

[sign up or log in](#) [find packages](#)

★ gulp-sprite-generator public

Plugin that generate sprites from your stylesheets (using [spritesmith](#)) and then updates the references.

npm package 0.2.3 build passing

Generate sprites from stylesheets.

Plugin that generate sprites from your stylesheets (using [spritesmith](#)) and then updates image references.

Getting started

If you haven't used [gulp](#) before, be sure to check out the [Getting Started](#) guide.

Install with [npm](#)

```
npm install --save-dev gulp-sprite-generator
```

Overview


Sprite generator is a gulp task, which accepts options object and returns two streams for style and image piping.

Here quick example of simple way usage:

Develop front-end JavaScript?
Help us make npm
better.



Tell us how you
develop, build, and ship
front-end JavaScript,
and we'll help make it
easier. [Take the survey »](#)

 `npm i gulp-sprite-gen`
[how? learn more](#)

 [gobwas](#) published 2 year...

0.2.3 is the latest of 10 releases

[github.com/gobwas/gulp-spr...](#)

MIT 

Collaborators



Stats

6 downloads in the last day

181 downloads in the last we...

554 downloads in the last m...

```
var gulp = require('gulp');
var sprite = require('gulp-sprite-generator');

gulp.task('sprites', function() {
  var spriteOutput;

  spriteOutput = gulp.src("./src/css/*.css")
    .pipe(sprite({
      baseUrl:          "./src/image",
      spriteSheetName:  "sprite.png",
      spriteSheetPath:  "/dist/image"
    }));

  spriteOutput.css.pipe(gulp.dest("./dist/css"));
  spriteOutput.img.pipe(gulp.dest("./dist/image"));
});
```

Of course you may need to have more flexible configuration for spriting. And this plugin could give you something more!

Options

Sprite generator options is an object, that mix **spritesmith** options and plugin specific options.

Spritesmith parameters *(all is optional)*:

property	necessary	type	plugin default value
[engine]	no	String	"pngsmith"
[algorithm]	no	String	"top-down"
[padding]	no	Number	0
[engineOpts]	no	Object	{}
[exportOpts]	no	Object	{}

More detailed explanation you can find on the **official page of spritesmith**.

2 open issues on GitHub

No open pull requests on Git..

Try it out

Test gulp-sprite-generat..

Keywords

css, spritesmith, stylesheet, png, generator, sprite, spritesheet, gulpplugin

Dependencies (8)

vinyl, through2, spritesmith, q, lodash, gulp-util, colors, async

[microapps](#) is hiring. View more...

Plugin options are:

property	necessary	type	plugin default value
spriteSheetName	yes	String	null
[spriteSheetPath]	no	String	null
[styleSheetName]	no	String	null
[baseUrl]	no	String	"./"
[retina]	no	Boolean	true
[filter]	no	Function[]	[]
[groupBy]	no	Function[]	[]
[accumulate]	no	Boolean	false
[verbose]	no	Boolean	false

More detailed explanation is below.

options.spriteSheetName

Type: String Default value: null

The one and last necessary parameter. Defines which *base* will have the name of the output sprite. Base means that if you will group your sprites by some criteria, name will change.

options.spriteSheetPath

Type: String Default value: null

Can define relative path of references in the output stylesheet.

options.styleSheetName

Type: String Default value: null

Defines the name of the output stylesheet.

options.baseUrl

Type: String Default value: ./

Defines where to find relatively defined image references in the input stylesheet.

options.retina

Type: Boolean Default value: true

Defines whether or not to search for retina mark in the filename. If true then it will look for `@{number}x` syntax. For example: `image@2x.png`.

options.filter

Type: Function[], Function Default value: []

Defines which filters apply to images found in the input stylesheet. Each filter called with image object, explained below. Each filter must return Boolean or **thenable Promise**, that will be resolved with Boolean. Each filter applies in series.

options.groupBy

Type: Function[], Function Default value: []

Defines logic of how to group images found in the input stylesheet. Each grouper called with image object, explained below. Each filter must return String|Null or **thenable Promise**, that will be resolved with String|Null. Each grouper applies in series.

options.accumulate

Type: Boolean Default value: false

Tells sprite-generator to accumulate images from multiple stylesheets. This mean, that images, found in stylesheet A.css and B.css will be accumulated and grouped in common sprite.

Note, that if `options.accumulate == true` then `options.styleSheetName` will not be used.

options.verbose

Type: Boolean Default value: false

Filtering and grouping

Sprite generator can filter and group images from the input stylesheet.

Built in filters:

- based on meta skip boolean flag;
- based on `fs.exists` method to check, whether file exists or not.

Built in groupers:

- based on @2x image naming syntax, will produce `sprite.@{number}x.png` naming. (@{number}x image group).

You can of course define your own filters or groupers. It will all based on main argument - the image object.

The Image object

Every filter or grouper is called with `image` object, that have these properties:

property	type	explanation
replacement	String	String, found by pattern in the input stylesheet
url	String	Url for image found in the input stylesheet
path	String	Resolved path for the image
group	String[]	List of string, representing groups of image
isRetina	Boolean	Boolean flag of retina image (@2x syntax)
retinaRatio	Number	Ratio of retina image (@2x, @3x => 2, 3)
meta	Object	Object of meta properties, defined in doc block (will explain below).

Doc block meta properties

You can also define some properties for the filters and groupers in doc block via this syntax:

```
{css definition} /* @meta {valid json} */
```

Example:

```
.my_class {
  background-image: url("/images/my.png"); /*
}
```

Important! Only object in sprite property of meta will be available in image object for filters and groupers.

Flexible example

```
var gulp    = require('gulp'),
    sprite  = require('gulp-sprite-generator'),
    Q       = require('q'),
    sizeof  = require('image-size');

gulp.task('sprites', function() {
  var spriteOutput;

  spriteOutput = gulp.src("./src/css/*.css")
    .pipe(sprite({
      baseUrl:          "./",
      spriteSheetName:  "sprite.png",
      spriteSheetPath:  "/dist/image",
      styleSheetName:   "stylesheet.css",

      filter: [
        // this is a copy of built in filter
        // do not forget to set it up in
        function(image) {
          return !image.meta.skip;
        }
      ],

      groupBy: [
        // group images by width
        // useful when building background
        function(image) {
          var deferred = Q.defer();
```

```

    sizeOf(image.path, function() {
      deferred.resolve(size.width);
    });

    return deferred.promise;
  }
]
});

spriteOutput.css.pipe(gulp.dest("./dist/css"));
spriteOutput.img.pipe(gulp.dest("./dist/images"));
});

```

License

MIT © **Sergey Kamardin**

You Need Help About npm Legal Stuff

Documentation	About npm, Inc	Terms of Use
Support / Contact Us	Jobs	Code of Conduct
Registry Status	npm Weekly	Package Name Disputes
Website Issues	Blog	Privacy Policy
CLI Issues	Twitter	Reporting Abuse
Security	GitHub	Other policies

npm loves you