

28

今日头条

转发

微博

Qzone

微信

首页 / 技术 / 正文

搜索站内资讯、视频或用户

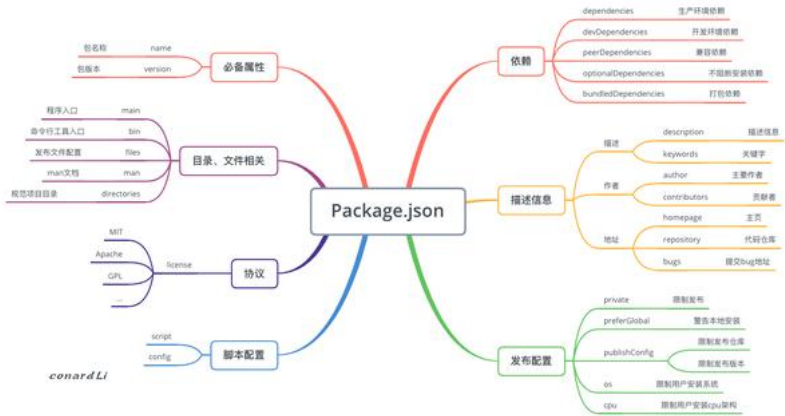
Echa攻城狮

+关注

让你减少加班的15条高效JS【实践】

详细讲解移动端开发的屏幕、布局的兼容适配【图文】

手把手教你深入讲解前端缓存最推荐7个很棒的JavaScript产品



在 Node.js 中，模块是一个库或框架，也是一个 Node.js 项目。Node.js 项目遵循模块化的架构，当我们创建了一个 Node.js 项目，意味着创建了一个模块，这个模块必须有一个描述文件，即 package.json。它是我们最常见的配置文件，但是它里面的配置你真的有详细了解过吗？配置一个合理的 package.json 文件直接决定着我们项目的质量，本章就带大家了解下 package.json 的各项详细配置。

#必备属性

package.json 中有非常多的属性，其中必须填写的只有两个：name 和 version，这两个属性组成一个 npm 模块的唯一标识。

其中版本我们在上一章已经介绍过了，有兴趣的可以查看 前端工程化（一）NPM如何管理依赖包版本？

#npm包命名规则

name 即模块名称，其命名时需要遵循官方的一些规范和建议：

- 包名会成为模块url、命令行中的一个参数或者一个文件夹名称，任何非url安全的字符在包名中都不能使用，可以使用 validate-npm-package-name 包来检测包名是否合法。
- 语义化包名，可以帮助开发者更快的找到需要的包，并且避免意外获取错误的包。
- 若包名称中存在一些符号，将符号去除后不得与现有包名重复

例如：由于react-native已经存在，react.native、reactnative都不要再创建。

- 如果你的包名与现有的包名太相近导致你不能发布这个包，那么推荐将这个包发布到你的作用域下。

例如：用户名 conard，那么作用域为 @conard，发布的包可以是@conard/react。

#查看包是否被占用

 28 转发 微博 Qzone 微信

name 是一个包的唯一标识，不得和其他包名重复，我们可以执行 `npm view packageName` 查看包是否被占用，并可以查看它的一些基本信息：

```
→ Desktop npm view conard
conard@1.1.8 | ISC | deps: 8 | versions: 18
https://github.com/ConardLi/awesome-cli#readme

bin: conard

dist
  .tarball: http://bnpm.byted.org/conard/download/conard-1.1.8.tgz
  .shasum: 20ad78ff24d2c9e69160961b71af56ede562c734

dependencies:
c-complexity: ^1.0.3    chalk: ^2.4.2    signal: ^1.4.0    update-notifier: ^3.0.1
c-line-sys: ^1.0.5    cli-spinner: ^0.2.10    table: ^5.4.6    yargs: ^14.0.0

maintainers:
- conardli <1009903985@qq.com>

dist-tags:
latest: 1.1.8

published a month ago by conardli <1009903985@qq.com>
```

若包名称从未被使用过，则会抛出 404 错误：

```
→ Desktop npm view react-2019
npm ERR! code E404
npm ERR! 404 Not found : react-2019
npm ERR! 404
npm ERR! 404 'react-2019' is not in the npm registry.
npm ERR! 404 You should bug the author to publish it (or use the name yourself!)
npm ERR! 404
npm ERR! 404 Note that you can also install from a
npm ERR! 404 tarball, folder, http url, or git url.

npm ERR! A complete log of this run can be found in:
npm ERR!   /Users/lishiqi/.npm/_logs/2019-11-22T12_24_27_170Z-debug.log
```

另外，你还可以去 <https://www.npmjs.com/> 查询更多更详细的包信息。

#描述信息

#基本描述

```
{
  "description": "An enterprise-class UI design language and React
components implementation",
  "keywords": [
    "ant",
    "component",
    "components",
    "design",
    "framework",
    "frontend",
    "react",
    "react-component",
    "ui"
  ]
}
```

description用于添加模块的描述信息，方便别人了解你的模块。

keywords用于给你的模块添加关键字。

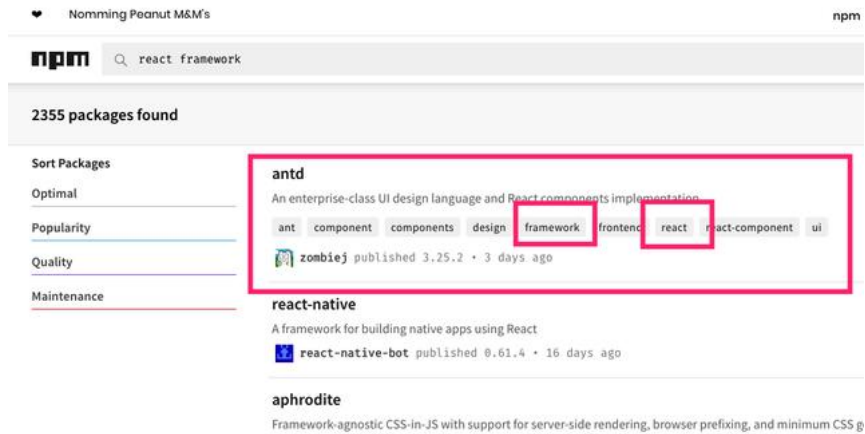
当然，他们的还有一个非常重要的作用，就是利于模块检索。当你使用 `npm search` 检索模块时，会到description 和 keywords 中进行匹配。写好 description 和 keywords 有利于你的模块获得更多更精准的曝光：

 28

 转发

 微博

 Qzone

 微信


#开发人员

描述开发人员的字段有两个：author 和 contributors，author 指包的主要作者，一个 author 对应一个人。contributors 指贡献者信息，一个 contributors 对应多个贡献者，值为数组，对人的描述可以是一个字符串，也可以是下面的结构：

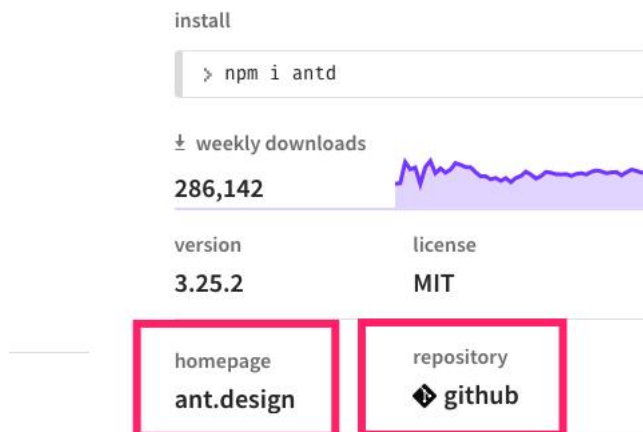
```
{
  "name": "ConardLi",
  "email": "lisqPersion@163.com",
  "url": "https://github.com/ConardLi"
}
```

#地址

```
{
  "homepage": "http://ant.design/",
  "bugs": {
    "url": "https://github.com/ant-design/ant-design/issues"
  },
  "repository": {
    "type": "git",
    "url": "https://github.com/ant-design/ant-design"
  },
}
```

homepage 用于指定该模块的主页。

repository 用于指定模块的代码仓库。



bugs 指定一个地址或者一个邮箱，对你的模块存在疑问的人可以到这里提出问题。



#依赖配置

我们的项目可能依赖一个或多个外部依赖包，根据依赖包的不同用途，我们将他们配置在下面几个属性下：dependencies、devDependencies、peerDependencies、bundledDependencies、optionalDependencies。

#配置规则

在介绍几种依赖配置之前，首先我们来看一下依赖的配置规则，你看到的依赖包配置可能是下面这样的：

```
"dependencies": {
  "antd": "ant-design/ant-design#4.0.0-alpha.8",
  "axios": "^1.2.0",
  "test-js": "file:../test",
  "test2-js": "http://cdn.com/test2-js.tar.gz",
  "core-js": "^1.1.5",
}
```

依赖配置遵循下面几种配置规则：

- 依赖包名称:VERSIONVERSION是一个遵循SemVer规范的版本号配置，npm install 时将要到npm服务器下载符合指定版本范围的包。
- 依赖包名称:DWONLOAD_URLDWONLOAD_URL 是一个可下载的tarball压缩包地址，模块安装时会把这个.tar下载并安装到本地。
- 依赖包名称:LOCAL_PATHLOCAL_PATH 是一个本地的依赖包路径，例如 file:../pacakges/pkgName。适用于你在本地测试一个npm包，不应该将这种方法应用于线上。
- 依赖包名称:GITHUB_URLGITHUB_URL 即 github 的 username/modulename 的写法，例如：ant-design/ant-design，你还可以在后面指定 tag 和 commit id。
- 依赖包名称:GIT_URLGIT_URL 即我们平时clone代码库的 git url，其遵循以下形式：

```
<protocol>://[<user>[:<password>]@]<hostname>[:<port>][:/]<path>[#<commit-ish> | #semver:<semver>]
```

其中 protocol 可以是以下几种形式：

- git://github.com/user/project.git#commit-ish
- git+ssh://user@hostname:project.git#commit-ish
- git+ssh://user@hostname/project.git#commit-ish
- git+http://user@hostname/project/blah.git#commit-ish
- git+https://user@hostname/project/blah.git#commit-ish

#dependencies

dependencies 指定了项目运行所依赖的模块，开发环境和生产环境的依赖模块都可以配置到这里，例如

```
"dependencies": {
  "lodash": "^4.17.13",
  "moment": "^2.24.0",
}
```

#devDependencies

 28 转发 微博 Qzone 微信

有一些包有可能你只是在开发环境中用到，例如你用于检测代码规范的 `eslint`，用于进行测试的 `jest`，用户使用你的包时即使不安装这些依赖也可以正常运行，反而安装他们会耗费更多的时间和资源，所以你可以把这些依赖添加到 `devDependencies` 中，这些依赖照样会在你本地进行 `npm install` 时被安装和管理，但是不会被安装到生产环境：

```
"devDependencies": {
  "jest": "^24.3.1",
  "eslint": "^6.1.0",
}
```

#peerDependencies

`peerDependencies` 用于指定你正在开发的模块所依赖的版本以及用户安装的依赖包版本的兼容性。

上面的说法可能有点太抽象，我们直接拿 `ant-design` 来举个例子，`ant-design` 的 `package.json` 中有如下配置：

```
"peerDependencies": {
  "react": ">=16.0.0",
  "react-dom": ">=16.0.0"
}
```

当你正在开发一个系统，使用了 `ant-design`，所以也肯定需要依赖 `React`。同时，`ant-design` 也是需要依赖 `React` 的，它要保持稳定运行所需要的 `React` 版本是 `16.0.0`，而你开发时依赖的 `React` 版本是 `15.x`：

这时，`ant-design` 要使用 `React`，并将其引入：

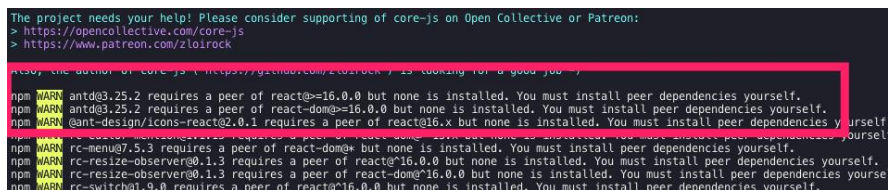
```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
```

这时取到的是宿主环境也就是你的环境中的 `React` 版本，这就可能造成一些问题。在 `npm2` 的时候，指定上面的 `peerDependencies` 将意味着强制宿主环境安装 `react@>=16.0.0` 和 `react-dom@>=16.0.0` 的版本。

`npm3` 以后不会再要求 `peerDependencies` 所指定的依赖包被强制安装，相反 `npm3` 会在安装结束后检查本次安装是否正确，如果不正确会给用户打印警告提示。

```
"dependencies": {
  "react": "15.6.0",
  "antd": "^3.22.0"
}
```

例如，我在项目中依赖了 `antd` 的最新版本，然后依赖了 `react` 的 `15.6.0` 版本，在进行依赖安装时将给出以下警告：



```
The project needs your help! Please consider supporting of core-js on Open Collective or Patreon:
> https://opencollective.com/core-js
> https://www.patreon.com/zloirock

antd, the owner of core-js (https://github.com/zloirock) is looking for a good job :)

npm WARN antd@3.25.2 requires a peer of react-dom@>=16.0.0 but none is installed. You must install peer dependencies yourself.
npm WARN antd@3.25.2 requires a peer of react-dom@>=16.0.0 but none is installed. You must install peer dependencies yourself.
npm WARN ant-design/icons-react@2.0.1 requires a peer of react@16.x but none is installed. You must install peer dependencies yourself.
npm WARN rc-menu@7.5.3 requires a peer of react-dom@>=16.0.0 but none is installed. You must install peer dependencies yourself.
npm WARN rc-resize-observer@0.1.3 requires a peer of react-dom@>=16.0.0 but none is installed. You must install peer dependencies yourself.
npm WARN rc-resize-observer@0.1.3 requires a peer of react-dom@>=16.0.0 but none is installed. You must install peer dependencies yourself.
npm WARN rc-switch@1.9.0 requires a peer of react@>=16.0.0 but none is installed. You must install peer dependencies yourself.
```

#optionalDependencies

某些场景下，依赖包可能不是强依赖的，这个依赖包的功能可有可无，当这个依赖包无法被获取到时，你希望 `npm install` 继续运行，而不会导致失败，你可以将这个依赖放到

 28 转发 微博 Qzone 微信

optionalDependencies 中，注意 optionalDependencies 中的配置将会覆盖掉 dependencies 所以只需在一个地方进行配置。

当然，引用 optionalDependencies 中安装的依赖时，一定要做好异常处理，否则在模块获取不到时会导致报错。

#bundledDependencies

和以上几个不同，bundledDependencies 的值是一个数组，数组里可以指定一些模块，这些模块将在这个包发布时被一起打包。

```
"bundledDependencies": ["package1", "package2"]
```

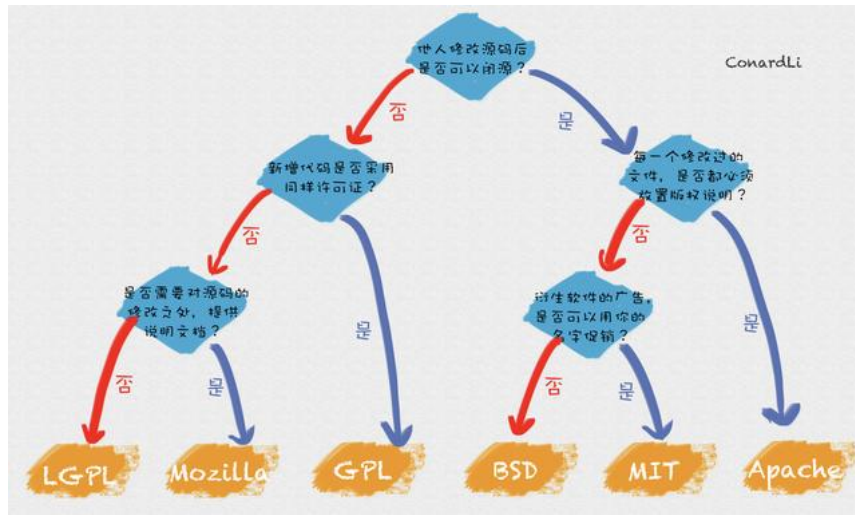
#协议

```
{  
  "license": "MIT"  
}
```

license 字段用于指定软件的开源协议，开源协议里面详尽表述了其他人获得你代码后拥有的权利，可以对你的代码进行何种操作，何种操作又是被禁止的。同一款协议有很多变种，协议太宽松会导致作者丧失对作品的很多权利，太严格又不便于使用者使用及作品的传播，所以开源作者要考虑自己对作品想保留哪些权利，放开哪些限制。


软件协议可分为开源和商业两类，对于商业协议，或者叫法律声明、许可协议，每个软件会有自己的一套行文，由软件作者或专门律师撰写，对于大多数人来说不必自己花时间和精力去写繁长的许可协议，选择一份广为流传的开源协议就是个不错的选择。

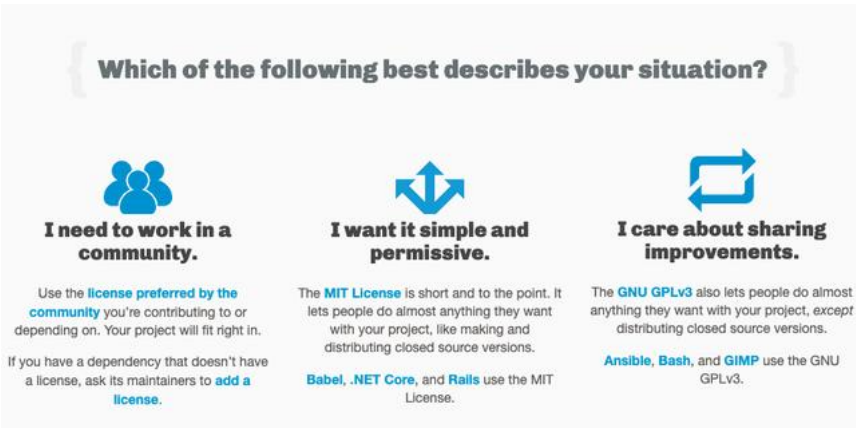
以下几种就是几种主流的开源协议：



- MIT：只要用户在项目副本中包含了版权声明和许可声明，他们就可以拿你的代码做任何想做的事情，你也无需承担任何责任。
- Apache：类似于 MIT，同时还包含了贡献者向用户提供专利授权相关的条款。
- GPL：修改项目代码的用户再次分发源码或二进制代码时，必须公布他的相关修改。

如果你对开源协议有更详细的要求，可以到 <https://choosealicense.com/> 获取更详细的开源协议说明。

-  28
-  转发
-  微博
-  Qzone
-  微信

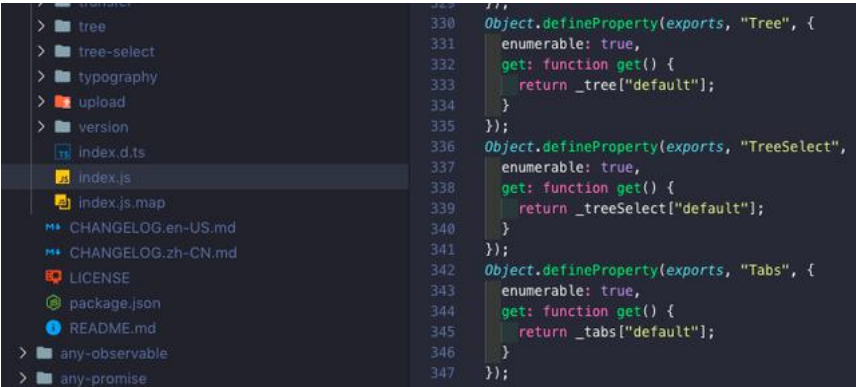


#目录、文件相关

#程序入口

```
{
  "main": "lib/index.js",
}
```

main 属性可以指定程序的主入口文件，例如，上面 antd 指定的模块入口 lib/index.js，当我们在代码引入 antd 时: import { notification } from 'antd'; 实际上引入的就是 lib/index.js 中暴露出去的模块。



#命令行工具入口

当你的模块是一个命令行工具时，你需要为命令行工具指定一个入口，即指定你的命令名称和本地可指定文件的对应关系。如果是全局安装，npm 将会使用符号链接把可执行文件链接到 /usr/local/bin，如果是本地安装，会链接到 ./node_modules/.bin/。

```
{
  "bin": {
    "conard": "./bin/index.js"
  }
}
```

例如上面的配置：当你的包安装到全局时：npm 会在 /usr/local/bin下创建一个以 conard 为名字的软链接，指向全局安装下来的 conard 包下面的 "./bin/index.js"。这时你在命令行执行 conard 则会调用链接到的这个js文件。

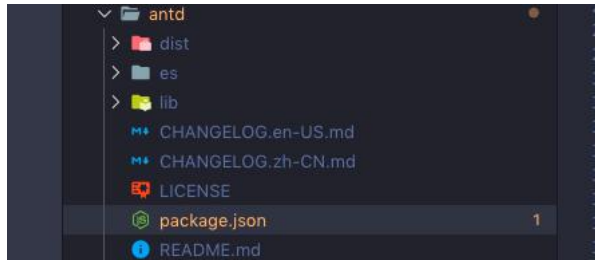
这里不再过多展开，更多内容在我后续的命令行工具文章中会进行详细讲解。

#发布文件配置

 28 转发 微博 Qzone 微信

```
{
  "files": [
    "dist",
    "lib",
    "es"
  ]
}
```

files 属性用于描述你 npm publish 后推送到 npm 服务器的文件列表，如果指定文件夹，则文件夹内的所有内容都会包含进来。我们可以看到下载后的包是下面的目录结构：



另外，你还可以通过配置一个 .npmignore 文件来排除一些文件，防止大量的垃圾文件推送到 npm，规则上和你用的 .gitignore 是一样的。.gitignore 文件也可以充当.npmignore 文件。

#man

man 命令是 Linux 下的帮助指令，通过 man 指令可以查看 Linux 中的指令帮助、配置文件帮助和编程帮助等信息。

如果你的 node.js 模块是一个全局的命令行工具，在 package.json 通过 man 属性可以指定 man 命令查找的文档地址。

man 文件必须以数字结尾，或者如果被压缩了，以 .gz 结尾。数字表示文件将被安装到 man 的哪个部分。如果 man 文件名称不是以模块名称开头的，安装的时候会给加上模块名称前缀。

例如下面这段配置：

```
{
  "man" : [
    "/Users/isaacs/dev/npm/cli/man/man1/npm-access.1",
    "/Users/isaacs/dev/npm/cli/man/man1/npm-audit.1"
  ]
}
```

在命令行输入 man npm-audit：

```
NAME
  npm-audit - Run a security audit

SYNOPSIS
  npm audit [--json|--parseable]
  npm audit fix [--force|--package-lock-only|--dry-run|--production|--only=dev]

EXAMPLES
  Scan your project for vulnerabilities and automatically install any compatible updates to vulnerable dependencies:

    $ npm audit fix

  Run audit fix without modifying node_modules, but still updating the pkglock:

    $ npm audit fix --package-lock-only

  Skip updating devDependencies:

    $ npm audit fix --only=prod
```

#规范项目目录



转发



微博



Qzone



微信

一个 node.js 模块是基于 CommonJS 模块化规范实现的，严格按照 CommonJS 规范，模块目录下除了必须包含包描述文件 package.json 以外，还需要包含以下目录：

- bin：存放可执行二进制文件的目录
- lib：存放js代码的目录
- doc：存放文档的目录
- test：存放单元测试用例代码的目录
- ...

在模块目录中你可能没有严格按照以上结构组织或命名，你可以通过在 package.json 指定 directories 属性来指定你的目录结构和上述的规范结构的对应情况。除此之外 directories 属性暂时没有其他应用。

```
{
  "directories": {
    "lib": "src/lib/",
    "bin": "src/bin/",
    "man": "src/man/",
    "doc": "src/doc/",
    "example": "src/example/"
  }
}
```

不过官方文档表示，虽然目前这个属性没有什么重要作用，未来可能会整出一些花样出来，例如：doc 中存放的 markdown 文件、example 中存放的示例文件，可能会友好的展示出来。

#脚本配置

#script

```
{
  "scripts": {
    "test": "jest --config .jest.js --no-cache",
    "dist": "antd-tools run dist",
    "compile": "antd-tools run compile",
    "build": "npm run compile && npm run dist"
  }
}
```

scripts 用于配置一些脚本命令的缩写，各个脚本可以互相组合使用，这些脚本可以覆盖整个项目的生命周期，配置后可使用 npm run command 进行调用。如果是 npm 关键字，则可以直接调用。例如，上面的配置制定了以下几个命令：npm run test、npm run dist、npm run compile、npm run build。

#config

config 字段用于配置脚本中使用的环境变量，例如下面的配置，可以在脚本中使用 process.env.npm_package_config_port 进行获取。

```
{
  "config": { "port": "8080" }
}
```

#发布配置



28



转发



微博



Qzone



微信

#preferGlobal

如果你的 node.js 模块主要用于安装到全局的命令行工具，那么该值设置为 true，当用户将该模块安装到本地时，将得到一个警告。这个配置并不会阻止用户安装，而是会提示用户防止错误使用而引发一些问题。

#private

如果将 private 属性设置为 true，npm 将拒绝发布它，这是为了防止一个私有模块被无意间发布出去。

```
npm notice === Tarball Details ===
npm notice name:          conard
npm notice version:       1.1.11
npm notice package size:  2.7 kB
npm notice unpacked size: 7.1 kB
npm notice shasum:        43f5c626d3f649c450bcc26722f47ba2a277c51d
npm notice integrity:      sha512-YwM9s/PVR5MEw[...]gS808BMSbWRvQ==
npm notice total files:   8
npm notice
npm ERR! This package has been marked as private
npm ERR! Remove the 'private' field from the package.json to publish it.
```

#publishConfig

```
"publishConfig": {
  "registry": "https://registry.npmjs.org/"
},
```

发布模块时更详细的配置，例如你可以配置只发布某个 tag、配置发布到的私有 npm 源。更详细的配置可以参考 npm-config

#os

假如你开发了一个模块，只能跑在 darwin 系统下，你需要保证 windows 用户不会安装到你的模块，从而避免发生不必要的错误。

使用 os 属性可以帮助你完成以上的需求，你可以指定你的模块只能被安装在某些系统下，或者指定一个不能安装的系统黑名单：

```
"os" : [ "darwin", "linux" ]
"os" : [ "!win32" ]
```

例如，我把一个测试模块指定一个系统黑名单："os": ["!darwin"]，当我在此系统下安装它时会爆出如下错误：

```
npm ERR! code EBADPLATFORM
npm ERR! notsup Unsupported platform for conard-test@1.0.1: wanted {"os": "!darwin", "arch": "any"} (current: {"os": "darwin", "arch": "x64"})
npm ERR! notsup Valid OS:   !darwin
npm ERR! notsup Valid Arch:  any
npm ERR! notsup Actual OS:   darwin
npm ERR! notsup Actual Arch: x64

npm ERR! A complete log of this run can be found in:
npm ERR!   /Users/Lishiqi/.npm/_logs/2019-11-28T12_17_51_779Z-debug.log
```


在node环境下可以使用 process.platform 来判断操作系统。

#cpu

和上面的 os 类似，我们可以用 cpu 属性更精准的限制用户安装环境：


```
"cpu" : [ "x64", "ia32" ]
"cpu" : [ "!arm", "!mips" ]
```

在node环境下可以使用 process.arch 来判断 cpu 架构。

 28

 转发

 微博

 Qzone

 微信

#参考

http://caibaojian.com/npm/files/package.json.html

☆ 收藏 ⓘ 举报

28 条评论



写下您的评论...

评论

-  Echa攻城狮 20天前
转发了
[回复](#) 0  ⓘ
-  pangpangjd 19天前
转发了
[回复 · 1条回复](#) 1  ⓘ
-  blanks299 20天前
转发了
[回复 · 1条回复](#) 0  ⓘ
-  cyq一百三十七分之一 20天前
转发了
[回复 · 1条回复](#) 0  ⓘ
-  -青梅爱上竹马 20天前
转发了
[回复 · 1条回复](#) 0  ⓘ

[查看更多评论](#)

相关推荐

- 

GitHub 十大顶级 JavaScript 开源项目

CSDN · 16评论 · [相关](#)

×
- 

「前端进阶」高性能渲染十万条数据(虚拟列表)

Echa攻城狮 · 47评论 · [相关](#)

×
- 

Vue源码中9个可借鉴的基础方法

web秀 · 31评论 · [相关](#)

×
- 

2020年了,再不会webpack敲得代码就不香了(近万字实战)

Echa攻城狮 · 107评论 · [相关](#)

×
- 

2020年前端三大顶级技术趋势是什么?


前端达人 · 19评论 · [相关](#)

×

 28



 转发

 微博

 Qzone

 微信

	<div>使用IntelliJ IDEA查看类的继承关系图形</div> <div> · 相关</div> <div>最详细的git教程</div> <div>代码接盘侠 · 78评论 · 相关</div>	×
	<div>轻量级Javascript全文搜索库——Lunr.js</div> <div>最美分享Coder · 29评论 · 相关</div>	×
	<div>面试官：聊聊对Vue.js框架的理解</div> <div>源码时代 · 27评论 · 相关</div>	×
	<div>InfoQ 2020 年 JavaScript 和 Web 开发趋势报告</div> <div>InfoQ · 8评论 · 相关</div>	×
	<div>何时使用 Map 来代替普通的 JS 对象</div> <div>前端小智 · 8评论 · 相关</div>	×
	<div>你值得拥有的 11 个前端开发利器</div> <div>CSDN · 34评论 · 相关</div>	×
	<div>看完这篇，再也不用焦虑如何写dockerfile了</div> <div>追逐仰望星空 · 41评论 · 相关</div>	×
	<div>手把手教你前端的各种文件上传攻略和大文件断点续传</div> <div>Echa攻城狮 · 73评论 · 相关</div>	×
	<div>生产环境关闭Swagger2</div> <div>Java实用技术 · 32评论 · 相关</div>	×
	<div>Git最全总结</div> <div>IT技术圈 · 14评论 · 相关</div>	×
	<div>用Vue和React构建相同应用程序，区别在哪？</div> <div>源码时代 · 47评论 · 相关</div>	×
	<div>ES6 完全使用手册附加案例实战讲解</div> <div>Echa攻城狮 · 49评论 · 相关</div>	×

-  28
-  转发
-  微博
-  Qzone
-  微信

推荐一个神器，MapStruct，你用过吗？

Java识堂 · 24评论 · [相关](#)

Vue.js + element-ui 扫盲(服务端对大前端的扫盲)

Java实用技术 · 57评论 · [相关](#)