



MOONでの

MONOREPO管理と

PACKEMONでのCJS/

ESMの**DUAL PACKAGE**



- » 一貫性を持ったmonorepo管理ツール for JavaScript
- » ハッシュを使ったファイルキャッシュ
- » プロジェクトのパッケージ間の依存関係の分析とタスク実行
- » Node.js/npmパッケージマネージャーのバージョン管理と一貫性
- » TypeScriptのProject Referencesの一貫性

比較

- » **Nx**: The Framework + Pluginで拡張する
 - » **Lerna**: Nx傘下、ライブラリ向け
- » **Turborepo**: タスク実行の最適化をする
- » **moon**: monorepoでの一貫性を提供する
 - » **Packemon**: moonと同じ作者。ESM/CJSのライブラリ向け

比較(言語)

» **Nx**: TypeScript + C++

» **Turborepo**: GoLang

» **moon**: Rust 🦀



» 将来的なカバー範囲の予想

» Nx >= 🌕 moon > Turborepo

» Moonでは、Remote caching/分散ビルドはこれから



MOONの特徴

» 一貫性



MOONのSTRUCTURE

- » Workspace: monorepoのルートのこと
 - » チーム開発なら1つのチームがいるスペース
 - » WorkspaceのNodeやパッケージマネージャーの環境を統一できる
- » Project: 各Packagesのこと
 - » client/server/commonみたいなパッケージなど
 - » Project間の依存関係を管理して、常に同期できる

Workspace (workspace.yml)

Project
(ts)

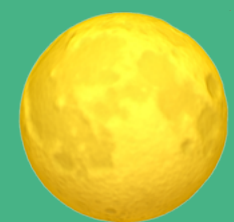
project.yml

Project
(js)

project.yml

Project
(bash)

project.yml



MOON 使い方のイメージ1

- » `.moon/workspace.yml`にワークスペースの設定を定義する
 - » どのディレクトリをProjectにするか
 - » Node/npm/TypeScript/VSC(git)などの共通設定
- » `.moon/project.yml`にプロジェクトの共通タスクを定義する
 - » `npm run-script`から`migrate`もできる
- » `<プロジェクト>/project.yml`にプロジェクト固有のタスクを設定する



MOON 使い方のイメージ2

- » `moon run <プロジェクト名>:<タスク名>` でタスクを実行できる
 - » `client`の`build`を実行するなら`moon run client:build`
 - » `moon run :build` で全てのプロジェクトの`build`を実行できる
(`lerna run <script>`相当)
- » 実行するタスクとそのプロジェクトがものは自動で実行される
 - » プロジェクト(`dependsOn`)とタスク(`deps`)に依存を定義できる
 - » タスクの入力ファイルと出力ファイルに関しても依存を定義できる



MOON 使い方のイメージ3

- » `moon run` でタスクを実行する前に、`workspace`の定義と実行して
る環境が一致するかをチェックしてる
- » 一致していないなら環境を一致させる`moon sync`が自動的に叩かれ
る
- » Node/npmのバージョン、TSの`Project References`の依存関
係、ローカルのキャッシュの状態などが意識せずに合うように同期
される
- » チーム開発での環境のばらつきが抑えられて一貫性が保てる



MOONの特徴

- » 意識しなくても常にmoon syncが行われている
 - » workspaceのNode.jsやnpmのバージョンを変更したら、自動的にバイナリがダウンロードされる
 - » workspaceにprojectを増やしたら、TSのProject Referencesが自動的に更新される
- » チーム開発で「変更入れたので、ローカルでこのコマンド実行しておいてください」みたいなのが減る

MOONの面白いポイント

- » `moon ci`というCI向けのコマンドが用意されている
 - » `Continuous integration | moon`
- » `project.yml`のタスクの定義に基づき、自動的にタスクが実行される
 - » タスク側に `runInCI: false` となければとりあえず実行される



PACKEMON

PACKEMON

- » 🌕 moonと作者は同じ
- » ESM/CJSのdual packageに対応したライブラリを公開する用途の bundler/build tool
- » Babel/rollupをいい感じにまとめて、package.jsonの設定も自動的に修正される

PACKEMONの特徴

- » `package.json`の`packemon`フィールドに出力形式を設定する と
- » → 自動的に `main/exports/types/type/engines`など公開するための設定が追加される

```
"packemon": [  
  {  
    "inputs": {  
      "index": "./src/index.ts"  
    },  
    "platform": "node",  
    "format": "cjs"  
  },  
  {  
    "inputs": {  
      "node": "./src/index.ts"  
    },  
    "platform": "node",  
    "format": "mjs"  
  }  
],
```

追加される

```
"types": "./dts/index.d.ts",
"main": "./cjs/index.cjs",
"engines": {
  "node": ">=14.15.0",
  "npm": ">=6.14.0"
},
"exports": {
  "./package.json": "./package.json",
  "/*": {
    "types": "./dts/*.d.ts",
    "node": {
      "import": "./mjs/*.mjs",
      "require": "./cjs/*.cjs"
    }
  },
  ".": {
    "types": "./dts/index.d.ts",
    "node": {
      "import": "./mjs/index.mjs",
      "require": "./cjs/index.cjs"
    }
  }
},
"files": [
  "cjs/**/*.{cjs,mjs,map}",
  "dts/**/*.{d.ts}",
  "mjs/**/*.{mjs,map}",
  "src/**/*.{ts,tsx,json}"
]
```


1256ce0b6ef25868ba77102d1e9ba40ba08307eb

},

"packemon": [

{

"inputs": {

"index": "./src/index.ts"

,

"platform": "node",

"format": "cjs"

,

{

"inputs": {

"node": "./src/index.ts"

,

"platform": "node",

"format": "mjs"

}

],

"author": "azu",

"license": "MIT",

"devDependencies": {

"packemon": "^2.3.2",

"typescript": "^4.7.4"

,

"dependencies": {

"pkg-dir": "^6.0.1"

}

}

9

32

10

33

11

34

12

35

13

36

14

37

15

38

16

39

17

40

18

41

19

42

20

43

21

44

22

45

23

46

24

47

25

48

26

49

27

50

28

51

29

52

30

53

31

54

32

55

33

56

>> 34

57

35

58

36

59

60

61

62

63

64

65

☒ Your version

"dependencies": {

"pkg-dir": "^6.0.1"

,

"types": "./dts/index.d.ts",

"main": "./cjs/index.cjs",

"engines": {

"node": ">=14.15.0",

"npm": ">=6.14.0"

,

"exports": {

"./package.json": "./package.json",

"./*": {

"types": "./dts/*.d.ts",

"node": {

"import": "./mjs/*.mjs",

"require": "./cjs/*.cjs"

}

,

"." : {

"types": "./dts/index.d.ts",

"node": {

"import": "./mjs/index.mjs",

"require": "./cjs/index.cjs"

}

,

"files": [

"cjs/**/*.{cjs,mjs,map}",

"dts/**/*.d.ts",

"mjs/**/*.{mjs,map}",

"src/**/*.{ts,tsx,json}"

]

}

Commit Message

rev

PACKEMON: その他

- » Babelを使ったCJS <-> MJSのinterop変換が実装されている
 - » [Features & optimizations | Packemon](#)
- » `index-wrapper.mjs`というCJSをMJSとしてラップして、`Dual package hazard`を回避時する実装も持っている
- » `require`と`import`で別のファイルを読み込むと、`instanceof`がコケる問題の回避

作ったもの

- » `azu/file-cache`: Node.js library that provide a cache for file metadata or file content.
- » 🌕 `moon`と`Packemon`を使い、MJSとCJSのdual packageとして公開している

まとめ

» 🌕 moonとPackemon

- » どちらもあまり意識せずに設定の同期などの一貫性を保ち、間違いを減らしてくれる
- » 一方で、抜け穴的な回避策が狭くなるので、理想と現実のギャップをどこまで埋められるかが体験に響きやすい
- » どちらもかなり現実のユースケースから作られていて、違和感がある動作は少ない〔ただ、バグはまだある〕