Node.js v23.1.0 documentation

► Table of contents | ► Index | ► Other versions | ► Options

▼ Table of contents

- `--experimental-default-type=type`
- `--experimental-transform-types`
- `--experimental-eventsource`
- `--experimental-import-meta-resolve`
- `--experimental-loader=module`
- `--experimental-network-inspection`
- `--experimental-permission`
- `--experimental-require-module`
- `--experimental-sea-config`
- `--experimental-shadow-realm`
- `--experimental-sqlite`
- `--experimental-strip-types`
- `--experimental-test-coverage`
- `--experimental-test-isolation=mode`
- `--experimental-test-module-mocks`
- `--experimental-test-snapshots`
- `--experimental-vm-modules`
- `--experimental-wasi-unstable-preview1`
- `--experimental-wasm-modules`
- `--experimental-webstorage`
- `--force-context-aware`
- `--force-fips`
- `--force-node-api-uncaught-exceptions-policy`
- `--frozen-intrinsics`
- `--heap-prof`
- `--heap-prof-dir`
- `--heap-prof-interval`
- `--heap-prof-name`
- `--heapsnapshot-near-heap-limit=max_count`
- `--heapsnapshot-signal=signal`
- `-h, --help`
- `--icu-data-dir=file`
- `--import=module`
- `--input-type=type`
- `--insecure-http-parser`
- `--inspect[=[host:]port]`
  - Warning: binding inspector to a public IP:port combination is insecure
- `--inspect-brk[=[host:]port]`
- `--inspect-port=[host:]port`
- `--inspect-publish-uid=stderr,http`
- `--inspect-wait[=[host:]port]`
- `-i, --interactive`
- `--jitless`

- `--localstorage-file=file`
- `--max-http-header-size=size`
- `--napi-modules`
- `--network-family-autoselection-attempt-timeout`
- `--no-addons`
- `--no-deprecation`
- `--no-experimental-detect-module`
- `--no-experimental-global-navigator`
- `--no-experimental-repl-await`
- `--no-experimental-require-module`
- `--no-experimental-websocket`
- `--no-extra-info-on-fatal-exception`
- `--no-force-async-hooks-checks`
- `--no-global-search-paths`
- `--no-network-family-autoselection`
- `--no-warnings`
- `--node-memory-debug`
- `--openssl-config=file`
- `--openssl-legacy-provider`
- `--openssl-shared-config`
- `--pending-deprecation`
- `--preserve-symlinks`
- `--preserve-symlinks-main`
- `-p, --print "script"`
- `--experimental-print-required-tla`
- `--prof`
- `--prof-process`
- `--redirect-warnings=file`
- `--report-compact`
- `--report-dir=directory, report-directory=directory`
- `--report-filename=filename`
- `--report-on-fatalerror`
- `--report-on-signal`
- `--report-signal=signal`
- `--report-uncaught-exception`
- `--report-exclude-network`
- `-r, --require module`
- `--run`
  - Intentional limitations
  - Environment variables
- `--secure-heap=n`
- `--secure-heap-min=n`
- `--snapshot-blob=path`

- `--test`
- `--test-concurrency`
- `--test-coverage-branches=threshold`
- `--test-coverage-exclude`
- `--test-coverage-functions=threshold`
- `--test-coverage-include`
- `--test-coverage-lines=threshold`
- `--test-force-exit`
- `--test-name-pattern`
- `--test-only`
- `--test-reporter`
- `--test-reporter-destination`
- `--test-shard`
- `--test-skip-pattern`
- `--test-timeout`
- `--test-update-snapshots`
- `--throw-deprecation`
- `--title=title`
- `--tls-cipher-list=list`
- `--tls-keylog=file`
- `--tls-max-v1.2`
- `--tls-max-v1.3`
- `--tls-min-v1.0`
- `--tls-min-v1.1`
- `--tls-min-v1.2`
- `--tls-min-v1.3`
- `--trace-deprecation`
- `--trace-event-categories`
- `--trace-event-file-pattern`
- `--trace-events-enabled`
- `--trace-exit`
- `--trace-sigint`
- `--trace-sync-io`
- `--trace-tls`
- `--trace-uncaught`
- `--trace-warnings`
- `--track-heap-objects`
- `--unhandled-rejections=mode`
- `--use-bundled-ca`, `--use-openssl-ca`
- `--use-largepages=mode`
- `--v8-options`
- `--v8-pool-size=num`
- `-v`, `--version`

- `--watch`
- `--watch-path`
- `--watch-preserve-output`
- `--zero-fill-buffers`

- Environment variables
  - `FORCE_COLOR=[1, 2, 3]`
  - `NO_COLOR=<any>`
  - `NODE_COMPILE_CACHE=dir`
  - `NODE_DEBUG=module[,…]`
  - `NODE_DEBUG_NATIVE=module[,…]`
  - `NODE_DISABLE_COLORS=1`
  - `NODE_DISABLE_COMPILE_CACHE=1`
  - `NODE_EXTRA_CA_CERTS=file`
  - `NODE_ICU_DATA=file`
  - `NODE_NO_WARNINGS=1`
  - `NODE_OPTIONS=options...`
  - `NODE_PATH=path[:…]`
  - `NODE_PENDING_DEPRECATION=1`
  - `NODE_PENDING_PIPE_INSTANCES=instances`
  - `NODE_PRESERVE_SYMLINKS=1`
  - `NODE_REDIRECT_WARNINGS=file`
  - `NODE_REPL_EXTERNAL_MODULE=file`
  - `NODE_REPL_HISTORY=file`
  - `NODE_SKIP_PLATFORM_CHECK=value`
  - `NODE_TEST_CONTEXT=value`
  - `NODE_TLS_REJECT_UNAUTHORIZED=value`
  - `NODE_V8_COVERAGE=dir`
    - Coverage output
    - Source map cache
  - `OPENSSL_CONF=file`
  - `SSL_CERT_DIR=dir`
  - `SSL_CERT_FILE=file`
  - `TZ`
  - `UV_THREADPOOL_SIZE=size`
- Useful V8 options
  - `--abort-on-uncaught-exception`
  - `--disallow-code-generation-from-strings`
  - `--enable-etw-stack-walking`
  - `--expose-gc`
  - `--harmony-shadow-realm`
  - `--jitless`
  - `--interpreted-frames-native-stack`
  - `--prof`

- `--perf-basic-prof`
- `--perf-basic-prof-only-functions`
- `--perf-prof`
- `--perf-prof-unwinding-info`
- `--max-old-space-size=SIZE` (in MiB)
- `--max-semi-space-size=SIZE` (in MiB)
- `--security-revert`
- `--stack-trace-limit=limit`

Command-line API

Node.js comes with a variety of CLI options. These options expose built-in debugging, multiple ways to execute scripts, and other helpful runtime options.

To view this documentation as a manual page in a terminal, run `man node`.

Synopsis

```
node [options] [V8 options] [<program-entry-point> | -e "script" | -] [--] [arguments]

node inspect [<program-entry-point> | -e "script" | <host>:<port>] …

node --v8-options
```

Execute without arguments to start the REPL.

For more info about `node inspect`, see the debugger documentation.

Program entry point

The program entry point is a specifier-like string. If the string is not an absolute path, it's resolved as a relative path from the current working directory. That path is then resolved by CommonJS module loader, or by the ES module loader if `--experimental-default-type=module` is passed. If no corresponding file is found, an error is thrown.

If a file is found, its path will be passed to the ES module loader under any of the following conditions:

- The program was started with a command-line flag that forces the entry point to be loaded with ECMAScript module loader, such as `--import` or `--experimental-default-type=module`.
- The file has an `.mjs` extension.
- The file does not have a `.cjs` extension, and the nearest parent `package.json` file contains a top-level `"type"` field with a value of `"module"`.

Otherwise, the file is loaded using the CommonJS module loader. See Modules loaders for more details.

ECMAScript modules loader entry point caveat

When loading, the ES module loader loads the program entry point, the `node` command will accept as input only files with `.js`, `.mjs`, or `.cjs` extensions; with `.wasm` extensions when `--experimental-wasm-modules` is enabled; and with no extension when `--experimental-default-type=module` is passed.

Options

▶ History

All options, including V8 options, allow words to be separated by both dashes (`-`) or underscores (`_`). For example, `--pending-deprecation` is equivalent to `--pending_deprecation`.

If an option that takes a single value (such as `--max-http-header-size`) is passed more than once, then the last passed value is used. Options from the command line take precedence over options passed through the `NODE_OPTIONS` environment variable.

`-`

Added in: v8.0.0

Alias for stdin. Analogous to the use of `-` in other command-line utilities, meaning that the script is read from stdin, and the rest of the options are passed to that script.

`--`

Added in: v6.11.0

Indicate the end of node options. Pass the rest of the arguments to the script. If no script filename or eval/print script is supplied prior to this, then the next argument is used as a script filename.

`--abort-on-uncaught-exception`

Added in: v0.10.8

Aborting instead of exiting causes a core file to be generated for post-mortem analysis using a debugger (such as `lldb`, `gdb`, and `mdb`).

If this flag is passed, the behavior can still be set to not abort through `process.setUncaughtExceptionCaptureCallback()` (and through usage of the `node:domain` module that uses it).

`--allow-addons`

Added in: v21.6.0, v20.12.0

Stability: 1.1 - Active development

When using the Permission Model, the process will not be able to use native addons by default. Attempts to do so will throw an `ERR_DLOPEN_DISABLED` unless the user explicitly passes the `--allow-addons` flag when starting Node.js.

Example:

```
// Attempt to require an native addon
require('nodejs-addon-example');
```

COPY

```
$ node --experimental-permission --allow-fs-read=* index.js
node:internal/modules/cjs/loader:1319
  return process.dlopen(module, path.toNamespacedPath(filename));
                 ^

Error: Cannot load native addon because loading addons is disabled.
    at Module._extensions..node (node:internal/modules/cjs/loader:1319:18)
    at Module.load (node:internal/modules/cjs/loader:1091:32)
    at Module._load (node:internal/modules/cjs/loader:938:12)
    at Module.require (node:internal/modules/cjs/loader:1115:19)
    at require (node:internal/modules/helpers:130:18)
    at Object.<anonymous> (/home/index.js:1:15)
    at Module._compile (node:internal/modules/cjs/loader:1233:14)
    at Module._extensions..js (node:internal/modules/cjs/loader:1287:10)
    at Module.load (node:internal/modules/cjs/loader:1091:32)
    at Module._load (node:internal/modules/cjs/loader:938:12) {
  code: 'ERR_DLOPEN_DISABLED'
}
```

COPY

--allow-child-process

Added in: v20.0.0

Stability: 1 .1 - Active development

When using the Permission Model , the process will not be able to spawn any child process by default. Attempts to do so will throw an `ERR_ACCESS_DENIED` unless the user explicitly passes the `--allow-child-process` flag when starting Node.js.

Example:

```
const childProcess = require('node:child_process');
// Attempt to bypass the permission
childProcess.spawn('node', ['-e', 'require("fs").writeFileSync("/new-file", "example")']);
```

COPY

```
$ node --experimental-permission --allow-fs-read=* index.js
node:internal/child_process:388
  const err = this._handle.spawn(options);
                           ^
Error: Access to this API has been restricted
    at ChildProcess.spawn (node:internal/child_process:388:28)
    at Object.spawn (node:child_process:723:9)
    at Object.<anonymous> (/home/index.js:3:14)
    at Module._compile (node:internal/modules/cjs/loader:1120:14)
    at Module._extensions..js (node:internal/modules/cjs/loader:1174:10)
    at Module.load (node:internal/modules/cjs/loader:998:32)
    at Module._load (node:internal/modules/cjs/loader:839:12)
    at Function.executeUserEntryPoint [as runMain] (node:internal/modules/run_main:81:12)
    at node:internal/main/run_main_module:17:47 {
  code: 'ERR_ACCESS_DENIED',
  permission: 'ChildProcess'
}
```

COPY

--allow-fs-read

▶ History

Stability: 1 .1 - Active development

This flag configures file system read permissions using the Permission Model .

The valid arguments for the `--allow-fs-read` flag are:

- `*` - To allow all `FileSystemRead` operations.
- Multiple paths can be allowed using multiple `--allow-fs-read` flags. Example `--allow-fs-read=/folder1/ --allow-fs-read=/folder1/`

Paths delimited by comma (`,`) are no longer allowed. When passing a single flag with a comma a warning will be displayed.

Examples can be found in the File System Permissions documentation.

Relative paths are NOT yet supported by the CLI flag.

The initializer module also needs to be allowed. Consider the following example:

```
$ node --experimental-permission t.js
node:internal/modules/cjs/loader:162
  const result = internalModuleStat(receiver, filename);
                 ^

Error: Access to this API has been restricted
    at stat (node:internal/modules/cjs/loader:162:18)
    at Module._findPath (node:internal/modules/cjs/loader:640:16)
    at resolveMainPath (node:internal/modules/run_main:15:25)
    at Function.executeUserEntryPoint [as runMain] (node:internal/modules/run_main:53:24)
    at node:internal/main/run_main_module:23:47 {
  code: 'ERR_ACCESS_DENIED',
  permission: 'FileSystemRead',
  resource: '/Users/rafaelgss/repos/os/node/t.js'
}
```

COPY

The process needs to have access to the `index.js` module:

```
node --experimental-permission --allow-fs-read=/path/to/index.js index.js
```

COPY

`--allow-fs-write`

▶ History

Stability: 1 .1 - Active development

This flag configures file system write permissions using the Permission Model .

The valid arguments for the `--allow-fs-write` flag are:

- `*` - To allow all `FileSystemWrite` operations.
- Multiple paths can be allowed using multiple `--allow-fs-write` flags. Example `--allow-fs-write=/folder1/ --allow-fs-write=/folder1/`

Paths delimited by comma (`,`) are no longer allowed. When passing a single flag with a comma a warning will be displayed.

Examples can be found in the File System Permissions documentation.

Relative paths are NOT supported through the CLI flag.

`--allow-wasi`

Added in: v22.3.0, v20.16.0

Stability: 1 .1 - Active development

When using the Permission Model , the process will not be capable of creating any WASI instances by default. For security reasons, the call will throw an `ERR_ACCESS_DENIED` unless the user explicitly passes the flag `--allow-wasi` in the main Node.js process.

Example:

```
const { WASI } = require('node:wasi');
// Attempt to bypass the permission
new WASI({
  version: 'preview1',
  // Attempt to mount the whole filesystem
  preopens: {
```

```
    '/': '/',
  },
});
```

COPY

```
$ node --experimental-permission --allow-fs-read=* index.js
node:wasi:99
    const wrap = new _WASI(args, env, preopens, stdio);
                 ^

Error: Access to this API has been restricted
    at new WASI (node:wasi:99:18)
    at Object.<anonymous> (/home/index.js:3:1)
    at Module._compile (node:internal/modules/cjs/loader:1476:14)
    at Module._extensions..js (node:internal/modules/cjs/loader:1555:10)
    at Module.load (node:internal/modules/cjs/loader:1288:32)
    at Module._load (node:internal/modules/cjs/loader:1104:12)
    at Function.executeUserEntryPoint [as runMain] (node:internal/modules/run_main:191:14)
    at node:internal/main/run_main_module:30:49 {
  code: 'ERR_ACCESS_DENIED',
  permission: 'WASI',
}
```

COPY

--allow-worker

Added in: v20.0.0

Stability: 1 .1 - Active development

When using the Permission Model , the process will not be able to create any worker threads by default. For security reasons, the call will throw an `ERR_ACCESS_DENIED` unless the user explicitly pass the flag `--allow-worker` in the main Node.js process.

Example:

```
const { Worker } = require('node:worker_threads');
// Attempt to bypass the permission
new Worker(__filename);
```

COPY

```
$ node --experimental-permission --allow-fs-read=* index.js
node:internal/worker:188
    this[kHandle] = new WorkerImpl(url,
                    ^

Error: Access to this API has been restricted
    at new Worker (node:internal/worker:188:21)
    at Object.<anonymous> (/home/index.js.js:3:1)
    at Module._compile (node:internal/modules/cjs/loader:1120:14)
    at Module._extensions..js (node:internal/modules/cjs/loader:1174:10)
    at Module.load (node:internal/modules/cjs/loader:998:32)
    at Module._load (node:internal/modules/cjs/loader:839:12)
    at Function.executeUserEntryPoint [as runMain] (node:internal/modules/run_main:81:12)
    at node:internal/main/run_main_module:17:47 {
  code: 'ERR_ACCESS_DENIED',
  permission: 'WorkerThreads'
}
```

COPY

--build-snapshot

Added in: v18.8.0

Stability: 1 - Experimental

Generates a snapshot blob when the process exits and writes it to disk, which can be loaded later with `--snapshot-blob`.

When building the snapshot, if `--snapshot-blob` is not specified, the generated blob will be written, by default, to `snapshot.blob` in the current working directory. Otherwise it will be written to the path specified by `--snapshot-blob`.

```
$ echo "globalThis.foo = 'I am from the snapshot'" > snapshot.js

# Run snapshot.js to initialize the application and snapshot the
# state of it into snapshot.blob.
$ node --snapshot-blob snapshot.blob --build-snapshot snapshot.js

$ echo "console.log(globalThis.foo)" > index.js

# Load the generated snapshot and start the application from index.js.
$ node --snapshot-blob snapshot.blob index.js
I am from the snapshot
```

COPY

The `v8.startupSnapshot` API can be used to specify an entry point at snapshot building time, thus avoiding the need of an additional entry script at deserialization time:

```
$ echo "require('v8').startupSnapshot.setDeserializeMainFunction(() => console.log('I am from the snaps
$ node --snapshot-blob snapshot.blob --build-snapshot snapshot.js
$ node --snapshot-blob snapshot.blob
I am from the snapshot
```

COPY

For more information, check out the `v8.startupSnapshot` API documentation.

Currently the support for run-time snapshot is experimental in that:

1. User-land modules are not yet supported in the snapshot, so only one single file can be snapshotted. Users can bundle their applications into a single script with their bundler of choice before building a snapshot, however.
2. Only a subset of the built-in modules work in the snapshot, though the Node.js core test suite checks that a few fairly complex applications can be snapshotted. Support for more modules are being added. If any crashes or buggy behaviors occur when building a snapshot, please file a report in the Node.js issue tracker and link to it in the tracking issue for user-land snapshots.

`--build-snapshot-config`

Added in: v21.6.0, v20.12.0

Stability: 1 - Experimental

Specifies the path to a JSON configuration file which configures snapshot creation behavior.

The following options are currently supported:

- `builder` `<string>` Required. Provides the name to the script that is executed before building the snapshot, as if `--build-snapshot` had been passed with `builder` as the main script name.
- `withoutCodeCache` `<boolean>` Optional. Including the code cache reduces the time spent on compiling functions included in the snapshot at the expense of a bigger snapshot size and potentially breaking portability of the snapshot.

When using this flag, additional script files provided on the command line will not be executed and instead be interpreted as regular command line arguments.

`-c`, `--check`

▶ History

Syntax check the script without executing.

`--completion-bash`

Added in: v10.12.0

Print source-able bash completion script for Node.js.

```
node --completion-bash > node_bash_completion
source node_bash_completion
```

COPY

`-C condition`, `--conditions=condition`

▶ History

Stability: 2 - Stable

Provide custom conditional exports resolution conditions.

Any number of custom string condition names are permitted.

The default Node.js conditions of `"node"`, `"default"`, `"import"`, and `"require"` will always apply as defined.

For example, to run a module with "development" resolutions:

```
node -C development app.js
```

COPY

`--cpu-prof`

▶ History

Stability: 2 - Stable

Starts the V8 CPU profiler on start up, and writes the CPU profile to disk before exit.

If `--cpu-prof-dir` is not specified, the generated profile is placed in the current working directory.

If `--cpu-prof-name` is not specified, the generated profile is named `CPU.${yyyymmdd}.${hhmmss}.${pid}.${tid}.${seq}.cpuprofile`.

```
$ node --cpu-prof index.js
$ ls *.cpuprofile
CPU.20190409.202950.15293.0.0.cpuprofile
```

COPY

`--cpu-prof-dir`

▶ History

Stability: 2 - Stable

Specify the directory where the CPU profiles generated by `--cpu-prof` will be placed.

The default value is controlled by the `--diagnostic-dir` command-line option.

--cpu-prof-interval

▶ History

Stability: 2 - Stable

Specify the sampling interval in microseconds for the CPU profiles generated by `--cpu-prof`. The default is 1000 microseconds.

--cpu-prof-name

▶ History

Stability: 2 - Stable

Specify the file name of the CPU profile generated by `--cpu-prof`.

--diagnostic-dir=directory

Set the directory to which all diagnostic output files are written. Defaults to current working directory.

Affects the default output directory of:

- `--cpu-prof-dir`
- `--heap-prof-dir`
- `--redirect-warnings`

--disable-warning=code-or-type

Stability: 1.1 - Active development

Added in: v21.3.0, v20.11.0

Disable specific process warnings by `code` or `type`.

Warnings emitted from `process.emitWarning()` may contain a `code` and a `type`. This option will not-emit warnings that have a matching `code` or `type`.

List of deprecation warnings.

The Node.js core warning types are: `DeprecationWarning` and `ExperimentalWarning`

For example, the following script will not emit DEP0025 `require('node:sys')` when executed with `node --disable-warning=DEP0025`:

```
const sys = require('node:sys');
```

COPY CJS ESM

For example, the following script will emit the DEP0025 `require('node:sys')`, but not any Experimental Warnings (such as ExperimentalWarning: `vm.measureMemory` is an experimental feature in <=v21) when executed with `node --disable-warning=ExperimentalWarning`:

```
const sys = require('node:sys');
const vm = require('node:vm');

vm.measureMemory();
```

CJS ESM

COPY

--disable-wasm-trap-handler

Added in: v22.2.0, v20.15.0

By default, Node.js enables trap-handler-based WebAssembly bound checks. As a result, V8 does not need to insert inline bound checks int the code compiled from WebAssembly which may speedup WebAssembly execution significantly, but this optimization requires allocating a big virtual memory cage (currently 10GB). If the Node.js process does not have access to a large enough virtual memory address space due to system configurations or hardware limitations, users won't be able to run any WebAssembly that involves allocation in this virtual memory cage and will see an out-of-memory error.

```
$ ulimit -v 5000000
$ node -p "new WebAssembly.Memory({ initial: 10, maximum: 100 });"
[eval]:1
new WebAssembly.Memory({ initial: 10, maximum: 100 });
^

RangeError: WebAssembly.Memory(): could not allocate memory
    at [eval]:1:1
    at runScriptInThisContext (node:internal/vm:209:10)
    at node:internal/process/execution:118:14
    at [eval]-wrapper:6:24
    at runScript (node:internal/process/execution:101:62)
    at evalScript (node:internal/process/execution:136:3)
    at node:internal/main/eval_string:49:3
```

COPY

`--disable-wasm-trap-handler` disables this optimization so that users can at least run WebAssembly (with less optimal performance) when the virtual memory address space available to their Node.js process is lower than what the V8 WebAssembly memory cage needs.

--disable-proto=mode

Added in: v13.12.0, v12.17.0

Disable the `Object.prototype.__proto__` property. If `mode` is `delete`, the property is removed entirely. If `mode` is `throw`, accesses to the property throw an exception with the code `ERR_PROTO_ACCESS`.

--disallow-code-generation-from-strings

Added in: v9.8.0

Make built-in language features like `eval` and `new Function` that generate code from strings throw an exception instead. This does not affect the Node.js `node:vm` module.

--expose-gc

Added in: v22.3.0, v20.18.0

Stability: 1 - Experimental. This flag is inherited from V8 and is subject to change upstream.

This flag will expose the gc extension from V8.

```
if (globalThis.gc) {
  globalThis.gc();
}
```

COPY

--dns-result-order=order

▶ History

Set the default value of `order` in `dns.lookup()` and `dnsPromises.lookup()`. The value could be:

- `ipv4first`: sets default `order` to `ipv4first`.
- `ipv6first`: sets default `order` to `ipv6first`.
- `verbatim`: sets default `order` to `verbatim`.

The default is `verbatim` and `dns.setDefaultResultOrder()` have higher priority than `--dns-result-order`.

--enable-fips

Added in: v6.0.0

Enable FIPS-compliant crypto at startup. (Requires Node.js to be built against FIPS-compatible OpenSSL.)

--enable-network-family-autoselection

Added in: v18.18.0

Enables the family autoselection algorithm unless connection options explicitly disables it.

--enable-source-maps

▶ History

Enable Source Map v3 support for stack traces.

When using a transpiler, such as TypeScript, stack traces thrown by an application reference the transpiled code, not the original source position. `--enable-source-maps` enables caching of Source Maps and makes a best effort to report stack traces relative to the original source file.

Overriding `Error.prepareStackTrace` may prevent `--enable-source-maps` from modifying the stack trace. Call and return the results of the original `Error.prepareStackTrace` in the overriding function to modify the stack trace with source maps.

```
const originalPrepareStackTrace = Error.prepareStackTrace;
Error.prepareStackTrace = (error, trace) => {
  // Modify error and trace and format stack trace with
  // original Error.prepareStackTrace.
  return originalPrepareStackTrace(error, trace);
};
```

COPY

Note, enabling source maps can introduce latency to your application when `Error.stack` is accessed. If you access `Error.stack` frequently in your application, take into account the performance implications of `--enable-source-maps`.

--entry-url

Added in: v23.0.0

Stability: 1 - Experimental

When present, Node.js will interpret the entry point as a URL, rather than a path.

Follows ECMAScript module resolution rules.

Any query parameter or hash in the URL will be accessible via `import.meta.url`.

```
node --entry-url 'file:///path/to/file.js?queryparams=work#and-hashes-too'
node --entry-url --experimental-strip-types 'file.ts?query#hash'
node --entry-url 'data:text/javascript,console.log("Hello")'
```

COPY

`--env-file=config`

Stability: 1 .1 - Active development

▶ History

Loads environment variables from a file relative to the current directory, making them available to applications on `process.env`. The environment variables which configure Node.js , such as `NODE_OPTIONS`, are parsed and applied. If the same variable is defined in the environment and in the file, the value from the environment takes precedence.

You can pass multiple `--env-file` arguments. Subsequent files override pre-existing variables defined in previous files.

An error is thrown if the file does not exist.

```
node --env-file=.env --env-file=.development.env index.js
```
COPY

The format of the file should be one line per key-value pair of environment variable name and value separated by `=`:

```
PORT=3000
```
COPY

Any text after a `#` is treated as a comment:

```
# This is a comment
PORT=3000 # This is also a comment
```
COPY

Values can start and end with the following quotes: `` ` ``, `"` or `'`. They are omitted from the values.

```
USERNAME="nodejs" # will result in `nodejs` as the value.
```
COPY

Multi-line values are supported:

```
MULTI_LINE="THIS IS
A MULTILINE"
# will result in `THIS IS\nA MULTILINE` as the value.
```
COPY

Export keyword before a key is ignored:

```
export USERNAME="nodejs" # will result in `nodejs` as the value.
```
COPY

If you want to load environment variables from a file that may not exist, you can use the `--env-file-if-exists` flag instead.

`--env-file-if-exists=config`

Added in: v22.9.0

Behavior is the same as `--env-file` , but an error is not thrown if the file does not exist.

`-e`, `--eval "script"`

▶ History

Evaluate the following argument as JavaScript. The modules which are predefined in the REPL can also be used in `script`.

On Windows, using `cmd.exe` a single quote will not work correctly because it only recognizes double `"` for quoting. In Powershell or Git bash, both `'` and `"` are usable.

It is possible to run code containing inline types by passing `--experimental-strip-types`.

`--experimental-async-context-frame`

Added in: v22.7.0

> Stability: 1 - Experimental

Enables the use of `AsyncLocalStorage` backed by `AsyncContextFrame` rather than the default implementation which relies on async_hooks. This new model is implemented very differently and so could have differences in how context data flows within the application. As such, it is presently recommended to be sure your application behaviour is unaffected by this change before using it in production.

`--experimental-default-type=type`

Added in: v21.0.0, v20.10.0, v18.19.0

> Stability: 1.0 - Early development

Define which module system, `module` or `commonjs`, to use for the following:

- String input provided via `--eval` or STDIN, if `--input-type` is unspecified.
- Files ending in `.js` or with no extension, if there is no `package.json` file present in the same folder or any parent folder.
- Files ending in `.js` or with no extension, if the nearest parent `package.json` field lacks a `"type"` field; unless the `package.json` folder or any parent folder is inside a `node_modules` folder.

In other words, `--experimental-default-type=module` flips all the places where Node.js currently defaults to CommonJS to instead default to ECMAScript modules, with the exception of folders and subfolders below `node_modules`, for backward compatibility.

Under `--experimental-default-type=module` and `--experimental-wasm-modules`, files with no extension will be treated as WebAssembly if they begin with the WebAssembly magic number (`\0asm`); otherwise they will be treated as ES module JavaScript.

`--experimental-transform-types`

Added in: v22.7.0

> Stability: 1.0 - Early development

Enables the transformation of TypeScript-only syntax into JavaScript code. Implies `--experimental-strip-types` and `--enable-source-maps`.

`--experimental-eventsource`

Added in: v22.3.0, v20.18.0

Enable exposition of EventSource Web API on the global scope.

`--experimental-import-meta-resolve`

▶ History

Enable experimental `import.meta.resolve()` parent URL support, which allows passing a second `parentURL` argument for contextual resolution.

Previously gated the entire `import.meta.resolve` feature.

--experimental-loader=module

▶ History

> This flag is discouraged and may be removed in a future version of Node.js. Please use `--import` with `register()` instead.

Specify the `module` containing exported module customization hooks. `module` may be any string accepted as an `import` specifier.

--experimental-network-inspection

Added in: v22.6.0, v20.18.0

Stability: 1 - Experimental

Enable experimental support for the network inspection with Chrome DevTools.

--experimental-permission

Added in: v20.0.0

Stability: 1.1 - Active development

Enable the Permission Model for current process. When enabled, the following permissions are restricted:

- File System - manageable through `--allow-fs-read`, `--allow-fs-write` flags
- Child Process - manageable through `--allow-child-process` flag
- Worker Threads - manageable through `--allow-worker` flag
- WASI - manageable through `--allow-wasi` flag
- Addons - manageable through `--allow-addons` flag

--experimental-require-module

▶ History

Stability: 1.1 - Active Development

Supports loading a synchronous ES module graph in `require()`.

See Loading ECMAScript modules using `require()`.

--experimental-sea-config

Added in: v20.0.0

Stability: 1 - Experimental

Use this flag to generate a blob that can be injected into the Node.js binary to produce a single executable application. See the documentation about this configuration for details.

--experimental-shadow-realm

Added in: v19.0.0, v18.13.0

Use this flag to enable ShadowRealm support.

--experimental-sqlite

Added in: v22.5.0

Enable the experimental `node:sqlite` module.

--experimental-strip-types

Added in: v22.6.0

Stability: 1.0 - Early development

Enable experimental type-stripping for TypeScript files. For more information, see the TypeScript type-stripping documentation.

--experimental-test-coverage

▶ History

When used in conjunction with the `node:test` module, a code coverage report is generated as part of the test runner output. If no tests are run, a coverage report is not generated. See the documentation on collecting code coverage from tests for more details.

--experimental-test-isolation=mode

Added in: v22.8.0

Stability: 1.0 - Early development

Configures the type of test isolation used in the test runner. When `mode` is `'process'`, each test file is run in a separate child process. When `mode` is `'none'`, all test files run in the same process as the test runner. The default isolation mode is `'process'`. This flag is ignored if the `--test` flag is not present. See the test runner execution model section for more information.

--experimental-test-module-mocks

Added in: v22.3.0, v20.18.0

Stability: 1.0 - Early development

Enable module mocking in the test runner.

--experimental-test-snapshots

Added in: v22.3.0

Stability: 1.0 - Early development

Enable snapshot testing in the test runner.

--experimental-vm-modules

Added in: v9.6.0

Enable experimental ES Module support in the `node:vm` module.

--experimental-wasi-unstable-preview1

▶ History

Enable experimental WebAssembly System Interface (WASI) support.

--experimental-wasm-modules

Added in: v12.3.0

Enable experimental WebAssembly module support.

--experimental-webstorage

Added in: v22.4.0

Enable experimental `Web Storage` support.

--force-context-aware

Added in: v12.12.0

Disable loading native addons that are not context-aware.

--force-fips

Added in: v6.0.0

Force FIPS-compliant crypto on startup. (Cannot be disabled from script code.) (Same requirements as `--enable-fips`.)

--force-node-api-uncaught-exceptions-policy

Added in: v18.3.0, v16.17.0

Enforces `uncaughtException` event on Node-API asynchronous callbacks.

To prevent from an existing add-on from crashing the process, this flag is not enabled by default. In the future, this flag will be enabled by default to enforce the correct behavior.

--frozen-intrinsics

Added in: v11.12.0

Stability: 1 - Experimental

Enable experimental frozen intrinsics like `Array` and `Object`.

Only the root context is supported. There is no guarantee that `globalThis.Array` is indeed the default intrinsic reference. Code may break under this flag.

To allow polyfills to be added, `--require` and `--import` both run before freezing intrinsics.

--heap-prof

▶ History

Stability: 2 - Stable

Starts the V8 heap profiler on start up, and writes the heap profile to disk before exit.

If `--heap-prof-dir` is not specified, the generated profile is placed in the current working directory.

If `--heap-prof-name` is not specified, the generated profile is named `Heap.${yyyymmdd}.${hhmmss}.${pid}.${tid}.${seq}.heapprofile`.

```
$ node --heap-prof index.js
$ ls *.heapprofile
Heap.20190409.202950.15293.0.001.heapprofile
```

COPY

--heap-prof-dir

▶ History

Stability: 2 - Stable

Specify the directory where the heap profiles generated by `--heap-prof` will be placed.

The default value is controlled by the `--diagnostic-dir` command-line option.

--heap-prof-interval

▶ History

Stability: 2 - Stable

Specify the average sampling interval in bytes for the heap profiles generated by `--heap-prof`. The default is 512 * 1024 bytes.

--heap-prof-name

▶ History

Stability: 2 - Stable

Specify the file name of the heap profile generated by `--heap-prof`.

--heapsnapshot-near-heap-limit=max_count

Added in: v15.1.0, v14.18.0

Stability: 1 - Experimental

Writes a V8 heap snapshot to disk when the V8 heap usage is approaching the heap limit. `count` should be a non-negative integer (in which case Node.js will write no more than `max_count` snapshots to disk).

When generating snapshots, garbage collection may be triggered and bring the heap usage down. Therefore multiple snapshots may be written to disk before the Node.js instance finally runs out of memory. These heap snapshots can be compared to determine what objects are being allocated during the time consecutive snapshots are taken. It's not guaranteed that Node.js will write exactly `max_count` snapshots to disk, but it will try its best to generate at least one and up to `max_count` snapshots before the Node.js instance runs out of memory when `max_count` is greater than `0`.

Generating V8 snapshots takes time and memory (both memory managed by the V8 heap and native memory outside the V8 heap). The bigger the heap is, the more resources it needs. Node.js will adjust the V8 heap to accommodate the additional V8 heap memory overhead, and try its best to avoid using up all the memory available to the process. When the process uses more memory than the system deems appropriate, the process may be terminated abruptly by the system, depending on the system configuration.

```
$ node --max-old-space-size=100 --heapsnapshot-near-heap-limit=3 index.js
Wrote snapshot to Heap.20200430.100036.49580.0.001.heapsnapshot
```

```
Wrote snapshot to Heap.20200430.100037.49580.0.002.heapsnapshot
Wrote snapshot to Heap.20200430.100038.49580.0.003.heapsnapshot

<--- Last few GCs --->

[49580:0x110000000]     4826 ms: Mark-sweep 130.6 (147.8) -> 130.5 (147.8) MB, 27.4 / 0.0 ms  (average
[49580:0x110000000]     4845 ms: Mark-sweep 130.6 (147.8) -> 130.6 (147.8) MB, 18.8 / 0.0 ms  (average


<--- JS stacktrace --->

FATAL ERROR: Ineffective mark-compacts near heap limit Allocation failed - JavaScript heap out of memor
....
```

COPY

--heapsnapshot-signal=signal

Added in: v12.0.0

Enables a signal handler that causes the Node.js process to write a heap dump when the specified signal is received. `signal` must be a valid signal name. Disabled by default.

```
$ node --heapsnapshot-signal=SIGUSR2 index.js &
$ ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
node         1  5.5  6.1 787252 247004 ?       Ssl  16:43   0:02 node --heapsnapshot-signal=SIGUSR2 inc
$ kill -USR2 1
$ ls
Heap.20190718.133405.15554.0.001.heapsnapshot
```

COPY

-h, --help

Added in: v0.1.3

Print node command-line options. The output of this option is less detailed than this document.

--icu-data-dir=file

Added in: v0.11.15

Specify ICU data load path. (Overrides `NODE_ICU_DATA`.)

--import=module

Added in: v19.0.0, v18.18.0

Stability: 1 - Experimental

Preload the specified module at startup. If the flag is provided several times, each module will be executed sequentially in the order they appear, starting with the ones provided in `NODE_OPTIONS`.

Follows ECMAScript module resolution rules. Use `--require` to load a CommonJS module. Modules preloaded with `--require` will run before modules preloaded with `--import`.

Modules are preloaded into the main thread as well as any worker threads, forked processes, or clustered processes.

--input-type=type

Added in: v12.0.0

This configures Node.js to interpret `--eval` or `STDIN` input as CommonJS or as an ES module. Valid values are `"commonjs"` or `"module"`. The default is `"commonjs"` unless `--experimental-default-type=module` is used.

The REPL does not support this option. Usage of `--input-type=module` with `--print` will throw an error, as `--print` does not support ES module syntax.

`--insecure-http-parser`

Added in: v13.4.0, v12.15.0, v10.19.0

Enable leniency flags on the HTTP parser. This may allow interoperability with non-conformant HTTP implementations.

When enabled, the parser will accept the following:

- Invalid HTTP headers values.
- Invalid HTTP versions.
- Allow message containing both `Transfer-Encoding` and `Content-Length` headers.
- Allow extra data after message when `Connection: close` is present.
- Allow extra transfer encodings after `chunked` has been provided.
- Allow `\n` to be used as token separator instead of `\r\n`.
- Allow `\r\n` not to be provided after a chunk.
- Allow spaces to be present after a chunk size and before `\r\n`.

All the above will expose your application to request smuggling or poisoning attack. Avoid using this option.

`--inspect[=[host:]port]`

Added in: v6.3.0

Activate inspector on `host:port`. Default is `127.0.0.1:9229`. If port `0` is specified, a random available port will be used.

V8 inspector integration allows tools such as Chrome DevTools and IDEs to debug and profile Node.js instances. The tools attach to Node.js instances via a tcp port and communicate using the Chrome DevTools Protocol. See V8 Inspector integration for Node.js for further explanation on Node.js debugger.

Warning: binding inspector to a public IP:port combination is insecure

Binding the inspector to a public IP (including `0.0.0.0`) with an open port is insecure, as it allows external hosts to connect to the inspector and perform a remote code execution attack.

If specifying a host, make sure that either:

- The host is not accessible from public networks.
- A firewall disallows unwanted connections on the port.

More specifically, `--inspect=0.0.0.0` is insecure if the port (`9229` by default) is not firewall-protected.

See the debugging security implications section for more information.

`--inspect-brk[=[host:]port]`

Added in: v7.6.0

Activate inspector on `host:port` and break at start of user script. Default `host:port` is `127.0.0.1:9229`. If port `0` is specified, a random available port will be used.

See V8 Inspector integration for Node.js for further explanation on Node.js debugger.

`--inspect-port=[host:]port`

Added in: v7.6.0

Set the `host:port` to be used when the inspector is activated. Useful when activating the inspector by sending the `SIGUSR1` signal.

Default host is `127.0.0.1`. If port `0` is specified, a random available port will be used.

See the security warning below regarding the `host` parameter usage.

`--inspect-publish-uid=stderr,http`

Specify ways of the inspector web socket url exposure.

By default inspector websocket url is available in stderr and under `/json/list` endpoint on `http://host:port/json/list`.

`--inspect-wait[=[host:]port]`

Added in: v22.2.0, v20.15.0

Activate inspector on `host:port` and wait for debugger to be attached. Default `host:port` is `127.0.0.1:9229`. If port `0` is specified, a random available port will be used.

See V8 Inspector integration for Node.js for further explanation on Node.js debugger.

`-i`, `--interactive`

Added in: v0.7.7

Opens the REPL even if stdin does not appear to be a terminal.

`--jitless`

Added in: v12.0.0

> Stability: 1 - Experimental. This flag is inherited from V8 and is subject to change upstream.

Disable runtime allocation of executable memory. This may be required on some platforms for security reasons. It can also reduce attack surface on other platforms, but the performance impact may be severe.

`--localstorage-file=file`

Added in: v22.4.0

The file used to store `localStorage` data. If the file does not exist, it is created the first time `localStorage` is accessed. The same file may be shared between multiple Node.js processes concurrently. This flag is a no-op unless Node.js is started with the `--experimental-webstorage` flag.

`--max-http-header-size=size`

▶ History

Specify the maximum size, in bytes, of HTTP headers. Defaults to 16 KiB.

`--napi-modules`

Added in: v7.10.0

This option is a no-op. It is kept for compatibility.

`--network-family-autoselection-attempt-timeout`

Added in: v22.1.0, v20.13.0

Sets the default value for the network family autoselection attempt timeout. For more information, see `net.getDefaultAutoSelectFamilyAttemptTimeout()`.

`--no-addons`

Added in: v16.10.0, v14.19.0

Disable the `node-addons` exports condition as well as disable loading native addons. When `--no-addons` is specified, calling `process.dlopen` or requiring a native C++ addon will fail and throw an exception.

--no-deprecation

Added in: v0.8.0

Silence deprecation warnings.

--no-experimental-detect-module

▶ History

Disable using syntax detection to determine module type.

--no-experimental-global-navigator

Added in: v21.2.0

Stability: 1 - Experimental

Disable exposition of Navigator API on the global scope.

--no-experimental-repl-await

Added in: v16.6.0

Use this flag to disable top-level await in REPL.

--no-experimental-require-module

▶ History

Stability: 1.1 - Active Development

Disable support for loading a synchronous ES module graph in `require()`.

See Loading ECMAScript modules using `require()`.

--no-experimental-websocket

Added in: v22.0.0

Disable exposition of `WebSocket` on the global scope.

--no-extra-info-on-fatal-exception

Added in: v17.0.0

Hide extra information on fatal exception that causes exit.

--no-force-async-hooks-checks

Added in: v9.0.0

Disables runtime checks for `async_hooks`. These will still be enabled dynamically when `async_hooks` is enabled.

--no-global-search-paths

Added in: v16.10.0

Do not search modules from global paths like `$HOME/.node_modules` and `$NODE_PATH`.

--no-network-family-autoselection

▶ History

Disables the family autoselection algorithm unless connection options explicitly enables it.

`--no-warnings`

Added in: v6.0.0

Silence all process warnings (including deprecations).

`--node-memory-debug`

Added in: v15.0.0, v14.18.0

Enable extra debug checks for memory leaks in Node.js internals. This is usually only useful for developers debugging Node.js itself.

`--openssl-config=file`

Added in: v6.9.0

Load an OpenSSL configuration file on startup. Among other uses, this can be used to enable FIPS-compliant crypto if Node.js is built against FIPS-enabled OpenSSL.

`--openssl-legacy-provider`

Added in: v17.0.0, v16.17.0

Enable OpenSSL 3.0 legacy provider. For more information please see OSSL_PROVIDER-legacy .

`--openssl-shared-config`

Added in: v18.5.0, v16.17.0, v14.21.0

Enable OpenSSL default configuration section, `openssl_conf` to be read from the OpenSSL configuration file. The default configuration file is named `openssl.cnf` but this can be changed using the environment variable `OPENSSL_CONF`, or by using the command line option `--openssl-config`. The location of the default OpenSSL configuration file depends on how OpenSSL is being linked to Node.js. Sharing the OpenSSL configuration may have unwanted implications and it is recommended to use a configuration section specific to Node.js which is `nodejs_conf` and is default when this option is not used.

`--pending-deprecation`

Added in: v8.0.0

Emit pending deprecation warnings.

Pending deprecations are generally identical to a runtime deprecation with the notable exception that they are turned *off* by default and will not be emitted unless either the `--pending-deprecation` command-line flag, or the `NODE_PENDING_DEPRECATION=1` environment variable, is set. Pending deprecations are used to provide a kind of selective "early warning" mechanism that developers may leverage to detect deprecated API usage.

`--preserve-symlinks`

Added in: v6.3.0

Instructs the module loader to preserve symbolic links when resolving and caching modules.

By default, when Node.js loads a module from a path that is symbolically linked to a different on-disk location, Node.js will dereference the link and use the actual on-disk "real path" of the module as both an identifier and as a root path to locate other dependency modules. In most cases, this default behavior is acceptable. However, when using symbolically linked peer dependencies, as illustrated in the example below, the default behavior causes an exception to be thrown if `moduleA` attempts to require `moduleB` as a peer dependency:

```
{appDir}
 ├── app
 │   ├── index.js
 │   └── node_modules
 │       ├── moduleA -> {appDir}/moduleA
```

```
│       └── moduleB
│           ├── index.js
│           └── package.json
└── moduleA
    ├── index.js
    └── package.json
```

COPY

The `--preserve-symlinks` command-line flag instructs Node.js to use the symlink path for modules as opposed to the real path, allowing symbolically linked peer dependencies to be found.

Note, however, that using `--preserve-symlinks` can have other side effects. Specifically, symbolically linked *native* modules can fail to load if those are linked from more than one location in the dependency tree (Node.js would see those as two separate modules and would attempt to load the module multiple times, causing an exception to be thrown).

The `--preserve-symlinks` flag does not apply to the main module, which allows `node --preserve-symlinks node_module/.bin/<foo>` to work. To apply the same behavior for the main module, also use `--preserve-symlinks-main`.

`--preserve-symlinks-main`

Added in: v10.2.0

Instructs the module loader to preserve symbolic links when resolving and caching the main module (`require.main`).

This flag exists so that the main module can be opted-in to the same behavior that `--preserve-symlinks` gives to all other imports; they are separate flags, however, for backward compatibility with older Node.js versions.

`--preserve-symlinks-main` does not imply `--preserve-symlinks`; use `--preserve-symlinks-main` in addition to `--preserve-symlinks` when it is not desirable to follow symlinks before resolving relative paths.

See `--preserve-symlinks` for more information.

`-p`, `--print "script"`

▶ History

Identical to `-e` but prints the result.

`--experimental-print-required-tla`

Added in: v22.0.0, v20.17.0

If the ES module being `require()`'d contains top-level `await`, this flag allows Node.js to evaluate the module, try to locate the top-level awaits, and print their location to help users find them.

`--prof`

Added in: v2.0.0

Generate V8 profiler output.

`--prof-process`

Added in: v5.2.0

Process V8 profiler output generated using the V8 option `--prof`.

`--redirect-warnings=file`

Added in: v8.0.0

Write process warnings to the given file instead of printing to stderr. The file will be created if it does not exist, and will be appended to if it does. If an error occurs while attempting to write the warning to the file, the warning

will be written to stderr instead.

The `file` name may be an absolute path. If it is not, the default directory it will be written to is controlled by the `--diagnostic-dir` command-line option.

`--report-compact`

Added in: v13.12.0, v12.17.0

Write reports in a compact format, single-line JSON, more easily consumable by log processing systems than the default multi-line format designed for human consumption.

`--report-dir=directory`, `report-directory=directory`

▶ History

Location at which the report will be generated.

`--report-filename=filename`

▶ History

Name of the file to which the report will be written.

If the filename is set to `'stdout'` or `'stderr'`, the report is written to the stdout or stderr of the process respectively.

`--report-on-fatalerror`

▶ History

Enables the report to be triggered on fatal errors (internal errors within the Node.js runtime such as out of memory) that lead to termination of the application. Useful to inspect various diagnostic data elements such as heap, stack, event loop state, resource consumption etc. to reason about the fatal error.

`--report-on-signal`

▶ History

Enables report to be generated upon receiving the specified (or predefined) signal to the running Node.js process. The signal to trigger the report is specified through `--report-signal`.

`--report-signal=signal`

▶ History

Sets or resets the signal for report generation (not supported on Windows). Default signal is `SIGUSR2`.

`--report-uncaught-exception`

▶ History

Enables report to be generated when the process exits due to an uncaught exception. Useful when inspecting the JavaScript stack in conjunction with native stack and other runtime environment data.

`--report-exclude-network`

Added in: v22.0.0, v20.13.0

Exclude `header.networkInterfaces` from the diagnostic report. By default this is not set and the network interfaces are included.

`-r`, `--require module`

Added in: v1.6.0

Preload the specified module at startup.

Follows `require()`'s module resolution rules. `module` may be either a path to a file, or a node module name.

Only CommonJS modules are supported. Use `--import` to preload an ECMAScript module. Modules preloaded with `--require` will run before modules preloaded with `--import`.

Modules are preloaded into the main thread as well as any worker threads, forked processes, or clustered processes.

`--run`

▶ History

Stability: 2 - Stable

This runs a specified command from a package.json's `"scripts"` object. If a missing `"command"` is provided, it will list the available scripts.

`--run` will traverse up to the root directory and finds a `package.json` file to run the command from.

`--run` prepends `./node_modules/.bin` for each ancestor of the current directory, to the `PATH` in order to execute the binaries from different folders where multiple `node_modules` directories are present, if `ancestor-folder/node_modules/.bin` is a directory.

`--run` executes the command in the directory containing the related `package.json`.

For example, the following command will run the `test` script of the `package.json` in the current folder:

```
$ node --run test
```

COPY

You can also pass arguments to the command. Any argument after `--` will be appended to the script:

```
$ node --run test -- --verbose
```

COPY

Intentional limitations

`node --run` is not meant to match the behaviors of `npm run` or of the `run` commands of other package managers. The Node.js implementation is intentionally more limited, in order to focus on top performance for the most common use cases. Some features of other `run` implementations that are intentionally excluded are:

- Running `pre` or `post` scripts in addition to the specified script.
- Defining package manager-specific environment variables.

Environment variables

The following environment variables are set when running a script with `--run`:

- `NODE_RUN_SCRIPT_NAME`: The name of the script being run. For example, if `--run` is used to run `test`, the value of this variable will be `test`.
- `NODE_RUN_PACKAGE_JSON_PATH`: The path to the `package.json` that is being processed.

`--secure-heap=n`

Added in: v15.6.0

Initializes an OpenSSL secure heap of `n` bytes. When initialized, the secure heap is used for selected types of allocations within OpenSSL during key generation and other operations. This is useful, for instance, to prevent sensitive information from leaking due to pointer overruns or underruns.

The secure heap is a fixed size and cannot be resized at runtime so, if used, it is important to select a large enough heap to cover all application uses.

The heap size given must be a power of two. Any value less than 2 will disable the secure heap.

The secure heap is disabled by default.

The secure heap is not available on Windows.

See `CRYPTO_secure_malloc_init` for more details.

`--secure-heap-min=n`

Added in: v15.6.0

When using `--secure-heap`, the `--secure-heap-min` flag specifies the minimum allocation from the secure heap. The minimum value is `2`. The maximum value is the lesser of `--secure-heap` or `2147483647`. The value given must be a power of two.

`--snapshot-blob=path`

Added in: v18.8.0

Stability: 1 - Experimental

When used with `--build-snapshot`, `--snapshot-blob` specifies the path where the generated snapshot blob is written to. If not specified, the generated blob is written to `snapshot.blob` in the current working directory.

When used without `--build-snapshot`, `--snapshot-blob` specifies the path to the blob that is used to restore the application state.

When loading a snapshot, Node.js checks that:

1. The version, architecture, and platform of the running Node.js binary are exactly the same as that of the binary that generates the snapshot.
2. The V8 flags and CPU features are compatible with that of the binary that generates the snapshot.

If they don't match, Node.js refuses to load the snapshot and exits with status code 1.

`--test`

▶ History

Starts the Node.js command line test runner. This flag cannot be combined with `--watch-path`, `--check`, `--eval`, `--interactive`, or the inspector. See the documentation on running tests from the command line for more details.

`--test-concurrency`

Added in: v21.0.0, v20.10.0, v18.19.0

The maximum number of test files that the test runner CLI will execute concurrently. If `--experimental-test-isolation` is set to `'none'`, this flag is ignored and concurrency is one. Otherwise, concurrency defaults to `os.availableParallelism() - 1`.

`--test-coverage-branches=threshold`

Added in: v22.8.0

Stability: 1 - Experimental

Require a minimum percent of covered branches. If code coverage does not reach the threshold specified, the process will exit with code `1`.

`--test-coverage-exclude`

Added in: v22.5.0

Stability: 1 - Experimental

Excludes specific files from code coverage using a glob pattern, which can match both absolute and relative file paths.

This option may be specified multiple times to exclude multiple glob patterns.

If both `--test-coverage-exclude` and `--test-coverage-include` are provided, files must meet **both** criteria to be included in the coverage report.

`--test-coverage-functions=threshold`

Added in: v22.8.0

Stability: 1 - Experimental

Require a minimum percent of covered functions. If code coverage does not reach the threshold specified, the process will exit with code `1`.

`--test-coverage-include`

Added in: v22.5.0

Stability: 1 - Experimental

Includes specific files in code coverage using a glob pattern, which can match both absolute and relative file paths.

This option may be specified multiple times to include multiple glob patterns.

If both `--test-coverage-exclude` and `--test-coverage-include` are provided, files must meet **both** criteria to be included in the coverage report.

`--test-coverage-lines=threshold`

Added in: v22.8.0

Stability: 1 - Experimental

Require a minimum percent of covered lines. If code coverage does not reach the threshold specified, the process will exit with code `1`.

`--test-force-exit`

Added in: v22.0.0, v20.14.0

Configures the test runner to exit the process once all known tests have finished executing even if the event loop would otherwise remain active.

`--test-name-pattern`

▶ History

A regular expression that configures the test runner to only execute tests whose name matches the provided pattern. See the documentation on filtering tests by name for more details.

If both `--test-name-pattern` and `--test-skip-pattern` are supplied, tests must satisfy **both** requirements in order to be executed.

--test-only

▶ History

Configures the test runner to only execute top level tests that have the `only` option set. This flag is not necessary when test isolation is disabled.

--test-reporter

▶ History

A test reporter to use when running tests. See the documentation on test reporters for more details.

--test-reporter-destination

▶ History

The destination for the corresponding test reporter. See the documentation on test reporters for more details.

--test-shard

Added in: v20.5.0, v18.19.0

Test suite shard to execute in a format of `<index>/<total>`, where

`index` is a positive integer, index of divided parts `total` is a positive integer, total of divided part This command will divide all tests files into `total` equal parts, and will run only those that happen to be in an `index` part.

For example, to split your tests suite into three parts, use this:

```
node --test --test-shard=1/3
node --test --test-shard=2/3
node --test --test-shard=3/3
```

COPY

--test-skip-pattern

Added in: v22.1.0

A regular expression that configures the test runner to skip tests whose name matches the provided pattern. See the documentation on filtering tests by name for more details.

If both `--test-name-pattern` and `--test-skip-pattern` are supplied, tests must satisfy **both** requirements in order to be executed.

--test-timeout

Added in: v21.2.0, v20.11.0

A number of milliseconds the test execution will fail after. If unspecified, subtests inherit this value from their parent. The default value is `Infinity`.

--test-update-snapshots

Added in: v22.3.0

Stability: 1.0 - Early development

Regenerates the snapshot files used by the test runner for snapshot testing. Node.js must be started with the `--experimental-test-snapshots` flag in order to use this functionality.

--throw-deprecation

Added in: v0.11.14

Throw errors for deprecations.

--title=title

Added in: v10.7.0

Set `process.title` on startup.

--tls-cipher-list=list

Added in: v4.0.0

Specify an alternative default TLS cipher list. Requires Node.js to be built with crypto support (default).

--tls-keylog=file

Added in: v13.2.0, v12.16.0

Log TLS key material to a file. The key material is in NSS `SSLKEYLOGFILE` format and can be used by software (such as Wireshark) to decrypt the TLS traffic.

--tls-max-v1.2

Added in: v12.0.0, v10.20.0

Set `tls.DEFAULT_MAX_VERSION` to 'TLSv1.2'. Use to disable support for TLSv1.3.

--tls-max-v1.3

Added in: v12.0.0

Set default `tls.DEFAULT_MAX_VERSION` to 'TLSv1.3'. Use to enable support for TLSv1.3.

--tls-min-v1.0

Added in: v12.0.0, v10.20.0

Set default `tls.DEFAULT_MIN_VERSION` to 'TLSv1'. Use for compatibility with old TLS clients or servers.

--tls-min-v1.1

Added in: v12.0.0, v10.20.0

Set default `tls.DEFAULT_MIN_VERSION` to 'TLSv1.1'. Use for compatibility with old TLS clients or servers.

--tls-min-v1.2

Added in: v12.2.0, v10.20.0

Set default `tls.DEFAULT_MIN_VERSION` to 'TLSv1.2'. This is the default for 12.x and later, but the option is supported for compatibility with older Node.js versions.

--tls-min-v1.3

Added in: v12.0.0

Set default `tls.DEFAULT_MIN_VERSION` to 'TLSv1.3'. Use to disable support for TLSv1.2, which is not as secure as TLSv1.3.

--trace-deprecation

Added in: v0.8.0

Print stack traces for deprecations.

`--trace-event-categories`

Added in: v7.7.0

A comma separated list of categories that should be traced when trace event tracing is enabled using `--trace-events-enabled`.

`--trace-event-file-pattern`

Added in: v9.8.0

Template string specifying the filepath for the trace event data, it supports `${rotation}` and `${pid}`.

`--trace-events-enabled`

Added in: v7.7.0

Enables the collection of trace event tracing information.

`--trace-exit`

Added in: v13.5.0, v12.16.0

Prints a stack trace whenever an environment is exited proactively, i.e. invoking `process.exit()`.

`--trace-sigint`

Added in: v13.9.0, v12.17.0

Prints a stack trace on SIGINT.

`--trace-sync-io`

Added in: v2.1.0

Prints a stack trace whenever synchronous I/O is detected after the first turn of the event loop.

`--trace-tls`

Added in: v12.2.0

Prints TLS packet trace information to `stderr`. This can be used to debug TLS connection problems.

`--trace-uncaught`

Added in: v13.1.0

Print stack traces for uncaught exceptions; usually, the stack trace associated with the creation of an `Error` is printed, whereas this makes Node.js also print the stack trace associated with throwing the value (which does not need to be an `Error` instance).

Enabling this option may affect garbage collection behavior negatively.

`--trace-warnings`

Added in: v6.0.0

Print stack traces for process warnings (including deprecations).

`--track-heap-objects`

Added in: v2.4.0

Track heap object allocations for heap snapshots.

`--unhandled-rejections=mode`

▶ History

Using this flag allows to change what should happen when an unhandled rejection occurs. One of the following modes can be chosen:

- `throw`: Emit `unhandledRejection`. If this hook is not set, raise the unhandled rejection as an uncaught exception. This is the default.
- `strict`: Raise the unhandled rejection as an uncaught exception. If the exception is handled, `unhandledRejection` is emitted.
- `warn`: Always trigger a warning, no matter if the `unhandledRejection` hook is set or not but do not print the deprecation warning.
- `warn-with-error-code`: Emit `unhandledRejection`. If this hook is not set, trigger a warning, and set the process exit code to 1.
- `none`: Silence all warnings.

If a rejection happens during the command line entry point's ES module static loading phase, it will always raise it as an uncaught exception.

`--use-bundled-ca`, `--use-openssl-ca`

Added in: v6.11.0

Use bundled Mozilla CA store as supplied by current Node.js version or use OpenSSL's default CA store. The default store is selectable at build-time.

The bundled CA store, as supplied by Node.js, is a snapshot of Mozilla CA store that is fixed at release time. It is identical on all supported platforms.

Using OpenSSL store allows for external modifications of the store. For most Linux and BSD distributions, this store is maintained by the distribution maintainers and system administrators. OpenSSL CA store location is dependent on configuration of the OpenSSL library but this can be altered at runtime using environment variables.

See `SSL_CERT_DIR` and `SSL_CERT_FILE`.

`--use-largepages=mode`

Added in: v13.6.0, v12.17.0

Re-map the Node.js static code to large memory pages at startup. If supported on the target system, this will cause the Node.js static code to be moved onto 2 MiB pages instead of 4 KiB pages.

The following values are valid for `mode`:

- `off`: No mapping will be attempted. This is the default.
- `on`: If supported by the OS, mapping will be attempted. Failure to map will be ignored and a message will be printed to standard error.
- `silent`: If supported by the OS, mapping will be attempted. Failure to map will be ignored and will not be reported.

`--v8-options`

Added in: v0.1.3

Print V8 command-line options.

`--v8-pool-size=num`

Added in: v5.10.0

Set V8's thread pool size which will be used to allocate background jobs.

If set to `0` then Node.js will choose an appropriate size of the thread pool based on an estimate of the amount of parallelism.

The amount of parallelism refers to the number of computations that can be carried out simultaneously in a given machine. In general, it's the same as the amount of CPUs, but it may diverge in environments such as VMs or containers.

`-v`, `--version`

Added in: v0.1.3

Print node's version.

`--watch`

▶ History

Stability: 2 - Stable

Starts Node.js in watch mode. When in watch mode, changes in the watched files cause the Node.js process to restart. By default, watch mode will watch the entry point and any required or imported module. Use `--watch-path` to specify what paths to watch.

This flag cannot be combined with `--check`, `--eval`, `--interactive`, or the REPL.

```
node --watch index.js
```

COPY

`--watch-path`

▶ History

Stability: 2 - Stable

Starts Node.js in watch mode and specifies what paths to watch. When in watch mode, changes in the watched paths cause the Node.js process to restart. This will turn off watching of required or imported modules, even when used in combination with `--watch`.

This flag cannot be combined with `--check`, `--eval`, `--interactive`, `--test`, or the REPL.

```
node --watch-path=./src --watch-path=./tests index.js
```

COPY

This option is only supported on macOS and Windows. An `ERR_FEATURE_UNAVAILABLE_ON_PLATFORM` exception will be thrown when the option is used on a platform that does not support it.

`--watch-preserve-output`

Added in: v19.3.0, v18.13.0

Disable the clearing of the console when watch mode restarts the process.

```
node --watch --watch-preserve-output test.js
```

COPY

`--zero-fill-buffers`

Added in: v6.0.0

Automatically zero-fills all newly allocated `Buffer` and `SlowBuffer` instances.

Environment variables

`FORCE_COLOR=[1, 2, 3]`

The `FORCE_COLOR` environment variable is used to enable ANSI colorized output. The value may be:

- `1`, `true`, or the empty string `''` indicate 16-color support,
- `2` to indicate 256-color support, or
- `3` to indicate 16 million-color support.

When `FORCE_COLOR` is used and set to a supported value, both the `NO_COLOR`, and `NODE_DISABLE_COLORS` environment variables are ignored.

Any other value will result in colorized output being disabled.

`NO_COLOR=<any>`

`NO_COLOR` is an alias for `NODE_DISABLE_COLORS`. The value of the environment variable is arbitrary.

`NODE_COMPILE_CACHE=dir`

Added in: v22.1.0

Stability: 1 .1 - Active Development

Enable the module compile cache for the Node.js instance. See the documentation of module compile cache for details.

`NODE_DEBUG=module[,...]`

Added in: v0.1.32

`','`-separated list of core modules that should print debug information.

`NODE_DEBUG_NATIVE=module[,...]`

`','`-separated list of core C++ modules that should print debug information.

`NODE_DISABLE_COLORS=1`

Added in: v0.3.0

When set, colors will not be used in the REPL.

`NODE_DISABLE_COMPILE_CACHE=1`

Added in: v22.8.0

Stability: 1 .1 - Active Development

Disable the module compile cache for the Node.js instance. See the documentation of module compile cache for details.

`NODE_EXTRA_CA_CERTS=file`

Added in: v7.3.0

When set, the well known "root" CAs (like VeriSign) will be extended with the extra certificates in `file`. The file should consist of one or more trusted certificates in PEM format. A message will be emitted (once) with `process.emitWarning()` if the file is missing or malformed, but any errors are otherwise ignored.

Neither the well known nor extra certificates are used when the `ca` options property is explicitly specified for a TLS or HTTPS client or server.

This environment variable is ignored when `node` runs as setuid root or has Linux file capabilities set.

The `NODE_EXTRA_CA_CERTS` environment variable is only read when the Node.js process is first launched. Changing the value at runtime using `process.env.NODE_EXTRA_CA_CERTS` has no effect on the current process.

NODE_ICU_DATA=file

Added in: v0.11.15

Data path for ICU (`Intl` object) data. Will extend linked-in data when compiled with small-icu support.

NODE_NO_WARNINGS=1

Added in: v6.11.0

When set to `1`, process warnings are silenced.

NODE_OPTIONS=options...

Added in: v8.0.0

A space-separated list of command-line options. `options...` are interpreted before command-line options, so command-line options will override or compound after anything in `options...`. Node.js will exit with an error if an option that is not allowed in the environment is used, such as `-p` or a script file.

If an option value contains a space, it can be escaped using double quotes:

```
NODE_OPTIONS='--require "./my path/file.js"'
```

COPY

A singleton flag passed as a command-line option will override the same flag passed into `NODE_OPTIONS`:

```
# The inspector will be available on port 5555
NODE_OPTIONS='--inspect=localhost:4444' node --inspect=localhost:5555
```

COPY

A flag that can be passed multiple times will be treated as if its `NODE_OPTIONS` instances were passed first, and then its command-line instances afterwards:

```
NODE_OPTIONS='--require "./a.js"' node --require "./b.js"
# is equivalent to:
node --require "./a.js" --require "./b.js"
```

COPY

Node.js options that are allowed are in the following list. If an option supports both --XX and --no-XX variants, they are both supported but only one is included in the list below.

- `--allow-addons`
- `--allow-child-process`
- `--allow-fs-read`
- `--allow-fs-write`
- `--allow-wasi`
- `--allow-worker`
- `--conditions`, `-C`
- `--diagnostic-dir`
- `--disable-proto`
- `--disable-warning`
- `--disable-wasm-trap-handler`
- `--dns-result-order`
- `--enable-fips`
- `--enable-network-family-autoselection`
- `--enable-source-maps`
- `--entry-url`
- `--experimental-abortcontroller`

- `--experimental-async-context-frame`
- `--experimental-default-type`
- `--experimental-detect-module`
- `--experimental-eventsource`
- `--experimental-import-meta-resolve`
- `--experimental-json-modules`
- `--experimental-loader`
- `--experimental-modules`
- `--experimental-permission`
- `--experimental-print-required-tla`
- `--experimental-require-module`
- `--experimental-shadow-realm`
- `--experimental-specifier-resolution`
- `--experimental-sqlite`
- `--experimental-strip-types`
- `--experimental-top-level-await`
- `--experimental-transform-types`
- `--experimental-vm-modules`
- `--experimental-wasi-unstable-preview1`
- `--experimental-wasm-modules`
- `--experimental-webstorage`
- `--force-context-aware`
- `--force-fips`
- `--force-node-api-uncaught-exceptions-policy`
- `--frozen-intrinsics`
- `--heap-prof-dir`
- `--heap-prof-interval`
- `--heap-prof-name`
- `--heap-prof`
- `--heapsnapshot-near-heap-limit`
- `--heapsnapshot-signal`
- `--http-parser`
- `--icu-data-dir`
- `--import`
- `--input-type`
- `--insecure-http-parser`
- `--inspect-brk`
- `--inspect-port`, `--debug-port`
- `--inspect-publish-uid`
- `--inspect-wait`
- `--inspect`
- `--localstorage-file`
- `--max-http-header-size`
- `--napi-modules`
- `--network-family-autoselection-attempt-timeout`
- `--no-addons`

- `--no-deprecation`
- `--no-experimental-global-navigator`
- `--no-experimental-repl-await`
- `--no-experimental-websocket`
- `--no-extra-info-on-fatal-exception`
- `--no-force-async-hooks-checks`
- `--no-global-search-paths`
- `--no-network-family-autoselection`
- `--no-warnings`
- `--node-memory-debug`
- `--openssl-config`
- `--openssl-legacy-provider`
- `--openssl-shared-config`
- `--pending-deprecation`
- `--preserve-symlinks-main`
- `--preserve-symlinks`
- `--prof-process`
- `--redirect-warnings`
- `--report-compact`
- `--report-dir`, `--report-directory`
- `--report-exclude-network`
- `--report-filename`
- `--report-on-fatalerror`
- `--report-on-signal`
- `--report-signal`
- `--report-uncaught-exception`
- `--require`, `-r`
- `--secure-heap-min`
- `--secure-heap`
- `--snapshot-blob`
- `--test-coverage-branches`
- `--test-coverage-exclude`
- `--test-coverage-functions`
- `--test-coverage-include`
- `--test-coverage-lines`
- `--test-name-pattern`
- `--test-only`
- `--test-reporter-destination`
- `--test-reporter`
- `--test-shard`
- `--test-skip-pattern`
- `--throw-deprecation`
- `--title`
- `--tls-cipher-list`
- `--tls-keylog`
- `--tls-max-v1.2`

- `--tls-max-v1.3`
- `--tls-min-v1.0`
- `--tls-min-v1.1`
- `--tls-min-v1.2`
- `--tls-min-v1.3`
- `--trace-deprecation`
- `--trace-event-categories`
- `--trace-event-file-pattern`
- `--trace-events-enabled`
- `--trace-exit`
- `--trace-sigint`
- `--trace-sync-io`
- `--trace-tls`
- `--trace-uncaught`
- `--trace-warnings`
- `--track-heap-objects`
- `--unhandled-rejections`
- `--use-bundled-ca`
- `--use-largepages`
- `--use-openssl-ca`
- `--v8-pool-size`
- `--watch-path`
- `--watch-preserve-output`
- `--watch`
- `--zero-fill-buffers`

V8 options that are allowed are:

- `--abort-on-uncaught-exception`
- `--disallow-code-generation-from-strings`
- `--enable-etw-stack-walking`
- `--expose-gc`
- `--interpreted-frames-native-stack`
- `--jitless`
- `--max-old-space-size`
- `--max-semi-space-size`
- `--perf-basic-prof-only-functions`
- `--perf-basic-prof`
- `--perf-prof-unwinding-info`
- `--perf-prof`
- `--stack-trace-limit`

`--perf-basic-prof-only-functions`, `--perf-basic-prof`, `--perf-prof-unwinding-info`, and `--perf-prof` are only available on Linux.

`--enable-etw-stack-walking` is only available on Windows.

`NODE_PATH=path[:...]`

Added in: v0.1.32

`':'`-separated list of directories prefixed to the module search path.

On Windows, this is a `';'`-separated list instead.

NODE_PENDING_DEPRECATION=1

Added in: v8.0.0

When set to `1`, emit pending deprecation warnings.

Pending deprecations are generally identical to a runtime deprecation with the notable exception that they are turned *off* by default and will not be emitted unless either the `--pending-deprecation` command-line flag, or the `NODE_PENDING_DEPRECATION=1` environment variable, is set. Pending deprecations are used to provide a kind of selective "early warning" mechanism that developers may leverage to detect deprecated API usage.

NODE_PENDING_PIPE_INSTANCES=instances

Set the number of pending pipe instance handles when the pipe server is waiting for connections. This setting applies to Windows only.

NODE_PRESERVE_SYMLINKS=1

Added in: v7.1.0

When set to `1`, instructs the module loader to preserve symbolic links when resolving and caching modules.

NODE_REDIRECT_WARNINGS=file

Added in: v8.0.0

When set, process warnings will be emitted to the given file instead of printing to stderr. The file will be created if it does not exist, and will be appended to if it does. If an error occurs while attempting to write the warning to the file, the warning will be written to stderr instead. This is equivalent to using the `--redirect-warnings=file` command-line flag.

NODE_REPL_EXTERNAL_MODULE=file

▶ History

Path to a Node.js module which will be loaded in place of the built-in REPL. Overriding this value to an empty string (`''`) will use the built-in REPL.

NODE_REPL_HISTORY=file

Added in: v3.0.0

Path to the file used to store the persistent REPL history. The default path is `~/.node_repl_history`, which is overridden by this variable. Setting the value to an empty string (`''` or `' '`) disables persistent REPL history.

NODE_SKIP_PLATFORM_CHECK=value

Added in: v14.5.0

If `value` equals `'1'`, the check for a supported platform is skipped during Node.js startup. Node.js might not execute correctly. Any issues encountered on unsupported platforms will not be fixed.

NODE_TEST_CONTEXT=value

If `value` equals `'child'`, test reporter options will be overridden and test output will be sent to stdout in the TAP format. If any other value is provided, Node.js makes no guarantees about the reporter format used or its stability.

NODE_TLS_REJECT_UNAUTHORIZED=value

If `value` equals `'0'`, certificate validation is disabled for TLS connections. This makes TLS, and HTTPS by extension, insecure. The use of this environment variable is strongly discouraged.

NODE_V8_COVERAGE=dir

When set, Node.js will begin outputting V8 JavaScript code coverage and Source Map data to the directory provided as an argument (coverage information is written as JSON to files with a `coverage` prefix).

`NODE_V8_COVERAGE` will automatically propagate to subprocesses, making it easier to instrument applications that call the `child_process.spawn()` family of functions. `NODE_V8_COVERAGE` can be set to an empty string, to prevent propagation.

Coverage output

Coverage is output as an array of ScriptCoverage objects on the top-level key `result`:

```
{
  "result": [
    {
      "scriptId": "67",
      "url": "internal/tty.js",
      "functions": []
    }
  ]
}
```

COPY

Source map cache

Stability: 1 - Experimental

If found, source map data is appended to the top-level key `source-map-cache` on the JSON coverage object.

`source-map-cache` is an object with keys representing the files source maps were extracted from, and values which include the raw source-map URL (in the key `url`), the parsed Source Map v3 information (in the key `data`), and the line lengths of the source file (in the key `lineLengths`).

```
{
  "result": [
    {
      "scriptId": "68",
      "url": "file:///absolute/path/to/source.js",
      "functions": []
    }
  ],
  "source-map-cache": {
    "file:///absolute/path/to/source.js": {
      "url": "./path-to-map.json",
      "data": {
        "version": 3,
        "sources": [
          "file:///absolute/path/to/original.js"
        ],
        "names": [
          "Foo",
          "console",
          "info"
        ],
        "mappings": "MAAMA,IACJC,YAAaC",
        "sourceRoot": "./"
      },
      "lineLengths": [
        13,
        62,
```

```
        38,
        27
      ]
    }
  }
}
```

COPY

OPENSSL_CONF=file

Added in: v6.11.0

Load an OpenSSL configuration file on startup. Among other uses, this can be used to enable FIPS-compliant crypto if Node.js is built with `./configure --openssl-fips`.

If the `--openssl-config` command-line option is used, the environment variable is ignored.

SSL_CERT_DIR=dir

Added in: v7.7.0

If `--use-openssl-ca` is enabled, this overrides and sets OpenSSL's directory containing trusted certificates.

Be aware that unless the child environment is explicitly set, this environment variable will be inherited by any child processes, and if they use OpenSSL, it may cause them to trust the same CAs as node.

SSL_CERT_FILE=file

Added in: v7.7.0

If `--use-openssl-ca` is enabled, this overrides and sets OpenSSL's file containing trusted certificates.

Be aware that unless the child environment is explicitly set, this environment variable will be inherited by any child processes, and if they use OpenSSL, it may cause them to trust the same CAs as node.

TZ

▶ History

The `TZ` environment variable is used to specify the timezone configuration.

While Node.js does not support all of the various ways that `TZ` is handled in other environments, it does support basic timezone IDs (such as `'Etc/UTC'`, `'Europe/Paris'`, or `'America/New_York'`). It may support a few other abbreviations or aliases, but these are strongly discouraged and not guaranteed.

```
$ TZ=Europe/Dublin node -pe "new Date().toString()"
Wed May 12 2021 20:30:48 GMT+0100 (Irish Standard Time)
```

COPY

UV_THREADPOOL_SIZE=size

Set the number of threads used in libuv's threadpool to `size` threads.

Asynchronous system APIs are used by Node.js whenever possible, but where they do not exist, libuv's threadpool is used to create asynchronous node APIs based on synchronous system APIs. Node.js APIs that use the threadpool are:

- all `fs` APIs, other than the file watcher APIs and those that are explicitly synchronous
- asynchronous crypto APIs such as `crypto.pbkdf2()`, `crypto.scrypt()`, `crypto.randomBytes()`, `crypto.randomFill()`, `crypto.generateKeyPair()`
- `dns.lookup()`
- all `zlib` APIs, other than those that are explicitly synchronous

Because libuv's threadpool has a fixed size, it means that if for whatever reason any of these APIs takes a long time, other (seemingly unrelated) APIs that run in libuv's threadpool will experience degraded performance. In

order to mitigate this issue, one potential solution is to increase the size of libuv's threadpool by setting the `'UV_THREADPOOL_SIZE'` environment variable to a value greater than `4` (its current default value). For more information, see the libuv threadpool documentation .

Useful V8 options

V8 has its own set of CLI options. Any V8 CLI option that is provided to `node` will be passed on to V8 to handle. V8's options have *no stability guarantee*. The V8 team themselves don't consider them to be part of their formal API, and reserve the right to change them at any time. Likewise, they are not covered by the Node.js stability guarantees. Many of the V8 options are of interest only to V8 developers. Despite this, there is a small set of V8 options that are widely applicable to Node.js, and they are documented here:

`--abort-on-uncaught-exception`

`--disallow-code-generation-from-strings`

`--enable-etw-stack-walking`

`--expose-gc`

`--harmony-shadow-realm`

`--jitless`

`--interpreted-frames-native-stack`

`--prof`

`--perf-basic-prof`

`--perf-basic-prof-only-functions`

`--perf-prof`

`--perf-prof-unwinding-info`

`--max-old-space-size=SIZE` (in MiB)

Sets the max memory size of V8's old memory section. As memory consumption approaches the limit, V8 will spend more time on garbage collection in an effort to free unused memory.

On a machine with 2 GiB of memory, consider setting this to 1536 (1.5 GiB) to leave some memory for other uses and avoid swapping.

```
node --max-old-space-size=1536 index.js
```

COPY

`--max-semi-space-size=SIZE` (in MiB)

Sets the maximum semi-space size for V8's scavenge garbage collector in MiB (mebibytes). Increasing the max size of a semi-space may improve throughput for Node.js at the cost of more memory consumption.

Since the young generation size of the V8 heap is three times (see `YoungGenerationSizeFromSemiSpaceSize` in V8) the size of the semi-space, an increase of 1 MiB to semi-space applies to each of the three individual semi-spaces and causes the heap size to increase by 3 MiB. The throughput improvement depends on your workload (see #42511).

The default value is 16 MiB for 64-bit systems and 8 MiB for 32-bit systems. To get the best configuration for your application, you should try different max-semi-space-size values when running benchmarks for your application.

For example, benchmark on a 64-bit systems:

```
for MiB in 16 32 64 128; do
    node --max-semi-space-size=$MiB index.js
done
```

COPY

--security-revert

--stack-trace-limit=limit

The maximum number of stack frames to collect in an error's stack trace. Setting it to 0 disables stack trace collection. The default value is 10.

```
node --stack-trace-limit=12 -p -e "Error.stackTraceLimit" # prints 12
```

COPY