Node.js JavaScript runtime CLI

NODE_OPTIONS environment defaults → **Node flags** node --watch ... → **Script** file.js or -e → **Arguments** process.argv

CONCEPTS

node runs a file as a CommonJS script, or as an ES module for .mjs, "type": "module" or --input-type=module.

Flags go before the script name; everything after it is passed to the script. Use -- to end node's own options.

RUN CODE

node *<file>*
Run a script

node
Start the interactive REPL

node -e "*<code>*"
Evaluate a string of code

node -p "*<expr>*"
Evaluate and print the result

node *<file> <args>*
Pass arguments to the script

node -
Read the script from stdin

node -c *<file>*
Syntax check without running

RUN CODE (cont.)

node --input-type=module -e "*<code>*"
Evaluate code as an ES module

node -v
Print the Node version

corepack enable
Activate pnpm and yarn shims

WATCH MODE

node --watch *<file>*
Restart when imported files change

node --watch-preserve-output *<file>*
Keep the console between restarts

TEST RUNNER

node --test
Run all discovered test files

node --test *<file>*
Run specific test files

node --test-name-pattern="*<regex>***"**
Run tests whose name matches

node --test-only
Run only tests marked only

TEST RUNNER (cont.)

node --test-reporter=*<name>*
Choose output: spec, tap, dot, junit

node --test --watch
Re-run tests on change

ENVIRONMENT AND CONFIG

node --env-file=*<file> <script>*
Load variables from a .env file

NODE_OPTIONS="*<flags>***" node** *<script>*
Set default flags via environment

NODE_ENV=production node *<script>*
Set a variable for one run

node --run *<script>*
Run a package.json script

DEBUGGING

node --inspect *<file>*
Start with the inspector open

node --inspect-brk *<file>*
Break on the first line

DEBUGGING (cont.)

node inspect *<file>*
Use the built-in CLI debugger

node --enable-source-maps *<file>*
Map stack traces to source files

node --trace-warnings *<file>*
Show where warnings come from

node --no-warnings *<file>*
Silence all process warnings

PERFORMANCE AND MEMORY

node --max-old-space-size=*<MiB> <file>*
Raise the heap limit

PERMISSIONS AND TYPESCRIPT

node --permission --allow-fs-read=*<path> <file>*
Restrict file access

node *<file>***.ts**
Run TypeScript by stripping types

REPL

Dot commands

Command	Effect
`.help`	List commands
`.editor`	Multi-line editor, Ctrl+D runs
`.load <file>`	Run a file in the session
`.save <file>`	Save the session to a file
`.break`	Abandon a multi-line input
`.exit`	Leave (or Ctrl+D)

Tab
Complete names and properties

_
Last evaluated result

Core modules like fs and path are available in the REPL and in -p/-e without require.

RECIPES

One-liners

```
# random UUID
node -p
"crypto.randomUUID()"
# today as ISO date
node -p "new
Date().toISOString()"
# usable CPU count
node -p
"os.availableParallelism()"
```