MOONでのMONOREPO管理とPACKEMONでのCJS/ESMのDUAL PACKAGE

MOON

» 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 Referendcesの一貫性

比較

- Nx: The Framework + Pluginで拡張する
  - Lerna: Nx傘下、ライブラリ公開向け
- 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のライブラリ公開向け

比較(言語)

» Nx: TypeScript + C++[1]

» Turborepo: GoLang

» moon: Rust 🦀

[1] Ref Nx and Turborepo

MOON

- 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は、プラグインというよりはコマンドなので、拡張性はNxの方が高そう

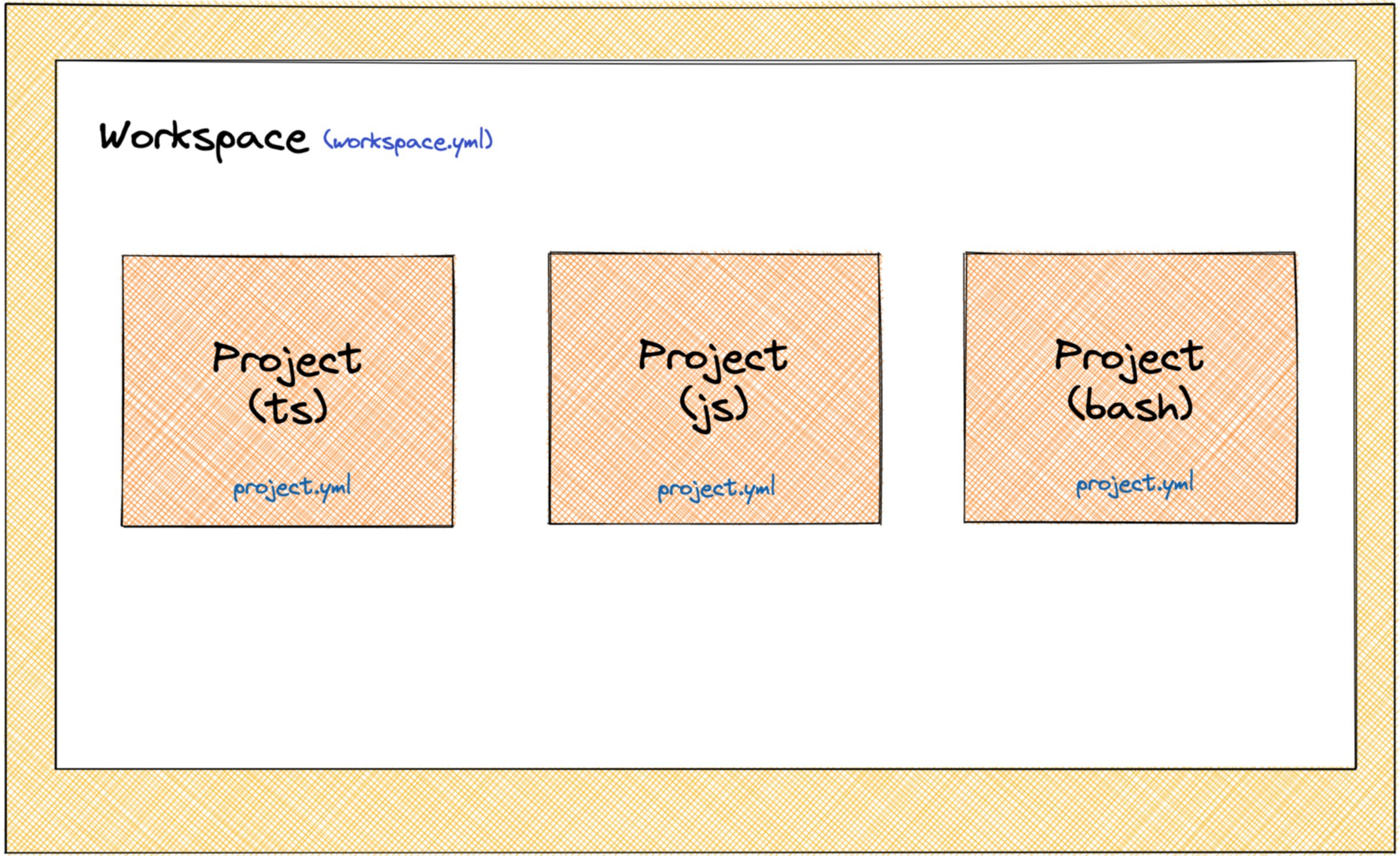
🌕 MOONの特徴

- 一貫性
- Node.jsやnpmといった実行環境に関してもmoonで管理できる
  - Voltaに近い仕組み

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

moon


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し.nvmrcとかに同期される(syncVersionManagerConfig)
  - workspaceにprojectを増やしたら、TSのProject Referencesが自動的に更新される(syncProjectWorkspaceDependencies)
  - Project間の依存は、package.jsonにも反映される(syncProjectReferences)
- 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

比較

» Nx: 全部入り、プラグインで拡張

» Turborepo: シンプル、npm run-scriptの拡張レイヤー

» 🌕 moon: 体験の一貫性

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

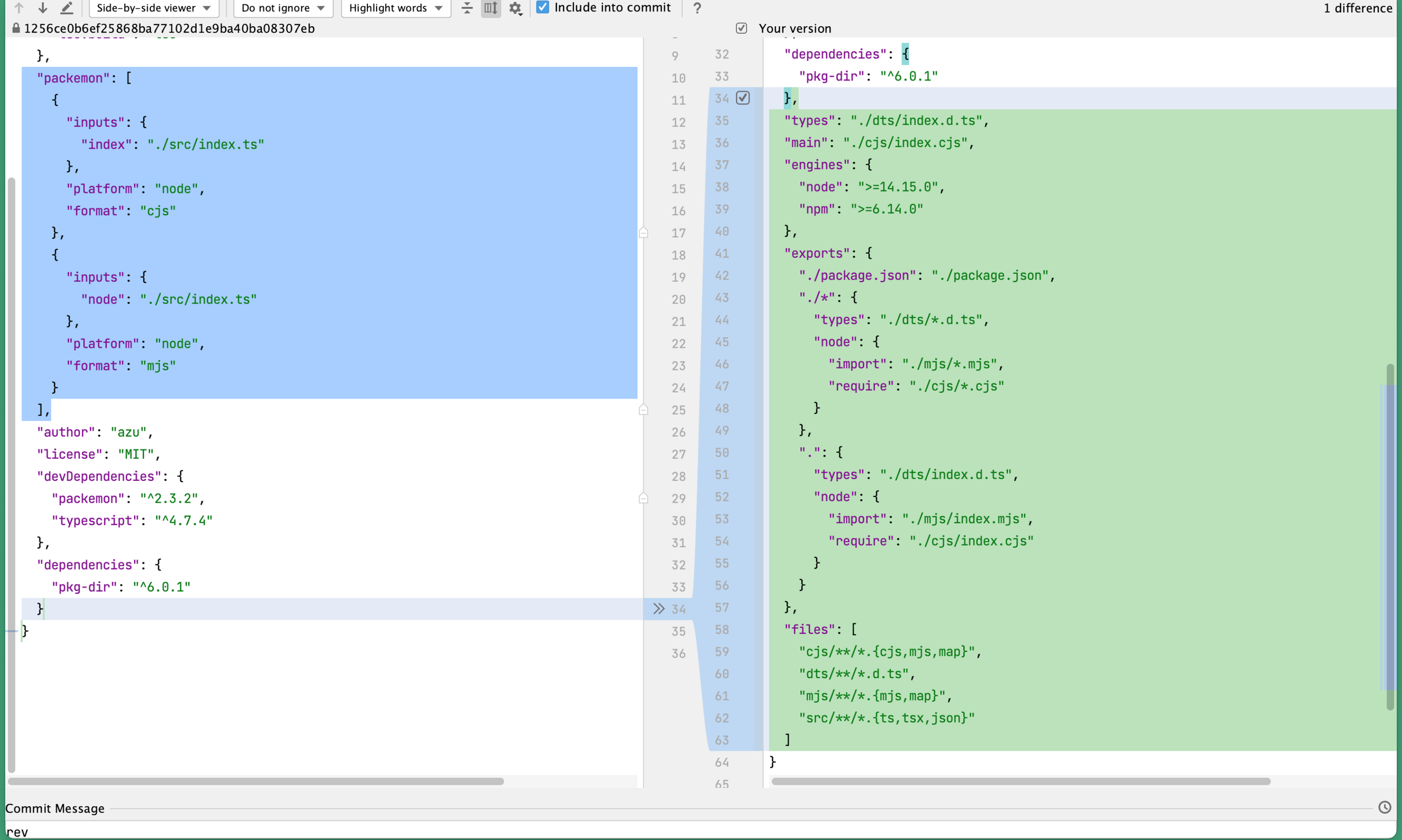
PACKEMONの特徴

» 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"
  }
],
```

```
"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}"
]
```


Side-by-side viewer
Do not ignore
Highlight words
Include into commit
1 difference
1256ce0b6ef25868ba77102d1e9ba40ba08307eb
Your version
},
"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"
}
}
"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、insteanceof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tsconfig.jsonやpackage.jsonと設定の一貫性を保ち、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)