包，如图 1-15 所示。

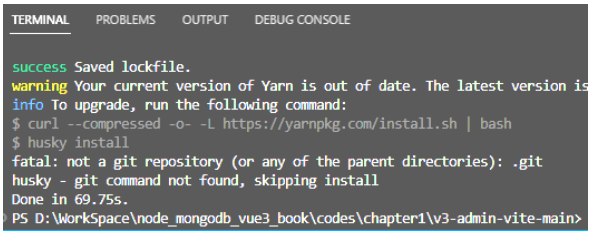

图 1-15

确保项目的依赖包安装完成之后，我们查看 package.json 文件，可以看到如下配置项：

```
  "scripts": {
    "dev": "vite",
    "build:stage": "vue-tsc --noEmit && vite build --mode staging",
    "build:prod": "vue-tsc --noEmit && vite build",
    "preview:stage": "pnpm build:stage && vite preview",
    "preview:prod": "pnpm build:prod && vite preview",
    "lint:eslint": "eslint --cache --max-warnings 0 \"{src,tests,types}/**/*.{vue,js,
jsx,ts,tsx}\" --fix",
     "lint:prettier": "prettier --write \"{src,tests,types}/**/*.{vue,js,jsx,ts,tsx,
json,css,less,scss,html,md}\"",
    "lint": "pnpm lint:eslint && pnpm lint:prettier",
    "prepare": "husky install",
    "test": "vitest"
  },
```

这就意味着我们可以在控制台直接运行这些脚本。例如执行 yarn run dev，结果如下：

```
PS D:\WorkSpace\node_mongodb_vue3_book\codes\chapter1\v3-admin-vite-main> yarn run dev
yarn run v1.22.10
$ vite

  VITE v4.4.9  ready in 2909 ms

  ➜  Local:   http://localhost:3333/
  ➜  Network: http://192.168.0.108:3333/
  ➜  press h to show help
```

步骤 03　直接按住 Ctrl 键并单击链接，就可以访问已运行的项目；或者复制这个地址，在浏览器中运行。

如果不想关闭或者中断当前运行的项目，而又需要在这个控制台终端执行新的任务，我们可以通过菜单栏再新开一个 Terminal，或者直接单击如图 1-16 所示的快捷图标进行操作。

图 1-16

1.5.4　Visual Studio Code 配置

在 IDE 的菜单栏中依次单击“File”→“Preferences”→“Settings”，可以对 Visual Studio Code 进行可视化的配置，如图 1-17 所示。

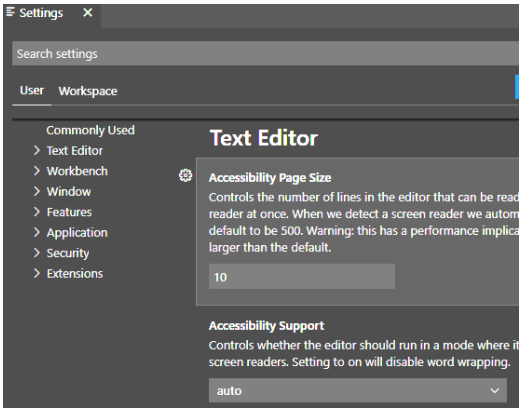

图 1-17

也可以通过 Ctrl+Shift+P 快捷键打开 JSON 配置文件 settings.json，直接在配置文件中进行配置，如图 1-18 和图 1-19 所示。

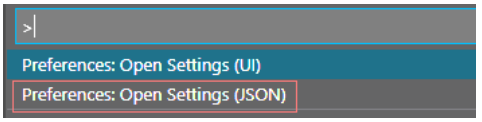

图 1-18

图 1-19

其实，当我们通过可视化的界面进行配置时，也会相应地修改 settings.json 配置文件。

1.5.5　搜索

在 Visual Studio Code 中进行搜索，有全局搜索和当前页面搜索两种方式。

1. 全局搜索

如图 1-20 所示，先单击左侧的搜索图标，然后输入内容，就可以进行全局模糊查找。当我们单击查找列表中的记录时，可以自动定位并打开查找到的文件。

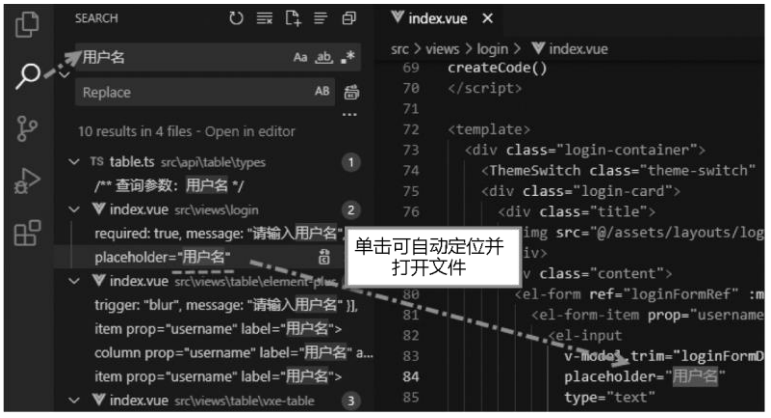

图 1-20

2. 当前页面搜索

在当前页面按 Ctrl+F 快捷键可直接在页面中进行搜索，如图 1-21 所示。

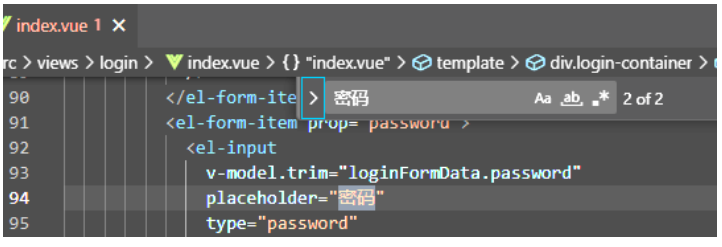

图 1-21