dest()

Creates a stream for writing Vinyl objects to the file system.

Usage

```
const { src, dest } = require('gulp');

function copy() {
  return src('input/*.js')
    .pipe(dest('output/'));
}

exports.copy = copy;
```

Signature

```
dest(directory, [options])
```

Parameters

parameter	type	note
directory (required)	string function	The path of the output directory where files will be written. If a function is used, the function will be called with each Vinyl object and must return a string directory path.
options	object	Detailed in Options below.

Returns

A stream that can be used in the middle or at the end of a pipeline to create files on the file system. Whenever a Vinyl object is passed through the stream, it writes the contents and other details out to the file system at the given directory. If the Vinyl object has a symlink property, a symbolic link will be created instead of writing the contents. After the file is created, its metadata will be updated to match the Vinyl object.
Whenever a file is created on the file system, the Vinyl object will be modified.
The cwd, base, and path properties will be updated to match the created file.
The stat property will be updated to match the file on the file system.
If the contents property is a stream, it will be reset so it can be read again.

Errors

When directory is an empty string, throws an error with the message, "Invalid dest() folder argument. Please specify a non-empty string or a function."
When directory is not a string or function, throws an error with the message, "Invalid dest() folder argument. Please specify a non-empty string or a function."
When directory is a function that returns an empty string or undefined, emits an error with the message, "Invalid output folder".

Options

For options that accept a function, the passed function will be called with each Vinyl object and must return a value of another listed type.

name	type	default	note
cwd	string function	process.cwd()	The directory that will be combined with any relative path to form an absolute path. Is ignored for absolute paths. Use to avoid combining directory with path.join().
mode	number function	stat.mode of the Vinyl object	The mode used when creating files. If not set and stat.mode is missing, the process' mode will be used instead.
dirMode	number function		The mode used when creating directories. If not set, the process' mode will be used.

name	type	default	note
overwrite	boolean function	true	When true, overwrites existing files with the same path.
append	boolean function	false	If true, adds contents to the end of the file, instead of replacing existing contents.
sourcemaps	boolean string function	false	If true, writes inline sourcemaps to the output file. Specifying a string path will write external sourcemaps at the given path.
relativeSymlinks	boolean function	false	When false, any symbolic links created will be absolute. Note: Ignored if a junction is being created, as they must be absolute.
useJunctions	boolean function	true	This option is only relevant on Windows and ignored elsewhere. When true, creates directory symbolic link as a junction. Detailed in Symbolic links on Windows below.

Metadata updates

Whenever the dest() stream creates a file, the Vinyl object's mode, mtime, and atimeare compared to the created file. If they differ, the created file will be updated to reflect the Vinyl object's metadata. If those properties are the same, or gulp doesn't have permissions to make changes, the attempt is skipped silently.
This functionality is disabled on Windows or other operating systems that don't support Node's process.getuid() or process.geteuid() methods. This is due to Windows having unexpected results through usage of fs.fchmod() and fs.futimes().
Note: The fs.futimes() method internally converts mtime and atime timestamps to seconds. This division by 1000 may cause some loss of precision on 32-bit operating systems.

Sourcemaps

Sourcemap support is built directly into src() and dest(), but it is disabled by default. Enable it to produce inline or external sourcemaps.
Inline sourcemaps:

```
const { src, dest } = require('gulp');
const uglify = require('gulp-uglify');

src('input/**/*.js', { sourcemaps: true })
  .pipe(uglify())
  .pipe(dest('output/', { sourcemaps: true }));
```

External sourcemaps:

```
const { src, dest } = require('gulp');
const uglify = require('gulp-uglify');

src('input/**/*.js', { sourcemaps: true })
  .pipe(uglify())
  .pipe(dest('output/', { sourcemaps: '.' }));
```

Symbolic links on Windows

When creating symbolic links on Windows, a type argument is passed to Node's fs.symlink() method which specifies the kind of target being linked. The link type is set to:
'file' when the target is a regular file
'junction' when the target is a directory
'dir' when the target is a directory and the user disables the useJunctions option

If you try to create a dangling (pointing to a non-existent target) link, the link type can't be determined automatically. In these cases, behavior will vary depending on whether the dangling link is being created via symlink() or via dest().
For dangling links created via symlink(), the incoming Vinyl object represents the target, so its stats will determine the desired link type. If isDirectory() returns false then a 'file' link is created, otherwise a 'junction' or a 'dir' link is created depending on the value of the useJunctions option.
For dangling links created via dest(), the incoming Vinyl object represents the link - typically loaded from disk via src(..., { resolveSymlinks: false }). In this case, the link type can't be reasonably determined and defaults to using 'file'. This may cause unexpected behavior if you are creating a dangling link to a directory. Avoid this scenario.