

【架构师（第二篇）】脚手架架构设计和框架搭建

一尾流莺 2022-02-17 11:01 1034

关注

脚手架架构设计和框架搭建

将收获什么

- 脚手架的实现原理
- Lerna 的常用方法
- 架构设计技巧和架构图绘制方法

主要内容

- 学习如何以架构师的角度思考基础架构问题
- 多 Package 项目管理痛点和解决方案，基于 Lerna 脚手架框架搭建
- 脚手架需求分析和架构设计，架构设计图
- 脚手架调试技巧
- Lerna 源码分析
- node 的 module 模块分析
- yargs 使用方法
- 剖析 Lerna 架构设计

开发脚手架的必要性

开发脚手架的核心目标是：**提升前端研发效能**

以下内容如果全靠脚手架进行自动化处理，可以提高相当大的研发效率了。

创建项目 + 通用代码

- 埋点
- `http` 请求
- 工具方法
- 组件库

git 操作

- 创建仓库
- 代码冲突
- 远程代码同步
- 创建版本
- 发布打 `tag`

构建 + 发布上线

- 依赖安装和构建
- 资源上传 `cdn`
- 域名绑定
- 测试/正式服务器

脚手架核心价值

将研发过程

- 自动化：项目重复代码拷贝、`git` 操作、发布上线操作
- 标准化：项目创建、`git flow`、发布流程、回滚流程
- 数据化：研发过程系统化、数据化、使得研发过程可量化

和自动化构建工具的区别

问题：`jenkins`，`travis` 等自动化构建工具已经很成熟了，为什么还要自研脚手架？

- 不满足需求：`jenkins`，`travis` 通常在 `git hooks` 中触发，需要在服务端执行，无法覆盖研发

且本地功能，如：创建项目自动化、本地 `... ..` 操作自动化等

- 定制复杂: `jenkins`, `travis` 定制过程需要开发插件, 其过程较为复杂, 需要使用 `java` 语言, 对前端同学不太友好。

| 什么是脚手架

脚手架本质是一个操作系统的客户端, 他通过命令行执行, 比如



js 复制代码

```
1 vue create vue-test-app
```

上面这条命令由 3 个部分组成:

- **主命令**: `vue`
- **command**: `create`
- **command 的 param**: `vue-test-app`

他表示创建一个 `vue` 项目, 项目的名称为 `vue-test-app`, 这是一个比较简单的脚手架命令, 但实际场景往往更加复杂, 比如:

当前目录已经有文件了, 我们需要覆盖当前目录的文件, 强制进行安装 `vue` 项目, 此时我们就可以输入



js 复制代码

```
1 vue create vue-test-app --force
```

这里的 `--force` 叫做 `option`, 用来辅助脚手架确认在特定场景下用户的选择 (可以理解为配置)。还有一种场景:

通过 `vue create` 创建项目时, 会自动执行 `npm install` 帮助用户安装依赖, 如果我们希望使用淘宝源来安装, 可以输入命令



js 复制代码

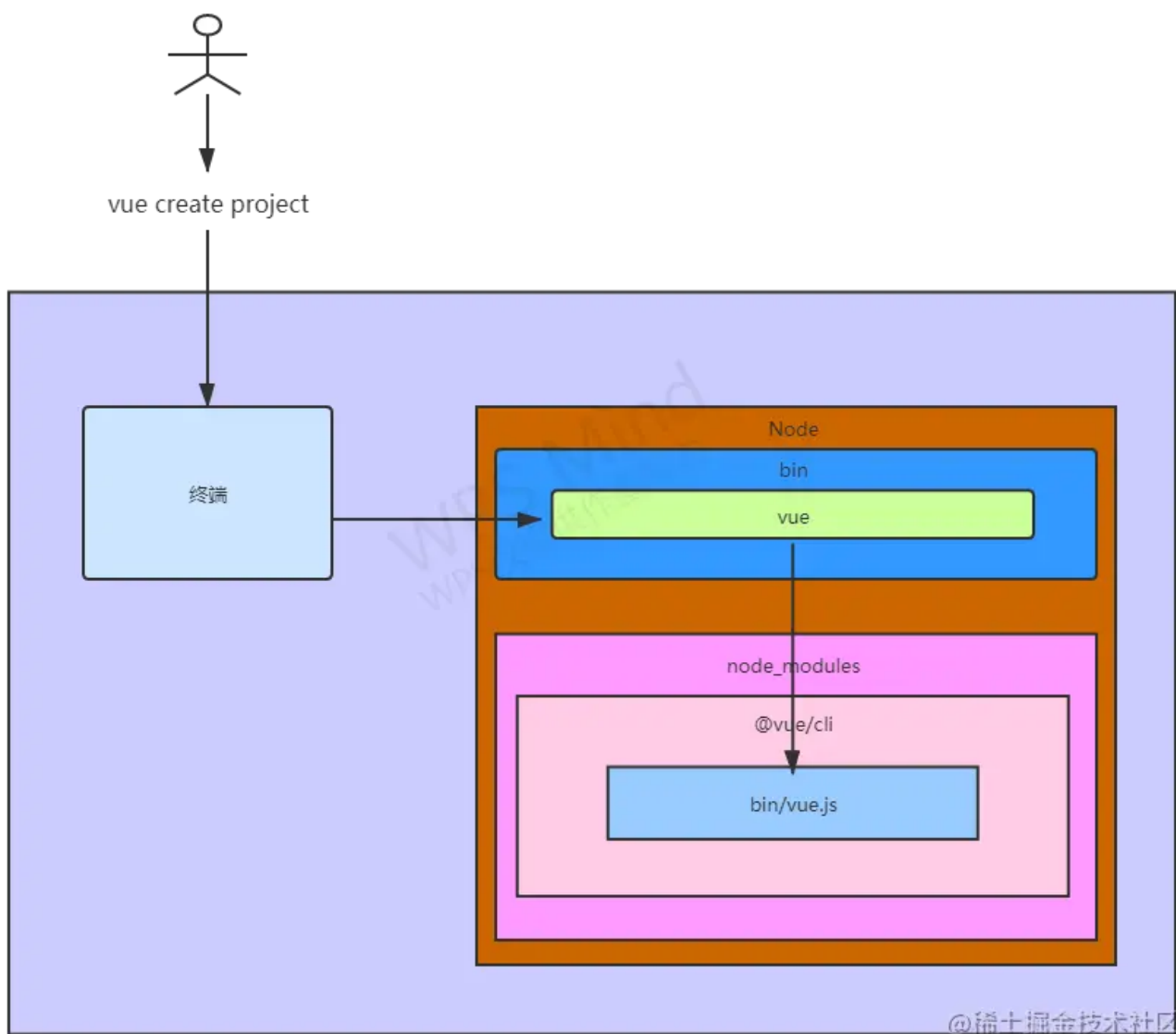
```
1 vue create vue-test-app --force -r https://registry.npm.taobao.org
```

这里的 `-r` 也叫做 `option`, 它与 `--force` 不同的是它使用 `-`, 并且使用简写, 这里的 `-r` 也可以替换成 `--registry`, 输入下面的命令就可以看到 `vue create` 支持的所有 `options`。

```
1 vue create --helps
```

-r 后面的 `https://registry.npm.taobao.org` 成为 option 的 param，其实 `--force` 可以理解为：`--force true`，简写为 `--force` 或 `-f`。

脚手架的执行原理



脚手架执行原理如下

- 在终端输入 `vue create project`
- 终端解析出 `vue`
- 在环境变量中通过 `which vue` 找到 `vue` 命令, 目录所在 `/node/bin/vue`，所以我们执行的 `vue`，实际上运行的是 `/node/bin/vue` 的这个 `vue`

- 这个 `vue` 只是一个链接，终端根据 `vue` 命令链接到实际文件
`/node/lib/node_modules/@vue/cli/bin/vue.js`
- 终端利用 `node` 执行 `vue.js`
- `vue.js` 解析 `command` 以及 `param`
- `vue.js` 执行 `command`
- 执行完毕，退出执行

如何开发一个脚手架

以 `vue-cli` 为例

- 开发一个 `npm` 项目，该项目中应包含一个 `bin/vue.js` 文件，并将这个项目发布到 `npm`；
- 将这个项目发布到 `npm`
- 将 `npm` 项目上的项目全局安装到 `node` 的 `lib/node_modules`
- 在 `node` 的 `bin` 目录下配置 `vue` 软链接指向 `lib/node_modules/@vue/cli/bin/vue.js`

这样我们在执行 `vue` 命令的时候就可以找到 `vue.js` 进行相关操作。

脚手架实现原理问题

为什么全局安装 `@vue/cli` 后会添加一个 `vue` 的命令呢？



js 复制代码

```
1 npm i -g @vue/cli
```

运行 `vue` 命令时，实际走的是 `node/bin/vue`，而这个文件只是一个软连接，指向
`lib/node_modules/@vue/cli/bin/vue.js`。

回到上级目录 `lib/node_modules/@vue/cli`，打开 `package.json` 文件，里面的 `bin` 字段定义了这样的绑定关系。



js 复制代码

```
1 // lib/node_modules/@vue/cli/package.json
2 {
3   "bin": {
4     "vue": "bin/vue.js"
5   }
6 }
```

总结：执行 `vue` 命令的时候，启动的是 `bin/vue` 这个文件，而这个文件指向 `lib/node_modules/@vue/cli/bin/vue.js`，所以最终启动的是 `lib/node_modules/@vue/cli/bin/vue.js`

全局安装 @vue/cli 的时候发生了什么？

- 把 `@vue/cli` 的包通过 `npm` 安装到 `node/lib/node_modules` 这个目录下。
- 解析 `package.json` 文件，根据文件中的 `bin` 字段，在 `/node/bin` 目录下创建软连接，软连接指向 `bin` 字段中规定的文件，也就是 `lib/node_modules/@vue/cli/bin/vue.js`。

执行 vue 命令时发生了什么？

- 根据 `which vue` 这条指令（在环境变量中查找），找到 `vue` 命令所在文件
- 运行这个文件，执行 `vue` 和执行 `node/bin/vue` 的结果是一样的
- 根据软连接，执行真实的 `lib/node_modules/@vue/cli/bin/vue.js` 文件

为什么 vue 指向一个 js 文件，我们却可以直接通过 vue 命令去执行它？

查看 `lib/node_modules/@vue/cli/bin/vue.js` 文件的源码，会发现第一行代码是这样的



js 复制代码

```
1 #!/usr/bin/env node
```

它的意思就是在环境变量中查找使用 `node` 命令来运行此文件。

为什么说脚手架本质是操作系统的客户端？

因为 `node` 本身是一个客户端，在 `windows` 系统下，可以看到 `node` 的安装目录中，`node` 是以 `node.exe` 的形式出现的。

而我们编写的脚手架文件，如 `vue.js` 只是 `node` 运行时的一个参数。

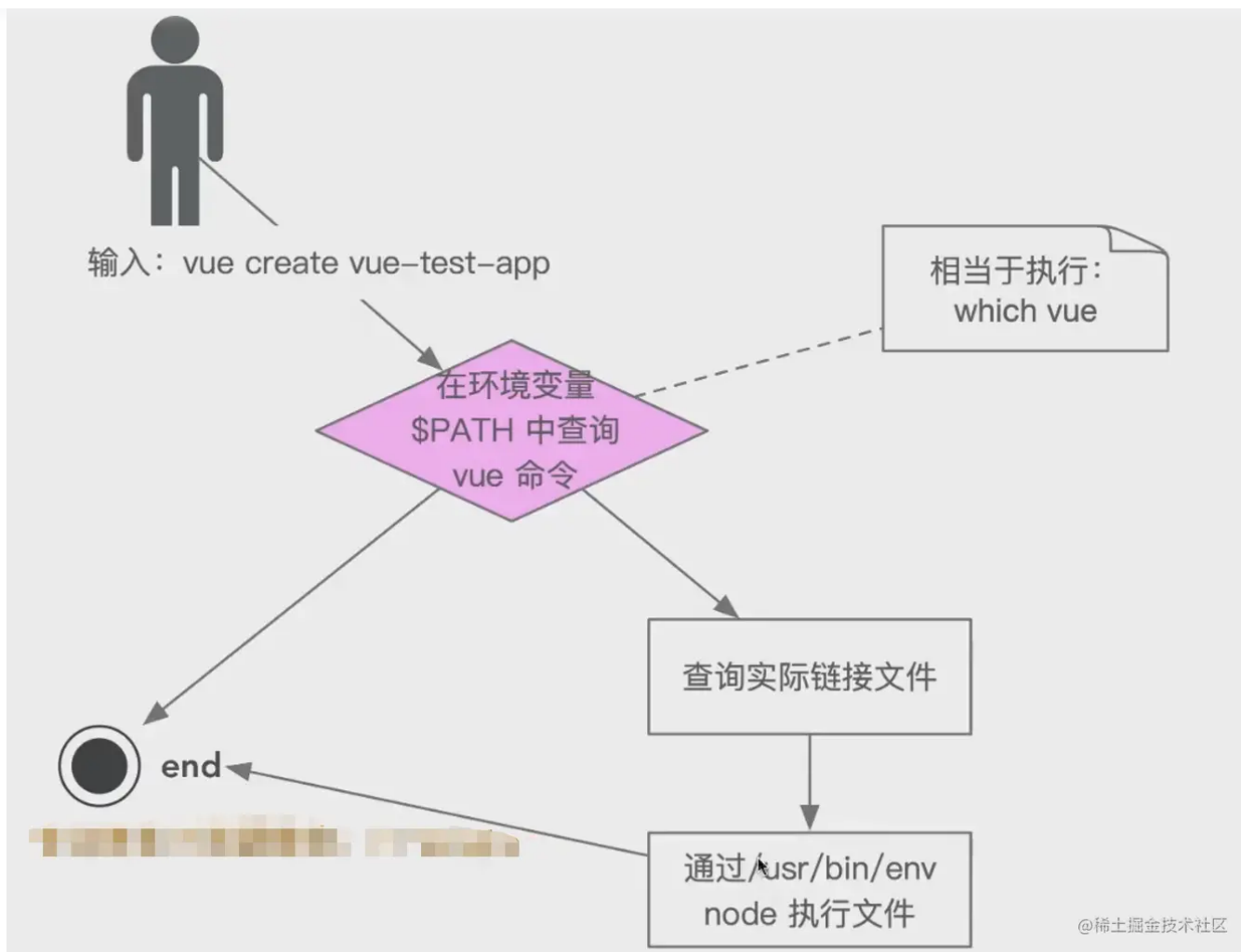


js 复制代码

```
1 node vue.js
```

如何为 node 脚手架创建别名？

软连接是可以嵌套的，只需让别名指向原来的名字即可。



脚手架开发流程

开发流程

- 创建 `npm` 项目
- 创建脚手架入口文件，最上方添加 `#!/usr/bin/env node`
- 配置 `package.json` 文件，添加 `bin` 属性，指定脚手架名称和入口文件地址
- 编写脚手架代码
- 将脚手架发布到 `npm`

使用流程

```
1 npm i -g @vue/cli
```

- 使用脚手架

```
1 vue create project
```

脚手架开发难点

- 分包：将复杂的系统拆分成多个模块
- 命令注册
- 参数解析
- 帮助文档
- 命令行交互
- 日志打印
- 命令行文字变色
- 网络通信： HTTP/WebSocket
- 文件处理
-

开发一个简单的脚手架

- 新建文件夹 `test-cli`

```
1 mkdir test-cli
```

- 进入到 `test-cli` ， - 初始化 `npm` 包，通过 `code .` 可以快速使用 `vscode` 打开当前文件夹。


```
1 cd test-cli
2 npm init -y
3 code .
```

- 添加 `bin/index.js` 文件，内容如下

```
1 // bin/index.js
2
3 #!/usr/bin/env node
4
5 console.log('🚀~ 脚手架开发 测试');
```

js 复制代码

- 修改 `package.json` 文件，添加 `bin` 属性，指定脚手架名称和入口文件地址

```
1 // package.json
2 {
3   "name": "test-cli-0174",
4   "version": "1.0.0",
5   "description": "",
6   "bin": {
7     "test-cli": "bin/index.js"
8   },
9   "main": "index.js",
10  "scripts": {
11    "test": "echo \"Error: no test specified\" && exit 1"
12  },
13  "keywords": [],
14  "author": "",
15  "license": "ISC"
16 }
```

js 复制代码

- 发布包到 `npm`

👉👉 [从0到1发布属于自己的库到npm](#)

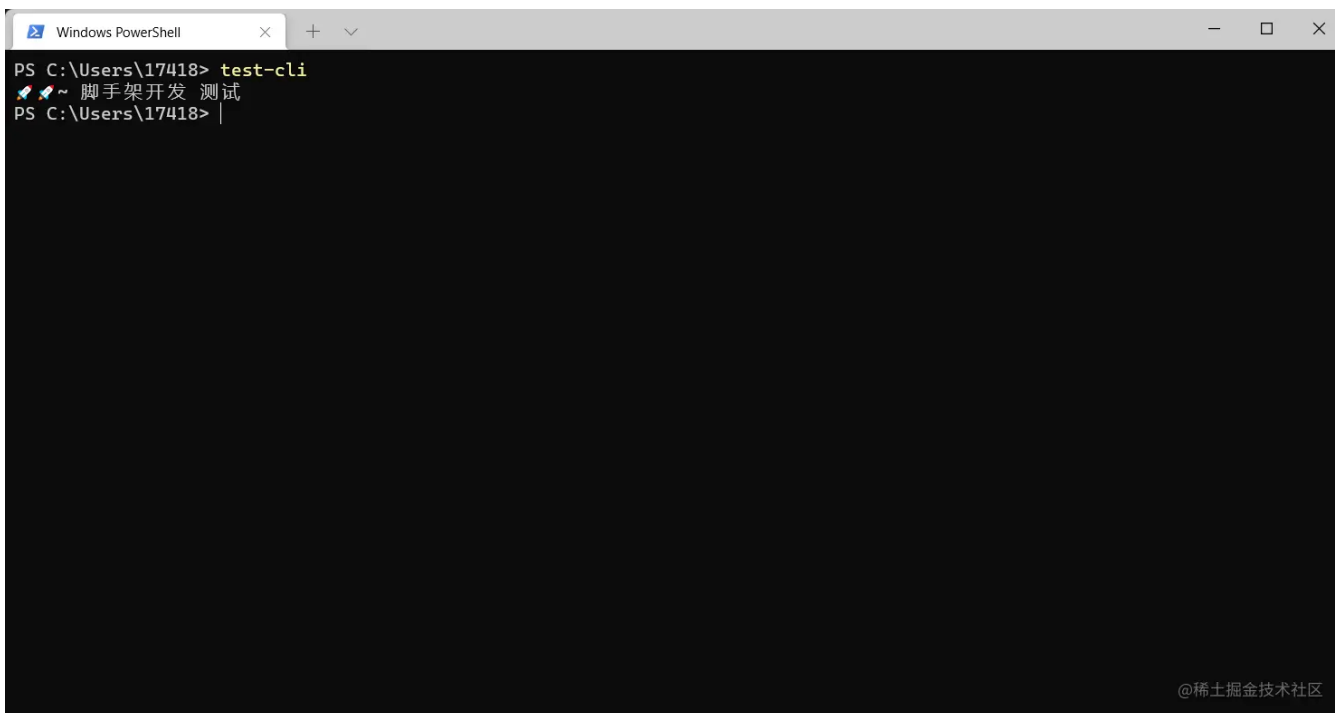
- 全局安装

```
1 npm i -g test-cli
```

js 复制代码

- 命令行执行命令 `test-cli`

结果如下，控制台输出  ~ 脚手架开发 测试



```
Windows PowerShell
PS C:\Users\17418> test-cli
🚀 ~ 脚手架开发 测试
PS C:\Users\17418> |
```

@稀土掘金技术社区

调试本地脚手架

进入到 `test-cli` 目录中

先全局移除之前通过 `npm` 安装的包，然后执行 `npm link`

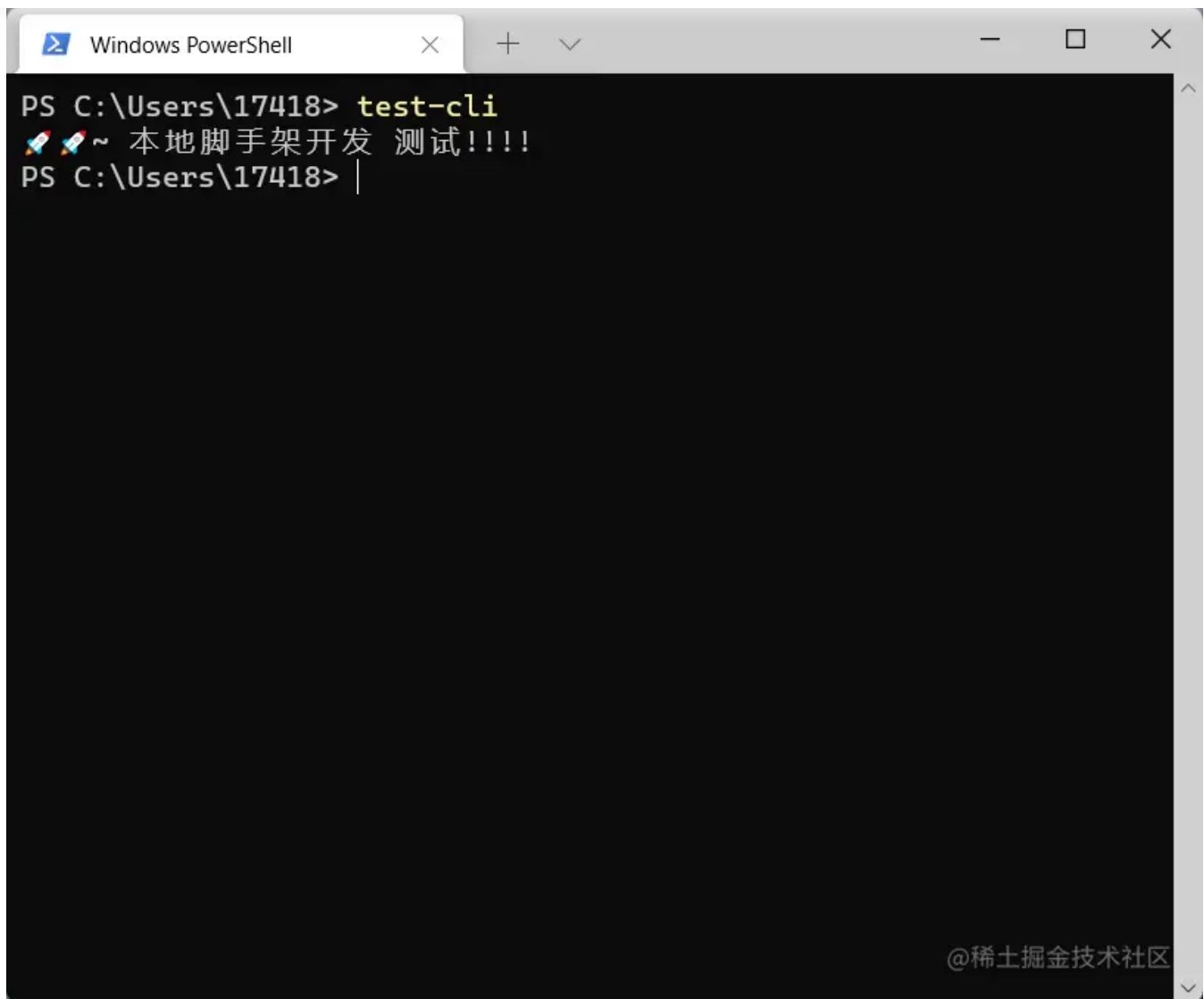


js 复制代码

- 1 `npm remove test-cli -g`
- 2 `npm link`

就会安装本地的脚手架了

随便修改本地代码后，然后再通过命令 `test-cli` 去启动脚手架



```
PS C:\Users\17418> test-cli
🚀🚀 ~ 本地脚手架开发 测试!!!!
PS C:\Users\17418> |
```

@稀土掘金技术社区

如果工程很复杂需要分包

在 `test-cli` 目录同级新建一个 `test-cli-lib` 目录，同样进行初始化。



js 复制代码

```
1 npm init -y
```

然后新建 `lib/index.js` 文件，写上一个方法。



js 复制代码

```
1 // lib/index.js
2 module.exports = {
3   sum(a, b) {
4     return a + b
5   }
6 }
```

修改 `package.json` 文件中的 `main` 属性



js 复制代码

```
1  "main": "lib/index.js",
```

进入 `test-cli-lib` 目录，执行 `npm link`，把这个包也安装到本地。

返回 `test-cli` 目录，执行



js 复制代码

```
1  npm link test-cli-lib
```

然后手动的修改 `package.json` 文件中的 `dependencies` 属性



js 复制代码

```
1  "dependencies": {  
2    "test-cli-lib"  
3  }
```

就可以把这个包连接起来了，然后再修改本地代码测试一下。



js 复制代码

```
1  // test-cli/bin/index.js  
2  #!/usr/bin/env node  
3  
4  const lib = require("test-cli-0174-lib")  
5  console.log(lib);  
6  console.log('🚀🚀~ 本地脚手架开发 测试!!!!');
```

运行 `test-cli` 命令

```
Windows PowerShell
PS D:\myGitHub\warlber-grow\架构师课程练习\脚手架练习\test-cli-0174> test-cli
{ sum: [Function: sum] }
🚀~ 本地脚手架开发 测试!!!!
PS D:\myGitHub\warlber-grow\架构师课程练习\脚手架练习\test-cli-0174> |
```

@稀土掘金技术社区

可以把函数正常的打印出来了。

注意：当开发完成后需要发布到 `npm` 上，然后通过 `npm` 安装的时候，需要执行



js 复制代码

```
1 npm unlink test-cli
2 npm unlink test-cli-lib
3 npm remove -g test-cli
4 npm remove -g test-cli-lib
```

然后再通过 `npm` 安装就行了



js 复制代码

```
1 npm i -g test-cli
2 npm i -g test-cli-lib
```

标签： 前端 架构

文章被收录于专栏：



我要当架构师

为了提升自己的能力，我在某课网买了一个架构师的课程。本专栏用来记录自己...

已关注

相关小册



VIP 玩转 CSS 的艺术之美
JowayYo...
¥19.9

4827购买



深入浅出 Vite
神三元
¥59.9

4614购买

找对——
属于你的
技术圈子

回复进群加入
掘金
微信交流群

评论

输入评论 (Enter换行, Ctrl + Enter发送)

全部评论 8

最新 最热



九号先生十号...

10月前

"dependencies": {
 "test-cli-lib"
} 直接这么写行吗？会报错的吧

点赞 2

我既然写了就不会呗

👍 点赞 💬 回复



九号先生十号少年

10月前

我这报错

“我既然写了就不会呗”

👍 点赞 💬 回复



buchiyu LV.2 JY.5

1年前

不会你也买了慕课的那个课程了吧

👍 点赞 💬 4



一尾流莺

1年前

是的兄弟，就是那个

👍 点赞 💬 回复



buchiyu

1年前

6666，一起学习，我最近也在看

👍 1 💬 回复

查看更多回复 ▾

相关推荐

一尾流莺 1年前

【架构师（第三篇）】脚手架开发之掌握Lerna操作流程

👁 709 👍 17 💬 7

一尾流莺 7月前

【架构师（第四十九篇）】服务端开发之认识 Docker-compose

👁 1151 👍 8 💬 评论

一尾流莺 1年前

【架构师（第五篇）】脚手架之import-local执行流程及简历设计



一尾流莺 1年前

【架构师（第二十一篇）】编辑器开发之需求分析和架构设计

👁 904 🍏 5 💬 4

一尾流莺 1年前

【架构师（第四十六篇）】服务端开发之安装 Docker

👁 2182 🍏 10 💬 评论

一尾流莺 1年前

【架构师（第三十八篇）】服务端开发之本地安装最新版 MySQL 数据库

👁 1831 🍏 7 💬 2

一尾流莺 1年前

【架构师（第四十一篇）】服务端开发之安装并连接 Redis数据库

👁 1572 🍏 10 💬 评论

一尾流莺 1年前

【架构师（第二十三篇）】编辑器开发之画布区域组件的渲染

👁 900 🍏 9 💬 评论

shenyWill 11月前

仿vue-cli从零搭建一个前端脚手架

👁 2129 🍏 44 💬 12

一尾流莺 7月前

【架构师（第五十篇）】服务端开发之自动发布到测试机

👁 1208 🍏 7 💬 评论

一尾流莺 1年前

【架构师（第四篇）】脚手架开发之Lerna源码分析

👁 3132 🍏 15 💬 9

一尾流莺 1年前

【架构师（第四十五篇）】服务端开发之认识 Github actions

👁 1133 🍏 6 💬 评论

一尾流莺 7月前

【架构师（第五十一篇）】服务端开发之技术方案设计

👁 1293 🍏 7 💬 评论

一尾流莺 1年前

👁 1977 👍 37 💬 6

👍 15

💬 8

★ 收藏