

03 Webpack 开发环境搭建

更新时间：2019-06-21 09:49:01



“

从不浪费时间的人，没有工夫抱怨时间不够。

——杰弗逊

”

本文将包括安装 Node、NPM、WebPack 开发环境，我使用的开发 IDE 是 [VScode](#)。因为 Webpack 实际是用 Node.js 写的，所以首先来介绍下 Node.js 的安装。已经准备好环境的，或者之前有过 Node.js 和 NPM 使用经验的可以直接跳过本小节内容。直接跳到下一小节的 Webpack 入门内容。

安装 Node.js

首先进入 [Node.js 的官网](#)，选择对应系统的 [下载包](#) 进行下载安装，对于 windows 用户，直接下载安装包安装即可，如果是 MacOS 用户，推荐使用 [brew](#) 进行安装。接下来分别介绍下 Node.js 的版本管理和包管理工具。

Node.js 版本管理

Node.js 版本众多，包括稳定版和开发版，可能不同的项目需要的 Node.js 版本不同，这里我推荐大家安装 8.9 以上版本，对于已经安装了 Node.js 的朋友，可以使用 [nvm](#)（[windows 版本](#)）对 Node.js 进行进行版本管理，（另外阿里有个 [tnvm](#)，也是管理 Node.js 版本的，增加了 alinode 版本系列的 Node.js）。

Node.js 包管理工具

Node.js 之所以这么流行，离不开庞大的社区建设，这里第一功劳就是 NPM 团队的贡献，使用 Node.js 写的代码，可以打包发布到 JavaScript 包管理平台 [npmjs.com](#)（这个存放包的地方一般也被称为仓库）上，当我们项目需要使用某个包（模块）时，可以直接使用包管理工具来安装（下载）对应的包，我们也可以免费注册一个账号，发布自己的公共包和私有包供其他人使用。

NPM 是围绕着 [语义版本控制（semver）](#) 思想而设计的，我们在软件版本中碰见的：rc、1.x.x、alpha、beta 等名词都可以在 [semver.org](#) 得到解释介绍，简单来说规范是 [主版本号.次版本号.修订号](#)（[MAJOR.MINOR.PATCH](#)）：

- 1.主版本号：当你做了不兼容的 API 修改；
- 2.次版本号：当你做了向下兼容的功能性新增；
- 3.修订号：当你做了向下兼容的问题修正；

NPM 中使用了一个命名为 `package.json` 的文件作为一个 NPM 包的描述文件，`package.json` 包含了包的基本信息（名称、版本号、描述、作者等）和依赖关系，例如：

```
{
  "name": "demo",
  "version": "1.0.0",
  "dependencies": {
    "webpack": "^4.29.6"
  }
}
```

除了 `version` 符合 `semver` 规范以外，再来看下其他两项：

- `name`：上面的代码就是表明了这个项目为 `demo`，这样如果我们将发布到 `npmjs.com` 会以这个来命名，除了这种方式的名称，还有一种命名的方式是 `@scope/name` 的方式，是 `作用域包`，例如我们用来转化 ES6 语法的 `@babel/core` 就是 `@babel` 的作用域，详细介绍可以查看 [package.json 的文档](#)
- `dependencies`：是 `demo` 这个项目的依赖，就是 `demo` 这个包内离开 `webpack` 这个包就不能使用了，对应的还有 `devdependencies`，开发以来，一般需要二次开发 `demo` 的时候需要安装的包，实际项目中，`webpack` 是构建工具，代码不会直接用 `webpack` 的 API，而只在开发和打包的时候采用，所以正确做法是放在 `devdependencies` 中。

注意到 `dependencies` 中 `webpack` 的后面版本号前面加了 `^`，意思是主版本是 `4` 的最新版本，每次执行安装命令的时候，会更新符合这个规则的最新版，可以在 [npm semver range](#) 部分看到更详细的介绍。

NPM 的常用命令

下面介绍下 NPM 的常用命令：安装、删除、初始化、配置。

安装和删除

安装某个 NPM 包，使用命令 `npm install packageName`，简写 `npm i packageName`，如果执行命令的目录下有 `package.json` 则可以直接用 `npm install` 安装 `package.json` 中的所有依赖。如果我们要安装某个版本的包，则可以使用命令 `npm i packageName@x.x.x` 格式。

如果我们安装依赖包并且将这个依赖写入 `package.json` 则可以使用命令 `npm i packageName --save`（简写 `npm i packageName -S`），如果希望写到 `package.json` 开发依赖中（`devdependencies`）则使用命令 `npm i packageName --save-dev`（简写 `npm i packageName -D`）

删除某个 NPM 包，则使用 `npm uninstall 包名`。

本地模式和全局模式

npm 的包安装，分为本地模式和全局模式，默认是本地模式，即在执行 `npm install` 命令的当前目录创建 `node_modules`，然后下载安装包及其依赖到 `node_modules` 目录。全局模式是指安装到全局路径的方式。在 Node.js 的 `require` 依赖之时，会优先查找自己当前文件的 `node_modules`，如果没有，则循环遍历上层的 `node_modules`，如果遍历到根目录还找不到，则会使用全局模式安装的模块，另外全局模式安装的包可以指定全局命令，只需要在 `package.json` 增加 `bin` 字段并且指向包内对应的文件即可。全局安装一个包，使用命令 `npm install --global`，`--global` 可以简写为 `-g`。

初始化一个 NPM 项目

`npm init` 用来初始化生成一个新的 `package.json` 文件。输入 `npm init` 并且根据对应的提示回答问题，会向用户提问一系列问题，如果你觉得不用修改默认配置，一路回车就可以了。

如果使用了 `-f`（代表 `force`）、`-y`（代表 `yes`），则跳过提问阶段，直接生成一个新的 `package.json` 文件。

设置 NPM 镜像

由于 NPM 网站经常不稳定，所以国内有很多镜像可以使用，[淘宝 NPM 镜像](#)是国内最大的一家 NPM 镜像网站，还有 `cnpm` 包可以替换官方 NPM 来使用，使用 `cnpm` 直接使用淘宝镜像安装 NPM 包。

单次使用镜像方法：

```
npm [命令] --registry=https://registry.npm.taobao.org
```

设置默认 npm 使用淘宝镜像方法：

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

使用下面的命令可以安装 `cnpm` 包，之后直接像使用 `npm` 一样使用 `cnpm` 即可，不需要添加 `register`

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

NPM 其他常用命令

- `npm set`: 设置环境变量，例如：`npm set init-author-name 'Your name'`，初始化的时候会使用默认环境变量；
- `npm info`: 查看某个包的信息，例如：`npm info lodash`；
- `npm search`: 查找 npm 仓库，后面可以跟字符串或者正则表达式，例如：`npm search webpack`；
- `npm list`: 树形的展现当前项目安装的所有模块，以及对应的依赖，例如：`npm list --global` 查看全局安装的模块。

NPM Scripts

NPM 不仅可以用于模块管理，还可以用于执行脚本。`package.json` 文件中可以添加 `scripts` 字段，用于指定脚本命令，供 NPM 直接调用。例如：

```
// package.json
{
  "scripts": {
    "build": "webpack",
    "start": "node src/scripts/dev.js"
  }
}
```

在 `package.json` 添加上面字段之后，可以直接使用 `npm run build` 和 `npm run start` 命令了，实际上：

- `npm run build`：相当于执行了当前项目中目录下的 `webpack` 命令；
- `npm run start`：相当于执行了 `node src/scripts/dev.js`。

另外 `npm run start` 还可以简写成 `npm start`。

Tips: 除了 `npm` 外，还有一些包管理工具，主要是针对 `npm` 的下载速度慢、`node_modules` 混乱等缺点设计的，例如 `yarn` 和 `pnpm`。

安装 Webpack-cli

`Webpack-cli` 是 `Webpack` 的 CLI（Command-line interface）工具，如果在项目中，我们可以使用下面的方式安装：

```
npm install webpack-cli --save-dev
```

如果想全局使用 `webpack` 的命令，可以使用 `npm install -g webpack-cli` 安装。

Tips: 这里建议在项目中安装 `webpack-cli` 并且使用 `--save-dev` 的配置将 `webpack-cli` 放到开发依赖中。

到此，我们就准备好 `Webpack` 的命令行开发环境了，下面小节开始介绍 `webpack-cli` 的零配置打包。

小结

本小节主要介绍了 `Webpack` 的开发环境搭建，从 `Node.js` 安装、`NPM` 的使用最基础开始讲解，指导学生手把手的上手 `Node.js` 开发环境。并且介绍了 `NPM` 相关的命令使用和 `NPM Scripts` 概念，`NPM Scripts` 在用 `NPM` 搭建项目开发命令时被广泛应用，接下来我们开始体验 `Webpack-cli` 的零配置打包吧！

本小节 `Webpack` 相关面试题：

1. 什么是 `NPM Scripts`？`NPM Scripts` 可以用来做什么？
2. `NPM` 的常用命令有哪些？