(/apps/
utm_sc
banner

# Yarn入门

FConfidence (/u/77dbd75f5ad9) +关注
2017.06.21 10:16* 字数 1271 阅读 6142 评论 4 喜欢 12
(/u/77dbd75f5ad9)

## Yarn

```
## yarn 推荐全局安装
npm install yarn -g
yarn help COMMAND
```

---

## 初始化一个项目

1. **yarn init** 相当 `npm init`

2. **yarn add**添加一个包 相当 `npm install [packageName] --save`

   **yarn add [package] --dev** 相当 `npm install [package] --save-dev`

   yarn add webpack

   yarn add [package]@[version]

   yarn add [package]@[tag]

   // yarn 安装package不推荐全局安装

3. **yarn upgrade**更新一个包 相当 `npm update webpack`

```
yarn upgrade webpack
yarn upgrade [package]@[version]
yarn upgrade [package]@[tag]
```

4. **--offline离线安装** 相当 `npm install [packageName] --save-dev`

```
yarn add browser-sync --offline
browser-sync start --server --files "**"
```

5. **yarn cache**检查本地是否有当前包

```
yarn cache ls
```

6. **yarn remve**移除本地依赖包 相当 `npm uninstall [package]`

```
yarn remove [package]
```

7. **yarn install**安装包依赖 相当 `npm install`

```
yarn 或者 yarn install
```

8. **yarn publish**发布到github上即项目仓库里面

```
yarn publish
```

## Yarn的工作流

1. 创建一个新项目工程

   ```
   yarn init
   ```

2. 添加/更新/删除依赖关系
3. 安装/重新安装你的依赖
4. 使用版本控制工具git
5. 持续集成

---

## 添加依赖关系

1. yarn add [package] == npm install [package]

2. 将会自动将package@version添加进 package.json文件中的 "dependencies"中

3. 自动更新.lock文件
   **dependencies**: 正常的运行过程中的依赖 浏览器跑的时候
   **devDependencies**: 开发模式 所用的依赖
   **peerDependencies**: 当你发布项目的时候 可以指定该依赖
   **optionalDependencies**: 可有可无,表示在安装失败的时候一个备选的依赖保证过程
   **bundleDependencies**: 发布项目的时候所用的依赖 不是从npm来的 一起打包发布到npm上

```
yarn add --dev to add to devDependencies
yarn add --peer to add to peerDependencies
yarn add --optional to add to optionalDependencies
```

4. 安装依赖 (同 npm install)

   - yarn
   - yarn install

     yarn install --flat ##安装一个且只有一个版本的包

yarn install --force ##强制加载获取所有的包

yarn install --production ## 只会安装dependencies, 不会安装devDependencies

## 这个时候的代码只提供运行 不提供开发



5. 版本控制器

    1. **yarn.lock** 保证每个包的版本的具体依赖 保证项目能通过安装命令执行正常

    2. Yarn Cache

```
yarn cache ls # 将打印出每个缓存方案。
yarn cache dir package# 安装在本地的什么位置
yarn cache clean # 对本地缓存进行强制清除 再执行yarn cache ls将找不到缓存

# 改变默认的缓存目录
yarn config set cache-folder <path>
yarn <command> --cache-folder <path>
```

    3. Yarn Config

```
yarn config set <key> <value> [-g|--global]  #设置配置项
yarn config set init-license BSD-2-Clause
yarn config set registry https://registry.npm.taobao.org  # 修改镜像获取位置'h
yarn config get <key> # 获取配置项信息
yarn config get init-license
yarn config delete <key>  # 删除某一配置项
yarn config delete test-key
yarn config list # 显示当前所有的配置
```

6. Yarn info

    显示包的详细信息 **yarn info <package> [<field>]**

```
yarn info react
yarn info react --json # 以json的格式显示
yarn info react@15.3.0 # 查看详细版本信息
yarn info react description # 只看描述信息
yarn info react time
## 如果指定的字段是又一个嵌套对象,返回子树
yarn info react readme 读取readme字段
```

7. Yarn Global

    **Yarn不推荐将依赖安装在全局环境下面**

```
yarn global <add/bin/ls/remove/upgrade> [--prefix]
yarn global add weback  ## 把包安装在全局的环境下面
```

8. Yarn Bin

    displays the location of the yarn bin folder. 显示yarn的安装目录bin文件夹

```
yarn global bin
```

9. Yarn Why

```
yarn why jest  # 确定为什么包已经安装,详细描述其他包依赖于它
# 可以是一个文件夹 也可以是一个js文件
yarn why node_modules/once
yarn why node_modules/once/once.js
```

10. yarn clean 清除npm包下面一些module下没有用的文件

```
yarn check # 检查包的完整性
yarn generate-lock-entry # 生成.lock文件
```

11. Yarn Run

    类似于npm run

```
package.json
{
  "name": "my-package",
  "scripts": {
   "build": "babel src -d lib",
   "test": "jest"
  }
}

yarn run [script] [-- <args>]
yarn run test
yarn run test -- -o --watch 执行jest -o --watch
yarn run 没有参数的的话 会列出所有的scripts里的执行脚本
```

12. Yarn Ls

```
yarn ls  ## 给出根目录下面已经安装的依赖
yarn global ls
yarn list [--depth]
yarn list --depth=0 ## 限制输出的深度
```

13. Yarn Link & Yarn Unlink

```
yarn link [package...]
## 将网络上的包连接到本地进行引用 调试
yarn unlink [packae...] 解除引用
```

14. Yarn Login & Yarn Logout

    登录npm的公共仓库, 发布一个包的时候 需要一个npm的账号

    登录之后 用yarn publish

15. Yarn Tag

```
yarn add your-package-name@stable
yarn add your-package-name@canary
```

- latest 当前版本的包
- stable包的最新稳定版本,通常长期支持(LTS)
- beta 最新发布前和/或稳定,用于共享即将改变之前完成
- canary 如果你的项目是频繁更新,许多人依靠你可能更早使用这个共享代码。
- dev 有时你希望能够测试出一个通过注册表修改当你还在研究的东西,这是非常有用的

16. Version和Package

1. 添加一个名为<tag>标记为一个特定的<version>

```
yarn add your-package-name@<version>
```

2. 列出所有的标签<包>。如果未指定的<包>将默认包你当前的目录内

```
yarn add your-package-name@<tag>
yarn tag add <package>@<version> <tag> <package>。
yarn tag rm <package> <tag>
yarn tag ls [<package>]
```



**小礼物走一走，来简书关注我**

赞赏支持