

GENERAL

[Getting Started \(/docs/getting-started\)](/docs/getting-started)

[Contributing \(/docs/contributing\)](/docs/contributing)

[Sample gulpfile \(/docs/sample-gulpfile\)](/docs/sample-gulpfile)

CLI

[Flags \(/docs/flags\)](/docs/flags)

[Running Tasks \(/docs/running-tasks\)](/docs/running-tasks)

[Compilers \(/docs/compilers\)](/docs/compilers)

[Examples \(/docs/examples\)](/docs/examples)

API

[gulp.src\(globs\[, options\]\) \(/docs/gulpsrcglobs-options\)](/docs/gulpsrcglobs-options)

[gulp.dest\(path\[, options\]\) \(/docs/gulpdestpath-options\)](/docs/gulpdestpath-options)

[gulp.symlink\(folder\[, options\]\) \(/docs/gulpsymlinkfolder-options\)](/docs/gulpsymlinkfolder-options)

[gulp.task\(\[name,\] fn\) \(/docs/gulptaskname-fn\)](/docs/gulptaskname-fn)

[gulp.parallel\(...tasks\) \(/docs/gulpparalleltasks\)](/docs/gulpparalleltasks)

[gulp.series\(...tasks\) \(/docs/gulpseriestasks\)](/docs/gulpseriestasks)

[gulp.watch\(glob\[, opts\], fn\) \(/docs/gulpwatchglob-opts-fn\)](/docs/gulpwatchglob-opts-fn)

WRITING A PLUGIN

[Guidelines \(/docs/guidelines\)](/docs/guidelines)

[Understanding Streams \(/docs/understanding-streams\)](/docs/understanding-streams)

[Files \(/docs/file\)](/docs/file)


RECIPES

[Using Browserify \(/docs/browserify-uglify2-with-sourcemaps\)](/docs/browserify-uglify2-with-sourcemaps)

[Pipeline Error Management \(/docs/combining-streams-to-handle-errors\)](/docs/combining-streams-to-handle-errors)

[Incremental Builds \(/docs/incremental-builds\)](/docs/incremental-builds)

Files

 [Suggest Edits \(/docs/file/edit\)](/docs/file/edit)

Each file object in the stream is an instance of a vinyl (<https://github.com/wearefractal/vinyl>) object.

JavaScript ()

```
var File = require('vinyl');

var coffeeFile = new File({
  cwd: "/",
  base: "/test/",
  path: "/test/file.coffee",
  contents: new Buffer("test = 123")
});
```

constructor(options)

options.cwd

Type: String

Default: process.cwd()

options.base

Used for relative pathing. Typically where a glob starts.

Type: String

Default: options.cwd

options.path

Full path to the file.

Type: String

Default: undefined

options.history

Path history. Has no effect if options.path is passed.

Type: Array

Default: options.path ? [options.path] : []

options.stat

The result of an `fs.stat` call. See `fs.Stats` (http://nodejs.org/api/fs.html#fs_class_fs_stats) for more information.

Type: `fs.Stats`

Default: `null`

options.contents

File contents.

Type: `Buffer`, `Stream`, or `null`

Default: `null`

isBuffer()

Returns true if `file.contents` is a `Buffer`.

isStream()

Returns true if `file.contents` is a `Stream`.

isNull()

Returns true if `file.contents` is `null`.

clone([opt])

Returns a new `File` object with all attributes cloned.

By default custom attributes are deep-cloned.

If `opt` or `opt.deep` is `false`, custom attributes will not be deep-cloned.

If `opt.contents` is `false`, it will copy `file.contents` `Buffer`'s reference.

pipe(stream[, opt])

If `file.contents` is a `Buffer`, it will write it to the stream.

If `file.contents` is a `Stream`, it will pipe it to the stream.

If `file.contents` is `null`, it will do nothing.

If `opt.end` is `false`, the destination stream will not be ended (same as node core).

Returns the stream.

inspect()

Returns a pretty String interpretation of the File. Useful for debugging via console.log.

contents

The Stream (https://nodejs.org/api/stream.html#stream_stream) or Buffer (https://nodejs.org/api/buffer.html#buffer_class_buffer) of the file as it was passed in via options, or as the result of modification.

For example:

JavaScript ()

```
// logs out the string of contents
if (file.isBuffer()) {
  console.log(file.contents.toString());
}
```

path

Absolute pathname string or `undefined`. Setting to a different value pushes the old value to `history`.

history

Array of `path` values the file object has had, from `history[0]` (original) through `history[history.length - 1]` (current). `history` and its elements should normally be treated as read-only and only altered indirectly by setting `path`.

relative

Returns `path.relative` for the file base and file path.

Example:

JavaScript ()

```
var file = new File({
  cwd: "/",
  base: "/test/",
  path: "/test/file.coffee"
});

console.log(file.relative); // file.coffee
```

dirname

Gets and sets path.dirname for the file path.

Example:

JavaScript ()

```
var file = new File({
  cwd: "/",
  base: "/test/",
  path: "/test/file.coffee"
});

console.log(file.dirname); // /test

file.dirname = '/specs';

console.log(file.dirname); // /specs
console.log(file.path); // /specs/file.coffee
```

basename

Gets and sets path.basename for the file path.

Example:

JavaScript ()

```
var file = new File({
  cwd: "/",
  base: "/test/",
  path: "/test/file.coffee"
});

console.log(file.basename); // file.coffee

file.basename = 'file.js';

console.log(file.basename); // file.js
console.log(file.path); // /test/file.js
```

extname

Gets and sets path.extname for the file path.

Example: