

npm クイックリファレンス

パッケージ管理、スクリプト、バージョン管理、公開、ワークスペース

インストール

npm と Node のインストール

node -v && npm -v	# インストール済みバージョンを確認
npm install -g npm@latest	# npm自身を更新
npm install --lts	# npmでNode LTSをインストール
npm use 20	# Node 20に切り替え

インストールコマンド

npm install	package.json から全依存関係をインストール
npm install pkg	パッケージを依存関係として追加
npm install -D pkg	パッケージを devDependency として追加
npm install -g pkg	パッケージをグローバルにインストール
npm install pkg@2.1.0	特定バージョンをインストール
npm ci	ロックファイルからクリーンインストール (CI/CD)
npm uninstall pkg	パッケージを削除

パッケージ管理

パッケージの管理

npm ls	# インストール済みパッケージを一覧表示
npm ls --depth=0	# トップレベルのみ
npm outdated	# 新しいバージョンを確認
npm update	# semverの範囲内で更新
npm audit	# 脆弱性を確認

管理コマンド

npm ls	インストール済みパッケージをツリー表示
npm outdated	新しいバージョンがあるパッケージを表示
npm update [pkg]	semverの範囲内でパッケージを更新
npm audit	依存関係の脆弱性を監査
npm audit fix	脆弱な依存関係を自動修正
npm prune	不要なパッケージを削除
npm dedupe	依存関係ツリーをフラット化して重複を削減

スクリプト

スクリプトの実行

npm run build	# "build"スクリプトを実行
npm test	# "test"スクリプトのショートカット
npm start	# "start"スクリプトのショートカット
npm run lint -- --fix	# スクリプトに引数を渡す
npm run dev &	# バックグラウンドで実行

スクリプトのライフサイクル

npm test / npm t	scripts.test を実行
npm start	scripts.start を実行
npm run <name>	任意のカスタムスクリプトを実行
pre<name>	<name>の前に自動実行
post<name>	<name>の後に自動実行
npm run	利用可能な全スクリプトを一覧表示

package.json

初期化とフィールド

npm init	# 対話式セットアップ
npm init -y	# 全てのデフォルトを受け入れ
npm pkg set name="my-app"	# フィールドを設定
npm pkg get version	# フィールドを読み込み

主要フィールド

name	パッケージ名 (小文字、スペースなし)
version	現在のバージョン (semver: major.minor.patch)
main	CommonJS のエントリーポイント (require)
module	ES モジュールのエントリーポイント (バンドラー用)
type	ESM 用に"module"、CJS 用に"commonjs" (デフォルト)
scripts	名前付きコマンド (build、test、start など)
dependencies	本番用依存関係
devDependencies	開発用のみの依存関係
engines	必要な Node/npm のバージョン範囲

バージョン管理

バージョンコマンド

npm version patch	# 1.0.0 → 1.0.1
npm version minor	# 1.0.1 → 1.1.0
npm version major	# 1.1.0 → 2.0.0
npm version 3.2.1	# 明示的なバージョンを設定
npm version prerelease --preid=beta	# 1.0.0-beta.0

semver の範囲

^1.2.3	互換性あり: >=1.2.3 <2.0.0 (デフォルト)
~1.2.3	パッチレベル: >=1.2.3 <1.3.0
1.2.3	完全一致のバージョンのみ
>=1.0.0 <2.0.0	明示的な範囲
*	任意のバージョン
1.x / 1.2.x	ワイルドカード範囲
latest	最新公開バージョンタグ

公開

公開ワークフロー

npm login	# レジストリに認証
npm publish	# パブリックパッケージを公開
npm publish --access public	# スコープ付きパッケージをパブリックとして公開
npm unpublish pkg@1.0.0	# 特定バージョンを削除
npm deprecate pkg@"<2" "v2を使用"	# 古いバージョンを非推奨化

公開リファレンス

npm login	npm レジストリで認証
npm publish	パッケージをレジストリに公開
npm pack	公開せずに tarball を作成
npm unpublish	公開済みバージョンを削除 (72 時間以内)
npm deprecate	バージョンを非推奨としてマーク
.npmignore	公開パッケージから除外するファイル
files (package.json)	パッケージに含めるファイルの許可リスト

ワークスペース

ワークスペースコマンド

npm init -w packages/core	# ワークスペースを作成
npm install -w packages/core lodash	# ワークスペースにインストール
npm run build -w packages	# 全ワークスペースで実行
npm run test -w packages/api	# 特定のワークスペースで実行
npm ls -w packages	# ワークスペースの依存関係を一覧表示

ワークスペース設定

workspaces (package.json)	ワークスペースの glob 配列: ["packages/*"]
-w / --workspace	特定のワークスペースを対象とする
--workspaces	全ワークスペースでコマンドを実行
--include-workspace-root	ワークスペース操作にルートパッケージを含める
npm install (ルート)	全ワークスペースの依存関係をインストール
ホイスティング	共有依存関係はルートの node_modules にホイス

npx

npx での実行

npx create-react-app my-app	# インストールせずに実行
npx tsc --init	# ローカルまたはリモートのバイナリを実行
npx -p typescript tsc file.ts	# パッケージを明示的に指定
npx --yes create-next-app	# インストールプロンプトをスキップ
npx node@18 -e "console.log('hi!')"	# 特定のNodeで実行

npx オプション

npx cmd	ローカルの node_modules/.bin またはリモートから cmd を実行
npx -p pkg cmd	pkg をインストールして cmd を実行
npx --yes cmd	インストールプロンプトを自動承認
npx --no cmd	インストールを拒否 - ローカルになければ失敗
npx -c 'cmd'	npx の PATH でシェルコマンドを実行
npx node@ver	特定の Node.js バージョンで実行

設定

設定コマンド

npm config list	# 現在の設定を表示
npm config set registry https://r.npmjs.com/	
npm config set init-author-name "Name"	
npm config get prefix	# グローバルインストールパス
npm config delete key	# 設定値を削除

設定リファレンス

.npmrc (プロジェクト)	プロジェクトごとの設定ファイル
~/.npmrc	ユーザーごとの設定ファイル
registry	パッケージレジストリ URL
save-exact	インストール時に完全一致バージョンを固定する場合は true
engine-strict	engines フィールドを強制する場合は true
fund	資金調達メッセージを抑制する場合は false
audit	インストール時の監査をスキップする場合は false

よくあるパターン

ワンライナー

npm ls --depth=0 --json jq '.dependencies keys[]'	
npm outdated --long	# 型とホームページを表示
npm cache clean --force	# npmキャッシュをクリア
npm explain pkg	# pkgがインストールされている理由
npm exec -- envinfo --system	# バグレポート用システム情報

npm クイックリファレンス

レシピ

ロックファイルのみ	<code>npm ci --package-lock.json</code> からクリーンインストール
ライセンス確認	<code>npx license-checker --summary</code>
未使用の依存関係を検索	<code>npx depcheck</code>
バンドルサイズ	<code>npx bundlephobia-cli pkg</code> - パッケージサイズを確認
全て更新	<code>npx npm-check-updates -u && npm install</code>
ローカルレジストリ	<code>npx verdaccio</code> - プライベートレジストリを実行