node-sass DT

7.0.1 • Public • Published 2 months ago

Readme

Explore BETA

15 Dependencies

11,944 Dependents

144 Versions

node-sass

Warning: LibSass and Node Sass are deprecated. While they will continue to receive maintenance releases indefinitely, there are no plans to add additional features or compatibility with any new CSS or Sass features. Projects that still use it should move onto Dart Sass.

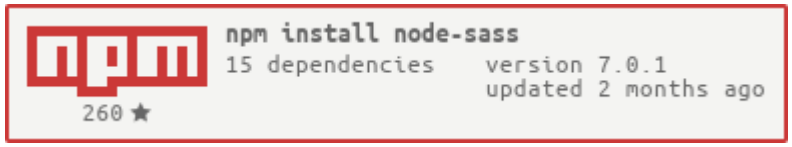
Node version support policy

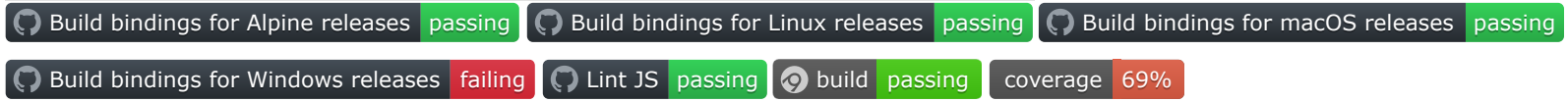
1. Supported Node.js versions vary by release, please consult the **releases page**.
2. Node versions that hit end of life **https://github.com/nodejs/Release**, will be dropped from support at each node-sass release (major, minor).
3. We will stop building binaries for unsupported releases, testing for breakages in dependency compatibility, but we will not block installations for those that want to support themselves.
4. New node release require minor internal changes along with support from CI providers (AppVeyor, GitHub Actions). We will open a single issue for interested parties to subscribe to, and close additional issues.

Below is a quick guide for minimum and maximum supported versions of node-sass:

NodeJS	Supported node-sass version	Node Module
Node 17	7.0+	102
Node 16	6.0+	93
Node 15	5.0+, <7.0	88
Node 14	4.14+	83
Node 13	4.13+, <5.0	79
Node 12	4.12+	72
Node 11	4.10+, <5.0	67

NodeJS	Supported node-sass version	Node Module
Node 10	4.9+, <6.0	64
Node 8	4.5.3+, <5.0	57
Node <8	<5.0	<57





Node-sass is a library that provides binding for Node.js to **LibSass**, the C version of the popular stylesheet preprocessor, Sass.

It allows you to natively compile .scss files to css at incredible speed and automatically via a connect middleware.

Find it on npm: **https://www.npmjs.com/package/node-sass**

Follow @nodesass on twitter for release updates: **https://twitter.com/nodesass**

Install

```
npm install node-sass
```

Some users have reported issues installing on Ubuntu due to `node` being registered to another package. **Follow the official NodeJS docs** to install NodeJS so that `#!/usr/bin/env node` correctly resolves.

Compiling on Windows machines requires the **node-gyp prerequisites**.

Are you seeing the following error? Check out our **Troubleshooting guide**.**

```
SyntaxError: Use of const in strict mode.
```

Having installation troubles? Check out our Troubleshooting guide.

Install from mirror in China

```
npm install -g mirror-config-china --registry=http://registry.npm.taobao.org
npm install node-sass
```

Usage

```
var sass = require('node-sass');
sass.render({
  file: scss_filename,
  [, options..]
}, function(err, result) { /*...*/ });
// OR
var result = sass.renderSync({
```

```
  data: scss_content
  [, options..]
});
```

Options

file

- Type: `String`
- Default: `null`

Special: `file` or `data` must be specified

Path to a file for **LibSass** to compile.

data

- Type: `String`
- Default: `null`

Special: `file` or `data` must be specified

A string to pass to **LibSass** to compile. It is recommended that you use `includePaths` in conjunction with this so that **LibSass** can find files when using the `@import` directive.

importer (>= v2.0.0) - *experimental*

This is an experimental LibSass feature. Use with caution.

- Type: `Function | Function[]` signature `function(url, prev, done)`
- Default: `undefined`

Function Parameters and Information:

- `url (String)` - the path in import **as-is**, which LibSass encountered
- `prev (String)` - the previously resolved path
- `done (Function)` - a callback function to invoke on async completion, takes an object literal containing
  - `file (String)` - an alternate path for LibSass to use **OR**
  - `contents (String)` - the imported contents (for example, read from memory or the file system)

Handles when **LibSass** encounters the `@import` directive. A custom importer allows extension of the **LibSass** engine in both a synchronous and asynchronous manner. In both cases, the goal is to either `return` or call `done()` with an object literal. Depending on the value of the object literal, one of two things will happen.

When returning or calling `done()` with `{ file: "String" }`, the new file path will be assumed for the `@import`. It's recommended to be mindful of the value of `prev` in instances where relative path resolution may be required.

When returning or calling `done()` with `{ contents: "String" }`, the string value will be used as if the file was read in through an external source.

Starting from v3.0.0:

- `this` refers to a contextual scope for the immediate run of `sass.render` or `sass.renderSync`

- importers can return error and LibSass will emit that error in response. For instance:

```
done(new Error('doesn\'t exist!'));
// or return synchronously
return new Error('nothing to do here');
```

- importer can be an array of functions, which will be called by LibSass in the order of their occurrence in array. This helps user specify special importer for particular kind of path (filesystem, http). If an importer does not want to handle a particular path, it should return `null`. See functions section for more details on Sass types.

functions (>= v3.0.0) - *experimental*

This is an experimental LibSass feature. Use with caution.

`functions` is an `Object` that holds a collection of custom functions that may be invoked by the sass files being compiled. They may take zero or more input parameters and must return a value either synchronously (`return ...;`) or asynchronously (`done();`). Those parameters will be instances of one of the constructors contained in the `require('node-sass').types` hash. The return value must be of one of these types as well. See the list of available types below:

types.Number(value [, unit = ""])

- `getValue()` / `setValue(value)` : gets / sets the numerical portion of the number
- `getUnit()` / `setUnit(unit)` : gets / sets the unit portion of the number

types.String(value)

- `getValue()` / `setValue(value)` : gets / sets the enclosed string

types.Color(r, g, b [, a = 1.0]) or types.Color(argb)

- `getR()` / `setR(value)` : red component (integer from `0` to `255`)
- `getG()` / `setG(value)` : green component (integer from `0` to `255`)
- `getB()` / `setB(value)` : blue component (integer from `0` to `255`)
- `getA()` / `setA(value)` : alpha component (number from `0` to `1.0`)

Example:

```
var Color = require('node-sass').types.Color,
  c1 = new Color(255, 0, 0),
  c2 = new Color(0xff0088cc);
```

types.Boolean(value)

- `getValue()` : gets the enclosed boolean
- `types.Boolean.TRUE` : Singleton instance of `types.Boolean` that holds "true"
- `types.Boolean.FALSE` : Singleton instance of `types.Boolean` that holds "false"

types.List(length [, commaSeparator = true])

- `getValue(index)` / `setValue(index, value)` : `value` must itself be an instance of one of the constructors in `sass.types` .
- `getSeparator()` / `setSeparator(isComma)` : whether to use commas as a separator
- `getLength()`

types.Map(length)

- `getKey(index)` / `setKey(index, value)`

- `getValue(index)` / `setValue(index, value)`
- `getLength()`

types.Null()

- `types.Null.NULL` : Singleton instance of `types.Null`.

Example

```
sass.renderSync({
  data: '#{headings(2,5)} { color: #08c; }',
  functions: {
    'headings($from: 0, $to: 6)': function(from, to) {
      var i, f = from.getValue(), t = to.getValue(),
          list = new sass.types.List(t - f + 1);

      for (i = f; i <= t; i++) {
        list.setValue(i - f, new sass.types.String('h' + i));
      }

      return list;
    }
  }
});
```

includePaths

- Type: `Array<String>`
- Default: `[]`

An array of paths that **LibSass** can look in to attempt to resolve your `@import` declarations. When using `data`, it is recommended that you use this.

indentedSyntax

- Type: `Boolean`
- Default: `false`

`true` values enable **Sass Indented Syntax** for parsing the data string or file.

Note: node-sass/libsass will compile a mixed library of scss and indented syntax (.sass) files with the Default setting (false) as long as .sass and .scss extensions are used in filenames.

indentType (>= v3.0.0)

- Type: `String`
- Default: `space`

Used to determine whether to use space or tab character for indentation.

indentWidth (>= v3.0.0)

- Type: `Number`
- Default: `2`
- Maximum: `10`

Used to determine the number of spaces or tabs to be used for indentation.

linefeed (>= v3.0.0)

- Type: `String`
- Default: `lf`

Used to determine whether to use `cr`, `crlf`, `lf` or `lfcr` sequence for line break.

omitSourceMapUrl

- Type: `Boolean`
- Default: `false`

Special: When using this, you should also specify `outFile` to avoid unexpected behavior.

`true` values disable the inclusion of source map information in the output file.

outFile

- Type: `String | null`
- Default: `null`

Special: Required when `sourceMap` is a truthy value

Specify the intended location of the output file. Strongly recommended when outputting source maps so that they can properly refer back to their intended files.

Attention enabling this option will **not** write the file on disk for you, it's for internal reference purpose only (to generate the map for example).

Example on how to write it on the disk

```
sass.render({
    ...
    outFile: yourPathTotheFile,
  }, function(error, result) { // node-style callback from v3.0.0 onwards
    if(!error){
      // No errors during the compilation, write this result on the disk
      fs.writeFile(yourPathTotheFile, result.css, function(err){
        if(!err){
          //file written on disk
        }
      });
    }
  });
});
```

outputStyle

- Type: `String`
- Default: `nested`
- Values: `nested`, `expanded`, `compact`, `compressed`

Determines the output format of the final CSS style.

precision

- Type: `Integer`

- Default: `5`

Used to determine how many digits after the decimal will be allowed. For instance, if you had a decimal number of `1.23456789` and a precision of `5`, the result will be `1.23457` in the final CSS.

sourceComments

- Type: `Boolean`
- Default: `false`

`true` Enables the line number and file where a selector is defined to be emitted into the compiled CSS as a comment. Useful for debugging, especially when using imports and mixins.

sourceMap

- Type: `Boolean | String | undefined`
- Default: `undefined`

Enables source map generation during `render` and `renderSync`.

When `sourceMap === true`, the value of `outFile` is used as the target output location for the source map with the suffix `.map` appended. If no `outFile` is set, `sourceMap` parameter is ignored.

When `typeof sourceMap === "string"`, the value of `sourceMap` will be used as the writing location for the file.

sourceMapContents

- Type: `Boolean`
- Default: `false`

`true` includes the `contents` in the source map information

sourceMapEmbed

- Type: `Boolean`
- Default: `false`

`true` embeds the source map as a data URI

sourceMapRoot

- Type: `String`
- Default: `undefined`

the value will be emitted as `sourceRoot` in the source map information

`render` Callback (>= v3.0.0)

node-sass supports standard node style asynchronous callbacks with the signature of `function(err, result)`. In error conditions, the `error` argument is populated with the error object. In success conditions, the `result` object is populated with an object describing the result of the render call.

Error Object

- `message` (String) - The error message.
- `line` (Number) - The line number of error.
- `column` (Number) - The column number of error.
- `status` (Number) - The status code.

- `file` (String) - The filename of error. In case `file` option was not set (in favour of `data`), this will reflect the value `stdin`.

Result Object

- `css` (Buffer) - The compiled CSS. Write this to a file, or serve it out as needed.
- `map` (Buffer) - The source map
- `stats` (Object) - An object containing information about the compile. It contains the following keys:
  - `entry` (String) - The path to the scss file, or `data` if the source was not a file
  - `start` (Number) - Date.now() before the compilation
  - `end` (Number) - Date.now() after the compilation
  - `duration` (Number) - *end* - *start*
  - `includedFiles` (Array) - Absolute paths to all related scss files in no particular order.

Examples

```
var sass = require('node-sass');
sass.render({
  file: '/path/to/myFile.scss',
  data: 'body{background:blue; a{color:black;}}',
  importer: function(url, prev, done) {
    // url is the path in import as is, which LibSass encountered.
    // prev is the previously resolved path.
    // done is an optional callback, either consume it or return value synchronously.
    // this.options contains this options hash, this.callback contains the node-style callba
    someAsyncFunction(url, prev, function(result){
      done({
        file: result.path, // only one of them is required, see section Special Behaviours.
```

```
        contents: result.data
      });
    });
    // OR
    var result = someSyncFunction(url, prev);
    return {file: result.path, contents: result.data};
  },
  includePaths: [ 'lib/', 'mod/' ],
  outputStyle: 'compressed'
}, function(error, result) { // node-style callback from v3.0.0 onwards
  if (error) {
    console.log(error.status); // used to be "code" in v2x and below
    console.log(error.column);
    console.log(error.message);
    console.log(error.line);
  }
  else {
    console.log(result.css.toString());

    console.log(result.stats);

    console.log(result.map.toString());
    // or better
    console.log(JSON.stringify(result.map)); // note, JSON.stringify accepts Buffer too
  }
```

```
});
// OR
var result = sass.renderSync({
  file: '/path/to/file.scss',
  data: 'body{background:blue; a{color:black;}}',
  outputStyle: 'compressed',
  outFile: '/to/my/output.css',
  sourceMap: true, // or an absolute or relative (to outFile) path
  importer: function(url, prev, done) {
    // url is the path in import as is, which LibSass encountered.
    // prev is the previously resolved path.
    // done is an optional callback, either consume it or return value synchronously.
    // this.options contains this options hash
    someAsyncFunction(url, prev, function(result){
      done({
        file: result.path, // only one of them is required, see section Special Behaviours.
        contents: result.data
      });
    });
    // OR
    var result = someSyncFunction(url, prev);
    return {file: result.path, contents: result.data};
  }
});
```

```
console.log(result.css);
console.log(result.map);
console.log(result.stats);
```

Special behaviours

- In the case that both `file` and `data` options are set, node-sass will give precedence to `data` and use `file` to calculate paths in sourcemaps.

Version information (>= v2.0.0)

Both `node-sass` and `libsass` version info is now exposed via the `info` method:

```
var sass = require('node-sass');

console.log(sass.info);

/*
  it will output something like:

  node-sass       2.0.1   (Wrapper)       [JavaScript]
  libsass         3.1.0   (Sass Compiler) [C/C++]
*/
```

Since node-sass >=v3.0.0 LibSass version is determined at run time.

Integrations

Listing of community uses of node-sass in build tools and frameworks.

Brackets extension

@jasonsanjose has created a Brackets extension based on node-sass: https://github.com/jasonsanjose/brackets-sass. When editing Sass files, the extension compiles changes on save. The extension also integrates with Live Preview to show Sass changes in the browser without saving or compiling.

Brunch plugin

Brunch's official sass plugin uses node-sass by default, and automatically falls back to ruby if use of Compass is detected: https://github.com/brunch/sass-brunch

Connect/Express middleware

Recompile `.scss` files automatically for connect and express based http servers.

This functionality has been moved to `node-sass-middleware` in node-sass v1.0.0

DocPad Plugin

@10xLaCroixDrinker wrote a DocPad plugin that compiles `.scss` files using node-sass: https://github.com/10xLaCroixDrinker/docpad-plugin-nodesass

Duo.js extension

@stephenway has created an extension that transpiles Sass to CSS using node-sass with duo.js https://github.com/duojs/sass

Grunt extension

@sindresorhus has created a set of grunt tasks based on node-sass: https://github.com/sindresorhus/grunt-sass

Gulp extension

@dlmanning has created a gulp sass plugin based on node-sass: https://github.com/dlmanning/gulp-sass

Harp

@sintaxi's Harp web server implicitly compiles `.scss` files using node-sass: https://github.com/sintaxi/harp

Metalsmith plugin

@stevenschobert has created a metalsmith plugin based on node-sass: https://github.com/stevenschobert/metalsmith-sass

Meteor plugin

@fourseven has created a meteor plugin based on node-sass: https://github.com/fourseven/meteor-scss

Mimosa module

@dbashford has created a Mimosa module for sass which includes node-sass: https://github.com/dbashford/mimosa-sass

Example App

There is also an example connect app here: https://github.com/andrew/node-sass-example

Rebuilding binaries

Node-sass includes pre-compiled binaries for popular platforms, to add a binary for your platform follow these steps:

Check out the project:

```
git clone --recursive https://github.com/sass/node-sass.git
cd node-sass
```

```
npm install
node scripts/build -f  # use -d switch for debug release
# if succeeded, it will generate and move
# the binary in vendor directory.
```

Command Line Interface

The interface for command-line usage is fairly simplistic at this stage, as seen in the following usage section.

Output will be sent to stdout if the `--output` flag is omitted.

Usage

`node-sass [options] <input> [output]` Or: `cat <input> | node-sass > output`

Example:

`node-sass src/style.scss dest/style.css`

Options:

```
    -w, --watch                 Watch a directory or file
    -r, --recursive             Recursively watch directories or files
    -o, --output                Output directory
    -x, --omit-source-map-url   Omit source map URL comment from output
    -i, --indented-syntax       Treat data from stdin as sass code (versus scss)
    -q, --quiet                 Suppress log output except on error
```

```
    -v, --version                Prints version info
    --output-style               CSS output style (nested | expanded | compact | compressed)
    --indent-type                Indent type for output CSS (space | tab)
    --indent-width               Indent width; number of spaces or tabs (maximum value: 10)
    --linefeed                   Linefeed style (cr | crlf | lf | lfcr)
    --source-comments            Include debug info in output
    --source-map                 Emit source map
    --source-map-contents        Embed include contents in map
    --source-map-embed           Embed sourceMappingUrl as data URI
    --source-map-root            Base path, will be emitted in source-map as is
    --include-path               Path to look for imported files
    --follow                     Follow symlinked directories
    --precision                  The amount of precision allowed in decimal numbers
    --error-bell                 Output a bell character on errors
    --importer                   Path to .js file containing custom importer
    --functions                  Path to .js file containing custom functions
    --help                       Print usage info
```

The `input` can be either a single `.scss` or `.sass`, or a directory. If the input is a directory the `--output` flag must also be supplied.

Also, note `--importer` takes the (absolute or relative to pwd) path to a js file, which needs to have a default `module.exports` set to the importer function. See our test **fixtures** for example.

The `--source-map` option accepts a boolean value, in which case it replaces destination extension with `.css.map`. It also accepts path to `.map` file and even path to the desired directory. When compiling a directory `--source-map` can either be a boolean value or a directory.

Binary configuration parameters

node-sass supports different configuration parameters to change settings related to the sass binary such as binary name, binary path or alternative download path. Following parameters are supported by node-sass:

Variable name	.npmrc parameter	Process argument	Value
SASS_BINARY_NAME	sass_binary_name	--sass-binary-name	path
SASS_BINARY_SITE	sass_binary_site	--sass-binary-site	URL
SASS_BINARY_PATH	sass_binary_path	--sass-binary-path	path
SASS_BINARY_DIR	sass_binary_dir	--sass-binary-dir	path
SASS_REJECT_UNAUTHORIZED	sass_reject_unauthorized	--sass-reject-unauthorized	value

These parameters can be used as environment variable:

- E.g. `export SASS_BINARY_SITE=http://example.com/`

As local or global **.npmrc** configuration file:

- E.g. `sass_binary_site=http://example.com/`

As a process argument:

- E.g. `npm install node-sass --sass-binary-site=http://example.com/`

If you are using self-signed certificates for your binary then `SASS_REJECT_UNAUTHORIZED` will override (rejectUnauthorized) [**https://nodejs.org/docs/latest/api/tls.html#tls_tls_createserver_options_secureconnectionlistener**].

Post-install Build

Install runs only two Mocha tests to see if your machine can use the pre-built **LibSass** which will save some time during install. If any tests fail it will build from source.

Maintainers

This module is brought to you and maintained by the following people:

- Michael Mifsud - Project Lead (Github / Twitter)
- Andrew Nesbitt (Github / Twitter)
- Dean Mao (Github / Twitter)
- Brett Wilkins (Github / Twitter)
- Keith Cirkel (Github / Twitter)
- Laurent Goderre (Github / Twitter)
- Nick Schonning (Github / Twitter)
- Adeel Mujahid (Github / Twitter)

Contributors

We <3 our contributors! A special thanks to all those who have clocked in some dev time on this project, we really appreciate your hard work. You can find **a full list of those people here.**

Note on Patches/Pull Requests

Check out our **Contributing guide**

Copyright

Keywords

css **libsass** **preprocessor** **sass** **scss** **style**

Install

```
> npm i node-sass
```

Repository

github.com/sass/node-sass

Homepage

github.com/sass/node-sass

Weekly Downloads

4,019,089

Version

7.0.1

License

MIT

Unpacked Size

4.65 MB

Total Files

340

Issues

125

Pull Requests

33

Last publish

2 months ago

Collaborators



>_Try on RunKit

Report malware

Support

Help

Advisories

Status

Contact npm

Company

About

Blog

Press

Terms & Policies

Policies

Terms of Use

Code of Conduct

Privacy