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. If no corresponding file is found, an error is thrown.

If a file is found, its path will be passed to the ECMAScript 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.
- 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 ECMAScript module loader loads the program entry point, the `node` command will only accept as input only files with `.js`, `.mjs`, or `.cjs` extensions; and with `.wasm` extensions when `--experimental-wasm-modules` is enabled.

Options

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.

`-`

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.

`--`

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`

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 `domain` module that uses it).

`--completion-bash`

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

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

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

> *Stability: 1 - Experimental*

Enable experimental support for 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
```

`--cpu-prof`

> *Stability: 1 - Experimental*

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
```

`--cpu-prof-dir`

> *Stability: 1 - Experimental*

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`

> *Stability: 1 - Experimental*

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

`--cpu-prof-name`

> *Stability: 1 - Experimental*

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-proto=mode`

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`

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 `vm` module.

`--dns-result-order=order`

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

- `ipv4first`: sets default `verbatim` `false`.
- `verbatim`: sets default `verbatim` `true`.

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

`--enable-fips`

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

`--enable-source-maps`

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` prevents `--enable-source-maps` from modifying the stack trace.

`--experimental-global-webcrypto`

Expose the Web Crypto API on the global scope.

`--experimental-import-meta-resolve`

Enable experimental `import.meta.resolve()` support.

`--experimental-loader=module`

Specify the `module` of a custom experimental ECMAScript module loader. `module` may be any string accepted as an `import` specifier.

`--experimental-network-imports`

> *Stability: 1 - Experimental*

Enable experimental support for the `https:` protocol in `import` specifiers.

`--experimental-policy`

Use the specified file as a security policy.

`--no-experimental-fetch`

Disable experimental support for the Fetch API.

`--no-experimental-repl-await`

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

`--experimental-specifier-resolution=mode`

Sets the resolution algorithm for resolving ES module specifiers. Valid options are `explicit` and `node`.

The default is `explicit`, which requires providing the full path to a module. The `node` mode enables support for optional file extensions and the ability to import a directory that has an index file.

See customizing ESM specifier resolution for example usage.

`--experimental-vm-modules`

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

`--experimental-wasi-unstable-preview1`

Enable experimental WebAssembly System Interface (WASI) support.

`--experimental-wasm-modules`

Enable experimental WebAssembly module support.

`--force-context-aware`

Disable loading native addons that are not context-aware.

`--force-fips`

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

`--frozen-intrinsics`

> *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` runs before freezing intrinsics.

`--heapsnapshot-near-heap-limit=max_count`

> *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 mu = 0.126, current mu = 0.034) allocation failure scavenge might
not succeed
[49580:0x110000000]     4845 ms: Mark-sweep 130.6 (147.8) -> 130.6 (147.8) MB, 18.8
/ 0.0 ms  (average mu = 0.088, current mu = 0.031) allocation failure scavenge might
not succeed


<--- JS stacktrace --->

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

`--heapsnapshot-signal=signal`

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 index.js
$ kill -USR2 1
$ ls
Heap.20190718.133405.15554.0.001.heapsnapshot
```

`--heap-prof`

> *Stability: 1 - Experimental*

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
```

`--heap-prof-dir`

> *Stability: 1 - Experimental*

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`

> *Stability: 1 - Experimental*

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

`--heap-prof-name`

> *Stability: 1 - Experimental*

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

`--icu-data-dir=file`

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

`--input-type=type`

This configures Node.js to interpret string input as CommonJS or as an ES module. String input is input via `--eval`, `--print`, or `STDIN`.

Valid values are `"commonjs"` and `"module"`. The default is `"commonjs"`.

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

Activate inspector on `host:port` and break at start of user script. Default `host:port` is `127.0.0.1:9229`.

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

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`.

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

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

Activate inspector on `host:port`. Default is `127.0.0.1:9229`.

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.

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-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`.

`--insecure-http-parser`

Use an insecure HTTP parser that accepts invalid HTTP headers. This may allow interoperability with non-conformant HTTP implementations. It may also allow request smuggling and other HTTP attacks that rely on invalid headers being accepted. Avoid using this option.

`--jitless`

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.

This flag is inherited from V8 and is subject to change upstream. It may disappear in a non-semver-major release.

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

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

`--napi-modules`

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

`--no-addons`

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`

Silence deprecation warnings.

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

Hide extra information on fatal exception that causes exit.

`--no-force-async-hooks-checks`

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

`--no-global-search-paths`

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

`--no-warnings`

Silence all process warnings (including deprecations).

`--node-memory-debug`

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`

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`

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

`--pending-deprecation`

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.

`--policy-integrity=sri`

> *Stability: 1 - Experimental*

Instructs Node.js to error prior to running any code if the policy does not have the specified integrity. It expects a Subresource Integrity string as a parameter.

`--preserve-symlinks`

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
```

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`

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.

`--prof`

Generate V8 profiler output.

`--prof-process`

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

`--redirect-warnings=file`

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`

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`

Location at which the report will be generated.

`--report-filename=filename`

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

`--report-on-fatalerror`

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`

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`

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

`--report-uncaught-exception`

Enables report to be generated on uncaught exceptions. Useful when inspecting the JavaScript stack in conjunction with native stack and other runtime environment data.

`--secure-heap=n`

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`

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.

`--test-only`

Configures the test runner to only execute top level tests that have the `only` option set.

`--throw-deprecation`

Throw errors for deprecations.

`--title=title`

Set `process.title` on startup.

`--tls-cipher-list=list`

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

`--tls-keylog=file`

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`

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

`--tls-max-v1.3`

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

`--tls-min-v1.0`

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

`--tls-min-v1.1`

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

`--tls-min-v1.2`

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`

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-atomics-wait`

Print short summaries of calls to `Atomics.wait()` to stderr. The output could look like this:

```
(node:15701) [Thread 0] Atomics.wait(&lt;address> + 0, 1, inf) started
(node:15701) [Thread 0] Atomics.wait(&lt;address> + 0, 1, inf) did not wait because
the values mismatched
(node:15701) [Thread 0] Atomics.wait(&lt;address> + 0, 0, 10) started
(node:15701) [Thread 0] Atomics.wait(&lt;address> + 0, 0, 10) timed out
(node:15701) [Thread 0] Atomics.wait(&lt;address> + 4, 0, inf) started
(node:15701) [Thread 1] Atomics.wait(&lt;address> + 4, -1, inf) started
(node:15701) [Thread 0] Atomics.wait(&lt;address> + 4, 0, inf) was woken up by
another thread
(node:15701) [Thread 1] Atomics.wait(&lt;address> + 4, -1, inf) was woken up by
another thread
```

The fields here correspond to:

- The thread id as given by `worker_threads.threadId`
- The base address of the `SharedArrayBuffer` in question, as well as the byte offset corresponding to the index passed to `Atomics.wait()`
- The expected value that was passed to `Atomics.wait()`
- The timeout passed to `Atomics.wait`

`--trace-deprecation`

Print stack traces for deprecations.

`--trace-event-categories`

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

`--trace-event-file-pattern`

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

`--trace-events-enabled`

Enables the collection of trace event tracing information.

`--trace-exit`

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

`--trace-sigint`

Prints a stack trace on SIGINT.

`--trace-sync-io`

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

`--trace-tls`

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

`--trace-uncaught`

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`

Print stack traces for process warnings (including deprecations).

`--track-heap-objects`

Track heap object allocations for heap snapshots.

`--unhandled-rejections=mode`

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`

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`

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`

Print V8 command-line options.

`--v8-pool-size=num`

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

If set to `0` then V8 will choose an appropriate size of the thread pool based on the number of online processors.

If the value provided is larger than V8's maximum, then the largest value will be chosen.

`--zero-fill-buffers`

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

`-c`, `--check`

Syntax check the script without executing.

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

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.

`-h`, `--help`

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

`-i`, `--interactive`

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

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

Identical to `-e` but prints the result.

`-r`, `--require module`

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. Attempting to preload a ES6 Module using `--require` will fail with an error.

`-v`, `--version`

Print node's version.

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.

`NODE_DEBUG=module[,…]`

`','`-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`

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

`NODE_EXTRA_CA_CERTS=file`

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`

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

`NODE_NO_WARNINGS=1`

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

`NODE_OPTIONS=options...`

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"'
```

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
```

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"
```

Node.js options that are allowed are:

- `--conditions`, `-C`
- `--diagnostic-dir`
- `--disable-proto`
- `--dns-result-order`
- `--enable-fips`
- `--enable-source-maps`
- `--experimental-abortcontroller`
- `--experimental-global-webcrypto`
- `--experimental-import-meta-resolve`
- `--experimental-json-modules`
- `--experimental-loader`
- `--experimental-modules`
- `--experimental-network-imports`
- `--experimental-policy`
- `--experimental-specifier-resolution`
- `--experimental-top-level-await`
- `--experimental-vm-modules`
- `--experimental-wasi-unstable-preview1`
- `--experimental-wasm-modules`
- `--force-context-aware`
- `--force-fips`
- `--frozen-intrinsics`
- `--heapsnapshot-near-heap-limit`
- `--heapsnapshot-signal`
- `--http-parser`
- `--icu-data-dir`
- `--input-type`
- `--insecure-http-parser`
- `--inspect-brk`
- `--inspect-port`, `--debug-port`
- `--inspect-publish-uid`
- `--inspect`
- `--max-http-header-size`
- `--napi-modules`
- `--no-addons`
- `--no-deprecation`
- `--no-experimental-fetch`
- `--no-experimental-repl-await`

- `--no-extra-info-on-fatal-exception`
- `--no-force-async-hooks-checks`
- `--no-global-search-paths`
- `--no-warnings`
- `--node-memory-debug`
- `--openssl-config`
- `--openssl-legacy-provider`
- `--pending-deprecation`
- `--policy-integrity`
- `--preserve-symlinks-main`
- `--preserve-symlinks`
- `--prof-process`
- `--redirect-warnings`
- `--report-compact`
- `--report-dir`, `--report-directory`
- `--report-filename`
- `--report-on-fatalerror`
- `--report-on-signal`
- `--report-signal`
- `--report-uncaught-exception`
- `--require`, `-r`
- `--secure-heap-min`
- `--secure-heap`
- `--test-only`
- `--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-atomics-wait`
- `--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`
- `--zero-fill-buffers`

V8 options that are allowed are:

- `--abort-on-uncaught-exception`
- `--disallow-code-generation-from-strings`
- `--huge-max-old-generation-size`
- `--interpreted-frames-native-stack`
- `--jitless`
- `--max-old-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.

`NODE_PATH=path[:…]`

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

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

`NODE_PENDING_DEPRECATION=1`

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`

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

`NODE_REDIRECT_WARNINGS=file`

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_HISTORY=file`

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_REPL_EXTERNAL_MODULE=file`

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_SKIP_PLATFORM_CHECK=value`

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_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": []
    }
  ]
}
```

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
      ]
    }
  }
}
```

`NO_COLOR=<any>`

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

`OPENSSL_CONF=file`

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`

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`

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`

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

While the Node.js support for `TZ` will not handle all of the various ways that `TZ` is handled in other environments, it will 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)
```

`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:

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

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 GB of memory, consider setting this to 1536 (1.5 GB) to leave some memory for other uses and avoid swapping.

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