# deno commands基础

## 版本

deno 1.25.3

A modern JavaScript and TypeScript runtime

## 文档

Docs: https://deno.land/manual@v1.25.3

Modules: https://deno.land/std/ https://deno.land/x/

Bugs: https://github.com/denoland/deno/issues

## 命令基础

|  |  |
| --- | --- |
| To start the REPL | deno |
| To execute a script | deno run https://deno.land/std/examples/welcome.ts |
| To evaluate code in the shell | deno eval "console.log(30933 + 404)" |
| USAGE | deno [OPTIONS] [SUBCOMMAND] |

# OPTIONS

|  |  |
| --- | --- |
| -h, --help | Print help information |
| -q, --quiet | Suppress diagnostic output |
| --unstable | Enable unstable features and APIs |
| -V, --version | Print version information |

# SUBCOMMANDS:

|  |  |
| --- | --- |
| bench | Run benchmarks |
| bundle | Bundle module and dependencies into single file |
| cache | Cache the dependencies |
| check | Type-check the dependencies |
| compile | UNSTABLE: Compile the script into a self contained executable |
| completions | Generate shell completions |
| coverage | Print coverage reports |
| doc | Show documentation for a module |
| eval | Eval script |
| fmt | Format source files |
| help | Print this message or the help of the given subcommand(s) |
| info | Show info about cache or info related to source file |
| init | Initialize a new project |
| install | Install script as an executable |
| lint | Lint source files |
| lsp | Start the language server |
| repl | Read Eval Print Loop |
| run | Run a JavaScript or TypeScript program |
| task | Run a task defined in the configuration file |
| test | Run tests |
| types | Print runtime TypeScript declarations |
| uninstall | Uninstall a script previously installed with deno install |
| upgrade | Upgrade deno executable to given version |
| vendor | Vendor remote modules into a local directory |

# ENVIRONMENT VARIABLES

|  |  |
| --- | --- |
| DENO\_AUTH\_TOKENS | A semi-colon separated list of bearer tokens and hostnames to use when fetching remote modules from private repositories (e.g. "abcde12345@deno.land;54321edcba@github.com") |
| DENO\_TLS\_CA\_STORE | Comma-separated list of order dependent certificate stores. Possible values: "system", "mozilla". Defaults to "mozilla". |
| DENO\_CERT | Load certificate authority from PEM encoded file |
| DENO\_DIR | Set the cache directory |
| DENO\_INSTALL\_ROOT | Set deno install's output directory (defaults to $HOME/.deno/bin) |
| DENO\_NO\_PROMPT | Set to disable permission prompts on access (alternative to passing --no-prompt on invocation) |
| DENO\_WEBGPU\_TRACE | Directory to use for wgpu traces |
| DENO\_JOBS | Number of parallel workers used for the --parallel flag with the test subcommand. Defaults to number of available CPUs. |
| HTTP\_PROXY | Proxy address for HTTP requests (module downloads, fetch) |
| HTTPS\_PROXY | Proxy address for HTTPS requests (module downloads, fetch) |
| NO\_COLOR | Set to disable color |
| NO\_PROXY | Comma-separated list of hosts which do not use a proxy (module downloads, fetch) |

# 命令详情

## bench

deno-bench

Run benchmarks using Deno's built-in bench tool.

Evaluate the given modules, run all benches declared with 'Deno.bench()' and

report results to standard output:

deno bench src/fetch\_bench.ts src/signal\_bench.ts

Directory arguments are expanded to all contained files matching the glob

{\*\_,\*.,}bench.{js,mjs,ts,mts,jsx,tsx}:

deno bench src/

USAGE:

deno bench [OPTIONS] [files]... [-- <SCRIPT\_ARG>...]

ARGS:

<files>...

List of file names to run

<SCRIPT\_ARG>...

Script arg

OPTIONS:

-A, --allow-all

Allow all permissions

--allow-env[=<allow-env>...]

Allow environment access

--allow-ffi[=<allow-ffi>...]

Allow loading dynamic libraries

--allow-hrtime

Allow high resolution time measurement

--allow-net[=<allow-net>...]

Allow network access

--allow-read[=<allow-read>...]

Allow file system read access

--allow-run[=<allow-run>...]

Allow running subprocesses

--allow-write[=<allow-write>...]

Allow file system write access

-c, --config <FILE>

The configuration file can be used to configure different aspects of

deno including TypeScript, linting, and code formatting. Typically

the configuration file will be called `deno.json` or `deno.jsonc`

and automatically detected; in that case this flag is not necessary.

See

https://deno.land/manual@v1.25.4/getting\_started/configuration\_file

--cached-only

Require that remote dependencies are already cached

--cert <FILE>

Load certificate authority from PEM encoded file

--check[=<CHECK\_TYPE>...]

Type-check modules.

Deno does not type-check modules automatically from v1.23 onwards.

Pass this flag to enable type-checking or use the 'deno check'

subcommand.

If the value of '--check=all' is supplied, diagnostic errors from

remote modules

will be included.

--filter <filter>

Run benchmarks with this string or pattern in the bench name

-h, --help

Print help information

--ignore=<ignore>

Ignore files

--import-map <FILE>

Load import map file from local file or remote URL.

Docs:

https://deno.land/manual@v1.25.4/linking\_to\_external\_code/import\_maps

Specification: https://wicg.github.io/import-maps/

Examples: https://github.com/WICG/import-maps#the-import-map

--location <HREF>

Value of 'globalThis.location' used by some web APIs

--lock <FILE>

Check the specified lock file

--lock-write

Write lock file (use with --lock)

--no-check[=<NO\_CHECK\_TYPE>...]

Skip type-checking. If the value of '--no-check=remote' is supplied,

diagnostic errors from remote modules will be ignored.

--no-clear-screen

Do not clear terminal screen when under watch mode

--no-config

Disable automatic loading of the configuration file.

--no-npm

Do not resolve npm modules

--no-prompt

Always throw if required permission wasn't passed

--no-remote

Do not resolve remote modules

--node-modules-dir

Creates a local node\_modules folder

-q, --quiet

Suppress diagnostic output

-r, --reload[=<CACHE\_BLOCKLIST>...]

Reload source code cache (recompile TypeScript)

--reload

Reload everything

--reload=https://deno.land/std

Reload only standard modules

--reload=https://deno.land/std/fs/utils.ts,https://deno.land/std/fmt/colors.ts

Reloads specific modules

--reload=npm:

Reload all npm modules

--reload=npm:chalk

Reload specific npm module

--seed <NUMBER>

Set the random number generator seed

--unsafely-ignore-certificate-errors[=<HOSTNAMES>...]

DANGER: Disables verification of TLS certificates

--unstable

Enable unstable features and APIs

--v8-flags=<v8-flags>

To see a list of all available flags use --v8-flags=--help.

--watch

Watch for file changes and restart process automatically. Only local

files from entry point module graph are watched.

## bundle

deno-bundle

Output a single JavaScript file with all dependencies.

deno bundle https://deno.land/std/examples/colors.ts colors.bundle.js

If no output file is given, the output is written to standard output:

deno bundle https://deno.land/std/examples/colors.ts

USAGE:

deno bundle [OPTIONS] <source\_file> [--] [out\_file]

ARGS:

<source\_file>

<out\_file>

OPTIONS:

-c, --config <FILE>

The configuration file can be used to configure different aspects of

deno including TypeScript, linting, and code formatting. Typically

the configuration file will be called `deno.json` or `deno.jsonc`

and automatically detected; in that case this flag is not necessary.

See

https://deno.land/manual@v1.25.4/getting\_started/configuration\_file

--cert <FILE>

Load certificate authority from PEM encoded file

--check[=<CHECK\_TYPE>...]

Type-check modules.

Deno does not type-check modules automatically from v1.23 onwards.

Pass this flag to enable type-checking or use the 'deno check'

subcommand.

If the value of '--check=all' is supplied, diagnostic errors from

remote modules

will be included.

-h, --help

Print help information

--import-map <FILE>

Load import map file from local file or remote URL.

Docs:

https://deno.land/manual@v1.25.4/linking\_to\_external\_code/import\_maps

Specification: https://wicg.github.io/import-maps/

Examples: https://github.com/WICG/import-maps#the-import-map

--lock <FILE>

Check the specified lock file

--lock-write

Write lock file (use with --lock)

--no-check[=<NO\_CHECK\_TYPE>...]

Skip type-checking. If the value of '--no-check=remote' is supplied,

diagnostic errors from remote modules will be ignored.

--no-clear-screen

Do not clear terminal screen when under watch mode

--no-config

Disable automatic loading of the configuration file.

--no-npm

Do not resolve npm modules

--no-remote

Do not resolve remote modules

--node-modules-dir

Creates a local node\_modules folder

-q, --quiet

Suppress diagnostic output

-r, --reload[=<CACHE\_BLOCKLIST>...]

Reload source code cache (recompile TypeScript)

--reload

Reload everything

--reload=https://deno.land/std

Reload only standard modules

--reload=https://deno.land/std/fs/utils.ts,https://deno.land/std/fmt/colors.ts

Reloads specific modules

--reload=npm:

Reload all npm modules

--reload=npm:chalk

Reload specific npm module

--unstable

Enable unstable features and APIs

--watch

Watch for file changes and restart process automatically. Only local

files from entry point module graph are watched.

## cache

deno-cache

Cache and compile remote dependencies recursively.

Download and compile a module with all of its static dependencies and save them

in the local cache, without running any code:

deno cache https://deno.land/std/http/file\_server.ts

Future runs of this module will trigger no downloads or compilation unless

--reload is specified.

USAGE:

deno cache [OPTIONS] <file>...

ARGS:

<file>...

OPTIONS:

-c, --config <FILE>

The configuration file can be used to configure different aspects of

deno including TypeScript, linting, and code formatting. Typically

the configuration file will be called `deno.json` or `deno.jsonc`

and automatically detected; in that case this flag is not necessary.

See

https://deno.land/manual@v1.25.4/getting\_started/configuration\_file

--cert <FILE>

Load certificate authority from PEM encoded file

--check[=<CHECK\_TYPE>...]

Type-check modules.

Deno does not type-check modules automatically from v1.23 onwards.

Pass this flag to enable type-checking or use the 'deno check'

subcommand.

If the value of '--check=all' is supplied, diagnostic errors from

remote modules

will be included.

-h, --help

Print help information

--import-map <FILE>

Load import map file from local file or remote URL.

Docs:

https://deno.land/manual@v1.25.4/linking\_to\_external\_code/import\_maps

Specification: https://wicg.github.io/import-maps/

Examples: https://github.com/WICG/import-maps#the-import-map

--lock <FILE>

Check the specified lock file

--lock-write

Write lock file (use with --lock)

--no-check[=<NO\_CHECK\_TYPE>...]

Skip type-checking. If the value of '--no-check=remote' is supplied,

diagnostic errors from remote modules will be ignored.

--no-config

Disable automatic loading of the configuration file.

--no-npm

Do not resolve npm modules

--no-remote

Do not resolve remote modules

--node-modules-dir

Creates a local node\_modules folder

-q, --quiet

Suppress diagnostic output

-r, --reload[=<CACHE\_BLOCKLIST>...]

Reload source code cache (recompile TypeScript)

--reload

Reload everything

--reload=https://deno.land/std

Reload only standard modules

--reload=https://deno.land/std/fs/utils.ts,https://deno.land/std/fmt/colors.ts

Reloads specific modules

--reload=npm:

Reload all npm modules

--reload=npm:chalk

Reload specific npm module

--unstable

Enable unstable features and APIs

## check

deno-check

Download and type-check without execution.

deno check https://deno.land/std/http/file\_server.ts

Unless --reload is specified, this command will not re-download already cached

dependencies.

USAGE:

deno check [OPTIONS] <file>...

ARGS:

<file>...

OPTIONS:

-c, --config <FILE>

The configuration file can be used to configure different aspects of

deno including TypeScript, linting, and code formatting. Typically

the configuration file will be called `deno.json` or `deno.jsonc`

and automatically detected; in that case this flag is not necessary.

See

https://deno.land/manual@v1.25.4/getting\_started/configuration\_file

--cert <FILE>

Load certificate authority from PEM encoded file

-h, --help

Print help information

--import-map <FILE>

Load import map file from local file or remote URL.

Docs:

https://deno.land/manual@v1.25.4/linking\_to\_external\_code/import\_maps

Specification: https://wicg.github.io/import-maps/

Examples: https://github.com/WICG/import-maps#the-import-map

--lock <FILE>

Check the specified lock file

--lock-write

Write lock file (use with --lock)

--no-config

Disable automatic loading of the configuration file.

--no-npm

Do not resolve npm modules

--no-remote

Do not resolve remote modules

--node-modules-dir

Creates a local node\_modules folder

-q, --quiet

Suppress diagnostic output

-r, --reload[=<CACHE\_BLOCKLIST>...]

Reload source code cache (recompile TypeScript)

--reload

Reload everything

--reload=https://deno.land/std

Reload only standard modules

--reload=https://deno.land/std/fs/utils.ts,https://deno.land/std/fmt/colors.ts

Reloads specific modules

--reload=npm:

Reload all npm modules

--reload=npm:chalk

Reload specific npm module

--remote

Type-check all modules, including remote

--unstable

Enable unstable features and APIs

## compile

deno-compile

UNSTABLE: Compiles the given script into a self contained executable.

deno compile -A https://deno.land/std/http/file\_server.ts

deno compile --output color\_util https://deno.land/std/examples/colors.ts

Any flags passed which affect runtime behavior, such as '--unstable',

'--allow-\*', '--v8-flags', etc. are encoded into the output executable and used

at runtime as if they were passed to a similar 'deno run' command.

The executable name is inferred by default: Attempt to take the file stem of the

URL path. The above example would become 'file\_server'. If the file stem is

something generic like 'main', 'mod', 'index' or 'cli', and the path has no

parent, take the file name of the parent path. Otherwise settle with the generic

name. If the resulting name has an '@...' suffix, strip it.

Cross-compiling to different target architectures is supported using the

`--target` flag. On the first invocation with deno will download proper binary

and cache it in $DENO\_DIR. The aarch64-apple-darwin target is not supported in

canary.

USAGE:

deno compile [OPTIONS] <SCRIPT\_ARG>...

ARGS:

<SCRIPT\_ARG>...

Script arg

OPTIONS:

-A, --allow-all

Allow all permissions

--allow-env[=<allow-env>...]

Allow environment access

--allow-ffi[=<allow-ffi>...]

Allow loading dynamic libraries

--allow-hrtime

Allow high resolution time measurement

--allow-net[=<allow-net>...]

Allow network access

--allow-read[=<allow-read>...]

Allow file system read access

--allow-run[=<allow-run>...]

Allow running subprocesses

--allow-write[=<allow-write>...]

Allow file system write access

-c, --config <FILE>

The configuration file can be used to configure different aspects of

deno including TypeScript, linting, and code formatting. Typically

the configuration file will be called `deno.json` or `deno.jsonc`

and automatically detected; in that case this flag is not necessary.

See

https://deno.land/manual@v1.25.4/getting\_started/configuration\_file

--cached-only

Require that remote dependencies are already cached

--cert <FILE>

Load certificate authority from PEM encoded file

--check[=<CHECK\_TYPE>...]

Type-check modules.

Deno does not type-check modules automatically from v1.23 onwards.

Pass this flag to enable type-checking or use the 'deno check'

subcommand.

If the value of '--check=all' is supplied, diagnostic errors from

remote modules

will be included.

-h, --help

Print help information

--import-map <FILE>

Load import map file from local file or remote URL.

Docs:

https://deno.land/manual@v1.25.4/linking\_to\_external\_code/import\_maps

Specification: https://wicg.github.io/import-maps/

Examples: https://github.com/WICG/import-maps#the-import-map

--location <HREF>

Value of 'globalThis.location' used by some web APIs

--lock <FILE>

Check the specified lock file

--lock-write

Write lock file (use with --lock)

--no-check[=<NO\_CHECK\_TYPE>...]

Skip type-checking. If the value of '--no-check=remote' is supplied,

diagnostic errors from remote modules will be ignored.

--no-config

Disable automatic loading of the configuration file.

--no-npm

Do not resolve npm modules

--no-prompt

Always throw if required permission wasn't passed

--no-remote

Do not resolve remote modules

--node-modules-dir

Creates a local node\_modules folder

-o, --output <output>

Output file (defaults to $PWD/<inferred-name>)

-q, --quiet

Suppress diagnostic output

-r, --reload[=<CACHE\_BLOCKLIST>...]

Reload source code cache (recompile TypeScript)

--reload

Reload everything

--reload=https://deno.land/std

Reload only standard modules

--reload=https://deno.land/std/fs/utils.ts,https://deno.land/std/fmt/colors.ts

Reloads specific modules

--reload=npm:

Reload all npm modules

--reload=npm:chalk

Reload specific npm module

--seed <NUMBER>

Set the random number generator seed

--target <target>

Target OS architecture

[possible values: x86\_64-unknown-linux-gnu, x86\_64-pc-windows-msvc,

x86\_64-apple-darwin, aarch64-apple-darwin]

--unsafely-ignore-certificate-errors[=<HOSTNAMES>...]

DANGER: Disables verification of TLS certificates

--unstable

Enable unstable features and APIs

--v8-flags=<v8-flags>

To see a list of all available flags use --v8-flags=--help.

## completions

deno-completions

Output shell completion script to standard output.

deno completions bash > /usr/local/etc/bash\_completion.d/deno.bash

source /usr/local/etc/bash\_completion.d/deno.bash

USAGE:

deno completions [OPTIONS] <shell>

ARGS:

<shell>

[possible values: bash, fish, powershell, zsh, fig]

OPTIONS:

-h, --help

Print help information

-q, --quiet

Suppress diagnostic output

--unstable

Enable unstable features and APIs

## coverage

deno-coverage

Print coverage reports from coverage profiles.

Collect a coverage profile with deno test:

deno test --coverage=cov\_profile

Print a report to stdout:

deno coverage cov\_profile

Include urls that start with the file schema:

deno coverage --include="^file:" cov\_profile

Exclude urls ending with test.ts and test.js:

deno coverage --exclude="test\.(ts|js)" cov\_profile

Include urls that start with the file schema and exclude files ending with

test.ts and test.js, for an url to match it must match the include pattern and

not match the exclude pattern:

deno coverage --include="^file:" --exclude="test\.(ts|js)" cov\_profile

Write a report using the lcov format:

deno coverage --lcov --output=cov.lcov cov\_profile/

Generate html reports from lcov:

genhtml -o html\_cov cov.lcov

USAGE:

deno coverage [OPTIONS] <files>...

ARGS:

<files>...

OPTIONS:

--exclude=<regex>...

Exclude source files from the report

[default: test\.(js|mjs|ts|jsx|tsx)$]

-h, --help

Print help information

--ignore=<ignore>

Ignore coverage files

--include=<regex>...

Include source files in the report

[default: ^file:]

--lcov

Output coverage report in lcov format

--output=<output>

Exports the coverage report in lcov format to the given file.

Filename should be passed along with '=' For example

'--output=foo.lcov' If no --output arg is specified then the report

is written to stdout.

-q, --quiet

Suppress diagnostic output

--unstable

Enable unstable features and APIs

## doc

deno-doc

Show documentation for a module.

Output documentation to standard output:

deno doc ./path/to/module.ts

Output private documentation to standard output:

deno doc --private ./path/to/module.ts

Output documentation in JSON format:

deno doc --json ./path/to/module.ts

Target a specific symbol:

deno doc ./path/to/module.ts MyClass.someField

Show documentation for runtime built-ins:

deno doc

deno doc --builtin Deno.Listener

USAGE:

deno doc [OPTIONS] [--] [ARGS]

ARGS:

<source\_file>

<filter>

Dot separated path to symbol

OPTIONS:

-h, --help

Print help information

--import-map <FILE>

Load import map file from local file or remote URL.

Docs:

https://deno.land/manual@v1.25.4/linking\_to\_external\_code/import\_maps

Specification: https://wicg.github.io/import-maps/

Examples: https://github.com/WICG/import-maps#the-import-map

--json

Output documentation in JSON format

--private

Output private documentation

-q, --quiet

Suppress diagnostic output

-r, --reload[=<CACHE\_BLOCKLIST>...]

Reload source code cache (recompile TypeScript)

--reload

Reload everything

--reload=https://deno.land/std

Reload only standard modules

--reload=https://deno.land/std/fs/utils.ts,https://deno.land/std/fmt/colors.ts

Reloads specific modules

--reload=npm:

Reload all npm modules

--reload=npm:chalk

Reload specific npm module

--unstable

Enable unstable features and APIs

## eval

deno-eval

Evaluate JavaScript from the command line.

deno eval "console.log('hello world')"

To evaluate as TypeScript:

deno eval --ext=ts "const v: string = 'hello'; console.log(v)"

This command has implicit access to all permissions (--allow-all).

USAGE:

deno eval [OPTIONS] <CODE\_ARG>...

ARGS:

<CODE\_ARG>...

Code arg

OPTIONS:

-c, --config <FILE>

The configuration file can be used to configure different aspects of

deno including TypeScript, linting, and code formatting. Typically

the configuration file will be called `deno.json` or `deno.jsonc`

and automatically detected; in that case this flag is not necessary.

See

https://deno.land/manual@v1.25.4/getting\_started/configuration\_file

--cached-only

Require that remote dependencies are already cached

--cert <FILE>

Load certificate authority from PEM encoded file

--check[=<CHECK\_TYPE>...]

Type-check modules.

Deno does not type-check modules automatically from v1.23 onwards.

Pass this flag to enable type-checking or use the 'deno check'

subcommand.

If the value of '--check=all' is supplied, diagnostic errors from

remote modules

will be included.

--ext <ext>

Set standard input (stdin) content type

[default: js]

[possible values: ts, tsx, js, jsx]

-h, --help

Print help information

--import-map <FILE>

Load import map file from local file or remote URL.

Docs:

https://deno.land/manual@v1.25.4/linking\_to\_external\_code/import\_maps

Specification: https://wicg.github.io/import-maps/

Examples: https://github.com/WICG/import-maps#the-import-map

--inspect[=<HOST:PORT>...]

Activate inspector on host:port (default: 127.0.0.1:9229)

--inspect-brk[=<HOST:PORT>...]

Activate inspector on host:port and break at start of user script

--location <HREF>

Value of 'globalThis.location' used by some web APIs

--lock <FILE>

Check the specified lock file

--lock-write

Write lock file (use with --lock)

--no-check[=<NO\_CHECK\_TYPE>...]

Skip type-checking. If the value of '--no-check=remote' is supplied,

diagnostic errors from remote modules will be ignored.

--no-config

Disable automatic loading of the configuration file.

--no-npm

Do not resolve npm modules

--no-remote

Do not resolve remote modules

--node-modules-dir

Creates a local node\_modules folder

-p, --print

print result to stdout

-q, --quiet

Suppress diagnostic output

-r, --reload[=<CACHE\_BLOCKLIST>...]

Reload source code cache (recompile TypeScript)

--reload

Reload everything

--reload=https://deno.land/std

Reload only standard modules

--reload=https://deno.land/std/fs/utils.ts,https://deno.land/std/fmt/colors.ts

Reloads specific modules

--reload=npm:

Reload all npm modules

--reload=npm:chalk

Reload specific npm module

--seed <NUMBER>

Set the random number generator seed

--unstable

Enable unstable features and APIs

--v8-flags=<v8-flags>

To see a list of all available flags use --v8-flags=--help.

## fmt

deno-fmt

Auto-format JavaScript, TypeScript, Markdown, and JSON files.

deno fmt

deno fmt myfile1.ts myfile2.ts

deno fmt --check

Format stdin and write to stdout:

cat file.ts | deno fmt -

Ignore formatting code by preceding it with an ignore comment:

// deno-fmt-ignore

Ignore formatting a file by adding an ignore comment at the top of the file:

// deno-fmt-ignore-file

USAGE:

deno fmt [OPTIONS] [files]...

ARGS:

<files>...

OPTIONS:

-c, --config <FILE>

The configuration file can be used to configure different aspects of

deno including TypeScript, linting, and code formatting. Typically

the configuration file will be called `deno.json` or `deno.jsonc`

and automatically detected; in that case this flag is not necessary.

See

https://deno.land/manual@v1.25.4/getting\_started/configuration\_file

--check

Check if the source files are formatted

--ext <ext>

Set standard input (stdin) content type

[default: ts]

[possible values: ts, tsx, js, jsx, md, json, jsonc]

-h, --help

Print help information

--ignore=<ignore>

Ignore formatting particular source files

--no-clear-screen

Do not clear terminal screen when under watch mode

--no-config

Disable automatic loading of the configuration file.

--options-indent-width <options-indent-width>

Define indentation width. Defaults to 2.

--options-line-width <options-line-width>

Define maximum line width. Defaults to 80.

--options-prose-wrap <options-prose-wrap>

Define how prose should be wrapped. Defaults to always.

[possible values: always, never, preserve]

--options-single-quote

Use single quotes. Defaults to false.

--options-use-tabs

Use tabs instead of spaces for indentation. Defaults to false.

-q, --quiet

Suppress diagnostic output

--unstable

Enable unstable features and APIs

--watch

Watch for file changes and restart process automatically. Only local

files from entry point module graph are watched.

## help

deno-help

Print this message or the help of the given subcommand(s)

USAGE:

deno help [OPTIONS] [SUBCOMMAND]...

ARGS:

<SUBCOMMAND>... The subcommand whose help message to display

OPTIONS:

-q, --quiet Suppress diagnostic output

--unstable Enable unstable features and APIs

## info

deno-info

Information about a module or the cache directories.

Get information about a module:

deno info https://deno.land/std/http/file\_server.ts

The following information is shown:

local: Local path of the file.

type: JavaScript, TypeScript, or JSON.

emit: Local path of compiled source code. (TypeScript only.)

dependencies: Dependency tree of the source file.

Without any additional arguments, 'deno info' shows:

DENO\_DIR: Directory containing Deno-managed files.

Remote modules cache: Subdirectory containing downloaded remote modules.

TypeScript compiler cache: Subdirectory containing TS compiler output.

USAGE:

deno info [OPTIONS] [--] [file]

ARGS:

<file>

OPTIONS:

-c, --config <FILE>

The configuration file can be used to configure different aspects of

deno including TypeScript, linting, and code formatting. Typically

the configuration file will be called `deno.json` or `deno.jsonc`

and automatically detected; in that case this flag is not necessary.

See

https://deno.land/manual@v1.25.4/getting\_started/configuration\_file

--cert <FILE>

Load certificate authority from PEM encoded file

-h, --help

Print help information

--import-map <FILE>

Load import map file from local file or remote URL.

Docs:

https://deno.land/manual@v1.25.4/linking\_to\_external\_code/import\_maps

Specification: https://wicg.github.io/import-maps/

Examples: https://github.com/WICG/import-maps#the-import-map

--json

UNSTABLE: Outputs the information in JSON format

--location <HREF>

Show files used for origin bound APIs like the Web Storage API when

running a script with '--location=<HREF>'

--no-config

Disable automatic loading of the configuration file.

-q, --quiet

Suppress diagnostic output

-r, --reload[=<CACHE\_BLOCKLIST>...]

Reload source code cache (recompile TypeScript)

--reload

Reload everything

--reload=https://deno.land/std

Reload only standard modules

--reload=https://deno.land/std/fs/utils.ts,https://deno.land/std/fmt/colors.ts

Reloads specific modules

--reload=npm:

Reload all npm modules

--reload=npm:chalk

Reload specific npm module

--unstable

Enable unstable features and APIs

## init

deno-init

Initialize a new project

USAGE:

deno init [OPTIONS] [dir]

ARGS:

<dir>

OPTIONS:

-h, --help Print help information

-q, --quiet Suppress diagnostic output

--unstable Enable unstable features and APIs

## install

deno-install

Installs a script as an executable in the installation root's bin directory.

deno install --allow-net --allow-read

https://deno.land/std/http/file\_server.ts

deno install https://deno.land/std/examples/colors.ts

To change the executable name, use -n/--name:

deno install --allow-net --allow-read -n serve

https://deno.land/std/http/file\_server.ts

The executable name is inferred by default:

- Attempt to take the file stem of the URL path. The above example would

become 'file\_server'.

- If the file stem is something generic like 'main', 'mod', 'index' or 'cli',

and the path has no parent, take the file name of the parent path. Otherwise

settle with the generic name.

- If the resulting name has an '@...' suffix, strip it.

To change the installation root, use --root:

deno install --allow-net --allow-read --root /usr/local

https://deno.land/std/http/file\_server.ts

The installation root is determined, in order of precedence:

- --root option

- DENO\_INSTALL\_ROOT environment variable

- $HOME/.deno

These must be added to the path manually if required.

USAGE:

deno install [OPTIONS] <cmd>...

ARGS:

<cmd>...

OPTIONS:

-A, --allow-all

Allow all permissions

--allow-env[=<allow-env>...]

Allow environment access

--allow-ffi[=<allow-ffi>...]

Allow loading dynamic libraries

--allow-hrtime

Allow high resolution time measurement

--allow-net[=<allow-net>...]

Allow network access

--allow-read[=<allow-read>...]

Allow file system read access

--allow-run[=<allow-run>...]

Allow running subprocesses

--allow-write[=<allow-write>...]

Allow file system write access

-c, --config <FILE>

The configuration file can be used to configure different aspects of

deno including TypeScript, linting, and code formatting. Typically

the configuration file will be called `deno.json` or `deno.jsonc`

and automatically detected; in that case this flag is not necessary.

See

https://deno.land/manual@v1.25.4/getting\_started/configuration\_file

--cached-only

Require that remote dependencies are already cached

--cert <FILE>

Load certificate authority from PEM encoded file

--check[=<CHECK\_TYPE>...]

Type-check modules.

Deno does not type-check modules automatically from v1.23 onwards.

Pass this flag to enable type-checking or use the 'deno check'

subcommand.

If the value of '--check=all' is supplied, diagnostic errors from

remote modules

will be included.

-f, --force

Forcefully overwrite existing installation

-h, --help

Print help information

--import-map <FILE>

Load import map file from local file or remote URL.

Docs:

https://deno.land/manual@v1.25.4/linking\_to\_external\_code/import\_maps

Specification: https://wicg.github.io/import-maps/

Examples: https://github.com/WICG/import-maps#the-import-map

--inspect[=<HOST:PORT>...]

Activate inspector on host:port (default: 127.0.0.1:9229)

--inspect-brk[=<HOST:PORT>...]

Activate inspector on host:port and break at start of user script

--location <HREF>

Value of 'globalThis.location' used by some web APIs

--lock <FILE>

Check the specified lock file

--lock-write

Write lock file (use with --lock)

-n, --name <name>

Executable file name

--no-check[=<NO\_CHECK\_TYPE>...]

Skip type-checking. If the value of '--no-check=remote' is supplied,

diagnostic errors from remote modules will be ignored.

--no-config

Disable automatic loading of the configuration file.

--no-npm

Do not resolve npm modules

--no-prompt

Always throw if required permission wasn't passed

--no-remote

Do not resolve remote modules

--node-modules-dir

Creates a local node\_modules folder

-q, --quiet

Suppress diagnostic output

-r, --reload[=<CACHE\_BLOCKLIST>...]

Reload source code cache (recompile TypeScript)

--reload

Reload everything

--reload=https://deno.land/std

Reload only standard modules

--reload=https://deno.land/std/fs/utils.ts,https://deno.land/std/fmt/colors.ts

Reloads specific modules

--reload=npm:

Reload all npm modules

--reload=npm:chalk

Reload specific npm module

--root <root>

Installation root

--seed <NUMBER>

Set the random number generator seed

--unsafely-ignore-certificate-errors[=<HOSTNAMES>...]

DANGER: Disables verification of TLS certificates

--unstable

Enable unstable features and APIs

--v8-flags=<v8-flags>

To see a list of all available flags use --v8-flags=--help.

## lint

deno-lint

Lint JavaScript/TypeScript source code.

deno lint

deno lint myfile1.ts myfile2.js

Print result as JSON:

deno lint --json

Read from stdin:

cat file.ts | deno lint -

cat file.ts | deno lint --json -

List available rules:

deno lint --rules

Ignore diagnostics on the next line by preceding it with an ignore comment and

rule name:

// deno-lint-ignore no-explicit-any

// deno-lint-ignore require-await no-empty

Names of rules to ignore must be specified after ignore comment.

Ignore linting a file by adding an ignore comment at the top of the file:

// deno-lint-ignore-file

USAGE:

deno lint [OPTIONS] [files]...

ARGS:

<files>...

OPTIONS:

-c, --config <FILE>

The configuration file can be used to configure different aspects of

deno including TypeScript, linting, and code formatting. Typically

the configuration file will be called `deno.json` or `deno.jsonc`

and automatically detected; in that case this flag is not necessary.

See

https://deno.land/manual@v1.25.4/getting\_started/configuration\_file

-h, --help

Print help information

--ignore=<ignore>

Ignore linting particular source files

--json

Output lint result in JSON format

--no-clear-screen

Do not clear terminal screen when under watch mode

--no-config

Disable automatic loading of the configuration file.

-q, --quiet

Suppress diagnostic output

--rules

List available rules

--rules-exclude=<rules-exclude>

Exclude lint rules

--rules-include=<rules-include>

Include lint rules

--rules-tags=<rules-tags>

Use set of rules with a tag

--unstable

Enable unstable features and APIs

--watch

Watch for file changes and restart process automatically. Only local

files from entry point module graph are watched.

## lsp

deno-lsp

The 'deno lsp' subcommand provides a way for code editors and IDEs to

interact with Deno using the Language Server Protocol. Usually humans do not

use this subcommand directly. For example, 'deno lsp' can provide IDEs with

go-to-definition support and automatic code formatting.

How to connect various editors and IDEs to 'deno lsp':

https://deno.land/manual@v1.25.4/getting\_started/setup\_your\_environment#editors-

and-ides

USAGE:

deno lsp [OPTIONS]

OPTIONS:

-h, --help

Print help information

-q, --quiet

Suppress diagnostic output

--unstable

Enable unstable features and APIs

## repl

deno-repl

Read Eval Print Loop

USAGE:

deno repl [OPTIONS]

OPTIONS:

-c, --config <FILE>

The configuration file can be used to configure different aspects of

deno including TypeScript, linting, and code formatting. Typically

the configuration file will be called `deno.json` or `deno.jsonc`

and automatically detected; in that case this flag is not necessary.

See

https://deno.land/manual@v1.25.4/getting\_started/configuration\_file

--cached-only

Require that remote dependencies are already cached

--cert <FILE>

Load certificate authority from PEM encoded file

--check[=<CHECK\_TYPE>...]

Type-check modules.

Deno does not type-check modules automatically from v1.23 onwards.

Pass this flag to enable type-checking or use the 'deno check'

subcommand.

If the value of '--check=all' is supplied, diagnostic errors from

remote modules

will be included.

--eval <code>

Evaluates the provided code when the REPL starts.

--eval-file=<eval-file>...

Evaluates the provided file(s) as scripts when the REPL starts.

Accepts file paths and URLs.

-h, --help

Print help information

--import-map <FILE>

Load import map file from local file or remote URL.

Docs:

https://deno.land/manual@v1.25.4/linking\_to\_external\_code/import\_maps

Specification: https://wicg.github.io/import-maps/

Examples: https://github.com/WICG/import-maps#the-import-map

--inspect[=<HOST:PORT>...]

Activate inspector on host:port (default: 127.0.0.1:9229)

--inspect-brk[=<HOST:PORT>...]

Activate inspector on host:port and break at start of user script

--location <HREF>

Value of 'globalThis.location' used by some web APIs

--lock <FILE>

Check the specified lock file

--lock-write

Write lock file (use with --lock)

--no-check[=<NO\_CHECK\_TYPE>...]

Skip type-checking. If the value of '--no-check=remote' is supplied,

diagnostic errors from remote modules will be ignored.

--no-config

Disable automatic loading of the configuration file.

--no-npm

Do not resolve npm modules

--no-remote

Do not resolve remote modules

--node-modules-dir

Creates a local node\_modules folder

-q, --quiet

Suppress diagnostic output

-r, --reload[=<CACHE\_BLOCKLIST>...]

Reload source code cache (recompile TypeScript)

--reload

Reload everything

--reload=https://deno.land/std

Reload only standard modules

--reload=https://deno.land/std/fs/utils.ts,https://deno.land/std/fmt/colors.ts

Reloads specific modules

--reload=npm:

Reload all npm modules

--reload=npm:chalk

Reload specific npm module

--seed <NUMBER>

Set the random number generator seed

--unsafely-ignore-certificate-errors[=<HOSTNAMES>...]

DANGER: Disables verification of TLS certificates

--unstable

Enable unstable features and APIs

--v8-flags=<v8-flags>

To see a list of all available flags use --v8-flags=--help.

## run

deno-run

Run a JavaScript or TypeScript program

By default all programs are run in sandbox without access to disk, network or

ability to spawn subprocesses.

deno run https://deno.land/std/examples/welcome.ts

Grant all permissions:

deno run -A https://deno.land/std/http/file\_server.ts

Grant permission to read from disk and listen to network:

deno run --allow-read --allow-net https://deno.land/std/http/file\_server.ts

Grant permission to read allow-listed files from disk:

deno run --allow-read=/etc https://deno.land/std/http/file\_server.ts

Specifying the filename '-' to read the file from stdin.

curl https://deno.land/std/examples/welcome.ts | deno run -

USAGE:

deno run [OPTIONS] <SCRIPT\_ARG>...

ARGS:

<SCRIPT\_ARG>...

Script arg

OPTIONS:

-A, --allow-all

Allow all permissions

--allow-env[=<allow-env>...]

Allow environment access

--allow-ffi[=<allow-ffi>...]

Allow loading dynamic libraries

--allow-hrtime

Allow high resolution time measurement

--allow-net[=<allow-net>...]

Allow network access

--allow-read[=<allow-read>...]

Allow file system read access

--allow-run[=<allow-run>...]

Allow running subprocesses

--allow-write[=<allow-write>...]

Allow file system write access

-c, --config <FILE>

The configuration file can be used to configure different aspects of

deno including TypeScript, linting, and code formatting. Typically

the configuration file will be called `deno.json` or `deno.jsonc`

and automatically detected; in that case this flag is not necessary.

See

https://deno.land/manual@v1.25.4/getting\_started/configuration\_file

--cached-only

Require that remote dependencies are already cached

--cert <FILE>

Load certificate authority from PEM encoded file

--check[=<CHECK\_TYPE>...]

Type-check modules.

Deno does not type-check modules automatically from v1.23 onwards.

Pass this flag to enable type-checking or use the 'deno check'

subcommand.

If the value of '--check=all' is supplied, diagnostic errors from

remote modules

will be included.

-h, --help

Print help information

--import-map <FILE>

Load import map file from local file or remote URL.

Docs:

https://deno.land/manual@v1.25.4/linking\_to\_external\_code/import\_maps

Specification: https://wicg.github.io/import-maps/

Examples: https://github.com/WICG/import-maps#the-import-map

--inspect[=<HOST:PORT>...]

Activate inspector on host:port (default: 127.0.0.1:9229)

--inspect-brk[=<HOST:PORT>...]

Activate inspector on host:port and break at start of user script

--location <HREF>

Value of 'globalThis.location' used by some web APIs

--lock <FILE>

Check the specified lock file

--lock-write

Write lock file (use with --lock)

--no-check[=<NO\_CHECK\_TYPE>...]

Skip type-checking. If the value of '--no-check=remote' is supplied,

diagnostic errors from remote modules will be ignored.

--no-clear-screen

Do not clear terminal screen when under watch mode

--no-config

Disable automatic loading of the configuration file.

--no-npm

Do not resolve npm modules

--no-prompt

Always throw if required permission wasn't passed

--no-remote

Do not resolve remote modules

--node-modules-dir

Creates a local node\_modules folder

-q, --quiet

Suppress diagnostic output

-r, --reload[=<CACHE\_BLOCKLIST>...]

Reload source code cache (recompile TypeScript)

--reload

Reload everything

--reload=https://deno.land/std

Reload only standard modules

--reload=https://deno.land/std/fs/utils.ts,https://deno.land/std/fmt/colors.ts

Reloads specific modules

--reload=npm:

Reload all npm modules

--reload=npm:chalk

Reload specific npm module

--seed <NUMBER>

Set the random number generator seed

--unsafely-ignore-certificate-errors[=<HOSTNAMES>...]

DANGER: Disables verification of TLS certificates

--unstable

Enable unstable features and APIs

--v8-flags=<v8-flags>

To see a list of all available flags use --v8-flags=--help.

--watch[=<FILES>...]

Watch for file changes and restart process automatically.

Local files from entry point module graph are watched by default.

Additional paths might be watched by passing them as arguments to

this flag.

## task

deno-task

Run a task defined in the configuration file

deno task build

USAGE:

deno task [OPTIONS] [task\_name\_and\_args]...

ARGS:

<task\_name\_and\_args>...

Task to be executed with any additional arguments passed to the task

OPTIONS:

-c, --config <FILE>

The configuration file can be used to configure different aspects of

deno including TypeScript, linting, and code formatting. Typically

the configuration file will be called `deno.json` or `deno.jsonc`

and automatically detected; in that case this flag is not necessary.

See

https://deno.land/manual@v1.25.4/getting\_started/configuration\_file

--cwd <DIR>

Specify the directory to run the task in

-h, --help

Print help information

-q, --quiet

Suppress diagnostic output

--unstable

Enable unstable features and APIs

## test

deno-test

Run tests using Deno's built-in test runner.

Evaluate the given modules, run all tests declared with 'Deno.test()' and

report results to standard output:

deno test src/fetch\_test.ts src/signal\_test.ts

Directory arguments are expanded to all contained files matching the glob

{\*\_,\*.,}test.{js,mjs,ts,mts,jsx,tsx}:

deno test src/

USAGE:

deno test [OPTIONS] [files]... [-- <SCRIPT\_ARG>...]

ARGS:

<files>...

List of file names to run

<SCRIPT\_ARG>...

Script arg

OPTIONS:

-A, --allow-all

Allow all permissions

--allow-env[=<allow-env>...]

Allow environment access

--allow-ffi[=<allow-ffi>...]

Allow loading dynamic libraries

--allow-hrtime

Allow high resolution time measurement

--allow-net[=<allow-net>...]

Allow network access

--allow-none

Don't return error code if no test files are found

--allow-read[=<allow-read>...]

Allow file system read access

--allow-run[=<allow-run>...]

Allow running subprocesses

--allow-write[=<allow-write>...]

Allow file system write access

-c, --config <FILE>

The configuration file can be used to configure different aspects of

deno including TypeScript, linting, and code formatting. Typically

the configuration file will be called `deno.json` or `deno.jsonc`

and automatically detected; in that case this flag is not necessary.

See

https://deno.land/manual@v1.25.4/getting\_started/configuration\_file

--cached-only

Require that remote dependencies are already cached

--cert <FILE>

Load certificate authority from PEM encoded file

--check[=<CHECK\_TYPE>...]

Type-check modules.

Deno does not type-check modules automatically from v1.23 onwards.

Pass this flag to enable type-checking or use the 'deno check'

subcommand.

If the value of '--check=all' is supplied, diagnostic errors from

remote modules

will be included.

--coverage=<DIR>

UNSTABLE: Collect coverage profile data into DIR

--doc

UNSTABLE: type-check code blocks

--fail-fast[=<N>...]

Stop after N errors. Defaults to stopping after first failure.

--filter <filter>

Run tests with this string or pattern in the test name

-h, --help

Print help information

--ignore=<ignore>

Ignore files

--import-map <FILE>

Load import map file from local file or remote URL.

Docs:

https://deno.land/manual@v1.25.4/linking\_to\_external\_code/import\_maps

Specification: https://wicg.github.io/import-maps/

Examples: https://github.com/WICG/import-maps#the-import-map

--inspect[=<HOST:PORT>...]

Activate inspector on host:port (default: 127.0.0.1:9229)

--inspect-brk[=<HOST:PORT>...]

Activate inspector on host:port and break at start of user script

--location <HREF>

Value of 'globalThis.location' used by some web APIs

--lock <FILE>

Check the specified lock file

--lock-write

Write lock file (use with --lock)

--no-check[=<NO\_CHECK\_TYPE>...]

Skip type-checking. If the value of '--no-check=remote' is supplied,

diagnostic errors from remote modules will be ignored.

--no-clear-screen

Do not clear terminal screen when under watch mode

--no-config

Disable automatic loading of the configuration file.

--no-npm

Do not resolve npm modules

--no-prompt

Always throw if required permission wasn't passed

--no-remote

Do not resolve remote modules

--no-run

Cache test modules, but don't run tests

--node-modules-dir

Creates a local node\_modules folder

--parallel

Run test modules in parallel. Parallelism defaults to the number of

available CPUs or the value in the DENO\_JOBS environment variable.

-q, --quiet

Suppress diagnostic output

-r, --reload[=<CACHE\_BLOCKLIST>...]

Reload source code cache (recompile TypeScript)

--reload

Reload everything

--reload=https://deno.land/std

Reload only standard modules

--reload=https://deno.land/std/fs/utils.ts,https://deno.land/std/fmt/colors.ts

Reloads specific modules

--reload=npm:

Reload all npm modules

--reload=npm:chalk

Reload specific npm module

--seed <NUMBER>

Set the random number generator seed

--shuffle[=<NUMBER>...]

(UNSTABLE): Shuffle the order in which the tests are run

--trace-ops

Enable tracing of async ops. Useful when debugging leaking ops in

test, but impacts test execution time.

--unsafely-ignore-certificate-errors[=<HOSTNAMES>...]

DANGER: Disables verification of TLS certificates

--unstable

Enable unstable features and APIs

--v8-flags=<v8-flags>

To see a list of all available flags use --v8-flags=--help.

--watch

Watch for file changes and restart process automatically. Only local

files from entry point module graph are watched.

## types

deno-types

Print runtime TypeScript declarations.

deno types > lib.deno.d.ts

The declaration file could be saved and used for typing information.

USAGE:

deno types [OPTIONS]

OPTIONS:

-h, --help

Print help information

-q, --quiet

Suppress diagnostic output

--unstable

Enable unstable features and APIs

## uninstall

deno-uninstall

Uninstalls an executable script in the installation root's bin directory.

deno uninstall serve

To change the installation root, use --root:

deno uninstall --root /usr/local serve

The installation root is determined, in order of precedence:

- --root option

- DENO\_INSTALL\_ROOT environment variable

- $HOME/.deno

USAGE:

deno uninstall [OPTIONS] <name>

ARGS:

<name>

OPTIONS:

-h, --help

Print help information

-q, --quiet

Suppress diagnostic output

--root <root>

Installation root

--unstable

Enable unstable features and APIs

## upgrade

deno-upgrade

Upgrade deno executable to the given version.

Defaults to latest.

The version is downloaded from

https://github.com/denoland/deno/releases

and is used to replace the current executable.

If you want to not replace the current Deno executable but instead download an

update to a different location, use the --output flag

deno upgrade --output $HOME/my\_deno

USAGE:

deno upgrade [OPTIONS]

OPTIONS:

--canary

Upgrade to canary builds

--cert <FILE>

Load certificate authority from PEM encoded file

--dry-run

Perform all checks without replacing old exe

-f, --force

Replace current exe even if not out-of-date

-h, --help

Print help information

--output <output>

The path to output the updated version to

-q, --quiet

Suppress diagnostic output

--unstable

Enable unstable features and APIs

--version <version>

The version to upgrade to

## vendor

deno-vendor

Vendor remote modules into a local directory.

Analyzes the provided modules along with their dependencies, downloads

remote modules to the output directory, and produces an import map that

maps remote specifiers to the downloaded files.

deno vendor main.ts

deno run --import-map vendor/import\_map.json main.ts

Remote modules and multiple modules may also be specified:

deno vendor main.ts test.deps.ts https://deno.land/std/path/mod.ts

USAGE:

deno vendor [OPTIONS] <specifiers>...

ARGS:

<specifiers>...

OPTIONS:

-c, --config <FILE>

The configuration file can be used to configure different aspects of

deno including TypeScript, linting, and code formatting. Typically

the configuration file will be called `deno.json` or `deno.jsonc`

and automatically detected; in that case this flag is not necessary.

See

https://deno.land/manual@v1.25.4/getting\_started/configuration\_file

--cert <FILE>

Load certificate authority from PEM encoded file

-f, --force

Forcefully overwrite conflicting files in existing output directory

-h, --help

Print help information

--import-map <FILE>

Load import map file from local file or remote URL.

Docs:

https://deno.land/manual@v1.25.4/linking\_to\_external\_code/import\_maps

Specification: https://wicg.github.io/import-maps/

Examples: https://github.com/WICG/import-maps#the-import-map

--lock <FILE>

Check the specified lock file

--no-config

Disable automatic loading of the configuration file.

--output <output>

The directory to output the vendored modules to

-q, --quiet

Suppress diagnostic output

-r, --reload[=<CACHE\_BLOCKLIST>...]

Reload source code cache (recompile TypeScript)

--reload

Reload everything

--reload=https://deno.land/std

Reload only standard modules

--reload=https://deno.land/std/fs/utils.ts,https://deno.land/std/fmt/colors.ts

Reloads specific modules

--reload=npm:

Reload all npm modules

--reload=npm:chalk

Reload specific npm module

--unstable

Enable unstable features and APIs