

npm 빠른 참조

패키지 관리, 스크립트, 버전 관리, 배포, 워크스페이스

설치

npm 및 Node 설치

```
node -v && npm -v # check installed versions
npm install -g npm@latest # update npm itself
npm install --lts # install Node LTS via npm
npm use 20 # switch to Node 20
```

설치 명령어

npm install	package.json의 모든 의존성 설치
npm install pkg	패키지를 의존성으로 추가
npm install -D pkg	패키지를 devDependency로 추가
npm install -g pkg	패키지 전역 설치
npm install pkg@2.1.0	특정 버전 설치
npm ci	잠금 파일로 클린 설치 (CI/CD)
npm uninstall pkg	패키지 제거

패키지 관리

패키지 관리

npm ls	# list installed packages
npm ls --depth=0	# top-level only
npm outdated	# check for newer versions
npm update	# update within semver range
npm audit	# check for vulnerabilities

관리 명령어

npm ls	설치된 패키지를 트리로 나열
npm outdated	새 버전이 있는 패키지 표시
npm update [pkg]	semver 범위 내에서 패키지 업데이트
npm audit	의존성 취약점 검사
npm audit fix	취약한 의존성 자동 수정
npm psize	불필요한 패키지 제거
npm dedupe	중복 줄이기 위해 의존성 트리 평탄화

스크립트

스크립트 실행

npm run build	# run "build" script
npm test	# shortcut for "test" script
npm start	# shortcut for "start" script
npm run lint -- --fix	# pass args to script
npm run dev &	# run in background

스크립트 생명주기

npm test / npm t	scripts.test 실행
npm start	scripts.start 실행
npm run <name>	사용자 정의 스크립트 실행
pre<name>	<name> 이전에 자동 실행
post<name>	<name> 이후에 자동 실행
npm run	사용 가능한 모든 스크립트 나열

package.json

초기화 및 필드

npm init	# interactive setup
npm init -y	# accept all defaults
npm pkg set name="my-app"	# set a field
npm pkg get version	# read a field

주요 필드

name	패키지 이름 (소문자, 공백 없음)
version	현재 버전 (semver: major.minor.patch)
main	CommonJS 진입점 (require)
module	ES 모듈 진입점 (번들러)
type	ESM은 "module", CJS는 "commonjs" (기본값)
scripts	이름 있는 명령어 (build , test , start 등)
dependencies	프로덕션 의존성
devDependencies	개발 전용 의존성
engines	필요한 Node/npm 버전 범위

버전 관리

버전 명령어

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	# set explicit version
npm version prerelease --preid=beta	# 1.0.0-beta.0

Semver 범위

^1.2.3	호환: >=1.2.3 <2.0.0 (기본값)
~1.2.3	패치 수준: >=1.2.3 <1.3.0
1.2.3	정확한 버전만
>=1.0.0 <2.0.0	명시적 범위
*	모든 버전
1.x / 1.2.x	와일드카드 범위
latest	최신 배포 버전 태그

배포

배포 워크플로

npm login	# authenticate to registry
npm publish	# publish public package
npm publish --access public	# scoped package as public
npm unpublish pkg@1.0.0	# remove specific version
npm deprecate pkg@<2>	# deprecate old versions

배포 참조

npm login	npm 레지스트리 인증
npm publish	레지스트리에 패키지 배포
npm pack	배포 없이 tarball 생성
npm unpublish	배포된 버전 제거 (72시간 이내)
npm deprecate	버전을 지원 중단으로 표시
.npmignore	배포 패키지에서 제외할 파일
files (package.json)	패키지에 포함할 파일 허용 목록

워크스페이스

워크스페이스 명령어

npm init -w packages/core	# create workspace
npm install -w packages/core lodash	# install in workspace
npm run build --workspaces	# run in all workspaces
npm run test -w packages/api	# run in specific workspace
npm ls --workspaces	# list workspace deps

워크스페이스 설정

workspaces (package.json)	워크스페이스 글로벌 배열: ["packages/*"]
-w / --workspace	특정 워크스페이스 대상 지정
--workspaces	모든 워크스페이스에서 명령 실행
--include-workspace-root	워크스페이스 작업에 루트 패키지 포함
npm install (root)	모든 워크스페이스 의존성 설치
Hoisting	공유 의존성은 루트 node_modules로 호이스팅

npx

npx로 실행

npx create-react-app my-app	# run without installing
npx tsc --init	# run local or remote bin
npx -p typescript tsc file.ts	# specify package explicitly
npx --yes create-next-app	# skip install prompt
npx node@18 -e "console.log('hi')"	# run with specific Node

npx 옵션

npx cmd	로컬 node_modules/.bin 또는 원격에서 cmd 실행
npx -p pkg cmd	pkg 설치 후 cmd 실행
npx --yes cmd	설치 확인 자동 승인
npx --no cmd	설치 거부 - 로컬에 없으면 실패
npx -c 'cmd'	npx PATH로 셸 명령 실행
npx node@ver	특정 Node.js 버전 실행

설정

설정 명령어

npm config list	# show current config
npm config set registry https://r.npmjs.com/	
npm config set init-author-name "Name"	
npm config get prefix	# global install path
npm config delete key	# remove a config value

설정 참조

.npmrc (project)	프로젝트별 설정 파일
~/npmrc	사용자별 설정 파일
registry	패키지 레지스트리 URL
save-exact	설치 시 정확한 버전 고정하려면 true
engine-strict	engines 필드 강제하려면 true
fund	후원 메시지 숨기려면 false
audit	설치 시 감사 건너뛰려면 false

일반 패턴

유용한 한 줄 명령

npm ls --depth=0 --json jq '.dependencies keys[]'	
npm outdated --long	# show type and homepage
npm cache clean --force	# clear npm cache
npm explain pkg	# why is pkg installed?
npm exec -- envinfo --system	# system info for bug reports

레시피

잠금 파일만 사용	npm ci - package-lock.json으로 클린 설치
라이선스 확인	npx license-checker --summary
미사용 의존성 찾기	npx depcheck
번들 크기	npx bundlephobia-cli pkg - 패키지 크기 확인
전체 업그레이드	npx npm-check-updates -u && npm install
로컬 레지스트리	npx verdaccio - 프라이빗 레지스트리 실행