入门指南

http://www.gulpjs.com.cn/docs/getting-started/

1. 全局安装 gulp：

$ npm install --global gulp

2. 作为项目的开发依赖（devDependencies）安装：

$ npm install --save-dev gulp

3. 在项目根目录下创建一个名为 gulpfile.js 的文件：

var gulp = require('gulp');

gulp.task('default', function() {

// 将你的默认的任务代码放在这

});

4. 运行 gulp：

$ gulp

默认的名为 default 的任务（task）将会被运行，在这里，这个任务并未做任何事情。

想要单独执行特定的任务（task），请输入 gulp <task> <othertask>。

下一步做什么呢？

你已经安装了所有必要的东西，并且拥有了一个空的 gulpfile。那怎样才算是\_\_真的\_\_入门了呢？可以查看这些 [秘籍](http://www.gulpjs.com.cn/docs/recipes/) 和这个 [文章列表](http://www.gulpjs.com.cn/docs/#articles/) 来学习更多的内容。

.src, .watch, .dest, CLI 参数 - 我该怎么去用这些东西呢？

要了解 API 规范文档，请查看 [API 文档](http://www.gulpjs.com.cn/docs/api/).

可用的插件

gulp 开发社区正在快速成长，每天都会有新的插件诞生。在 [主站](http://gulpjs.com/plugins/) 上可以查看完整的列表。

````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````

输出（Emits）符合所提供的匹配模式（glob）或者匹配模式的数组（array of globs）的文件。 将返回一个 [Vinyl files](https://github.com/wearefractal/vinyl-fs) 的 [stream](http://nodejs.org/api/stream.html) 它可以被 [piped](http://nodejs.org/api/stream.html#stream_readable_pipe_destination_options) 到别的插件中。

gulp.src('client/templates/\*.jade')

.pipe(jade())

.pipe(minify())

.pipe(gulp.dest('build/minified\_templates'));

glob 请参考 [node-glob 语法](https://github.com/isaacs/node-glob) 或者，你也可以直接写文件的路径。

#### globs

类型： String 或 Array

所要读取的 glob 或者包含 globs 的数组。

#### options

类型： Object

通过 [glob-stream](https://github.com/wearefractal/glob-stream) 所传递给 [node-glob](https://github.com/isaacs/node-glob) 的参数。

除了 [node-glob](https://github.com/isaacs/node-glob#options) 和 [glob-stream](https://github.com/wearefractal/glob-stream) 所支持的参数外，gulp 增加了一些额外的选项参数：

#### options.buffer

类型： Boolean 默认值： true

如果该项被设置为 false，那么将会以 stream 方式返回 file.contents 而不是文件 buffer 的形式。这在处理一些大文件的时候将会很有用。\*\*注意：\*\*插件可能并不会实现对 stream 的支持。

#### options.read

类型： Boolean 默认值： true

如果该项被设置为 false， 那么 file.contents 会返回空值（null），也就是并不会去读取文件。

#### options.base

类型： String 默认值： 将会加在 glob 之前 (请看 [glob2base](https://github.com/wearefractal/glob2base))

如, 请想像一下在一个路径为 client/js/somedir 的目录中，有一个文件叫 somefile.js ：

gulp.src('client/js/\*\*/\*.js') // 匹配 'client/js/somedir/somefile.js' 并且将 `base` 解析为 `client/js/`

.pipe(minify())

.pipe(gulp.dest('build')); // 写入 'build/somedir/somefile.js'

gulp.src('client/js/\*\*/\*.js', { base: 'client' })

.pipe(minify())

.pipe(gulp.dest('build')); // 写入 'build/js/somedir/somefile.js'

### gulp.dest(path[, options])

能被 pipe 进来，并且将会写文件。并且重新输出（emits）所有数据，因此你可以将它 pipe 到多个文件夹。如果某文件夹不存在，将会自动创建它。

gulp.src('./client/templates/\*.jade')

.pipe(jade())

.pipe(gulp.dest('./build/templates'))

.pipe(minify())

.pipe(gulp.dest('./build/minified\_templates'));

文件被写入的路径是以所给的相对路径根据所给的目标目录计算而来。类似的，相对路径也可以根据所给的 base 来计算。 请查看上述的 gulp.src 来了解更多信息。

#### path

类型： String or Function

文件将被写入的路径（输出目录）。也可以传入一个函数，在函数中返回相应路径，这个函数也可以由 [vinyl 文件实例](https://github.com/wearefractal/vinyl) 来提供。

#### options

类型： Object

#### options.cwd

类型： String 默认值： process.cwd()

输出目录的 cwd 参数，只在所给的输出目录是相对路径时候有效。

#### options.mode

类型： String 默认值： 0777

八进制权限字符，用以定义所有在输出目录中所创建的目录的权限。

### gulp.task(name[, deps], fn)

定义一个使用 [Orchestrator](https://github.com/robrich/orchestrator) 实现的任务（task）。

gulp.task('somename', function() {

// 做一些事

});

#### name

任务的名字，如果你需要在命令行中运行你的某些任务，那么，请不要在名字中使用空格。

#### deps

类型： Array

一个包含任务列表的数组，这些任务会在你当前任务运行之前完成。

gulp.task('mytask', ['array', 'of', 'task', 'names'], function() {

// 做一些事

});

**注意：** 你的任务是否在这些前置依赖的任务完成之前运行了？请一定要确保你所依赖的任务列表中的任务都使用了正确的异步执行方式：使用一个 callback，或者返回一个 promise 或 stream。

#### fn

该函数定义任务所要执行的一些操作。通常来说，它会是这种形式：gulp.src().pipe(someplugin())。

#### 异步任务支持

任务可以异步执行，如果 fn 能做到以下其中一点：

##### 接受一个 callback

// 在 shell 中执行一个命令

var exec = require('child\_process').exec;

gulp.task('jekyll', function(cb) {

// 编译 Jekyll

exec('jekyll build', function(err) {

if (err) return cb(err); // 返回 error

cb(); // 完成 task

});

});

##### 返回一个 stream

gulp.task('somename', function() {

var stream = gulp.src('client/\*\*/\*.js')

.pipe(minify())

.pipe(gulp.dest('build'));

return stream;

});

##### 返回一个 promise

var Q = require('q');

gulp.task('somename', function() {

var deferred = Q.defer();

// 执行异步的操作

setTimeout(function() {

deferred.resolve();

}, 1);

return deferred.promise;

});

**注意：** 默认的，task 将以最大的并发数执行，也就是说，gulp 会一次性运行所有的 task 并且不做任何等待。如果你想要创建一个序列化的 task 队列，并以特定的顺序执行，你需要做两件事：

* 给出一个提示，来告知 task 什么时候执行完毕，
* 并且再给出一个提示，来告知一个 task 依赖另一个 task 的完成。

对于这个例子，让我们先假定你有两个 task，"one" 和 "two"，并且你希望它们按照这个顺序执行：

1. 在 "one" 中，你加入一个提示，来告知什么时候它会完成：可以再完成时候返回一个 callback，或者返回一个 promise 或 stream，这样系统会去等待它完成。
2. 在 "two" 中，你需要添加一个提示来告诉系统它需要依赖第一个 task 完成。

因此，这个例子的实际代码将会是这样：

var gulp = require('gulp');

// 返回一个 callback，因此系统可以知道它什么时候完成

gulp.task('one', function(cb) {

// 做一些事 -- 异步的或者其他的

cb(err); // 如果 err 不是 null 或 undefined，则会停止执行，且注意，这样代表执行失败了

});

// 定义一个所依赖的 task 必须在这个 task 执行之前完成

gulp.task('two', ['one'], function() {

// 'one' 完成后

});

gulp.task('default', ['one', 'two']);

### gulp.watch(glob [, opts], tasks) 或 gulp.watch(glob [, opts, cb])

监视文件，并且可以在文件发生改动时候做一些事情。它总会返回一个 EventEmitter 来发射（emit） change 事件。

### gulp.watch(glob[, opts], tasks)

#### glob

类型： String or Array

一个 glob 字符串，或者一个包含多个 glob 字符串的数组，用来指定具体监控哪些文件的变动。

#### opts

类型： Object

传给 [gaze](https://github.com/shama/gaze) 的参数。

#### tasks

类型： Array

需要在文件变动后执行的一个或者多个通过 gulp.task() 创建的 task 的名字，

var watcher = gulp.watch('js/\*\*/\*.js', ['uglify','reload']);

watcher.on('change', function(event) {

console.log('File ' + event.path + ' was ' + event.type + ', running tasks...');

});

### gulp.watch(glob[, opts, cb])

#### glob

类型： String or Array

一个 glob 字符串，或者一个包含多个 glob 字符串的数组，用来指定具体监控哪些文件的变动。

#### opts

类型： Object

传给 [gaze](https://github.com/shama/gaze) 的参数。

#### cb(event)

类型： Function

每次变动需要执行的 callback。

gulp.watch('js/\*\*/\*.js', function(event) {

console.log('File ' + event.path + ' was ' + event.type + ', running tasks...');

});

callback 会被传入一个名为 event 的对象。这个对象描述了所监控到的变动：

##### event.type

类型： String

发生的变动的类型：added, changed 或者 deleted。

##### event.path

类型： String

触发了该事件的文件的路径。