

GitHub

This repository Search

Explore

Features

Enterprise

Pricing

Sign up

Sign in

gulpjs / gulp-util

Watch 17

★ Star 542

🍴 Fork 65

<> Code

! Issues 5

🔗 Pull requests 3

📶 Pulse

📊 Graphs

Utilities for gulp plugins

📦 210 commits

🌿 4 branches

📦 12 releases

32 contributors

Branch: master ▾

New pull request

New file

Find file

HTTPS ▾

https://github.com/gulpjs



Download ZIP



Blaine Bublitz 3.0.7

Latest commit b74a5ff on 20 Oct 2015

lib	add branching logic to support new gulplog stuff	6 months ago
test	missing quotes on previous test and space for object in current	11 months ago
.gitignore	Added additional tests to satisfy code coverage	2 years ago
.jshintrc	update dotfiles	a year ago
.travis.yml	Update .travis.yml	a year ago
LICENSE	license fixes, dep updates, badges	2 years ago
README.md	Update node version in readme	9 months ago
index.js	Merge pull request #72 from gulpjs/use-beeper-module	a year ago
package.json	3.0.7	4 months ago

📖 README.md

gulp-util

npm package 3.0.7

build passing

coverage 98%

dependencies up to date

Information

Package	gulp-util
Description	Utility functions for gulp plugins
Node Version	>= 0.10

Usage

```
var gutil = require('gulp-util');

gutil.log('stuff happened', 'Really it did', gutil.colors.magenta('123'));
gutil.beep();

gutil.replaceExtension('file.coffee', '.js'); // file.js

var opt = {
  name: 'todd',
  file: someGulpFile
};
gutil.template('test <%= name %> <%= file.path %>', opt) // test todd /js/hi.js
```

log(msg...)

Logs stuff. Already prefixed with [gulp] and all that. If you pass in multiple arguments it will join them by a space.

The default gulp coloring using `gutil.colors`:

```
values (files, module names, etc.) = cyan  
numbers (times, counts, etc) = magenta
```

colors

Is an instance of [chalk](#).

replaceExtension(path, newExtension)

Replaces a file extension in a path. Returns the new path.

isStream(obj)

Returns true or false if an object is a stream.

isBuffer(obj)

Returns true or false if an object is a Buffer.

template(string[, data])

This is a `lodash.template` function wrapper. You must pass in a valid gulp file object so it is available to the user or it will

error. You can not configure any of the delimiters. Look at the [lodash docs](#) for more info.

new File(obj)

This is just [vinyl](#)

```
var file = new gutil.File({
  base: path.join(__dirname, './fixtures/'),
  cwd: __dirname,
  path: path.join(__dirname, './fixtures/test.coffee')
});
```

noop()

Returns a stream that does nothing but pass data straight through.

```
// gulp should be called like this :
// $ gulp --type production
gulp.task('scripts', function() {
  gulp.src('src/**/*.js')
    .pipe(concat('script.js'))
    .pipe(gutil.env.type === 'production' ? uglify() : gutil.noop())
    .pipe(gulp.dest('dist/'));
});
```

buffer(cb)

This is similar to `es.wait` but instead of buffering text into one string it buffers anything into an array (so very useful for file objects).

Returns a stream that can be piped to.

The stream will emit one data event after the stream piped to it has ended. The data will be the same array passed to the callback.

Callback is optional and receives two arguments: error and data

```
gulp.src('stuff/*.js')
  .pipe(gutil.buffer(function(err, files) {

  }));
```

new PluginError(pluginName, message[, options])

- `pluginName` should be the module name of your plugin
- `message` can be a string or an existing error
- By default the stack will not be shown. Set `options.showStack` to true if you think the stack is important for your error.
- If you pass an error in as the message the stack will be pulled from that, otherwise one will be created.
- Note that if you pass in a custom stack string you need to include the message along with that.
- Error properties will be included in `err.toString()`. Can be omitted by including `{showProperties: false}` in the options.

These are all acceptable forms of instantiation:

```
var err = new gutil.PluginError('test', {
```

```
    message: 'something broke'
  });

  var err = new gutil.PluginError({
    plugin: 'test',
    message: 'something broke'
  });

  var err = new gutil.PluginError('test', 'something broke');

  var err = new gutil.PluginError('test', 'something broke', {showStack: true});

  var existingError = new Error('OMG');
  var err = new gutil.PluginError('test', existingError, {showStack: true});
```

