Deprecated APIs

Node.js APIs might be deprecated for any of the following reasons:

- Use of the API is unsafe.
- An improved alternative API is available.
- Breaking changes to the API are expected in a future major release.

Node.js uses three kinds of Deprecations:

- Documentation-only
- Runtime
- End-of-Life

A Documentation-only deprecation is one that is expressed only within the Node.js API docs. These generate no side-effects while running Node.js. Some Documentation-only deprecations trigger a runtime warning when launched with `--pending-deprecation` flag (or its alternative, `NODE_PENDING_DEPRECATION=1` environment variable), similarly to Runtime deprecations below. Documentation-only deprecations that support that flag are explicitly labeled as such in the list of Deprecated APIs.

A Runtime deprecation will, by default, generate a process warning that will be printed to `stderr` the first time the deprecated API is used. When the `--throw-deprecation` command-line flag is used, a Runtime deprecation will cause an error to be thrown.

An End-of-Life deprecation is used when functionality is or will soon be removed from Node.js.

Revoking deprecations

Occasionally, the deprecation of an API might be reversed. In such situations, this document will be updated with information relevant to the decision. However, the deprecation identifier will not be modified.

List of deprecated APIs

DEP0001: `http.OutgoingMessage.prototype.flush`

Type: End-of-Life

`OutgoingMessage.prototype.flush()` has been removed. Use `OutgoingMessage.prototype.flushHeaders()` instead.

DEP0002: `require('_linklist')`

Type: End-of-Life

The `_linklist` module is deprecated. Please use a userland alternative.

DEP0003: `_writableState.buffer`

Type: End-of-Life

The `_writableState.buffer` has been removed. Use `_writableState.getBuffer()` instead.

DEP0004: `CryptoStream.prototype.readyState`

Type: End-of-Life

The `CryptoStream.prototype.readyState` property was removed.

DEP0005: `Buffer()` constructor

Type: Runtime (supports `--pending-deprecation`)

The `Buffer()` function and `new Buffer()` constructor are deprecated due to API usability issues that can lead to accidental security issues.

As an alternative, use one of the following methods of constructing `Buffer` objects:

- `Buffer.alloc(size[, fill[, encoding]])`: Create a `Buffer` with *initialized* memory.
- `Buffer.allocUnsafe(size)`: Create a `Buffer` with *uninitialized* memory.
- `Buffer.allocUnsafeSlow(size)`: Create a `Buffer` with *uninitialized* memory.
- `Buffer.from(array)`: Create a `Buffer` with a copy of `array`
- `Buffer.from(arrayBuffer[, byteOffset[, length]])` - Create a `Buffer` that wraps the given `arrayBuffer`.
- `Buffer.from(buffer)`: Create a `Buffer` that copies `buffer`.
- `Buffer.from(string[, encoding])`: Create a `Buffer` that copies `string`.

Without `--pending-deprecation`, runtime warnings occur only for code not in `node_modules`. This means there will not be deprecation warnings for `Buffer()` usage in dependencies. With `--pending-deprecation`, a runtime warning results no matter where the `Buffer()` usage occurs.

DEP0006: `child_process` `options.customFds`

Type: End-of-Life

Within the `child_process` module's `spawn()`, `fork()`, and `exec()` methods, the `options.customFds` option is deprecated. The `options.stdio` option should be used instead.

DEP0007: Replace `cluster` `worker.suicide` with `worker.exitedAfterDisconnect`

Type: End-of-Life

In an earlier version of the Node.js `cluster`, a boolean property with the name `suicide` was added to the `Worker` object. The intent of this property was to provide an indication of how and why the `Worker` instance exited. In Node.js 6.0.0, the old property was deprecated and replaced with a new `worker.exitedAfterDisconnect` property. The old property name did not precisely describe the actual semantics and was unnecessarily emotion-laden.

DEP0008: `require('constants')`

Type: Documentation-only

The `constants` module is deprecated. When requiring access to constants relevant to specific Node.js builtin modules, developers should instead refer to the `constants` property exposed by the relevant module. For instance, `require('fs').constants` and `require('os').constants`.

DEP0009: `crypto.pbkdf2` without digest

Type: End-of-Life

Use of the `crypto.pbkdf2()` API without specifying a digest was deprecated in Node.js 6.0 because the method defaulted to using the non-recommended `'SHA1'` digest. Previously, a deprecation warning was printed. Starting in Node.js 8.0.0, calling `crypto.pbkdf2()` or `crypto.pbkdf2Sync()` with `digest` set to `undefined` will throw a `TypeError`.

Beginning in Node.js v11.0.0, calling these functions with `digest` set to `null` would print a deprecation warning to align with the behavior when `digest` is `undefined`.

Now, however, passing either `undefined` or `null` will throw a `TypeError`.

DEP0010: `crypto.createCredentials`

Type: End-of-Life

The `crypto.createCredentials()` API was removed. Please use `tls.createSecureContext()` instead.

DEP0011: `crypto.Credentials`

Type: End-of-Life

The `crypto.Credentials` class was removed. Please use `tls.SecureContext` instead.

DEP0012: `Domain.dispose`

Type: End-of-Life

`Domain.dispose()` has been removed. Recover from failed I/O actions explicitly via error event handlers set on the domain instead.

DEP0013: `fs` asynchronous function without callback

Type: End-of-Life

Calling an asynchronous function without a callback throws a `TypeError` in Node.js 10.0.0 onwards. See https://github.com/nodejs/node/pull/12562.

DEP0014: `fs.read` legacy String interface

Type: End-of-Life

The `fs.read()` legacy `String` interface is deprecated. Use the `Buffer` API as mentioned in the documentation instead.

DEP0015: `fs.readSync` legacy String interface

Type: End-of-Life

The `fs.readSync()` legacy `String` interface is deprecated. Use the `Buffer` API as mentioned in the documentation instead.

DEP0016: `GLOBAL` / `root`

Type: End-of-Life

The `GLOBAL` and `root` aliases for the `global` property were deprecated in Node.js 6.0.0 and have since been removed.

DEP0017: `Intl.v8BreakIterator`

Type: End-of-Life

`Intl.v8BreakIterator` was a non-standard extension and has been removed. See `Intl.Segmenter`.

DEP0018: Unhandled promise rejections

Type: End-of-Life

Unhandled promise rejections are deprecated. By default, promise rejections that are not handled terminate the Node.js process with a non-zero exit code. To change the way Node.js treats unhandled rejections, use the `--unhandled-rejections` command-line option.

DEP0019: `require('.')` resolved outside directory

Type: End-of-Life

In certain cases, `require('.')` could resolve outside the package directory. This behavior has been removed.

DEP0020: `Server.connections`

Type: End-of-Life

The `Server.connections` property was deprecated in Node.js v0.9.7 and has been removed. Please use the `Server.getConnections()` method instead.

DEP0021: `Server.listenFD`

Type: End-of-Life

The `Server.listenFD()` method was deprecated and removed. Please use `Server.listen({fd: <number>})` instead.

DEP0022: `os.tmpDir()`

Type: End-of-Life

The `os.tmpDir()` API was deprecated in Node.js 7.0.0 and has since been removed. Please use `os.tmpdir()` instead.

DEP0023: `os.getNetworkInterfaces()`

Type: End-of-Life

The `os.getNetworkInterfaces()` method is deprecated. Please use the `os.networkInterfaces()` method instead.

DEP0024: `REPLServer.prototype.convertToContext()`

Type: End-of-Life

The `REPLServer.prototype.convertToContext()` API has been removed.

DEP0025: `require('sys')`

Type: Runtime

The `sys` module is deprecated. Please use the `util` module instead.

DEP0026: `util.print()`

Type: End-of-Life

`util.print()` has been removed. Please use `console.log()` instead.

DEP0027: `util.puts()`

Type: End-of-Life

`util.puts()` has been removed. Please use `console.log()` instead.

DEP0028: `util.debug()`

Type: End-of-Life

`util.debug()` has been removed. Please use `console.error()` instead.

DEP0029: `util.error()`

Type: End-of-Life

`util.error()` has been removed. Please use `console.error()` instead.

DEP0030: `SlowBuffer`

Type: Documentation-only

The `SlowBuffer` class is deprecated. Please use `Buffer.allocUnsafeSlow(size)` instead.

DEP0031: `ecdh.setPublicKey()`

Type: Documentation-only

The `ecdh.setPublicKey()` method is now deprecated as its inclusion in the API is not useful.

DEP0032: `domain` module

Type: Documentation-only

The `domain` module is deprecated and should not be used.

DEP0033: `EventEmitter.listenerCount()`

Type: Documentation-only

The `events.listenerCount(emitter, eventName)` API is deprecated. Please use `emitter.listenerCount(eventName)` instead.

DEP0034: `fs.exists(path, callback)`

Type: Documentation-only

The `fs.exists(path, callback)` API is deprecated. Please use `fs.stat()` or `fs.access()` instead.

DEP0035: `fs.lchmod(path, mode, callback)`

Type: Documentation-only

The `fs.lchmod(path, mode, callback)` API is deprecated.

DEP0036: `fs.lchmodSync(path, mode)`

Type: Documentation-only

The `fs.lchmodSync(path, mode)` API is deprecated.

DEP0037: `fs.lchown(path, uid, gid, callback)`

Type: Deprecation revoked

The `fs.lchown(path, uid, gid, callback)` API was deprecated. The deprecation was revoked because the requisite supporting APIs were added in libuv.

DEP0038: `fs.lchownSync(path, uid, gid)`

Type: Deprecation revoked

The `fs.lchownSync(path, uid, gid)` API was deprecated. The deprecation was revoked because the requisite supporting APIs were added in libuv.

DEP0039: `require.extensions`

Type: Documentation-only

The `require.extensions` property is deprecated.

DEP0040: `punycode` module

Type: Documentation-only (supports `--pending-deprecation`)

The `punycode` module is deprecated. Please use a userland alternative instead.

DEP0041: `NODE_REPL_HISTORY_FILE` environment variable

Type: End-of-Life

The `NODE_REPL_HISTORY_FILE` environment variable was removed. Please use `NODE_REPL_HISTORY` instead.

DEP0042: `tls.CryptoStream`

Type: End-of-Life

The `tls.CryptoStream` class was removed. Please use `tls.TLSSocket` instead.

DEP0043: `tls.SecurePair`

Type: Documentation-only

The `tls.SecurePair` class is deprecated. Please use `tls.TLSSocket` instead.

DEP0044: `util.isArray()`

Type: Documentation-only

The `util.isArray()` API is deprecated. Please use `Array.isArray()` instead.

DEP0045: `util.isBoolean()`

Type: Documentation-only

The `util.isBoolean()` API is deprecated.

DEP0046: `util.isBuffer()`

Type: Documentation-only

The `util.isBuffer()` API is deprecated. Please use `Buffer.isBuffer()` instead.

DEP0047: `util.isDate()`

Type: Documentation-only

The `util.isDate()` API is deprecated.

DEP0048: `util.isError()`

Type: Documentation-only

The `util.isError()` API is deprecated.

DEP0049: `util.isFunction()`

Type: Documentation-only

The `util.isFunction()` API is deprecated.

DEP0050: `util.isNull()`

Type: Documentation-only

The `util.isNull()` API is deprecated.

DEP0051: `util.isNullOrUndefined()`

Type: Documentation-only

The `util.isNullOrUndefined()` API is deprecated.

DEP0052: `util.isNumber()`

Type: Documentation-only

The `util.isNumber()` API is deprecated.

DEP0053: `util.isObject()`

Type: Documentation-only

The `util.isObject()` API is deprecated.

DEP0054: `util.isPrimitive()`

Type: Documentation-only

The `util.isPrimitive()` API is deprecated.

DEP0055: `util.isRegExp()`

Type: Documentation-only

The `util.isRegExp()` API is deprecated.

DEP0056: `util.isString()`

Type: Documentation-only

The `util.isString()` API is deprecated.

DEP0057: `util.isSymbol()`

Type: Documentation-only

The `util.isSymbol()` API is deprecated.

DEP0058: `util.isUndefined()`

Type: Documentation-only

The `util.isUndefined()` API is deprecated.

DEP0059: `util.log()`

Type: Documentation-only

The `util.log()` API is deprecated.

DEP0060: `util._extend()`

Type: Documentation-only

The `util._extend()` API is deprecated.

DEP0061: `fs.SyncWriteStream`

Type: End-of-Life

The `fs.SyncWriteStream` class was never intended to be a publicly accessible API and has been removed. No alternative API is available. Please use a userland alternative.

DEP0062: `node --debug`

Type: End-of-Life

`--debug` activates the legacy V8 debugger interface, which was removed as of V8 5.8. It is replaced by Inspector which is activated with `--inspect` instead.

DEP0063: `ServerResponse.prototype.writeHeader()`

Type: Documentation-only

The `http` module `ServerResponse.prototype.writeHeader()` API is deprecated. Please use `ServerResponse.prototype.writeHead()` instead.

The `ServerResponse.prototype.writeHeader()` method was never documented as an officially supported API.

DEP0064: `tls.createSecurePair()`

Type: Runtime

The `tls.createSecurePair()` API was deprecated in documentation in Node.js 0.11.3. Users should use `tls.Socket` instead.

DEP0065: `repl.REPL_MODE_MAGIC` and `NODE_REPL_MODE=magic`

Type: End-of-Life

The `repl` module's `REPL_MODE_MAGIC` constant, used for `replMode` option, has been removed. Its behavior has been functionally identical to that of `REPL_MODE_SLOPPY` since Node.js 6.0.0, when V8 5.0 was imported. Please use `REPL_MODE_SLOPPY` instead.

The `NODE_REPL_MODE` environment variable is used to set the underlying `replMode` of an interactive `node` session. Its value, `magic`, is also removed. Please use `sloppy` instead.

DEP0066: `OutgoingMessage.prototype._headers`, `OutgoingMessage.prototype._headerNames`

Type: Runtime

The `http` module `OutgoingMessage.prototype._headers` and `OutgoingMessage.prototype._headerNames` properties are deprecated. Use one of the public methods (e.g. `OutgoingMessage.prototype.getHeader()`, `OutgoingMessage.prototype.getHeaders()`, `OutgoingMessage.prototype.getHeaderNames()`, `OutgoingMessage.prototype.getRawHeaderNames()`, `OutgoingMessage.prototype.hasHeader()`, `OutgoingMessage.prototype.removeHeader()`, `OutgoingMessage.prototype.setHeader()`) for working with outgoing headers.

The `OutgoingMessage.prototype._headers` and `OutgoingMessage.prototype._headerNames` properties were never documented as officially supported properties.

DEP0067: `OutgoingMessage.prototype._renderHeaders`

Type: Documentation-only

The `http` module `OutgoingMessage.prototype._renderHeaders()` API is deprecated.

The `OutgoingMessage.prototype._renderHeaders` property was never documented as an officially supported API.

DEP0068: `node debug`

Type: End-of-Life

`node debug` corresponds to the legacy CLI debugger which has been replaced with a V8-inspector based CLI debugger available through `node inspect`.

DEP0069: `vm.runInDebugContext(string)`

Type: End-of-Life

DebugContext has been removed in V8 and is not available in Node.js 10+.

DebugContext was an experimental API.

DEP0070: `async_hooks.currentId()`

Type: End-of-Life

`async_hooks.currentId()` was renamed to `async_hooks.executionAsyncId()` for clarity.

This change was made while `async_hooks` was an experimental API.

DEP0071: `async_hooks.triggerId()`

Type: End-of-Life

`async_hooks.triggerId()` was renamed to `async_hooks.triggerAsyncId()` for clarity.

This change was made while `async_hooks` was an experimental API.

DEP0072: `async_hooks.AsyncResource.triggerId()`

Type: End-of-Life

`async_hooks.AsyncResource.triggerId()` was renamed to `async_hooks.AsyncResource.triggerAsyncId()` for clarity.

This change was made while `async_hooks` was an experimental API.

DEP0073: Several internal properties of `net.Server`

Type: End-of-Life

Accessing several internal, undocumented properties of `net.Server` instances with inappropriate names is deprecated.

As the original API was undocumented and not generally useful for non-internal code, no replacement API is provided.

DEP0074: `REPLServer.bufferedCommand`

Type: End-of-Life

The `REPLServer.bufferedCommand` property was deprecated in favor of `REPLServer.clearBufferedCommand()`.

DEP0075: `REPLServer.parseREPLKeyword()`

Type: End-of-Life

`REPLServer.parseREPLKeyword()` was removed from userland visibility.

DEP0076: `tls.parseCertString()`

Type: End-of-Life

`tls.parseCertString()` was a trivial parsing helper that was made public by mistake. While it was supposed to parse certificate subject and issuer strings, it never handled multi-value Relative Distinguished Names correctly.

Earlier versions of this document suggested using `querystring.parse()` as an alternative to `tls.parseCertString()`. However, `querystring.parse()` also does not handle all certificate subjects correctly and should not be used.

DEP0077: `Module._debug()`

Type: Runtime

`Module._debug()` is deprecated.

The `Module._debug()` function was never documented as an officially supported API.

DEP0078: `REPLServer.turnOffEditorMode()`

Type: End-of-Life

`REPLServer.turnOffEditorMode()` was removed from userland visibility.

DEP0079: Custom inspection function on objects via `.inspect()`

Type: End-of-Life

Using a property named `inspect` on an object to specify a custom inspection function for `util.inspect()` is deprecated. Use `util.inspect.custom` instead. For backward compatibility with Node.js prior to version 6.4.0, both can be specified.

DEP0080: `path._makeLong()`

Type: Documentation-only

The internal `path._makeLong()` was not intended for public use. However, userland modules have found it useful. The internal API is deprecated and replaced with an identical, public `path.toNamespacedPath()` method.

DEP0081: `fs.truncate()` using a file descriptor

Type: Runtime

`fs.truncate()` `fs.truncateSync()` usage with a file descriptor is deprecated. Please use `fs.ftruncate()` or `fs.ftruncateSync()` to work with file descriptors.

DEP0082: `REPLServer.prototype.memory()`

Type: End-of-Life

`REPLServer.prototype.memory()` is only necessary for the internal mechanics of the `REPLServer` itself. Do not use this function.

DEP0083: Disabling ECDH by setting `ecdhCurve` to `false`

Type: End-of-Life.

The `ecdhCurve` option to `tls.createSecureContext()` and `tls.TLSSocket` could be set to `false` to disable ECDH entirely on the server only. This mode was deprecated in preparation for migrating to OpenSSL 1.1.0 and consistency with the client and is now unsupported. Use the `ciphers` parameter instead.

DEP0084: requiring bundled internal dependencies

Type: End-of-Life

Since Node.js versions 4.4.0 and 5.2.0, several modules only intended for internal usage were mistakenly exposed to user code through `require()`. These modules were:

- `v8/tools/codemap`
- `v8/tools/consarray`
- `v8/tools/csvparser`
- `v8/tools/logreader`
- `v8/tools/profile_view`
- `v8/tools/profile`
- `v8/tools/SourceMap`
- `v8/tools/splaytree`
- `v8/tools/tickprocessor-driver`
- `v8/tools/tickprocessor`
- `node-inspect/lib/_inspect` (from 7.6.0)
- `node-inspect/lib/internal/inspect_client` (from 7.6.0)
- `node-inspect/lib/internal/inspect_repl` (from 7.6.0)

The `v8/*` modules do not have any exports, and if not imported in a specific order would in fact throw errors. As such there are virtually no legitimate use cases for importing them through `require()`.

On the other hand, `node-inspect` can be installed locally through a package manager, as it is published on the npm registry under the same name. No source code modification is necessary if that is done.

DEP0085: AsyncHooks sensitive API

Type: End-of-Life

The AsyncHooks sensitive API was never documented and had various minor issues. Use the `AsyncResource` API instead. See https://github.com/nodejs/node/issues/15572.

DEP0086: Remove `runInAsyncIdScope`

Type: End-of-Life

`runInAsyncIdScope` doesn't emit the `'before'` or `'after'` event and can thus cause a lot of issues. See https://github.com/nodejs/node/issues/14328.

DEP0089: `require('assert')`

Type: Deprecation revoked

Importing assert directly was not recommended as the exposed functions use loose equality checks. The deprecation was revoked because use of the `assert` module is not discouraged, and the deprecation caused developer confusion.

DEP0090: Invalid GCM authentication tag lengths

Type: End-of-Life

Node.js used to support all GCM authentication tag lengths which are accepted by OpenSSL when calling `decipher.setAuthTag()`. Beginning with Node.js v11.0.0, only authentication tag lengths of 128, 120, 112, 104, 96, 64, and 32 bits are allowed. Authentication tags of other lengths are invalid per NIST SP 800-38D.

DEP0091: `crypto.DEFAULT_ENCODING`

Type: Runtime

The `crypto.DEFAULT_ENCODING` property is deprecated.

DEP0092: Top-level `this` bound to `module.exports`

Type: Documentation-only

Assigning properties to the top-level `this` as an alternative to `module.exports` is deprecated. Developers should use `exports` or `module.exports` instead.

DEP0093: `crypto.fips` is deprecated and replaced

Type: Documentation-only

The `crypto.fips` property is deprecated. Please use `crypto.setFips()` and `crypto.getFips()` instead.

DEP0094: Using `assert.fail()` with more than one argument

Type: Runtime

Using `assert.fail()` with more than one argument is deprecated. Use `assert.fail()` with only one argument or use a different `assert` module method.

DEP0095: `timers.enroll()`

Type: Runtime

`timers.enroll()` is deprecated. Please use the publicly documented `setTimeout()` or `setInterval()` instead.

DEP0096: `timers.unenroll()`

Type: Runtime

`timers.unenroll()` is deprecated. Please use the publicly documented `clearTimeout()` or `clearInterval()` instead.

DEP0097: `MakeCallback` with `domain` property

Type: Runtime

Users of `MakeCallback` that add the `domain` property to carry context, should start using the `async_context` variant of `MakeCallback` or `CallbackScope`, or the high-level `AsyncResource` class.

DEP0098: AsyncHooks embedder `AsyncResource.emitBefore` and `AsyncResource.emitAfter` APIs

Type: End-of-Life

The embedded API provided by AsyncHooks exposes `.emitBefore()` and `.emitAfter()` methods which are very easy to use incorrectly which can lead to unrecoverable errors.

Use `asyncResource.runInAsyncScope()` API instead which provides a much safer, and more convenient, alternative. See https://github.com/nodejs/node/pull/18513.

DEP0099: Async context-unaware `node::MakeCallback` C++ APIs

Type: Compile-time

Certain versions of `node::MakeCallback` APIs available to native modules are deprecated. Please use the versions of the API that accept an `async_context` parameter.

DEP0100: `process.assert()`

Type: Runtime

`process.assert()` is deprecated. Please use the `assert` module instead.

This was never a documented feature.

DEP0101: `--with-lttng`

Type: End-of-Life

The `--with-lttng` compile-time option has been removed.

DEP0102: Using `noAssert` in `Buffer#(read|write)` operations

Type: End-of-Life

Using the `noAssert` argument has no functionality anymore. All input is verified regardless of the value of `noAssert`. Skipping the verification could lead to hard-to-find errors and crashes.

DEP0103: `process.binding('util').is[...]` typechecks

Type: Documentation-only (supports `--pending-deprecation`)

Using `process.binding()` in general should be avoided. The type checking methods in particular can be replaced by using `util.types`.

This deprecation has been superseded by the deprecation of the `process.binding()` API (DEP0111).

DEP0104: `process.env` string coercion

Type: Documentation-only (supports `--pending-deprecation`)

When assigning a non-string property to `process.env`, the assigned value is implicitly converted to a string. This behavior is deprecated if the assigned value is not a string, boolean, or number. In the future, such assignment might result in a thrown error. Please convert the property to a string before assigning it to `process.env`.

DEP0105: `decipher.finaltol`

Type: End-of-Life

`decipher.finaltol()` has never been documented and was an alias for `decipher.final()`. This API has been removed, and it is recommended to use `decipher.final()` instead.

DEP0106: `crypto.createCipher` and `crypto.createDecipher`

Type: Runtime

Using `crypto.createCipher()` and `crypto.createDecipher()` should be avoided as they use a weak key derivation function (MD5 with no salt) and static initialization vectors. It is recommended to derive a key using `crypto.pbkdf2()` or `crypto.scrypt()` and to use `crypto.createCipheriv()` and `crypto.createDecipheriv()` to obtain the `Cipher` and `Decipher` objects respectively.

DEP0107: `tls.convertNPNProtocols()`

Type: End-of-Life

This was an undocumented helper function not intended for use outside Node.js core and obsoleted by the removal of NPN (Next Protocol Negotiation) support.

DEP0108: `zlib.bytesRead`

Type: Runtime

Deprecated alias for `zlib.bytesWritten`. This original name was chosen because it also made sense to interpret the value as the number of bytes read by the engine, but is inconsistent with other streams in Node.js that expose values under these names.

DEP0109: `http`, `https`, and `tls` support for invalid URLs

Type: End-of-Life

Some previously supported (but strictly invalid) URLs were accepted through the `http.request()`, `http.get()`, `https.request()`, `https.get()`, and `tls.checkServerIdentity()` APIs because those were accepted by the legacy `url.parse()` API. The mentioned APIs now use the WHATWG URL parser that requires strictly valid URLs. Passing an invalid URL is deprecated and support will be removed in the future.

DEP0110: `vm.Script` cached data

Type: Documentation-only

The `produceCachedData` option is deprecated. Use `script.createCachedData()` instead.

DEP0111: `process.binding()`

Type: Documentation-only (supports `--pending-deprecation`)

`process.binding()` is for use by Node.js internal code only.

DEP0112: `dgram` private APIs

Type: Runtime

The `dgram` module previously contained several APIs that were never meant to accessed outside of Node.js core: `Socket.prototype._handle`, `Socket.prototype._receiving`, `Socket.prototype._bindState`, `Socket.prototype._queue`, `Socket.prototype._reuseAddr`, `Socket.prototype._healthCheck()`, `Socket.prototype._stopReceiving()`, and `dgram._createSocketHandle()`.

DEP0113: `Cipher.setAuthTag()`, `Decipher.getAuthTag()`

Type: End-of-Life

`Cipher.setAuthTag()` and `Decipher.getAuthTag()` are no longer available. They were never documented and would throw when called.

DEP0114: `crypto._toBuf()`

Type: End-of-Life

The `crypto._toBuf()` function was not designed to be used by modules outside of Node.js core and was removed.

DEP0115: `crypto.prng()`, `crypto.pseudoRandomBytes()`, `crypto.rng()`

Type: Documentation-only (supports `--pending-deprecation`)

In recent versions of Node.js, there is no difference between `crypto.randomBytes()` and `crypto.pseudoRandomBytes()`. The latter is deprecated along with the undocumented aliases `crypto.prng()` and `crypto.rng()` in favor of `crypto.randomBytes()` and might be removed in a future release.

DEP0116: Legacy URL API

Type: Deprecation revoked

The Legacy URL API is deprecated. This includes `url.format()`, `url.parse()`, `url.resolve()`, and the legacy `urlObject`. Please use the WHATWG URL API instead.

DEP0117: Native crypto handles

Type: End-of-Life

Previous versions of Node.js exposed handles to internal native objects through the `_handle` property of the `Cipher`, `Decipher`, `DiffieHellman`, `DiffieHellmanGroup`, `ECDH`, `Hash`, `Hmac`, `Sign`, and `Verify` classes. The `_handle` property has been removed because improper use of the native object can lead to crashing the application.

DEP0118: `dns.lookup()` support for a falsy host name

Type: Runtime

Previous versions of Node.js supported `dns.lookup()` with a falsy host name like `dns.lookup(false)` due to backward compatibility. This behavior is undocumented and is thought to be unused in real world apps. It will become an error in future versions of Node.js.

DEP0119: `process.binding('uv').errname()` private API

Type: Documentation-only (supports `--pending-deprecation`)

`process.binding('uv').errname()` is deprecated. Please use `util.getSystemErrorName()` instead.

DEP0120: Windows Performance Counter support

Type: End-of-Life

Windows Performance Counter support has been removed from Node.js. The undocumented `COUNTER_NET_SERVER_CONNECTION()`, `COUNTER_NET_SERVER_CONNECTION_CLOSE()`, `COUNTER_HTTP_SERVER_REQUEST()`, `COUNTER_HTTP_SERVER_RESPONSE()`, `COUNTER_HTTP_CLIENT_REQUEST()`, and `COUNTER_HTTP_CLIENT_RESPONSE()` functions have been deprecated.

DEP0121: `net._setSimultaneousAccepts()`

Type: Runtime

The undocumented `net._setSimultaneousAccepts()` function was originally intended for debugging and performance tuning when using the `child_process` and `cluster` modules on Windows. The function is not generally useful and is being removed. See discussion here: https://github.com/nodejs/node/issues/18391

DEP0122: `tls` `Server.prototype.setOptions()`

Type: Runtime

Please use `Server.prototype.setSecureContext()` instead.

DEP0123: setting the TLS ServerName to an IP address

Type: Runtime

Setting the TLS ServerName to an IP address is not permitted by RFC 6066. This will be ignored in a future version.

DEP0124: using `REPLServer.rli`

Type: End-of-Life

This property is a reference to the instance itself.

DEP0125: `require('_stream_wrap')`

Type: Runtime

The `_stream_wrap` module is deprecated.

DEP0126: `timers.active()`

Type: Runtime

The previously undocumented `timers.active()` is deprecated. Please use the publicly documented `timeout.refresh()` instead. If re-referencing the timeout is necessary, `timeout.ref()` can be used with no performance impact since Node.js 10.

DEP0127: `timers._unrefActive()`

Type: Runtime

The previously undocumented and "private" `timers._unrefActive()` is deprecated. Please use the publicly documented `timeout.refresh()` instead. If unreferencing the timeout is necessary, `timeout.unref()` can be used with no performance impact since Node.js 10.

DEP0128: modules with an invalid `main` entry and an `index.js` file

Type: Runtime

Modules that have an invalid `main` entry (e.g., `./does-not-exist.js`) and also have an `index.js` file in the top level directory will resolve the `index.js` file. That is deprecated and is going to throw an error in future Node.js versions.

DEP0129: `ChildProcess._channel`

Type: Runtime

The `_channel` property of child process objects returned by `spawn()` and similar functions is not intended for public use. Use `ChildProcess.channel` instead.

DEP0130: `Module.createRequireFromPath()`

Type: End-of-Life

Use `module.createRequire()` instead.

DEP0131: Legacy HTTP parser

Type: End-of-Life

The legacy HTTP parser, used by default in versions of Node.js prior to 12.0.0, is deprecated and has been removed in v13.0.0. Prior to v13.0.0, the `--http-parser=legacy` command-line flag could be used to revert to using the legacy parser.

DEP0132: `worker.terminate()` with callback

Type: Runtime

Passing a callback to `worker.terminate()` is deprecated. Use the returned `Promise` instead, or a listener to the worker's `'exit'` event.

DEP0133: `http` `connection`

Type: Documentation-only

Prefer `response.socket` over `response.connection` and `request.socket` over `request.connection`.

DEP0134: `process._tickCallback`

Type: Documentation-only (supports `--pending-deprecation`)

The `process._tickCallback` property was never documented as an officially supported API.

DEP0135: `WriteStream.open()` and `ReadStream.open()` are internal

Type: Runtime

`WriteStream.open()` and `ReadStream.open()` are undocumented internal APIs that do not make sense to use in userland. File streams should always be opened through their corresponding factory methods `fs.createWriteStream()` and `fs.createReadStream()`) or by passing a file descriptor in options.

DEP0136: `http` `finished`

Type: Documentation-only

`response.finished` indicates whether `response.end()` has been called, not whether `'finish'` has been emitted and the underlying data is flushed.

Use `response.writableFinished` or `response.writableEnded` accordingly instead to avoid the ambiguity.

To maintain existing behavior `response.finished` should be replaced with `response.writableEnded`.

DEP0137: Closing fs.FileHandle on garbage collection

Type: Runtime

Allowing a `fs.FileHandle` object to be closed on garbage collection is deprecated. In the future, doing so might result in a thrown error that will terminate the process.

Please ensure that all `fs.FileHandle` objects are explicitly closed using `FileHandle.prototype.close()` when the `fs.FileHandle` is no longer needed:

```
const fsPromises = require('fs').promises;
async function openAndClose() {
  let filehandle;
  try {
    filehandle = await fsPromises.open('thefile.txt', 'r');
  } finally {
    if (filehandle !== undefined)
      await filehandle.close();
  }
}
```

DEP0138: `process.mainModule`

Type: Documentation-only

`process.mainModule` is a CommonJS-only feature while `process` global object is shared with non-CommonJS environment. Its use within ECMAScript modules is unsupported.

It is deprecated in favor of `require.main`, because it serves the same purpose and is only available on CommonJS environment.

DEP0139: `process.umask()` with no arguments

Type: Documentation-only

Calling `process.umask()` with no argument causes the process-wide umask to be written twice. This introduces a race condition between threads, and is a potential security vulnerability. There is no safe, cross-platform alternative API.

DEP0140: Use `request.destroy()` instead of `request.abort()`

Type: Documentation-only

Use `request.destroy()` instead of `request.abort()`.

DEP0141: `repl.inputStream` and `repl.outputStream`

Type: Documentation-only (supports `--pending-deprecation`)

The `repl` module exported the input and output stream twice. Use `.input` instead of `.inputStream` and `.output` instead of `.outputStream`.

DEP0142: `repl._builtinLibs`

Type: Documentation-only

The `repl` module exports a `_builtinLibs` property that contains an array with native modules. It was incomplete so far and instead it's better to rely upon `require('module').builtinModules`.

DEP0143: `Transform._transformState`

Type: Runtime `Transform._transformState` will be removed in future versions where it is no longer required due to simplification of the implementation.

DEP0144: `module.parent`

Type: Documentation-only (supports `--pending-deprecation`)

A CommonJS module can access the first module that required it using `module.parent`. This feature is deprecated because it does not work consistently in the presence of ECMAScript modules and because it gives an inaccurate representation of the CommonJS module graph.

Some modules use it to check if they are the entry point of the current process. Instead, it is recommended to compare `require.main` and `module`:

```
if (require.main === module) {
  // Code section that will run only if current file is the entry point.
}
```

When looking for the CommonJS modules that have required the current one, `require.cache` and `module.children` can be used:

```
const moduleParents = Object.values(require.cache)
  .filter((m) => m.children.includes(module));
```

DEP0145: `socket.bufferSize`

Type: Documentation-only

`socket.bufferSize` is just an alias for `writable.writableLength`.

DEP0146: `new crypto.Certificate()`

Type: Documentation-only

The `crypto.Certificate()` constructor is deprecated. Use static methods of `crypto.Certificate()` instead.

DEP0147: `fs.rmdir(path, { recursive: true })`

Type: Runtime

In future versions of Node.js, `recursive` option will be ignored for `fs.rmdir`, `fs.rmdirSync`, and `fs.promises.rmdir`.

Use `fs.rm(path, { recursive: true, force: true })`, `fs.rmSync(path, { recursive: true, force: true })` or `fs.promises.rm(path, { recursive: true, force: true })` instead.

DEP0148: Folder mappings in `"exports"` (trailing `"/"`)

Type: Runtime

Using a trailing `"/"` to define subpath folder mappings in the subpath exports or subpath imports fields is deprecated. Use subpath patterns instead.

DEP0149: `http.IncomingMessage#connection`

Type: Documentation-only.

Prefer `message.socket` over `message.connection`.

DEP0150: Changing the value of `process.config`

Type: Runtime

The `process.config` property provides access to Node.js compile-time settings. However, the property is mutable and therefore subject to tampering. The ability to change the value will be removed in a future version of Node.js.

DEP0151: Main index lookup and extension searching

Type: Runtime

Previously, `index.js` and extension searching lookups would apply to `import 'pkg'` main entry point resolution, even when resolving ES modules.

With this deprecation, all ES module main entry point resolutions require an explicit `"exports"` or `"main"` entry with the exact file extension.

DEP0152: Extension PerformanceEntry properties

Type: Runtime

The `'gc'`, `'http2'`, and `'http'` {PerformanceEntry} object types have additional properties assigned to them that provide additional information. These properties are now available within the standard `detail` property of the `PerformanceEntry` object. The existing accessors have been deprecated and should no longer be used.

DEP0153: `dns.lookup` and `dnsPromises.lookup` options type coercion

Type: Runtime

Using a non-nullish non-integer value for `family` option, a non-nullish non-number value for `hints` option, a non-nullish non-boolean value for `all` option, or a non-nullish non-boolean value for `verbatim` option in `dns.lookup()` and `dnsPromises.lookup()` is deprecated.

DEP0154: RSA-PSS generate key pair options

Type: Documentation-only (supports `--pending-deprecation`)

The `'hash'` and `'mgf1Hash'` options are replaced with `'hashAlgorithm'` and `'mgf1HashAlgorithm'`.

DEP0155: Trailing slashes in pattern specifier resolutions

Type: Runtime

The remapping of specifiers ending in `"/"` like `import 'pkg/x/'` is deprecated for package `"exports"` and `"imports"` pattern resolutions.

DEP0156: `.aborted` property and `'abort'`, `'aborted'` event in `http`

Type: Documentation-only

Move to {Stream} API instead, as the `http.ClientRequest`, `http.ServerResponse`, and `http.IncomingMessage` are all stream-based. Check `stream.destroyed` instead of the `.aborted` property, and listen for `'close'` instead of `'abort'`, `'aborted'` event.

The `.aborted` property and `'abort'` event are only useful for detecting `.abort()` calls. For closing a request early, use the Stream `.destroy([error])` then check the `.destroyed` property and `'close'` event should have the same effect. The receiving end should also check the `readable.readableEnded` value on `http.IncomingMessage` to get whether it was an aborted or graceful destroy.

DEP0157: Thenable support in streams

Type: Documentation-only

An undocumented feature of Node.js streams was to support thenables in implementation methods. This is now deprecated, use callbacks instead and avoid use of async function for streams implementation methods.

This feature caused users to encounter unexpected problems where the user implements the function in callback style but uses e.g. an async method which would cause an error since mixing promise and callback semantics is not valid.

```
const w = new Writable({
  async final(callback) {
    await someOp();
    callback();
  }
});
```

DEP0158: `buffer.slice(start, end)`

Type: Documentation-only

This method was deprecated because it is not compatible with `Uint8Array.prototype.slice()`, which is a superclass of `Buffer`.

Use `buffer.subarray` which does the same thing instead.

DEP0159: `ERR_INVALID_CALLBACK`

Type: End-of-Life

This error code was removed due to adding more confusion to the errors used for value type validation.

DEP0160: `process.on('multipleResolves', handler)`

Type: Runtime.

This event was deprecated because it did not work with V8 promise combinators which diminished its usefulness.

DEP0161: `process._getActiveRequests()` and `process._getActiveHandles()`

Type: Documentation-only

The `process._getActiveHandles()` and `process._getActiveRequests()` functions are not intended for public use and can be removed in future releases.

Use `process.getActiveResourcesInfo()` to get a list of types of active resources and not the actual references.

DEP0162: `fs.write()`, `fs.writeFileSync()` coercion to string

Type: Documentation-only

Implicit coercion of objects with own `toString` property, passed as second parameter in `fs.write()`, `fs.writeFile()`, `fs.appendFile()`, `fs.writeFileSync()`, and `fs.appendFileSync()` is deprecated. Convert them to primitive strings.