

npm

全称: `node package manager`

1. 查询包的信息

<https://registry.npmjs.org/vue/0.7.1> 进行查询, 该地址返回该包的 详细信息(`json`)

2. 发包

1. 新建文件夹 `cd` 进入 `bao`
2. 生成 `package.json`

```
{
  "name": "bao-zhenshuai", // 包的名字  npm install bao-zhenshuai
  "version": "1.0.0", // 版本 1.0.0  大改.新增的功能.小的bug
  "description": "", // 描述
  "main": "index.js", // 入口文件
  "scripts": { // 命令 npm run dev
    "dev": "",
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [], // 搜索包的关键字
  "author": "bingyu", // 作者
  "license": "ISC" // 许可  MIT  ISC
}
```

3. 新建一个文件 `index.js` 入口文件


3. 发布的步骤

1. 去npm 官网注册一个账号 <https://www.npmjs.com/>
2. `npm config set registry` 地址

3. npm login 输入用户名密码邮箱

```
bogon:bao sairitsutakara$ npm login
Username: bingyu~
Password:
Email: (this IS public) cuilibao123@gmail.com
Logged in as bingyu~ on https://registry.npmjs.org/.
bogon:bao sairitsutakara$
```

4. npm publish

```
bogon:bao sairitsutakara$ npm publish
npm notice
npm notice  bao@1.0.0
npm notice === Tarball Contents ===
npm notice 41B index.js
npm notice 229B package.json
npm notice === Tarball Details ===
npm notice name:          bao
npm notice version:       1.0.0
npm notice package size:  352 B
npm notice unpacked size: 270 B
npm notice shasum:        4e27bd6c6db5b690e21e87020a9440f8000f6ff7
npm notice integrity:     sha512-uzrwnjAySobJx[...]7FVnzt04rD/Cw==
npm notice total files:   2
npm notice
npm ERR! code E401
npm ERR! 401 Unauthorized - PUT https://registry.npm.taobao.org/bao -
[unauthorized] Login first

npm ERR! A complete log of this run can be found in:
npm ERR!     /Users/sairitsutakara/.npm/_logs/2020-07-23T02_09_21_086Z-
debug.log
```

403: 包名重读 / 邮箱没有验证

如果出现上述的错误 说明设置了淘宝镜像

解决方案: 将镜像设置回初始镜像

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

如果下载第三方包 再改回到淘宝镜像

包的安装的时候都发生了什么

npm i 包

- 找到下载地址：<https://registry.npmjs.org/vue/0.7.1>

```
"dist": {
  "shasum": "17a6ea20a5660c8614636387e15521530ff48c50",
  "tarball": "https://registry.npmjs.org/vue/-/vue-0.7.1.tgz"
},
```

- 在 `node_modules` 中多了一个文件
- 根据参数 放到对应环境中
 - S 生产环境 -D 开发环境 -g: 全局
- 根据这个包决定是否生成 `.cmd` 的可执行文件

比如 `npx create-react-app`

回忆哪些会生成: webpack babel vue...

- 下载环境的差异

下载到本地 就会在本地的 `node_modules` 生成 `.bin` 目录

下载到全局 通过命令 `npm config get prefix` 查看全局的路径

```
E:\project\classCode\1911A\gj_node\test (master -> origin)
λ npm config get prefix
C:\Users\Administrator\AppData\Roaming\npm
```

- 执行命令的时候 先从本地找 如果没有 再从全局找
- `.cmd` 就是一个可执行程序

练习

@babel/cli @babel/core

下载到本地

执行命令 `npx babel --version`

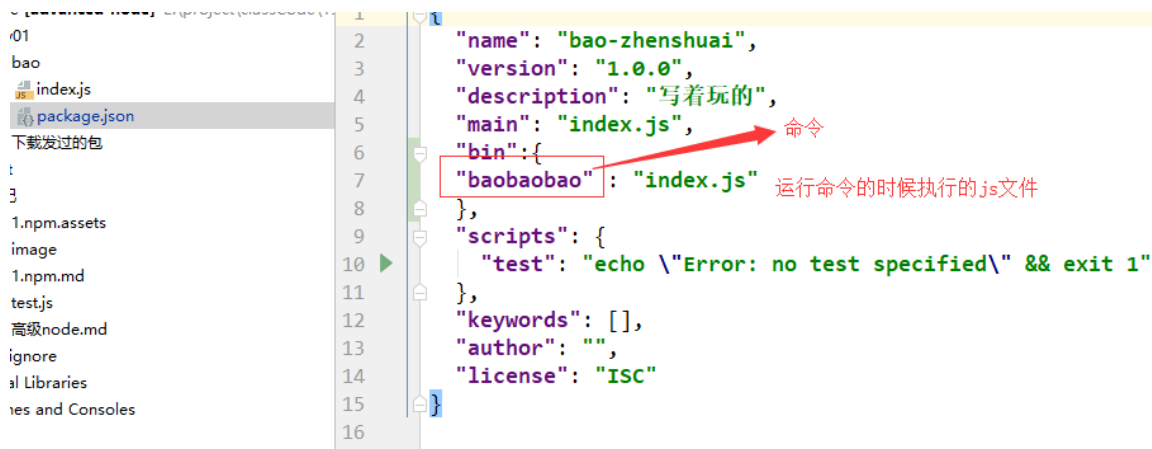
卸载 `npx babel --version`

下载全局 `npx i @babel/cli @babel/core -g`

执行命令 `babel --version`

生成自己的.cmd文件

1. 在package.json中加入bin字段

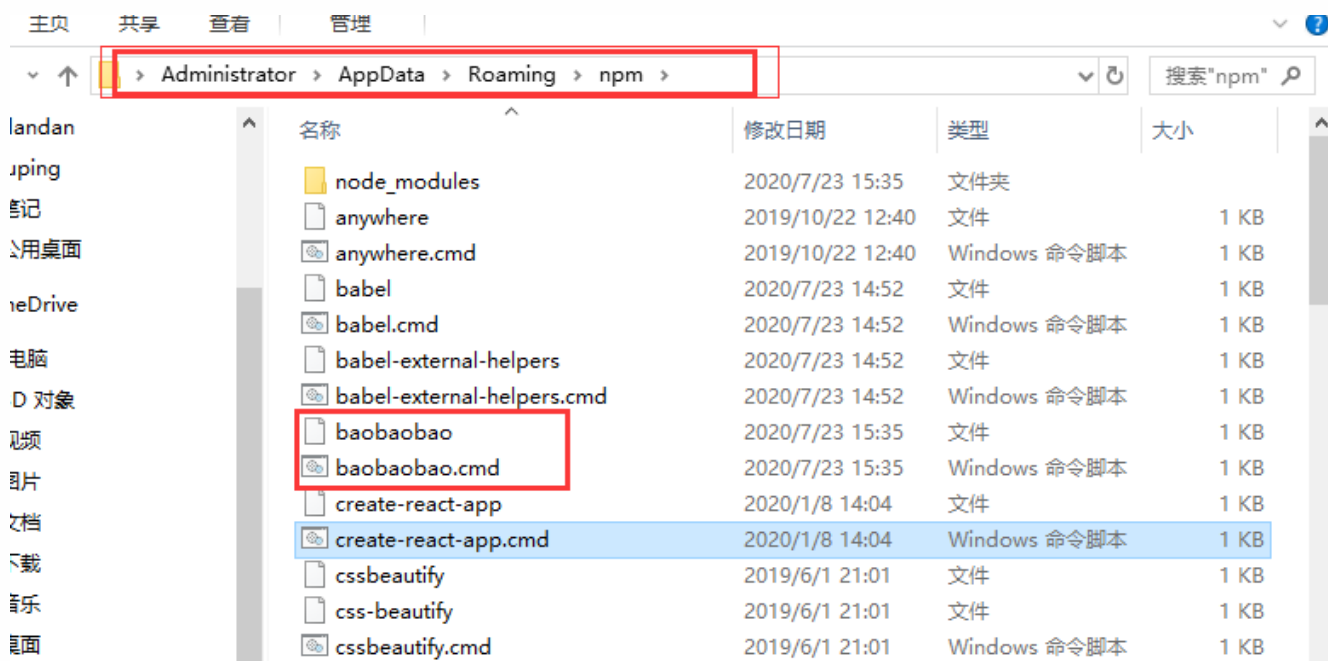


2. 在js文件中最上面添加一句话

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

这句话的意思是在node环境中执行这个文件

3. 通过 `npm link` 这个命令 生成一个全局的cmd文件

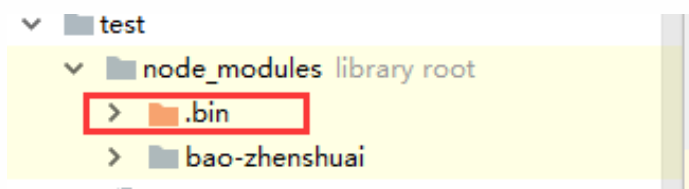


4. 取消全局cmd文件

掌握npm unlink的用法，取消npm link的作用

发包之后别人下载的时候生成.bin文件

- 把淘宝镜像改回来
- npm login
- npm publish
- 别人下载 npm i 包
- 生成.bin文件



附录

package.json是什么

每个项目的根目录下面，一般都有一个package.json文件，定义了这个项目所需要的各种模块，以及项目的配置信息（比如名称、版本、许可证等元数据）。npm install命令根据这个配置文件，自动下载所需的模块，也就是配置项目所需的运行和开发环境。

下面是一个最简单的package.json文件，只定义两项元数据：项目名称和项目版本。

```
{
  "name" : "xxx",
  "version" : "0.0.0",
}
```

package.json文件就是一个**JSON**对象，该对象的每一个成员就是当前项目的一项设置。比如**name**就是项目名称，**version**是版本（遵守“大版本.次要版本.小版本”的格式）。

package.json配置说明

下面就以我的博客项目的package.json文件的配置作一个简单的说明，分为**必须字段**和**可选字段**

```
{
```

```

"name": "wblearn-blog",
"title": "Wblearn Blog",
"author": "Wblearn <1275660493@qq.com>",
"version": "1.7.0",
"homepage": "https://wblearn.github.io",
"repository": {
  "type": "git",
  "url": "https://github.com/wblearn/wblearn.github.io"
},
"bugs": "https://github.com/wblearn/wblearn.github.io/issues",
"devDependencies": {
  "grunt": "~0.4.5",
  "grunt-contrib-less": "~0.11.4",
  "grunt-contrib-watch": "~0.6.1",
  "grunt-banner": "~0.2.3",
  "grunt-contrib-uglify": "~0.5.1"
},
"scripts": {
  "preview": "cd _site; python -m SimpleHTTPServer 8020",
  "py3view": "cd _site; python3 -m http.server 8020",
  "watch" : "grunt watch & npm run preview & jekyll serve -w",
  "py3wa" : "grunt watch & npm run py3view & jekyll serve -w",
  "boil" : "git push boilerplate boilerplate:master",
  "push" : "git push origin master --tag",
  "cafe" : "git co gitcafe-pages; git merge master; git push
gitcafe gitcafe-pages:gitcafe-pages --tag; git co master;"
}
}

```

必须字段

1.Name 项目名称

全部小写，没有空格，可以使用下划线或者横线

2.Version 项目版本号

x.x.x 的格式 符合“语义化版本规则”

可选字段

1.title 标题 **2.author** author是一个人，contributors是一组人。格式设置如下：

```
{ "name" : "wblearn"  
  , "email" : "1275660493@qq.com"  
  , "url" : "[https://wblearn.github.io](https://wblearn.github.io)"  
}
```

也可以像我博客中的格式一样：

```
"author": "Wblearn <1275660493@qq.com>",
```

3.homepage 项目url主页

4.repository 用于指示代码存放的位置。

```
"repository": {  
  "type": "git",  
  "url": "https://github.com/wblearn/wblearn.github.io"  
}
```

```
>"repository": {  
  "type": "svn",  
  "url": "https://github.com/wblearn/wblearn.github.io"  
}
```

5.bugs 问题追踪系统的URL或邮箱地址；npm bugs用的上。比如我的

```
"bugs": "https://github.com/wblearn/wblearn.github.io/issues"
```

6.devDependencies 指定项目开发所需要的模块，如果只需要下载使用某些模块，而不下载这些模块的测试和文档框架，放在这个下面比较不错。

```
"devDependencies": {
  "grunt": "~0.4.5",
  "grunt-contrib-less": "~0.11.4",
  "grunt-contrib-watch": "~0.6.1",
  "grunt-banner": "~0.2.3",
  "grunt-contrib-uglify": "~0.5.1"
}
```

7.scripts **object** **Key**是生命周期事件名，**value**是在事件点要跑的命令。参考npm-scripts。**scripts**指定了运行脚本命令的npm命令行缩写，比如**push**指定了运行**npm run push**时，所要执行的命令。下面的设置指定了**npm run preview**、**npm run watch**、**npm run push**、**npm run cafe**时，所要执行的命令。

```
"scripts": {
  "preview": "cd _site; python -m SimpleHTTPServer 8020",
  "py3view": "cd _site; python3 -m http.server 8020",
  "watch" : "grunt watch & npm run preview & jekyll serve -w",
  "py3wa" : "grunt watch & npm run py3view & jekyll serve -w",
  "boil" : "git push boilerplate boilerplate:master",
  "push" : "git push origin master --tag",
  "cafe" : "git co gitcafe-pages; git merge master; git push
gitcafe gitcafe-pages:gitcafe-pages --tag; git co master;"
}
```

其它字段

1.Dependencies 指示当前包所依赖的其他包。


```
{ "dependencies" :
  { "foo" : "1.0.0 - 2.9999.9999"
  , "bar" : ">=1.0.2 <2.1.2"
  , "baz" : ">1.0.2 <=2.3.4"
  , "boo" : "2.0.1"
  , "qux" : "<1.0.0 || >=2.3.1 <2.4.5 || >=2.5.2 <3.0.0"
  , "asd" : "http://asdf.com/asdf.tar.gz"
  , "til" : "~1.2"
  , "elf" : "~1.2.3"
  , "two" : "2.x"
  , "thr" : "3.3.x"
  }
}
```

版本格式可以是下面任一种：

- `ersion` 完全匹配
- `>version` 大于这个版本
- `>=version` 大于或等于这个版本
- `<version`
- `<=version`
- `~version` 非常接近这个版本
- `^version` 与当前版本兼容
- `1.2.x` X代表任意数字，因此1.2.1, 1.2.3等都可以
- `http://...` Unix系统下使用的tarball的URL。
- `*` 任何版本都可以
- `"` 任何版本都可以
- `version1 - version2` 等价于 `>=version1 <=version2` .
- `range1 || range2` 满足任意一个即可
- `git...` Git地址
- `user/repo`

2.License 授权方式，如果是使用一个普遍的license，比如BSD-3-Clause或MIT，直接使用：

```
{ "license" : "BSD-3-Clause" }
```

3.main main字段指定了加载的入口文件，`require('moduleName')`就会加载这个文件。这个字段的默认值是模块根目录下面的index.js。

4.Config object Config对象中的值在Scripts的整个周期中皆可用，专门用于给Scripts提供配置参数。

5.Keywords 字符串数组。人们使用 `npm search` 搜索时发现你的项目

6.Description 必须是字符串。 `npm search` 的时候会用到。

7.Bin bin项用来指定各个内部命令对应的可执行文件的位置。很多的包都会有执行文件需要安装到PATH中去。这个字段对应的是一个Map，每个元素对应一个{ 命令名: 文件名 }。 `{ "bin" : { "npm" : "./cli.js" } }`

8.Engines engines字段指明了该模块运行的平台，比如 Node 的某个版本或者浏览器 既可以指定node版本： `{ "engines" : { "node" : ">=0.10.3 <0.12" } }` 也可以指定npm版本： `{ "engines" : { "npm" : "~1.0.20" } }`

写在最后

对于学习nodejs或者npm的同学，详细了解package.json的一些字段不管是搭建自己的博客或者项目都有好处，当然，以上只列出package.json文件的部分主要字段，如果还想了解更多，可以参考阮一峰的[package.json文件](#)或者[package.json字段全解](#)。

作业

使用 `npm` 发一个包 包名叫 `自己姓名-algorithm`

实现冒泡排序

实现随机生成一个数组进行查找

下载完包后 使用命令 `algorithm_find` 显示所有方法