NAME

node - server-side JavaScript runtime

SYNOPSIS

node [*options*] [*v8 options*] [<*program-entry-point*> | **-e** *string* | **--**] [*arguments* ...]
node inspect, [<*program-entry-point*> | **-e** *string* | <*host*>:<*port*>] ...
node [**--v8-options**]

DESCRIPTION

Node.js is a set of libraries for JavaScript which allows it to be used outside of the browser. It is primarily focused on creating simple, easy-to-build network clients and servers.

Execute **node** without arguments to start a REPL.

OPTIONS

- 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 **node:domain** module that uses it).

--allow-addons

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');

$ node --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'
}
```

--allow-child-process

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")']);

$ node --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 node:internal/main/run_main_module:17:47 {
  code: 'ERR_ACCESS_DENIED',
  permission: 'ChildProcess'
}
```

The **child_process.fork()** API inherits the execution arguments from the parent process. This means that if Node.js is started with the Permission Model enabled and the **--allow-child-process** flag is set, any child process created using **child_process.fork()** will automatically receive all relevant Permission Model flags. This behavior also applies to **child_process.spawn()**, but in that case, the flags are propagated via the **NODE_OPTIONS** environment variable rather than directly through the process arguments.

--allow-fs-read

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/**

Examples can be found in the File System Permissions documentation. The initializer module and custom **--require** modules has a implicit read permission.

$ node --permission -r custom-require.js -r custom-require-2.js index.js

- The **custom-require.js**, **custom-require-2.js**, and **index.js** will be by default in the allowed read list.

process.has('fs.read', 'index.js'); // true
process.has('fs.read', 'custom-require.js'); // true
process.has('fs.read', 'custom-require-2.js'); // true

--allow-fs-write

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.

--allow-inspector

When using the Permission Model, the process will not be able to connect through inspector protocol. Attempts to do so will throw an **ERR_ACCESS_DENIED** unless the user explicitly passes the **--allow-inspector** flag when starting Node.js. Example:

const { Session } = require('node:inspector/promises');

const session = new Session();

```
session.connect();

$ node --permission index.js
Error: connect ERR_ACCESS_DENIED Access to this API has been restricted. Use --allow-inspector to manag
  code: 'ERR_ACCESS_DENIED',
}
```

--allow-net

When using the Permission Model, the process will not be able to access network by default. Attempts to do so will throw an **ERR_ACCESS_DENIED** unless the user explicitly passes the **--allow-net** flag when starting Node.js. Example:

```
const http = require('node:http');
// Attempt to bypass the permission
const req = http.get('http://example.com', () => {});

req.on('error', (err) => {
  console.log('err', err);
});

$ node --permission index.js
Error: connect ERR_ACCESS_DENIED Access to this API has been restricted. Use --allow-net to manage perm
  code: 'ERR_ACCESS_DENIED',
}
```

--allow-wasi

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

```
$ node --permission --allow-fs-read=* index.js

Error: Access to this API has been restricted
    at node:internal/main/run_main_module:30:49 {
  code: 'ERR_ACCESS_DENIED',
  permission: 'WASI',
}
```

--allow-worker

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);

$ node --permission --allow-fs-read=* index.js

Error: Access to this API has been restricted
    at node:internal/main/run_main_module:17:47 {
  code: 'ERR_ACCESS_DENIED',
  permission: 'WorkerThreads'
}
```

--build-sea=*config*

Generates a single executable application from a JSON configuration file. The argument must be a path to the configuration file. If the path is not absolute, it is resolved relative to the current working directory. For configuration fields, cross-platform notes, and asset APIs, see the single executable application documentation.

--build-snapshot

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
```

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 snapshot'))
$ node --snapshot-blob snapshot.blob --build-snapshot snapshot.js
$ node --snapshot-blob snapshot.blob
I am from the snapshot
```

For more information, check out the **v8.startupSnapshot** API documentation. The snapshot currently only supports loding a single entrypoint during the snapshot building process, which can load built-in modules, but not additional user-land modules. Users can bundle their applications into a single script with their bundler of choice before building a snapshot. As it's complicated to ensure the serializablility of all built-in modules, which are also growing over time, only a subset of the built-in modules are well tested to be serializable during the snapshot building process. The Node.js core test suite checks that a few fairly complex applications can be snapshotted. The list of built-in modules being captured by the built-in snapshot of Node.js is considered supported. When the snapshot builder encounters a built-in module that cannot be serialized, it may crash the snapshot building process. In that case a typical workaround would be to delay loading that module until runtime, using either **v8.startupSnapshot.setDeserializeMainFunction()** or **v8.startupSnapshot.addDeserializeCallback()**. If serialization for an additional module during the snapshot building process is needed, please file a request in the Node.js issue tracker and link to it in the tracking issue for user-land snapshots.

--build-snapshot-config

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**

Syntax check the script without executing.

--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*

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

--cpu-prof

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

If **--cpu-prof-name** is specified, the provided value is used as a template for the file name. The following placeholder is supported and will be substituted at runtime:

- **${pid}** -- the current process ID

$ node --cpu-prof --cpu-prof-name 'CPU.${pid}.cpuprofile' index.js
$ ls *.cpuprofile
CPU.15293.cpuprofile

--cpu-prof-dir

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

--cpu-prof-name
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**.

--disable-sigusr1
Disable the ability of starting a debugging session by sending a **SIGUSR1** signal to the process.

--disable-warning=*code-or-type*
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**:

import sys from 'node:sys';

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

```
import sys from 'node:sys';
import vm from 'node:vm';

vm.measureMemory();

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

vm.measureMemory();
```

--disable-wasm-trap-handler

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
```

--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.

--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 **node:vm** module.

--dns-result-order=*order*

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

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

--enable-network-family-autoselection

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

--enable-source-maps

Enable Source Map 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);
};
```

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

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 'file.ts?query#hash'
node --entry-url 'data:text/javascript,console.log("Hello")'

--env-file-if-exists=*file*

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

--env-file=*file*

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

PORT=3000
Any text after a # is treated as a comment:

# This is a comment
PORT=3000 # This is also a comment
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.
Multi-line values are supported:

MULTI_LINE="THIS IS
A MULTILINE"
will result in `THIS IS\nA MULTILINE` as the value.
Export keyword before a key is ignored:

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

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

-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. It is possible to run code containing inline types unless the **--no-strip-types** flag is provided.

--experimental-addon-modules

Enable experimental import support for **.node** addons.

--experimental-config-file=*config*

If present, Node.js will look for a configuration file at the specified path. Node.js will read the configuration file and apply the settings. The configuration file should be a JSON file with the following structure. **vX.Y.Z** in the **$schema** must be replaced with the version of Node.js you are using.

```
{
  "$schema": "https://nodejs.org/dist/vX.Y.Z/docs/node-config-schema.json",
  "nodeOptions": {
    "import": [
      "amaro/strip"
    ],
    "watch-path": "src",
    "watch-preserve-output": true
  },
  "test": {
    "test-isolation": "process"
  },
  "watch": {
    "watch-preserve-output": true
  }
}
```

The configuration file supports namespace-specific options:

- The **nodeOptions** field contains CLI flags that are allowed in **NODE_OPTIONS**.
- Namespace fields like **test**, **watch**, and **permission** contain configuration specific to that

subsystem.

When a namespace is present in the configuration file, Node.js automatically enables the corresponding flag (e.g., **--test**, **--watch**, **--permission**). This allows you to configure subsystem-specific options without explicitly passing the flag on the command line. For example:

```
{
  "test": {
    "test-isolation": "process"
  }
}
```

is equivalent to:

```
node --test --test-isolation=process
```

To disable the automatic flag while still using namespace options, you can explicitly set the flag to **false** within the namespace:

```
{
  "test": {
    "test": false,
    "test-isolation": "process"
  }
}
```

No-op flags are not supported. Not all V8 flags are currently supported. It is possible to use the official JSON schema to validate the configuration file, which may vary depending on the Node.js version. Each key in the configuration file corresponds to a flag that can be passed as a command-line argument. The value of the key is the value that would be passed to the flag. For example, the configuration file above is equivalent to the following command-line arguments:

```
node --import amaro/strip --watch-path=src --watch-preserve-output --test-isolation=process
```

The priority in configuration is as follows:

- NODE_OPTIONS and command-line options
- Configuration file
- Dotenv NODE_OPTIONS

Values in the configuration file will not override the values in the environment variables and command-line options, but will override the values in the **NODE_OPTIONS** env file parsed by the **--env-file** flag. Keys cannot be duplicated within the same or different namespaces. The configuration parser will throw an error if the configuration file contains unknown keys or keys

that cannot be used in a namespace. Node.js will not sanitize or perform validation on the user-provided configuration, so **NEVER** use untrusted configuration files.

--experimental-default-config-file

If the **--experimental-default-config-file** flag is present, Node.js will look for a **node.config.json** file in the current working directory and load it as a as configuration file.

--experimental-eventsource

Enable exposition of EventSource Web API on the global scope.

--experimental-import-meta-resolve

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-inspector-network-resource

Enable experimental support for inspector network resources.

--experimental-loader=*module*

Specify the **module** containing exported asynchronous module customization hooks. **module** may be any string accepted as an **import** specifier. This feature requires **--allow-worker** if used with the Permission Model.

--experimental-network-inspection

Enable experimental support for the network inspection with Chrome DevTools.

--experimental-print-required-tla

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.

--experimental-quic

Enable experimental support for the QUIC protocol.

--experimental-sea-config

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

Use this flag to enable ShadowRealm support.

--experimental-storage-inspection
Enable experimental support for storage inspection

--experimental-test-coverage
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-module-mocks
Enable module mocking in the test runner. This feature requires **--allow-worker** if used with the Permission Model.

--experimental-transform-types
Enables the transformation of TypeScript-only syntax into JavaScript code. Implies **--enable-source-maps**.

--experimental-vm-modules
Enable experimental ES Module support in the **node:vm** module.

--experimental-wasi-unstable-preview1
Enable experimental WebAssembly System Interface (WASI) support.

--experimental-worker-inspection
Enable experimental support for the worker inspection with Chrome DevTools.

--expose-gc
This flag will expose the gc extension from V8.

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

--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**.)

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

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

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

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

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

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

--heap-prof-name

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

--heapsnapshot-near-heap-limit=*max_count*

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


<--- 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
```

-h, **--help**

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

--icu-data-dir=*file*

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

--import=*module*

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*

This configures Node.js to interpret **--eval** or **STDIN** input as CommonJS or as an ES module. Valid values are **"commonjs"**, **"module"**, **"module-typescript"** and **"commonjs-typescript"**. The **"-typescript"** values are not available with the flag **--no-strip-types**. The default is no value, or **"commonjs"** if **--no-experimental-detect-module** is passed. If **--input-type** is not provided, Node.js will try to detect the syntax with the following steps:

- Run the input as CommonJS.
- If step 1 fails, run the input as an ES module.
- If step 2 fails with a SyntaxError, strip the types.
- If step 3 fails with an error code **ERR_UNSUPPORTED_TYPESCRIPT_SYNTAX** or **ERR_INVALID_TYPESCRIPT_SYNTAX**, throw the error from step 2, including the TypeScript error in the message, else run as CommonJS.
- If step 4 fails, run the input as an ES module.

To avoid the delay of multiple syntax detection passes, the **--input-type=type** flag can be used to specify how the **--eval** input should be interpreted. 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

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-brk=*[[host:]port]*

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*

Set the **host:port** to be used when the inspector is activated. Useful when activating the inspector by sending the **SIGUSR1** signal. Except when **--disable-sigusr1** is passed. 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]*

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.

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

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.

-i, **--interactive**

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

--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.

--localstorage-file=*file*

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.

--max-http-header-size=*size*

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

--max-old-space-size-percentage=*percentage*

Sets the maximum memory size of V8's old memory section as a percentage of available system memory. This flag takes precedence over **--max-old-space-size** when both are specified. The **percentage** parameter must be a number greater than 0 and up to 100, representing the percentage of available system memory to allocate to the V8 heap. **Note:** This flag utilizes **--max-old-space-size**, which may be unreliable on 32-bit platforms due to integer overflow issues.

```
# Using 50% of available system memory
node --max-old-space-size-percentage=50 index.js

# Using 75% of available system memory
node --max-old-space-size-percentage=75 index.js
```

--napi-modules

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

--network-family-autoselection-attempt-timeout

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

--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-async-context-frame

Disables the use of **AsyncLocalStorage** backed by **AsyncContextFrame** and uses the prior implementation which relied on async_hooks. The previous model is retained for compatibility with Electron and for cases where the context flow may differ. However, if a difference in flow is found please report it.

--no-deprecation

Silence deprecation warnings.

--no-experimental-detect-module

Disable using syntax detection to determine module type.

--no-experimental-global-navigator

Disable exposition of Navigator API on the global scope.

--no-experimental-repl-await

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

--no-experimental-require-module

Legacy alias for **--no-require-module**.

--no-experimental-sqlite

Disable the experimental **node:sqlite** module.

--no-experimental-websocket

Disable exposition of **<WebSocket>** on the global scope.

--no-experimental-webstorage

Disable **Web Storage** support.

--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-network-family-autoselection

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

--no-require-module

Disable support for loading a synchronous ES module graph in **require()**. See Loading ECMAScript modules using **require()**.

--no-strip-types

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

--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.

--openssl-shared-config

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

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.

--permission

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
- Network - manageable through **--allow-net** flag
- 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

--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.

-p, **--print** *script*

Identical to **-e** but prints the result.

--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*, **-eport-directory**=*directory*

Location at which the report will be generated.

--report-exclude-env

When **--report-exclude-env** is passed the diagnostic report generated will not contain the **environmentVariables** data.

--report-exclude-network

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

--report-filename=*filename*

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

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 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.

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

$ node --run test -- --verbose

--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.

--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.

--snapshot-blob=*path*

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:

- The version, architecture, and platform of the running Node.js binary are exactly the same as that of the binary that generates the snapshot.

- 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 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
The maximum number of test files that the test runner CLI will execute concurrently. If **--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*
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
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. By default all the matching test files are excluded from the coverage report. Specifying this option will override the default behavior.

--test-coverage-functions=*threshold*
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
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*
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
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-global-setup=*module*

Specify a module that will be evaluated before all tests are executed and can be used to setup global state or fixtures for tests. See the documentation on global setup and teardown for more details.

--test-isolation=*mode*

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.

--test-name-pattern

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

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

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

--test-reporter-destination

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

--test-rerun-failures

A path to a file allowing the test runner to persist the state of the test suite between runs. The test runner will use this file to determine which tests have already succeeded or failed, allowing for re-running of failed tests without having to re-run the entire test suite. The test runner will create this file if it does not exist. See the documentation on test reruns for more details.

--test-shard

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

--test-skip-pattern

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

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

Regenerates the snapshot files used by the test runner for snapshot testing.

--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-deprecation

Print stack traces for deprecations.

--trace-env

Print information about any access to environment variables done in the current Node.js instance to stderr, including:

- The environment variable reads that Node.js does internally.
- Writes in the form of **process.env.KEY = "SOME VALUE"**.
- Reads in the form of **process.env.KEY**.
- Definitions in the form of **Object.defineProperty(process.env, 'KEY', {...})**.
- Queries in the form of **Object.hasOwn(process.env, 'KEY')**, **process.env.hasOwnProperty('KEY')** or **'KEY' in process.env**.
- Deletions in the form of **delete process.env.KEY**.
- Enumerations inf the form of **...process.env** or **Object.keys(process.env)**.

Only the names of the environment variables being accessed are printed. The values are not printed. To print the stack trace of the access, use **--trace-env-js-stack** and/or **--trace-env-native-stack**.

--trace-env-js-stack

In addition to what **--trace-env** does, this prints the JavaScript stack trace of the access.

--trace-env-native-stack

In addition to what **--trace-env** does, this prints the native stack trace of the access.

--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-require-module=*mode*

Prints information about usage of Loading ECMAScript modules using **require()**. When **mode** is **all**, all usage is printed. When **mode** is **no-node-modules**, usage from the **node_modules** folder is excluded.

--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-env-proxy

When enabled, Node.js parses the **HTTP_PROXY**, **HTTPS_PROXY** and **NO_PROXY** environment variables during startup, and tunnels requests over the specified proxy. This is equivalent to setting the **NODE_USE_ENV_PROXY=1** environment variable. When both are

set, **--use-env-proxy** takes precedence.

--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.

--use-system-ca

Node.js uses the trusted CA certificates present in the system store along with the **--use-bundled-ca** option and the **NODE_EXTRA_CA_CERTS** environment variable. On platforms other than Windows and macOS, this loads certificates from the directory and file trusted by OpenSSL, similar to **--use-openssl-ca**, with the difference being that it caches the certificates after first load. On Windows and macOS, the certificate trust policy is similar to Chromium's policy for locally trusted certificates, but with some differences: On macOS, the following settings are respected:

- Default and System Keychains.Bl -bullet
- Trust:.Bl -bullet
- Any certificate where the "When using this certificate" flag is set to "Always Trust" or
- Any certificate where the "Secure Sockets Layer (SSL)" flag is set to "Always Trust".

The certificate must also be valid, with "X.509 Basic Policy" set to "Always Trust".
On Windows, the following settings are respected:

- Local Machine (accessed via **certlm.msc**).Bl -bullet
- Trust:.Bl -bullet
- Trusted Root Certification Authorities

- Trusted People
- Enterprise Trust -> Enterprise -> Trusted Root Certification Authorities
- Enterprise Trust -> Enterprise -> Trusted People
- Enterprise Trust -> Group Policy -> Trusted Root Certification Authorities
- Enterprise Trust -> Group Policy -> Trusted People

Current User (accessed via **certmgr.msc**).Bl -bullet
Trust:.Bl -bullet
Trusted Root Certification Authorities
Enterprise Trust -> Group Policy -> Trusted Root Certification Authorities On Windows and macOS, Node.js would check that the user settings for the trusted certificates do not forbid them for TLS server authentication before using them. Node.js currently does not support distrust/revocation of certificates from another source based on system settings. On other systems, Node.js loads certificates from the default certificate file (typically **/etc/ssl/cert.pem**) and default certificate directory (typically **/etc/ssl/certs**) that the version of OpenSSL that Node.js links to respects. This typically works with the convention on major Linux distributions and other Unix-like systems. If the overriding OpenSSL environment variables (typically **SSL_CERT_FILE** and **SSL_CERT_DIR**, depending on the configuration of the OpenSSL that Node.js links to) are set, the specified paths will be used to load certificates instead. These environment variables can be used as workarounds if the conventional paths used by the version of OpenSSL Node.js links to are not consistent with the system configuration that the users have for some reason.
Print V8 command-line options.
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.
Print node's version.
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. Note: The **--watch** flag requires a file path as an argument and is incompatible with **--run** or inline script input, as **--run** takes precedence and ignores watch mode. If no file is provided, Node.js will exit with status code **9**.

node --watch index.js
Customizes the signal sent to the process on watch mode restarts.

node --watch --watch-kill-signal SIGINT test.js
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. Note: Using **--watch-path** implicitly enables **--watch**, which requires a file path and is incompatible with **--run**, as **--run** takes precedence and ignores watch mode.

node --watch-path=./src --watch-path=./tests index.js
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.
Disable the clearing of the console when watch mode restarts the process.

node --watch --watch-preserve-output test.js
Automatically zero-fills all newly allocated **Buffer** instances.

ENVIRONMENT

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_COMPILE_CACHE *dir*

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

NODE_COMPILE_CACHE_PORTABLE *1*

When set to 1, the module compile cache can be reused across different directory locations as long as the module layout relative to the cache directory remains the same.

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_DISABLE_COMPILE_CACHE *1*

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

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 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-inspector**
- **--allow-net**
- **--allow-wasi**
- **--allow-worker**
- **--conditions**, **-C**
- **--cpu-prof-dir**
- **--cpu-prof-interval**
- **--cpu-prof-name**
- **--cpu-prof**
- **--diagnostic-dir**
- **--disable-proto**

- **--disable-sigusr1**
- **--disable-warning**
- **--disable-wasm-trap-handler**
- **--dns-result-order**
- **--enable-fips**
- **--enable-network-family-autoselection**
- **--enable-source-maps**
- **--entry-url**
- **--experimental-abortcontroller**
- **--experimental-addon-modules**
- **--experimental-detect-module**
- **--experimental-eventsource**
- **--experimental-import-meta-resolve**
- **--experimental-json-modules**
- **--experimental-loader**
- **--experimental-modules**
- **--experimental-print-required-tla**
- **--experimental-quic**
- **--experimental-require-module**
- **--experimental-shadow-realm**

- **--experimental-specifier-resolution**
- **--experimental-test-isolation**
- **--experimental-top-level-await**
- **--experimental-transform-types**
- **--experimental-vm-modules**
- **--experimental-wasi-unstable-preview1**
- **--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**
- **--max-old-space-size-percentage**
- **--napi-modules**
- **--network-family-autoselection-attempt-timeout**
- **--no-addons**
- **--no-async-context-frame**
- **--no-deprecation**
- **--no-experimental-global-navigator**
- **--no-experimental-repl-await**
- **--no-experimental-sqlite**
- **--no-experimental-strip-types**
- **--no-experimental-websocket**
- **--no-experimental-webstorage**

- **--no-extra-info-on-fatal-exception**
- **--no-force-async-hooks-checks**
- **--no-global-search-paths**
- **--no-network-family-autoselection**
- **--no-strip-types**
- **--no-warnings**
- **--no-webstorage**
- **--node-memory-debug**
- **--openssl-config**
- **--openssl-legacy-provider**
- **--openssl-shared-config**
- **--pending-deprecation**
- **--permission**
- **--preserve-symlinks-main**
- **--preserve-symlinks**
- **--prof-process**
- **--redirect-warnings**
- **--report-compact**
- **--report-dir**, **--report-directory**
- **--report-exclude-env**

- **--report-exclude-network**
- **--report-filename**
- **--report-on-fatalerror**
- **--report-on-signal**
- **--report-signal**
- **--report-uncaught-exception**
- **--require-module**
- **--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-global-setup**
- **--test-isolation**
- **--test-name-pattern**
- **--test-only**

- **--test-reporter-destination**
- **--test-reporter**
- **--test-rerun-failures**
- **--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-env-js-stack**
- **--trace-env-native-stack**
- **--trace-env**
- **--trace-event-categories**

- **--trace-event-file-pattern**
- **--trace-events-enabled**
- **--trace-exit**
- **--trace-require-module**
- **--trace-sigint**
- **--trace-sync-io**
- **--trace-tls**
- **--trace-uncaught**
- **--trace-warnings**
- **--track-heap-objects**
- **--unhandled-rejections**
- **--use-bundled-ca**
- **--use-env-proxy**
- **--use-largepages**
- **--use-openssl-ca**
- **--use-system-ca**
- **--v8-pool-size**
- **--watch-kill-signal**
- **--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[:<?>]*

':'-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_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_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_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_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_USE_ENV_PROXY *1*

When enabled, Node.js parses the **HTTP_PROXY**, **HTTPS_PROXY** and **NO_PROXY** environment variables during startup, and tunnels requests over the specified proxy. This can also be enabled using the **--use-env-proxy** command-line flag. When both are set, **--use-env-proxy** takes precedence.

NODE_USE_SYSTEM_CA *1*

Node.js uses the trusted CA certificates present in the system store along with the **--use-bundled-ca** option and the **NODE_EXTRA_CA_CERTS** environment variable. This can also be enabled using the **--use-system-ca** command-line flag. When both are set, **--use-system-ca** takes precedence.

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.

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, or if **--use-system-ca** is enabled on platforms other than macOS and Windows, 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, or if **--use-system-ca** is enabled on platforms other than macOS

and Windows, 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 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)
```

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). However, setting this from inside the process using **process.env.UV_THREADPOOL_SIZE=size** is not guranteed to work as the threadpool would have been created as part of the runtime initialisation much before user code is run. For more information, see the libuv threadpool documentation.

BUGS

Bugs are tracked in GitHub Issues: **https://github.com/nodejs/node/issues**

COPYRIGHT

SEE ALSO

Website: **https://nodejs.org/**

Documentation: **https://nodejs.org/api/**

GitHub repository and issue tracker: **https://github.com/nodejs/node**