

【架构师（第十二篇）】脚手架之命令行交互工具 inquirer.js 使用方法

一尾流莺 2022-04-11 09:04 👁 2039

[关注](#)

inquirer.js 基本用法

| 什么是 inquirer.js

`inquirer.js` 是一个用来实现命令行交互式界面的工具集合。它帮助我们实现与用户的交互式交流，比如给用户提一个问题，用户给我们一个答案，我们根据用户的答案来做一些事情。

它包含以下功能：

- 可以向用户提出问题
- 解析用户输入的答案
- 对用户的输入进行验证
- 提供错误回调

| 安装

js 复制代码

```
1 npm i inquirer -S
```

| 使用

APP内打开

```
3 // 启动 inquirer
4 inquirer
5 // 交互内容
6 .prompt([
7 ])
8 // 收集用户答案后的回调，会以键值对的方式存储在这里
9 .then((answers) => {
10 })
11 // 捕获错误的回调
12 .catch((error) => {
13 });
```

`default` , `choices` , `validate` , `filter` 以及 `when` 的值为函数时，可以异步调用，也可以返回一个 `Promise` 或者使用 `this.async()` 方法来获得一个回调，然后使用最终值来调用它。



js 复制代码

```
1 {
2   filter() {
3     return new Promise();
4   },
5
6   validate: function (input) {
7     var done = this.async();
8     setTimeout(function() {
9       if (typeof input !== 'number') {
10         done('You need to provide a number');
11         return;
12       }
13       done(null, true);
14     }, 3000);
15   }
16 }
```

属性

这里暂时只做介绍，后面会结合代码演示所有属性的使用方法。

type

APP内打开

- **confirm**
- **list**
- **rawlist**
- **expand**
- **checkbox**
- **password**
- **editor**

name

存储当前问题回答的变量；

message

问题的描述；

default

默认值；

choices

列表选项，在某些 **type** 下可用，并且包含一个分隔符(**separator**)；

validate

对用户的回答进行校验，只有返回 **true** 的时候才会向下进行；

APP内打开

对用户的回答进行过滤处理，返回处理后的值，会修改用户提交的值；

transformer

对用户回答的显示效果进行处理(如：修改回答的字体或背景颜色)，但不会影响最终的答案的内容；

when

根据前面问题的回答，判断当前问题是否需要被回答；

pageSize

修改某些 `type` 类型下的渲染行数；

prefix

修改 `message` 默认前缀；

suffix

修改 `message` 默认后缀；

mask

修改 `type` 为 `password` 时的显示模式。

APP内打开

输入交互，完整代码如下，但是我会一个个属性的添加进行演示。

[js 复制代码](#)

```

1  inquirer
2  .prompt([
3    {
4      type: 'input',
5      message: '请输入姓名',
6      name: 'name',
7      default: '一尾流莺',
8      validate: (answer) => {
9        if (answer.length < 2) {
10          return '名字不可以少于两个字符';
11        }
12        console.log();
13        return true;
14      },
15      transformer: (a) => {
16        return `最帅的人是 : ${a}`;
17      },
18      filter: (a) => {
19        return `最帅的人是 : ${a}`;
20      },
21      prefix: '●',
22      suffix: '●',
23    },
24  ])
25  .then((answers) => {
26    console.log('🚀🚀 ~ answers', answers);
27  })
28  .catch((error) => {
29    console.log('🚀🚀 ~ error', error);
30  });

```

当前只设置了如下属性

- **type**: 交互类型，输入类
- **message**: 问题描述，命令行展示的内容
- **name**: 字段值，将来会把用户的输入

[APP内打开](#)
[定义的字段](#)

接下来我们设置一个 `default` 属性，可以看到问题后面出现了括号，括号里面就是我们设定的默认值，如果什么都不输入，直接回车的话，就会使用默认值作为答案。

- **default: 默认值**

```
warbler@DESKTOP-57C47H7 D: > myGithub > warbler-grow > 架构师课程练习 > 脚手架练习 > test-cli-0174
└─main
> test-cli
```

接下来我们设置一个 `validate` 属性，用来对答案进行验证，不符合条件的时候，会进行提示，符合条件就可以通过。

- **validate: 对用户的回答进行校验，只**

APP内打开

提示输出在页面中

@稀土掘金技术社区

接下来我们再设置一个 `transformer` 属性，这个属性用于修改答案内容的显示，但不会修改实际的答案，可以看到最终回调中输出的答案依旧是“一尾流莺”，但是界面显示的“最帅的人是：一尾流莺”。

- **transformer: 对用户回答的显示效果进行处理，但不会影响最终的答案的内容**

```
warbler@DESKTOP-57C47H7 D: > myGithub > warbler-grow > 架构师课程练习 > 脚手架练习 > test-cli-0174  
└─main  
> test-cli
```

@稀土掘金技术社区

接下来我们设置一个 `filter` 属性，这个属性会修改实际的答案，可以看到我们输入的是“一尾流莺”，但是回调中输出的是“最帅的人是：一尾流莺”。

- **filter: 对用户的回答进行过滤处理，返**

APP内打开

- 会修改用户提交的值**

接下来我们分别设置前缀和后缀，可以看到我在问题的前面加了一个绿色的圆，问题的后面加了一个黄色的圆。

- **prefix:** 修改 message 默认前缀
- **suffix:** 修改 message 默认后缀



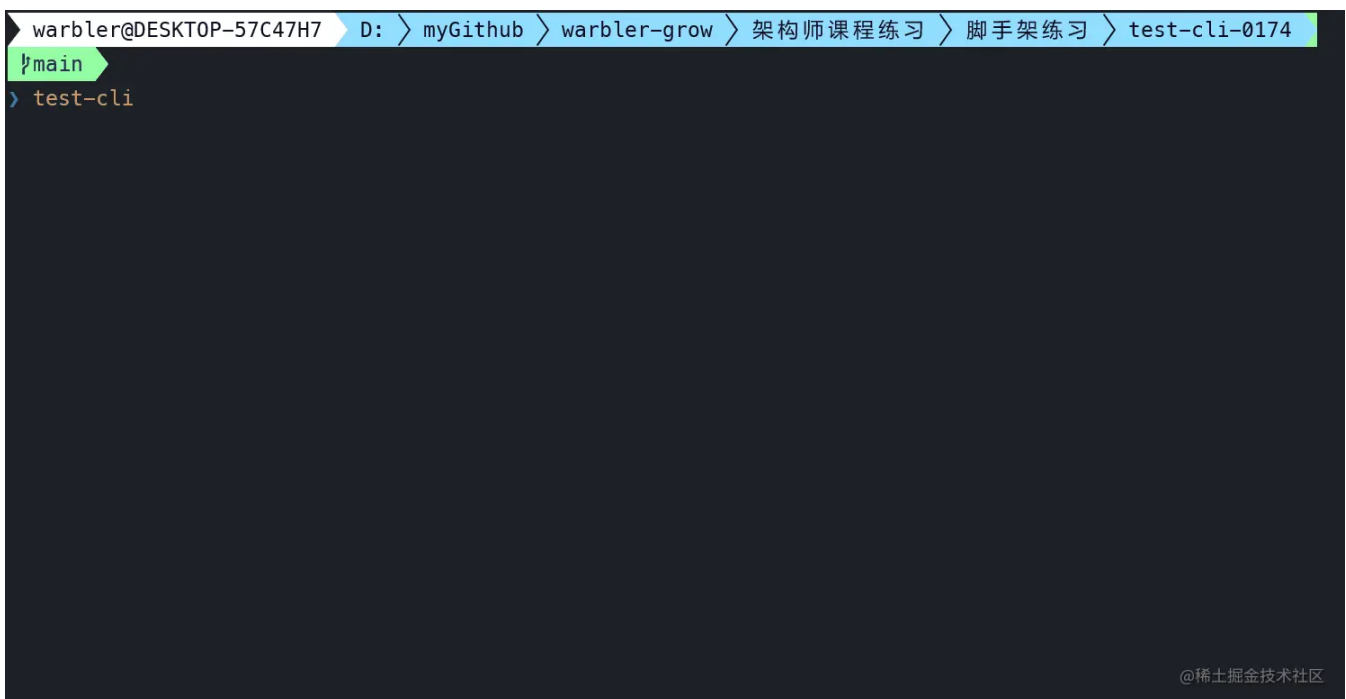
```
D: hzw-cli-dev  C:\Windows\system32\cmd.exe  D:
warbler@DESKTOP-57C47H7 D: > myGithub > warbler-grow > 架构师课程练习 > 脚手架练习 > test-cli-0174
└─main
> test-cli
● 请输入姓名 ● (一尾流莺) 最帅的人是 : |
```

number

APP内打开




```
1 inquirer
2   .prompt([
3     {
4       type: 'number',
5       message: '请输入年龄',
6       name: 'age',
7     },
8   ])
9   .then((answers) => {
10     console.log('🚀 ~ answers', answers);
11   })
12   .catch((error) => {
13     console.log('🚀 ~ error', error);
14   });
```



PS: 这里可以利用 `validate` 属性对用户输入进行校验。

confirm

确认交互，默认值在界面上会以大写字母的方式进行显示。

APP内打开

```
3    {
4      type: 'confirm',
5      message: '是否单身?',
6      name: 'single',
7      default: 'true',
8    },
9  ])
10 .then((answers) => {
11   console.log('🚀🚀 ~ answers', answers);
12 })
13 .catch((error) => {
14   console.log('🚀🚀 ~ error', error);
15 });
```

```
warbler@DESKTOP-57C47H7 D: > myGithub > warbler-grow > 架构师课程练习 > 脚手架练习 > test-cli-0174
└─main
> test-cli
```

@稀土掘金技术社区

list

列表选择交互，这里涉及到了两个新属性。



js 复制代码

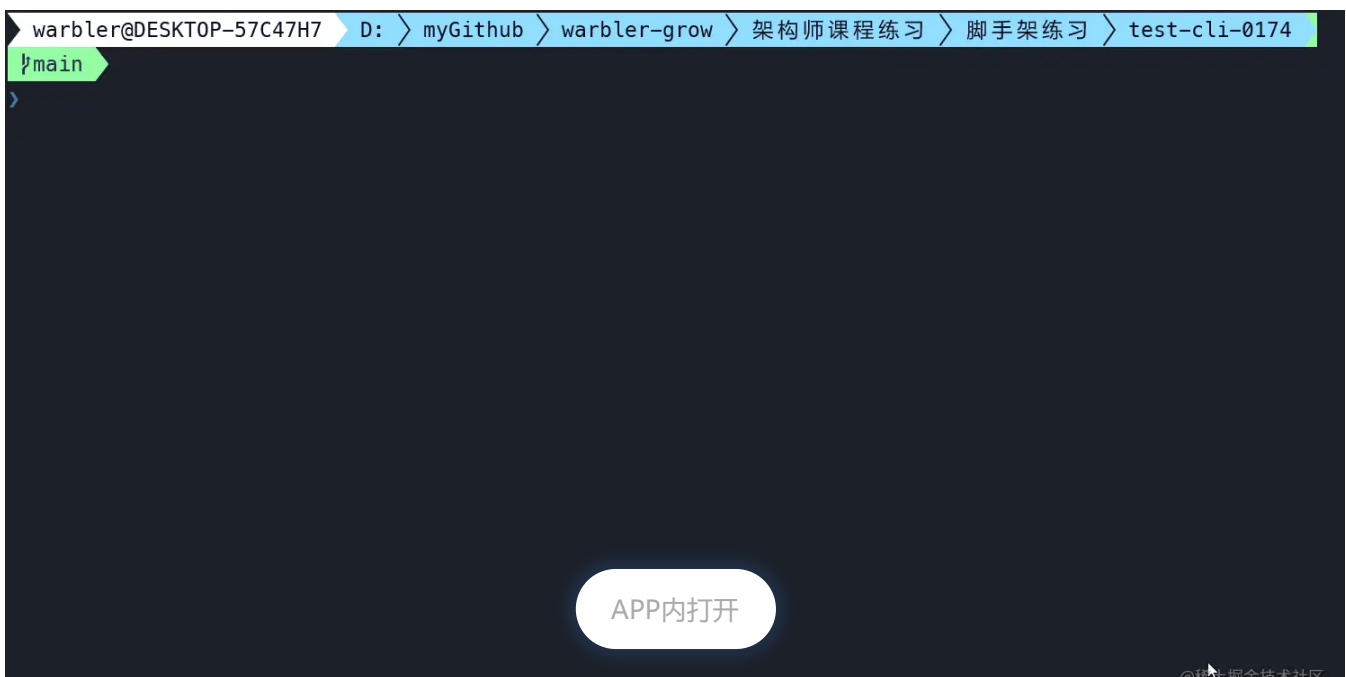
```
1 inquirer
2   .prompt([
3     {
4       type: 'list',
```

APP内打开

 33 8 收藏

```
10     value: 1,
11     name: '赵今麦',
12   },
13   {
14     value: 2,
15     name: '谭松韵',
16   },
17   {
18     value: 3,
19     name: '毛晓彤',
20   },
21   {
22     value: 4,
23     name: '新垣结衣',
24   },
25 ],
26 pageSize: 2,
27 },
28 ])
29 .then((answers) => {
30   console.log('🚀🚀 ~ answers', answers);
31 })
32 .catch((error) => {
33   console.log('🚀🚀 ~ error', error);
34 });
```

- **choices**: 列表选项, 在某些 **type** 下可用, 并且可以包含一个分隔符(**separator**)



APP内打开

@掘金技术社区



```
warbler@DESKTOP-57C47H7 D: > myGithub > warbler-grow > 架构师课程练习 > 脚手架练习 > test-cli-0174
└─main
> test-cli
```

@稀土掘金技术社区

- **pageSize:** 修改某些 **type** 类型下的渲染行数

如果设置了 **pageSize**，剩下的选项会隐藏，并且通过上下键进行切换。

```
warbler@DESKTOP-57C47H7 D: > myGithub > warbler-grow > 架构师课程练习 > 脚手架练习 > test-cli-0174
└─main
>
```

@稀土掘金技术社区

rawlist

APP内打开

这个和 **list** 差不多，只不过多了一个可以通过输入索引来进行选择的功能，当然也可以通过上下按



```
1 inquirer
2   .prompt([
3     {
4       type: 'rawlist',
5       message: '请选择你的老婆',
6       name: 'wife',
7       default: 0,
8       choices: [
9         {
10          value: 1,
11          name: '赵今麦',
12        },
13        {
14          value: 2,
15          name: '谭松韵',
16        },
17        {
18          value: 3,
19          name: '毛晓彤',
20        },
21        {
22          value: 4,
23          name: '新垣结衣',
24        },
25      ],
26    },
27  ])
28  .then((answers) => {
29    console.log('🚀🚀 ~ answers', answers);
30  })
31  .catch((error) => {
32    console.log('🚀🚀 ~ error', error);
33  });
```

APP内打开

expand

这个也和 `list` 差不多，提供简写，但是语法有一些差别。输入 `h` 可以查看所有选项。



js 复制代码

```
1 inquirer
2   .prompt([
3     {
4       type: 'expand',
5       message: '请选择一个颜色',
6       name: 'color',
7       default: 'R',
8       choices: [
9         {
10          key: 'R',
11          value: 'red',
12        },
13        {
14          key: 'G',
15          value: 'green',
16        },
17        {
18          key: 'B',
19          value: 'blue',
20        },
21      ],
```

APP内打开



```
26    })
27    .catch((error) => {
28      console.log('🚀 ~ error', error);
29    });
```

```
warbler@DESKTOP-57C47H7 D: > myGithub > warbler-grow > 架构师课程练习 > 脚手架练习 > test-cli-0174
main
> test-cli
```

@稀土掘金技术社区

checkbox

多选交互，通过 **空格** 进行选中操作，**回车** 进行确认操作，通过给选项添加 **checked: true** 来进行默认选中的操作。通过 **a** 进行全选，通过 **i** 进行反选。



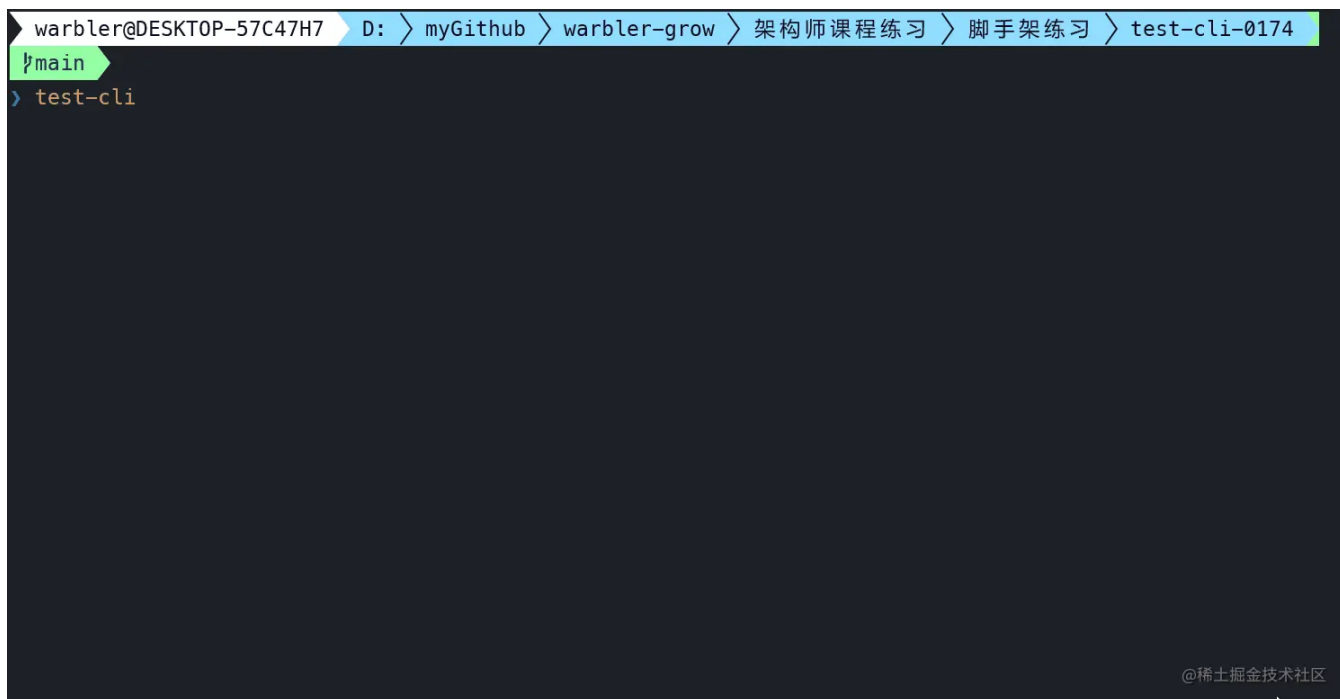
js 复制代码

```
1 inquirer
2   .prompt([
3     {
4       type: 'checkbox',
5       message: '请选择你的老婆们:',
6       name: 'wives',
7       choices: [
8         {
9           value: 1,
10          name: '赵今麦',
11          checked: true,
12        },
```

APP内打开



```
17     },
18     value: 3,
19     name: '毛晓彤',
20   },
21   {
22     value: 4,
23     name: '新垣结衣',
24   },
25 ],
26 },
27 ])
28 .then((answers) => {
29   console.log('🚀🚀 ~ answers', answers);
30 })
31 .catch((error) => {
32   console.log('🚀🚀 ~ error', error);
33 });
```



```
warbler@DESKTOP-57C47H7 D: > myGithub > warbler-grow > 架构师课程练习 > 脚手架练习 > test-cli-0174
main
> test-cli
```

@稀土掘金技术社区

password

密码交互，可以隐藏用户输入的内容，通过 `mask` 属性来修改显示方式，一种是 隐藏 输入，一种是用 `*` 代替输入。

[APP内打开](#)[js 复制代码](#) 33 8 收藏


```
3    {
4      type: 'password',
5      message: '请输入密码1:',
6      name: 'password1',
7    },
8    {
9      type: 'password',
10     message: '请输入密码2:',
11     name: 'password2',
12     mask: true,
13   },
14 ]
15 .then((answers) => {
16   console.log('🚀🚀 ~ answers', answers);
17 })
18 .catch((error) => {
19   console.log('🚀🚀 ~ error', error);
20 });
```

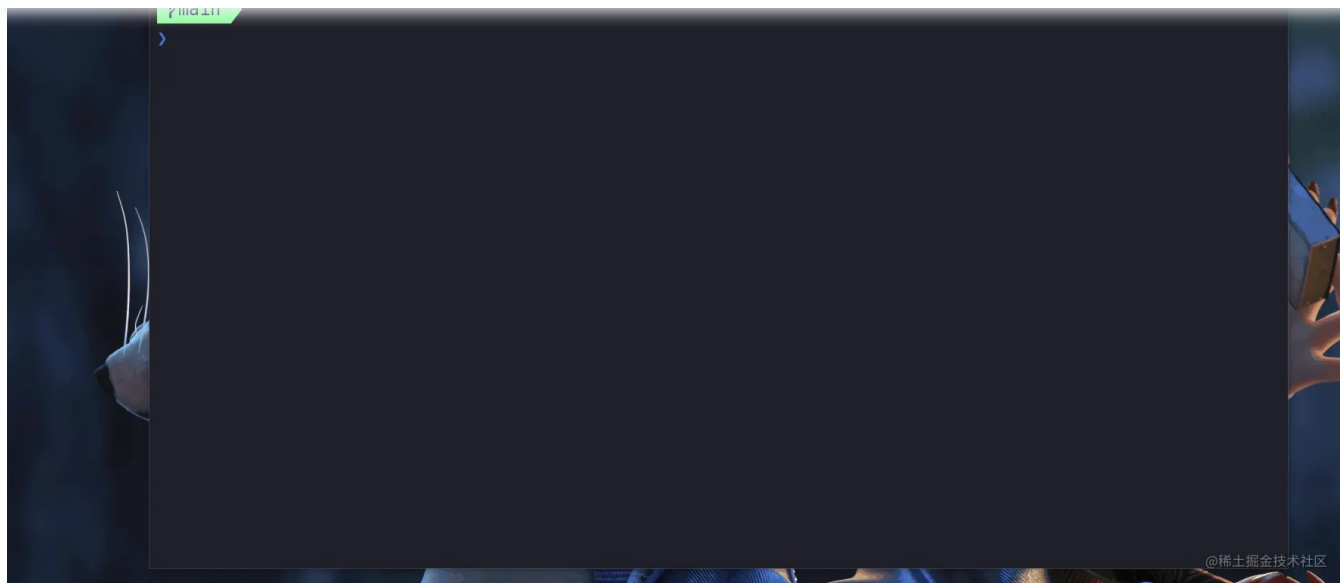
```
warbler@DESKTOP-57C47H7 D: > myGithub > warbler-grow > 架构师课程练习 > 脚手架练习 > test-cli-0174
└─main
> test-cli
```

@稀土掘金技术社区

editor

打开一个文本编辑器，用来输入一些复杂的内容。这里我们打开编辑器。

APP内打开



@稀土掘金技术社区

when

最后来学习一下 `when` 属性的用法。

`when` 的回调参数是前面所有问题的答案，只有返回值为 `true` 的时候，才会出现 `when` 类型的交互。



js 复制代码

```
1 inquirer
2   .prompt([
3     {
4       type: 'confirm',
5       message: '是否单身? ',
6       name: 'single',
7       default: true,
8     },
9     {
10      type: 'confirm',
11      message: '是否需要女朋友? ',
12      name: 'girl',
13      when: function (answers) {
14        // 当 single 为 true 的时候才会提问当前问题
15        return answers.single;
16      },
17    },
18  ])
19   .then((answers) => {
```

APP内打开

 33 8 收藏



```
warbler@DESKTOP-57C47H7 D: > myGithub > warbler-grow > 架构师课程练习 > 脚手架练习 > test-cli-0174
main
> test-cli
```

@稀土掘金技术社区

`inquirer` 的用法还是比较简单的，到这里基本就可以满足日常开发中大多数的交互需求了。

标签： 前端 架构 前端框架

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...

已订阅

相关小册



Nuxt 3.0 全栈开发

杨村长 LV.6

¥129

863购买



趣学 Node.js

死月 LV.3

APP内打开

2611购买

👍 33

💬 8

★ 收藏

输入评论 (Enter换行, Ctrl + Enter发送)

全部评论 8

🕒 最新

🔥 最热



懒懒子  

12月前

你好，我在我的脚手架里面用`const Inquirer = require("inquirer");`引入inquirer会报错，inquirer的版本是9.0.0， 报的错的意思就是ES模块(我看inquirer包是以es方式导出的)不能用require的形式引入，但是为什么你的不会报错呀？希望能帮忙解答一下

👍 点赞 💬 2



一尾流莺 

12月前

看一下你那个版本的源码,如果是ES module ,就降低版本试试。如果没有处理的话，在commonjs模块不能使用ES module 的第三方库。

👍 点赞 💬 回复



爱吃零食的猫

10月前

官方有说这个问题，你需要降低版本到8.0

👍 点赞 💬 回复



戴欧巴  

1年前

then写成了when
分隔符是咋设置的没看到呢

👍 点赞 💬 2



一尾流莺 

1年前

就是when。。。一个错了可能是失误，全都错了可能就是没错了。

👍 点赞 💬 回复

APP内打开



👍 点赞 💬 回复



csdn来挖墙脚 LV.4 JY.5

1年前

见证架构师的诞生

👍 点赞 💬 1



一尾流莺 🔒

1年前

可以,必须见证

👍 1 💬 回复

相关推荐

执着的烙印 4月前

升级node版本报错

👁 61 👍 点赞 💬 评论

某时橙 2年前

nodemon的安装与使用

👁 2119 👍 3 💬 评论

su_rm_rf 3月前

React全家桶 + TS + 后端Koa Server + MongoDB - 全栈实战（一）

👁 114 👍 点赞 💬 评论

MrGaoGang 2年前

推荐：vscode编写typescript的两个插件

👁 8048 👍 11 💬 评论

前端啊 4年前

面试官：请简述一下vue-cli命令行工具，你能自己手写一个吗？

👁 8376 👍 145 💬 36

counterxing 5年前

APP内打开

👍 33

💬 8

★ 收藏

CacheMei 3年前

TypeScript 优点总结

👁 263 🍏 点赞 💬 评论

| 囧 | 2月前

当你在.d.ts文件中写declare global 声明全局类型无效时,可以进来看看,也许会解决你的问题

👁 226 🍏 3 💬 2

西二在线 4年前

基于koa实现一个装饰器风格的框架

👁 1762 🍏 15 💬 3

LuckDay 4年前

vue-cli3.0 多页面配置

👁 2.1w 🍏 65 💬 14

Neal_yang 5年前

实战react技术栈+express前后端博客项目（5）-- 前后端实现登录功能

👁 1771 🍏 38 💬 1

前端_小姜 3月前

配置package.json中scripts命令脚本

👁 765 🍏 1 💬 1

小胖子学习笔记 3年前

Node入门---后台服务与访问接口

👁 2422 🍏 1 💬 评论

Bug终结者_ 2月前

Mockjs模拟接口实现增删改查、分页、多条件查询

👁 50 🍏 点赞 💬 评论

APP内打开