Bun JS runtime and toolkit

CONCEPTS

Bun is one binary: runtime, package manager, test runner and bundler. It runs TypeScript and JSX directly, no build step.

package.json declared ranges
↓ bun install
bun.lock pinned versions
↓ fetch + link
node_modules global cache
↓ bun run / test
Your code TS, JSX, JS

Put Bun flags right after bun: 'bun --watch run dev'. After the script name they go to the script.

Dependency lifecycle scripts (postinstall) are blocked unless the package is in trustedDependencies.

INSTALL AND ADD

bun install
Install all dependencies

bun add *<pkg>*
Add a dependency

bun add *<pkg>*@*<version>*
Add a specific version or tag

bun add -d *<pkg>*
Add a dev dependency

bun add --exact *<pkg>*
Pin the exact version

bun add github:*<user>*/*<repo>*
Add from a GitHub repo

bun add -g *<pkg>*
Install a CLI tool globally

bun install --frozen-lockfile
Fail if lockfile is out of date

bun ci
Same as install --frozen-lockfile

bun install --production
Skip devDependencies

bun install --filter *<pattern>*
Install only for matching workspaces

REMOVE AND UPDATE

bun remove *<pkg>*
Remove a dependency

bun update
Update within package.json ranges

bun update *<pkg>*
Update one package

bun update --latest
Ignore ranges, take latest

bun update -i
Pick updates interactively

bun outdated
Show packages with newer versions

bun audit
Check for known vulnerabilities

RUN AND WATCH

bun *<file>*
Run a TS, JS or JSX file

bun run *<script>*
Run a package.json script

bun run
List available scripts

bun --watch *<file>*
Restart on file change

bun --hot *<file>*
Reload in place, keep state

bun --env-file=*<file>* *<file>*
Load a specific .env file

bun run --bun *<bin>*
Run a Node-shebang binary with Bun

bunx *<pkg>*
Run a package binary without installing

bunx *<pkg>*@*<version>*
Run a specific version

bunx -p *<pkg>* *<bin>*
Binary name differs from package

bunx --bun *<pkg>*
Run with Bun instead of Node

TEST

bun test
Run all tests

TEST (cont.)

bun test *<path>*
Run files matching a path filter

bun test -t '*<regex>***'**
Run tests by name pattern

bun test --watch
Re-run on file change

bun test --coverage
Print a coverage report

bun test --bail
Stop at the first failure

bun test --timeout *<ms>*
Set per-test timeout

bun test -u
Update snapshots

BUILD AND SCAFFOLD

bun init -y
Scaffold with default answers

bun build *<entry>* **--outdir** *<dir>*
Bundle to a directory

bun build *<entry>* **--outfile** *<file>*
Bundle to one file

bun build *<entry>* **--compile --outfile** *<name>*
Make a standalone executable

bun build *<entry>* **--target** *<env>*
Target browser, bun or node

bun build *<entry>* **--minify**
Minify the output

WORKSPACES AND PACKAGES

bun run --filter '*<pattern>***'** *<script>*
Run a script in matching workspaces

bun run --workspaces *<script>*
Run a script in all workspaces

bun add *<pkg>* **--filter** *<workspace>*
Add a dependency to one workspace

bun why *<pkg>*
Explain why a package is installed